**UNIVERSITY OF READING**

SCHOOL OF MATHEMATICAL AND PHYSICAL SCIENCES

DEPARTMENT OF METEOROLOGY

# How does the skill of global model precipitation forecasts over Europe depend on spatial scale?

---

**Bethan Jones**

**August 2014**

A dissertation submitted in partial fulfilment of the requirements for the degree of MSc. Applied Meteorology.

# *Abstract*

The Fractions Skill Score (FSS) is a method that has been used to analyse the skill of convective-allowing models at forecasting precipitation on a spatial scale. This report looks at applying the FSS to a sub-area of Met Office global model data, covering Eastern Europe. The scale at which FSS first reaches 0.5 is defined as the useful scale, or the scale at which the forecast is considered to have a greater than 50% chance in being correct. This useful scale has been compared for different forecast ranges (from T+24 to T+48) and also for the different seasons. The time range analysed is from 1$^{st}$ June 2013 to 31$^{st}$ May 2014.

Code was written specifically in python to analyse the data using the FSS at a range of neighbourhood sizes. The code was tested and checked to ensure that results were in line with results found in research using the same method.

It has been shown that the useful scale is smaller for winter and autumn implying that it is easier to forecast accurately during these seasons compared to spring and summer. For T+24 forecasts, spring is the most unreliable whereas for T+48, it is the summer season. It is proposed that this is to do with the timescales on which convective showers in spring evolve.

Over the whole year, 92% of T+24 forecasts can be considered skillful at grid scale (25km), dropping to 52% at T+48. 93% of T+48 forecasts are skillful at scale of 75km (3 grid squares) or less.

# *Acknowledgments*

I would like to thank my supervisors Bob Plant and Nigel Roberts for all their invaluable support and help they have given me in completing this project.

I would also like to thank my family for encouraging me and supporting me.

Finally, I would like to thank my fellow MSc students for their help and willingness to help talk through problems both meteorological and code-based.

*Contents*

# 1. Introduction

## 1.1 Traditional skill score methods for precipitation

How the skill of a precipitation forecast is measured has been a highly researched area for many years (Jolliffe and Stephenson, 2012). Traditionally, the skill is measured by comparing the forecast quantity with the observed amount at point locations (Mittermaier, 2014). Statistical methods include mean and RMS values, or 'categorical scores' such as the Equitable Threat Score. Such methods are used by national meteorological services like the UK Met Office and have been in place since Numerical Weather Prediction (NWP) models have been run. Whilst these methods are easy to implement and seem intuitively sensible, they are not without their problems. One of the most basic problems is called the 'double penalty problem', discussed later, which means that forecasts that give a good spatial representation score just as poorly as the forecasts that place occurrences of precipitation further from the observed location. This issue has come to light in recent years because of the development of 'convection-permitting' models, as will be explained.

## 1.2 Model resolution

In recent years, the numerical weather prediction models used by organisations such as the Met Office have been constantly upgraded. Global models such as the Met Office's Global Model now run at a higher resolution of 25km (Met Office, 2014). This means that more parameters can be modelled explicitly rather than rely on parameterisations, as was the case in previous models (Davies, 2004).

## 1.3 Importance of site specific forecasting

The weather has a large effect on people's lives, from leisure and sport (Thorne, 1977) to health (Kunkel *et al*, 1999) to job interviews (Simonsohn, 2009). There is an increasing demand for site specific forecasting for events such as Wimbledon or the Olympic games (Golding, 2012) as well as for everyday uses including energy availability at wind farms (Landberg, 1999). Event managers want to know what the chances of events being cancelled or delayed due to rain or even just what to prepare for. Like in big multinational shops, sales at events are heavily influenced by the weather (Murray *et al*, 2010). We have to accept that forecasts are never perfect – with some locations harder to forecast for than others, but in some cases, even a forecast that does not exactly match the observations in intensity or placing can provide important information. For example, if storms are forecast nearby then that indicates a threat at the location of interest.

## *1.4 Problems with traditional methods*

An important question for meteorologists is how accurate can a forecast be at that scale and at what scale can forecasts be considered more useful than guessing. Using the traditional skill score methods, the skill of the higher resolution models is lower than that of their coarser predecessors (Mittermaier *et al*, 2011).

One of the reasons for the decrease in skill is what is commonly referred to as the 'double penalty problem'. This refers to the fact that when looking over a spatial scale, the precipitation forecast is compared with that observed for each grid square. When the forecast does not exactly correspond to the observations, as shown in Figure 1, then a double penalty occurs, one where the observation says rain but the forecast says not and another where the observations say no rain but the forecast says rain.

**Figure 1: The double penalty problem, means that errors in the location of a band of precipitation are counted twice, once where the model shows the precipitation and once where the observations locate the precipitation. The observations are shown in blue and the forecast in purple. Therefore, both forecasts shown in this figure are considered equally incorrect and the best results would come from not providing a forecast at all.**

## *1.5 The Global Model*

Many of the national meteorological organisations run their own global models. The Met Office Unified Model (UM) is one such model. The skill of the models can be compared against each other using traditional skill score methods. Such a comparison can be seen in Figure 2, which shows the root-mean-square-error for the T+72 forecast of the Northern Hemisphere sea level pressure (Met Office, 2011). The global models tend to run at a lower resolution than the more specialised regional and local models used to produce the forecasts for the areas of interest to national meteorological organisations but provide important information to these nested models.

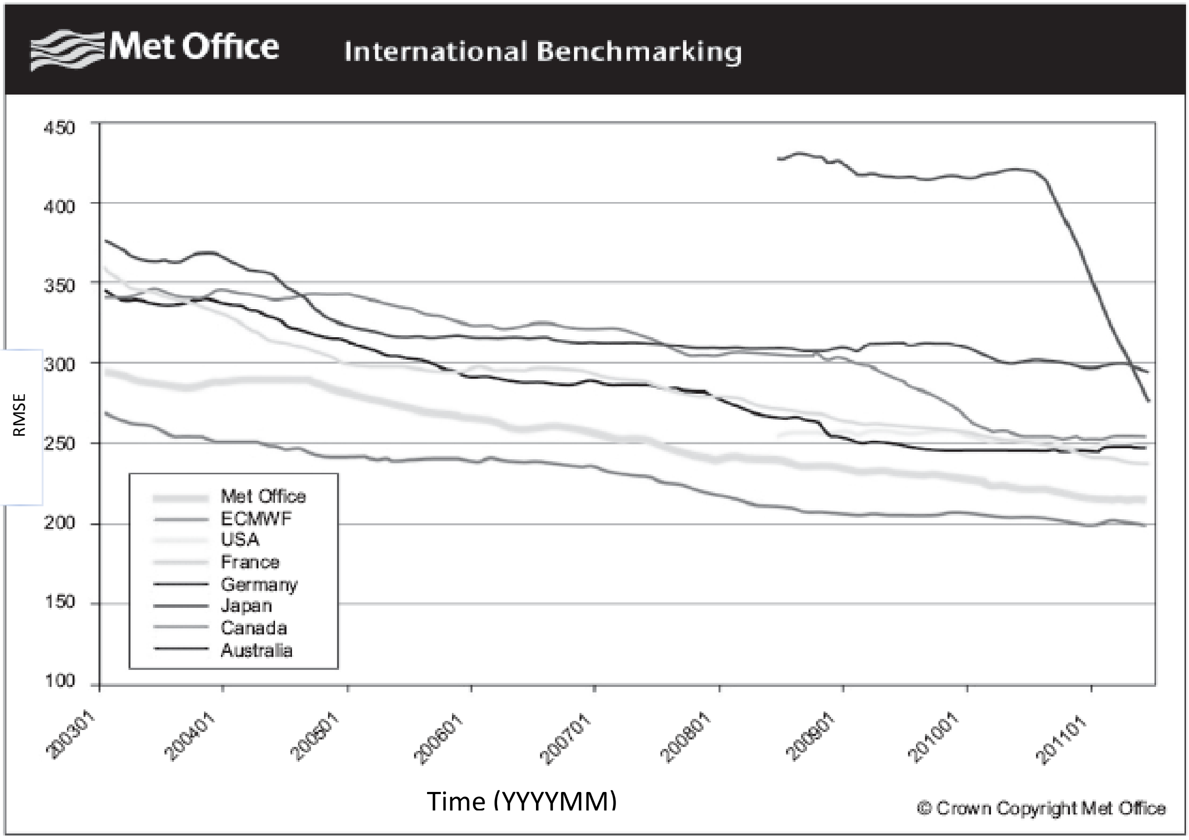



**Figure 2: Comparison of global models run by different Meteorological Organisations, including the Met Office and ECMWF (Source: Met Office 2011).**

## *1.6 Global Scale*

Whilst much research has been conducted into how to measure the skill on a local or regional scale (Gilleland, 2009) there has not been a lot of research on a global scale. Working on a global scale brings its own set of problems in that the shape of the earth means that grid squares are not consistent in size. Verification tests often use rainfall accumulations at point locations, measured using gauges, to test against (Mittermaier, 2014). These scores tend to be poor unless long accumulation times are examined. Therefore, a test which verifies precipitation rates, such as radar or global analyses would be considered a stringent test on such a scale.

## *1.7 Observations on a global scale*

On a global scale one of the major problems is the lack of global observations. Over the ocean there are only infrequent observations from ships, planes, satellite. There are also sparse observations in some landed areas, particularly those that are difficult to access or where funding is poor, as seen in Figure 3. Many methods use radar observations as the observational dataset but there is no global coverage.

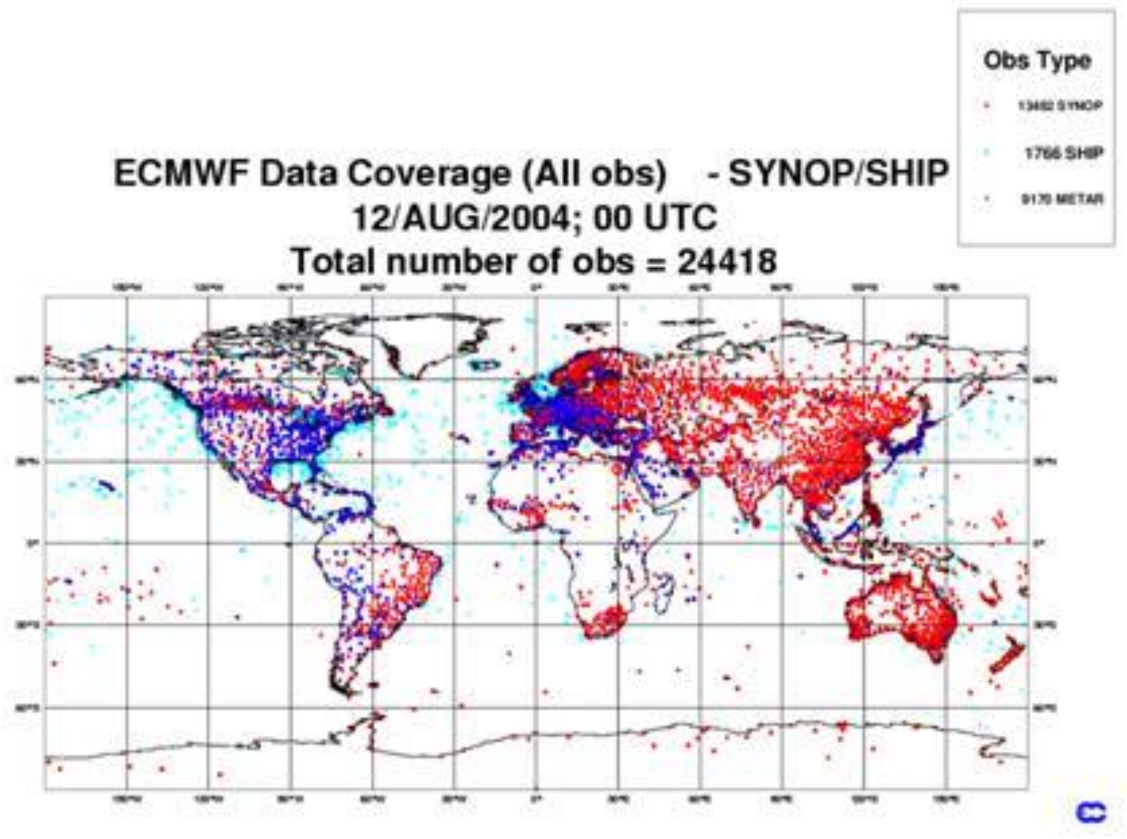



**Figure 3: Global Data coverage from synoptic stations (red), METAR (dark blue) and ships (light blue) in 2004 (ECMWF, 2014). Shows lack of data over oceans and some land masses such as Africa, Greenland.**

## *1.8 Uses of the global model*

The global model is used in a variety of situations; as a forecasting tool for areas not covered by higher resolution regional or national scale models, to edge data for regional models (Brandt, 2005). When used to give the boundary conditions of nested high resolution models, it is important to know the reliability of the global model and the scale to which it is reliable. Large displacement errors in the global models can have a large effect on the skill of the nested regional or local scale model as events such as fronts are not in the right place.

## *1.9 Usefulness*

A forecast is deemed to be 'useful' if it accurate more than 50% of the time (Mittermaier and Roberts, 2010). Below this, a random forecast deciding by a coin toss would be correct more often and therefore the forecast is deemed to be not of any use. The purpose of this project is to attempt to show the scale at which forecasts become useful at different forecast lead times. Throughout this work, the term 'useful scale' is used to define the scale at which the skill score first reaches 0.5.

# 2. Literature Review

## 2.1 UM Models

The global model (25km grid spacing (Met Office, 2014)), or any model with grid spacing of greater than 4km, uses convection parameterisation to model convective precipitation (Gregory and Rowntree (1990), Gregory *et al* (1990)). This means that there are some differences in the models, as described by Lean *et al* (2008) and in the distribution of precipitation (Holloway *et al*, 2012). The convection permitting models were introduced with the aim of improving forecasts of convective and orographic rain, with the hope of a better Quantitive Precipitation Forecast (See section 2.2).

Rather than improving the skill of precipitation forecasts as expected, traditional verification methods showed little improvement or even a decrease in skill when going to a convection permitting model, as can be seen in Figure 4 (Mittermaier *et al*, 2013). The equitable threat score (ETS) is a skill score used to define the skill of precipitation forecasts at locating precipitation above a threshold in the correct place (Mesinger, 2008).

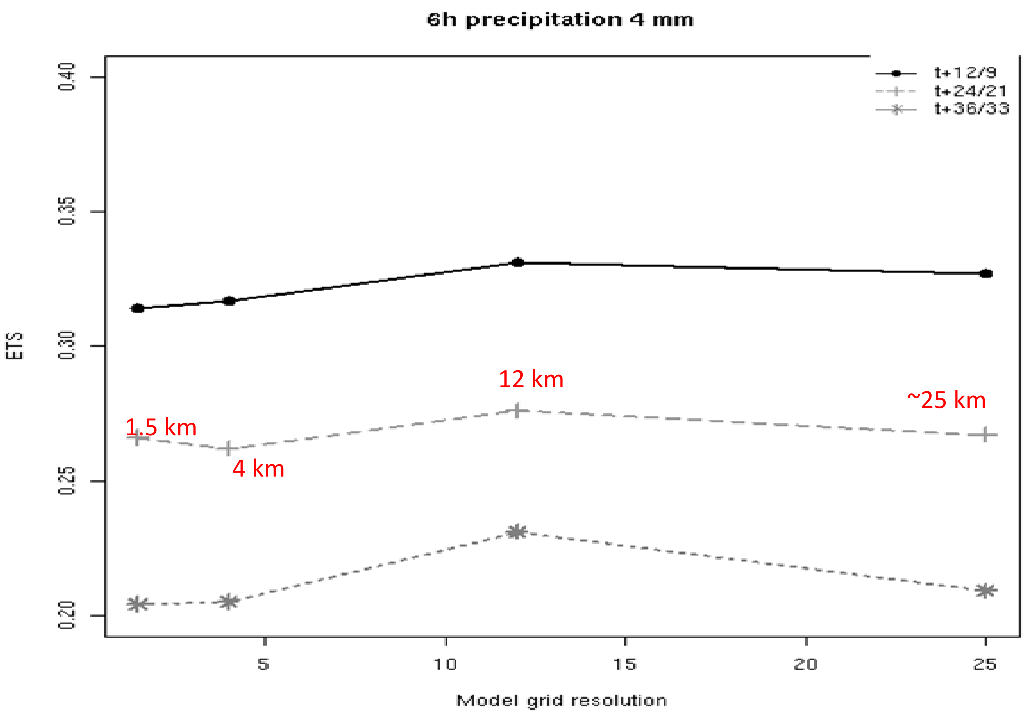



**Figure 4: Equitable Threat score at different model resolutions (Mittermaier *et al*, 2011)**

This lack of improvement is not because the new convection-permitting models are poorer but because traditional verification methods that measured skill are no longer appropriate because they do not take any account of one forecast being closer than the other (Ebert, 2008). This was recognised and therefore new methods were developed. It was also recognised that probabilistic forecasts were needed.

## *2.2 Quantitive Precipitation Forecasting*

In recent years there has been an increase in probabilistic forecasting, particularly from ensemble methods but also from a deterministic model. Quantitive precipitation forecasting (QPF) has a high impact on decision-making and allows the end user to make an informed decision even if they are not necessarily a meteorologist. Research falls into two main sources, weather radar and NWP models (Wang *et al*, 2008).

Theis *et al* (2005) propose a low-budget post processing procedure to create a pseudo-ensemble and probabilistic forecast. Their method uses the concept of a 'neighbourhood', the group of pixels surrounding the pixel in question in space and/or time, an example is shown in Figure 5. Theis *et al* make one key assumption that *"Model precipitation forecasts at grid points within the neighbourhood are assumed to constitute a sample from the unknown probability density function of the precipitation forecast at location ($x_0$, $y_0$) and forecast lead time $T_0$."* By comparing each pixel to a threshold, a percentage of pixels within the neighbourhood which exceed the threshold can be calculated. This is the value returned as the percentage probability of precipitation. Roberts (2003) also shows the need for probabilistic post processing. The techniques used to produce a QPF can be expanded to also provide verification of forecast skill.

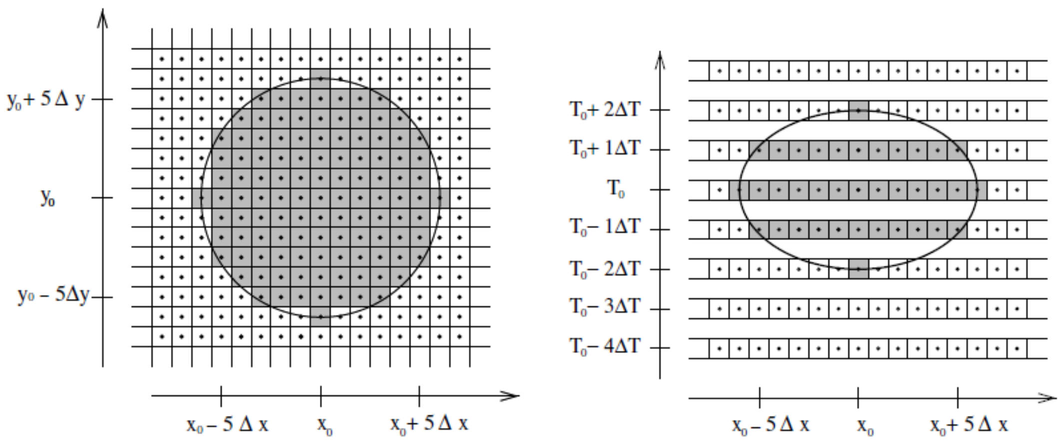



**Figure 5: Example of a spatial neighbourhood is shown on the left and a spatio-temporal neighbourhood on the right. (Theis *et al*, 2005).**

## *2.3 Traditional Skill Score Methods*

Reasons for forecast verification fall under 3 main headings- administrative, scientific and economic (Brier and Allen, 1951). The purpose of this project lies under the scientific verification heading. Traditional skill score methods such as Root Mean Square error or the equitable threat score are used to determine the skill of a forecast (Inness and Dorling, 2013).

Another method is the Briar Skill Score (BSS) which is commonly used for comparing forecast probability against climatological probability of an event (Jolliffe and Stephenson, 2012) However, such methods do not account for spatial errors and therefore can represent the accuracy of the forecast unreliably in high resolution models.

## *2.4 Alternative Skill Score Methods*

The apparent lack of improvement shown by traditional methods for convection-resolving models is to do with the problem of using traditional skill score methods for measuring the skill of precipitation forecasts at grid scale. High resolution precipitation fields contain more noise and therefore are more prone to missing correctly positioning the precipitation even if they give realistic-looking forecasts. The closeness and realism of forecasts is not rewarded and this has led to the development of new spatial and spatio-temporal methods. Gilleland *et al* (2009) grouped these into four categories for easy comparison (shown in Figure 6). The four categories are; feature-based, field deformation, neighbourhood and scale-separation. The first two categories both work by looking at the displacement of the forecast from the observations, but whilst field deformation looks at the entire field, features-based approaches look at specific features of interest such as storm cells, each analysed separately. Neighbourhood and scale separation are also similar in that they apply a spatial filter to a field to calculate verification statistics.

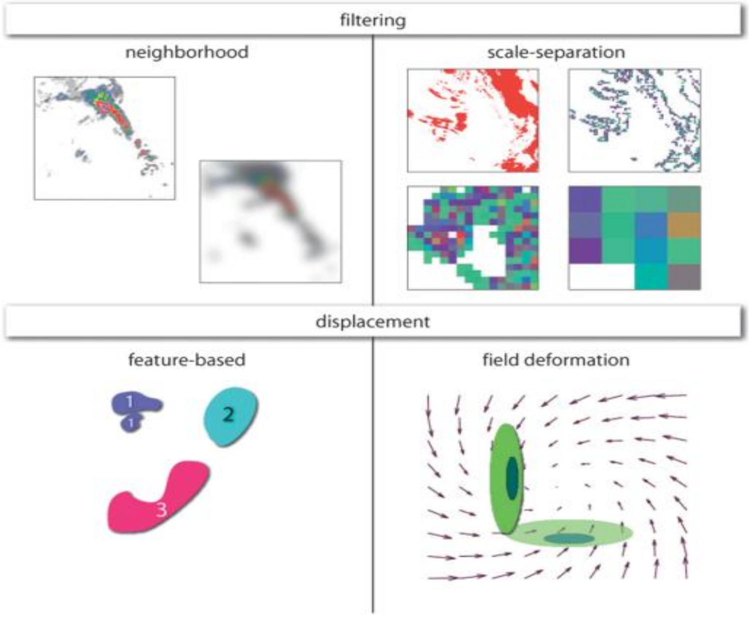



**Figure 6: The four groups of skill score methods, Neighbourhood, scale-separation, feature-based and field deformation (Gilleland *et al*, 2009).**

Gilleland *et al* (2009) go on to discuss the situations these methods are used for and discuss the pros and cons of various methods within each group.

Some methods such as that used by Mittermaier (2014) use traditional methods such as the Brier score, ranked probability or continuous ranked probability scores , applied at different scales to verify Near-Convection-Resolving model forecasts against observing sites. Casati *et al* (2004) use the Mean Square Error (MSE) for binary images on different spatial scales to verify QPFs.

Weusthoff *et al* (2010) used a neighbourhood verification method to assess the skill of convection-permitting models by evaluating three different pairs of models. They concluded that *"The differences between the models are significant and robust against small changes in the verification settings. An evaluation based on individual months shows that high-resolution models give better results, particularly with regard to convective, more localized precipitation events.*

## *2.5 Fractions Skill Score*

The Fractions Skill Score, or FSS (Roberts (2005), Roberts and Lean (2008)) is a method in the neighbourhood group and is the verification method used in this project. The method does not require the identification of features, as others do, so can run through data without human input. The method is designed to show how the skill varies with neighbourhood size, and determine the smallest scale at which the forecasts are deemed useful. The FSS is a variation of the Briar Skill Score

(Brier, 1950), but used to make a spatial comparison, rather than point comparisons between forecast and observation.

The model and observational data, usually from radar, are projected onto an identical verification grid. These are compared against suitable thresholds to create binary fields such that:

$$I_O = \begin{cases} 1 \; O_r \geq q \\ 0 \; O_r \leq q \end{cases} \quad , \quad I_M = \begin{cases} 1 \; M_r \geq q \\ 0 \; M_r \leq q \end{cases}$$

Where $I_O$ is the pixel value in the observations binary grid, $I_M$ is the pixel value in the forecast binary grid, $O_r$ is the pixel value in the observations grid, $M_r$ is the pixel value in the forecast grid and q is the threshold value.

Roberts and Lean (2008) go on to recommend that a percentile, rather than an accumulation threshold is used. A percentile threshold takes the top percentage of grid locations with the heaviest rainfall and compares these. A 90% threshold of a 100 cell grid for example, would take the highest 10 rainfall rates over the whole grid. This allows for the effect of spatial scale to be observed by negating the effects of bias in the forecast (e.g. the model produces too much heavy rain). At larger scales, a biased forecast always gives lower FSS values and the same is usually true at smaller scales. However, it is possible at smaller scales for a higher score to be given by biased forecasts, in certain cases. This can be reduced by using a large sample of forecasts (Mittermaier and Roberts, 2010).

From this a series of grids can be produced each showing the fraction of pixels around it that contain rain above the threshold (=1) on different scales, e.g. 1,3,5 grid cells. This fraction is the percentage precipitation forecast that Theis *et al* (2005) proposed. This can be written in equation form as:

$$O(n)(i,j) = \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} I_0\left[i+k-1-\frac{(n-1)}{2}, j+l-1-\frac{(n-1)}{2}\right], \textbf{(Eq. 1)}$$

$$M(n)(i,j) = \frac{1}{n^2}\sum_{k=1}^{n}\sum_{l=1}^{n} I_M\left[i+k-1-\frac{(n-1)}{2}, j+l-1-\frac{(n-1)}{2}\right], \textbf{(Eq. 2)}$$

For each of these grids, a MSE error can be found using:

$$\boldsymbol{MSE_{(n)}} = \frac{\mathbf{1}}{N_x N_y}\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\left[\boldsymbol{O_{(n)(i,j)}} - \boldsymbol{M_{(n)(i,j)}}\right]^{\mathbf{2}} \quad \textbf{(Eq. 3)}$$

and from this the FSS can be calculated using:

$$FSS = 1 - \frac{MSE_{(n)}}{MSE_{(n)ref}} \quad \textbf{(Eq. 4)}$$

where: $MSE_{(n)}$ is the mean square error of the model to observations (Equation 3) and $MSE_{ref}$ is defined as (Roberts and Lean, 2008):

$$MSE_{(n)} = \frac{1}{N_x N_y}\left[\sum_{i=1}^{N_x}\sum_{j=1}^{N_y} O_{(n)(i,j)}^2 + \sum_{i=1}^{N_x}\sum_{j=1}^{N_y} M_{(n)(i,j)}^2\right] \textbf{ (Eq. 5)}$$

An alternative view of this is:

$$FSS = 1 - \frac{FBS_{mod}}{FBS_{worst}} \textbf{(Eq. 6)}$$

Where FBS is the Fractions Brier Score and $FBS_{worst}$ is the '" *largest Fractions Brier Score (FBS) that could be obtained from the forecast and observed fractions when there is no collocation of non-zero fractions and therefore the worst possible FBS*" (Roberts, 2008).

The FSS can then be plotted against scale, resulting in a graph similar to the example in Figure 7.

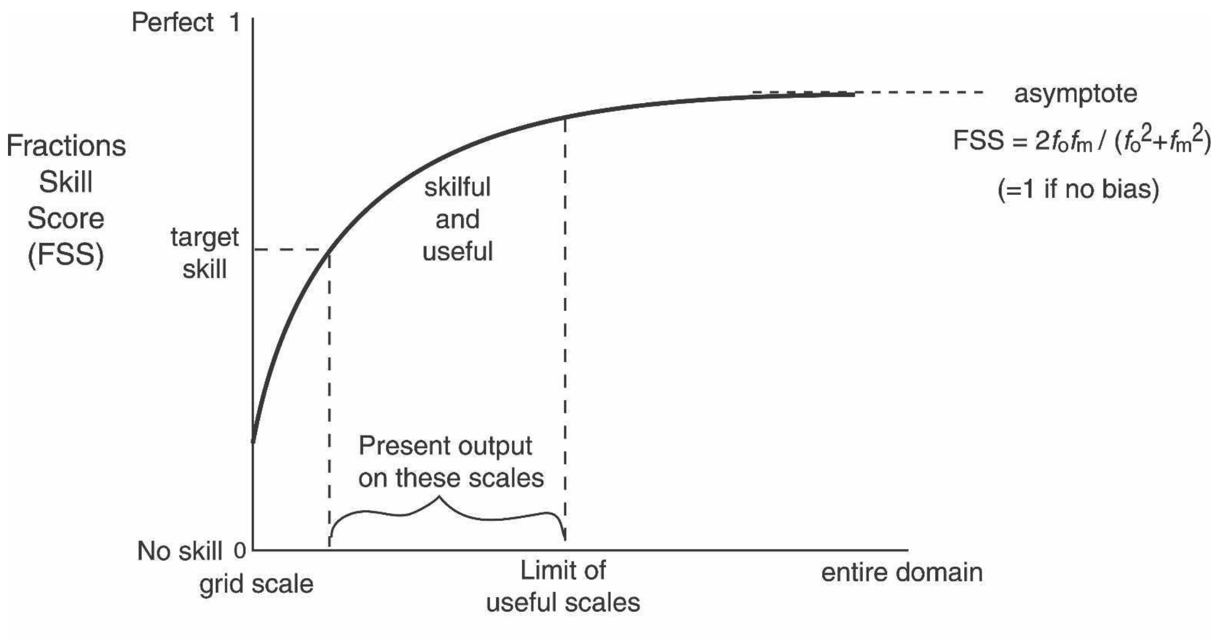



**Figure 7: FSS against scale (Source: Roberts and Lean, 2008).**

## *2.6 Results from Regional Models*

Like the QPF method used by Theis *et al* (2005), in Section 2.1, the FSS can be used over both a spatial and spatio-temporal neighbourhood. Mittermaier (2013) shows that "the 'convection permitting' (4km) Unified Model (MetUM) forecasts are better than the 12km MetUM , significant at the 5% level." This is the result that was expected but not shown from the traditional skill score methods. The method can also be used on QPF forecasts (Zacharov and Rezacova, 2009)

At local scale, the forecasts can be compared with an identical verification grid using radar data. The UK radar network covers most of the UK to a resolution of 5km, making it suitable for comparison. However, the use of radar data comes with its own problems, including beam blockage from obstacles, attenuation and brightband (Harrison *et al* (2000), Wang *et al* (2008), Met Office (2009)).

## *2.7 Global model forecasts*

On the global scale, it has been shown that the global model forecast skill deteriorates with forecast lead time, but like regional and local models, this has improved in the last few decades (Figure 8).

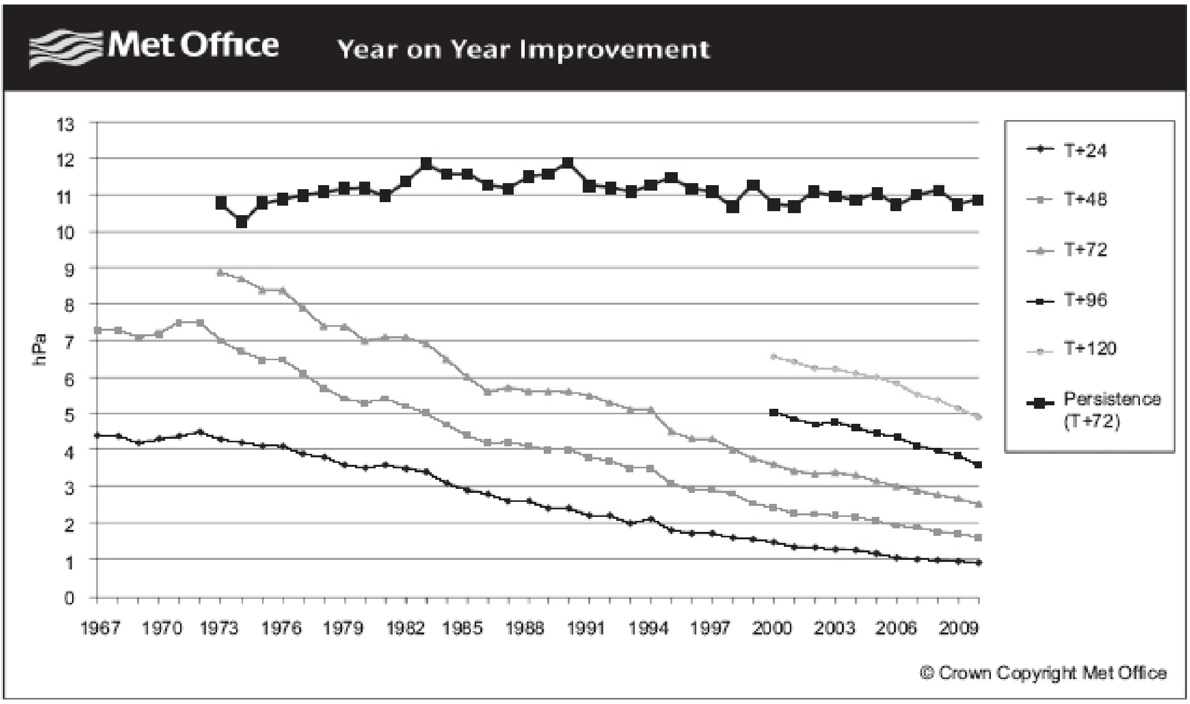



**Figure 8: Improvement of forecasts of Sea Surface Pressure since 1967 (Source: Met Office, 2011).**

However, at the global scale there are difficulties in measuring the skill. The global observation network is patchy at best. Whilst some areas, mostly in developed western countries, observations are plentiful, over other areas, such as West Africa, observation stations are more spread out (ECMWF, 2011). Over the ocean, observations are reduced to those made by ships and planes. Main shipping or flying routes therefore may be well covered but other parts are not so well covered. Observation points fed into the WMO database can be seen in Figure 3, highlighting the lack of observations over the oceans. New initiatives such as floating buoys and Argo floats mean that more data is being provided (Roemmich *et al*, 2009). Data can also be received from satellites orbiting the

Earth, however complete global coverage at any one time is problematic. It is possible to verify the results of the global model over a regional area using observational data such as ARM data in the tropical Western Pacific (Culverwell, 2000).

The skill of a global model can only be detected for smooth fields like the 500hPa geopotential height or surface pressure or for rainfall accumulations over 24 hours (that are very smoothed). Whilst this is useful to a certain extent, it does not tell us about the quality of the forecast for the middle of the next day, or the day after. It also can mean that, like at lower scales, the double penalty problem can imply that two forecasts are equally poor, when in reality one is much closer in position than the other.

It should be noted that so far, no research has been published using the FSS, or any other spatial approach, on a global scale, that the author is aware of.

# 3. Methodology

## 3.1 Data

The data used in this project is from the Met Office Global Model, from 1st July 2013 to 31st May 2014. It was extracted from the Met Office operational forecast archive. Unfortunately, within this period the forecast data for some days are missing, notably the 10th-12th December 2013. The field being examined is the precipitation rate at 12UTC. The precipitation rate includes rain and snow. Precipitation is either represented explicitly on the model grid or output from the convection parameterisation scheme. The data fits a grid of 769 by 1024 grid squares, which covers the whole globe. This is approximately 25km grid spacing at UK latitude, however the spherical shape of the earth means that the grid size varies with latitude, particularly at the poles. It was not possible to apply the FSS methodology to the whole global grid and therefore a sub-area of 128 by 150 grid squares has been used, covering Western Europe, as seen in Figure 9. This area is between 30 and 60 degrees north and proceeds east from the Greenwich meridian line. For each day the precipitation rate from the forecast 'analysis', time t=0, is used as the observational dataset against which the forecast precipitation rates are compared. Because these analyses are constructed using a previous forecast combined with new observational data, through the process of data assimilation, they are considered more accurate over land areas where there are more observations comparatively than over the ocean. This was an important factor in the choice of the sub-domain, which is largely over land. Each forecast starting at 12UTC was run for 7 days, which meant that there were 7 forecast times (T+24, T+48, T+72, T+96, T+120, T+144 and T+168) Analysis of the data showed that these were stored in a non-sequential order within the file so care was taken to ensure that the forecast lead time was correctly assigned within the code and that the correct forecasts were compared.

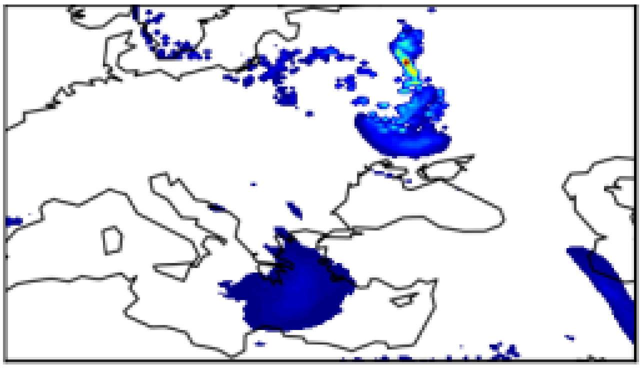


**Figure 9: Sub-area over Western Europe.**

### *3.2 Precipitation Rates*

Previous verification methods tested precipitation accumulations against gauge accumulations measured at point locations. (Mittermaier, 2014) However, as the observational grid being used to calculate the FSS is the global model analysis, which is naturally smoothed to the model grid and also because it is the spatial scale that is being observed, it is possible to verify precipitation rates at a snapshot in time, which is of interest.

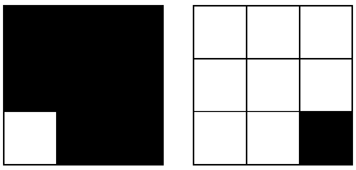
## *3.3 Calculating the Fractions Skill Score*

In order to calculate the FSS for large amounts of data, it was necessary to first create a series of Python programs, which were designed to loop through all the days of the year-long period and calculate FSS on all scales up to the maximum grid size, for the different forecast times. The method is in line with that used by Roberts and Lean (2005), discussed in Section 2.5. The procedure for doing this will now be described and the complete code can be found in Appendix A.

### *3.3.1 Thresholding*

First, a threshold was applied to pick out the model grid squares that exceed a particular value (e.g. 1 mm/hour) on both the forecast and analysis grid. A 99% (90%) threshold means that the top 1% (10%) of grid squares (sampled from the whole domain including zeroes) are deemed to have exceeded the threshold. All the grid squares that exceed the threshold are assigned a value of 1 and the others a value of 0.

a)



b)

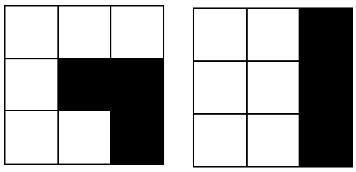


**Figure 10: Actual vs percentage thresholds. The example on the left (a) shows the forecast (far left) and observations binary grid for a threshold of 2mm/hr On the right (b) are the binary grids produced taking the top 33% (highest 3 values). These show a much better match between forecast and observations.**

The percentile thresholds used are 99% and 90%, unless otherwise stated. By using percentage thresholds, the FSS must asymptote to a value of 1 at a neighbourhood of the whole grid. A benefit of using a percentile threshold is that any over- or under-prediction of the rain in the forecasts compared to the analyses is removed in the comparison so that the focus can be on the spatial distribution of the precipitation. Using a set value as a rainfall rate can be problematic if either most of the rainfall is above that threshold or most of it below. For example, in Figure 10a, the forecast grid shows 8 out of 9 grid squares to be over 2mm/hr whereas the observations show just 1 out of 9. This would result in a low FSS value. If the top 33% were to be taken in each grid (Figure 10b), the FSS value would be higher, as the shape of the rainfall is close although the forecast is biased to higher rainfall amounts, just exceeding the set threshold whilst the observations show just under the threshold.

Once the model binary grid of 1s and 0s has been constructed, fractions can be computed. The first step in computing the fractions is to define a “neighbourhood” size. As an example, consider a square neighbourhood size of 3x3 pixels. Then consider a single target pixel on the model grid. The fraction for that pixel of interest is the number of pixels exceeding the threshold in the 3x3 neighbourhood centred at that pixel divided by the total number of pixels in the 3x3 neighbourhood. If the neighbourhood included pixels that were off the edge of the model grid then these pixels were assumed to be 0, an example is shown in Figure 11.

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| 2/9 | 2/9 | 3/9 | 2/9 | 2/9 |
| 3/9 | 3/9 | 3/9 | 2/9 | 2/9 |
| 3/9 | 4/9 | 3/9 | 2/9 | 1/9 |
| 2/9 | 3/9 | 2/9 | 2/9 | 1/9 |
| 1/9 | 2/9 | 2/9 | 2/9 | 1/9 |

**Figure 11: Fractions on a 5x5 grid for 3x3 neighbourhoods. The value of the shaded square is the fraction of the pixels in the neighbourhood shown on the right.**

However, when this process is scaled up to work for larger model grids and bigger neighbourhoods then it became highly inefficient to add up all the pixels in the neighbourhood s for each target pixel. A more efficient method, which is a common image-processing trick involving neighbourhoods and blurring was adapted to reduce the resources required (Russ, 2007).

This method uses the binary model grid to create a corresponding summed grid where each pixel takes the value of the sum of all the pixels above and to the left of it (Figure 12).

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 2 | 2 |
| 1 | 2 | 2 | 4 | 4 |
| 2 | 3 | 3 | 5 | 5 |
| 2 | 4 | 5 | 7 | 7 |
| 2 | 4 | 5 | 7 | 8 |

**Figure 12: Summation of binary field. The value of each pixel is the sum of all the pixels above and to the left of the pixel. The original binary field is shown on the left and the summed field on the right.**

From this summed grid, the fraction for any target pixel for any neighbourhood size can be calculated using just four numbers. To calculate the sum of the pixels in the neighbourhood, shown in Figure 13 for a target pixel shaded red the sum would simply be 7-2-2+0=3. Looking at the binary grid on the left shows three pixels with the value 1 within the neighbourhood. The fraction is simply this number divided by the total number of pixels in the neighbourhood.

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 |

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 1 | 2 | 2 |
| 1 | 2 | 2 | 4 | 4 |
| 2 | 3 | 3 | 5 | 5 |
| 2 | 4 | 5 | 7 | 7 |
| 2 | 4 | 5 | 7 | 8 |

**Figure 13: Summation of binary field. To calculate the sum of the neighbourhood for the target pixel (red) use: green pixel-blue pixels + orange pixel.**

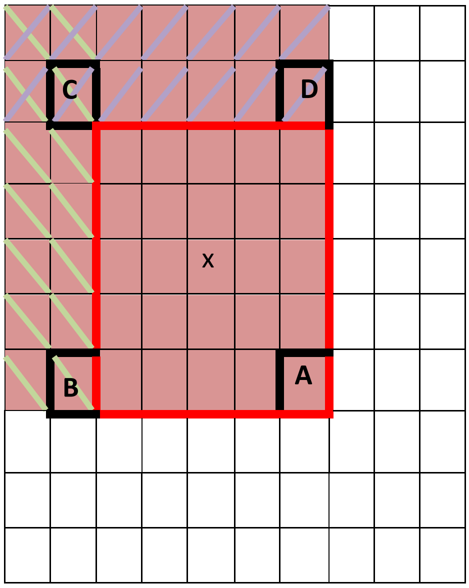


**Figure 14: Summing the values in a neighbourhood of a 5 by 5 grid.**

This is can be expanded to look at any neighbourhood size on any model grid, as shown in Figure 14, where the sum of the values of the pixels within the 5x5 neighbourhood outlined in red can be calculated by A-B-D+C. The area shaded red shows the area included in the sum for pixel A. To discount the areas not included in the red lined neighbourhood, the green dashed area (B) and the purple dashed area (D) are removed. This means that the area dashed both green and purple (C) has been discounted twice so must be added on again.

The same method can be followed to create a corresponding fractions grid from the observational dataset. From these fractions grids a single RMS value can be calculated by taking the squared difference between the model and observational fractions grids for each corresponding pixel and averaging these (Eq3, Section 2.5). The FSS can then be calculated by dividing this by the reference MSE calculated using Eq.5, and subtracting from 1 (Eq.4).

This is completed for all neighbourhood sizes between 1 and the model grid size for each threshold analysed.

### *3.3.2 More about Neighbourhoods*

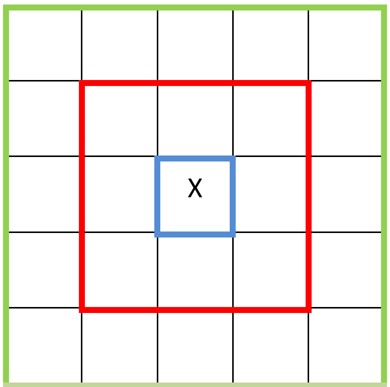



Figure 15: Neighbourhoods for pixel X, 1 grid square is shown in blue, 3 grid squares in red and 5 grid squares in green

As previously discussed in section 2, neighbourhoods can be either spatial or spatio-temporal. In this project, only spatial neighbourhoods have been used, future work. The data used is for lead times of T+24, T+48 up to T+120 for 12UTC runs. By using this side by side with data from the 00UTC or the 18UTC runs, a spatio-temporal neighbourhood could be conceived for future work. The neighbourhoods used increase in increments of 2 grid squares each time, from 1 to 3 to 5 to 7 and so on. This allows the grid to remain centred on the pixel in question by adding on one pixel on each side (Figure 15).

### *3.3.3 Dealing with a regional grid*

Due to time constraints and the nature of the latitude-longitude grid it would be unfeasible to calculate the FSS for large areas of the globe because of the elongation of pixels at the pole compared to lower latitudes and the wrap around.

Therefore a regional grid of 128 by 150 grid squares is used rather than the whole globe. This means that there are edges to deal with. For points close to the edge the neighbourhood will go outside of the area being examined. As the neighbourhoods get larger this problem will affect more pixels. This is called the ‘edge effect’. To deal with this, it was decided to treat pixels outside the edge of the regional verification area as if there was no rain there. When the neighbourhood goes off the edge of the grid, the missing pixels are counted as ‘0’ or below the threshold. This is a reasonable thing to do because both the analysis observational dataset and the forecast fields are treated the same way and so the fractions from the analyses and forecasts can be compared in the FSS calculation. It might unfortunately have some effect on the FSS particularly in situations where a large area of rain is seen near the edge of the grid in either the forecast or observations but is just off the grid in the other. However, it is unlikely to be a dramatic effect for a large sample as long as the spatial errors in the forecast are small compared to the size of the verification area. Another approach is to nest the FSS grid into a larger grid and include the data from these pixels (outside the verification region) in the neighbourhoods but not calculate the FSS for those points. However, this means that the percentage thresholds may be skewed so the FSS will not tend to 1. So far, little work has been published on this problem so the full impacts are yet to be discovered. Each method has its advantages and disadvantages as including the outside data can skew the thresholds. Not including the outside data can imply poor forecasts if precipitation consistently occurs just inside the grid in the model but is just outside in the observational dataset, or vice versa.

## *3.4 The code*

The code uses the fractions grids at each neighbourhood size to calculate the FSS for each threshold (and forecast time) and returns these in the form of an array. By looping over multiple days and different forecast periods then a large number of arrays can be produced and further analysed.

To allow the code to run smoothly with the large amounts of data, the data has been split into 'seasons' of Summer (June, July, August), Autumn (September, October, November), Winter (December, January, February) and Spring (March, April,  May). Whilst reducing calculation time, this also allows an interesting comparison between the skill of forecasts in different seasons. However, this does mean that there are some gaps in the results were the code is unable to compare the forecast from a date at the very end of one season with the analysis charts from a day at the very beginning of the next season.

### *3.4.1 Graphical Output*

There are 3 basic types of plots shown in Section 4. The first shows the change in FSS with scale. On these plots the line where FSS is equal to 0.5 is shown as a blue dotted line. This is of particular interest as at this skill level the forecasts are said to be 'useful'. Below this then the forecast is of little use as the user will have more chance of being right by guessing either rain or no rain.

The second type of plot shows the FSS or useful scale plotted for each season and threshold. This is to show the differences between the season and the thresholds.

The third type of plot is a bar plot showing the absolute values of the thresholds used. This gives an idea of the amount of rain falling and the intensity. For these plots the precipitation rates are shown in mm/hr, the most common unit (WMO, 2008).

The rainfall plots are used throughout the project as a way to visually check the code output. By plotting the rainfall, binary and fractions grids the selected data that is being compared (e.g. pixels above the threshold from the binary grids) can be compared visually. Although we cannot say exactly how skilful the forecast is by sight, it can be seen whether the model is generally a 'good' fit or a 'poor' fit. If the final output suggests differently then this is a sign that something is wrong.

Likewise, if the plots show completely different scenarios, this can show that an error has been made in choosing the data files to compare.

### *3.4.2 Useful Scale*

The code is designed to locate and store the useful scale for each day. The scale is saved by the date for which the forecast has been made, so a T+24 forecast from 1st March 2014 would be stored under 2nd March. This means that comparison can be made between different forecast ranges easily as all forecasts for a day are saved with that date.

### *3.4.3 Tests*

To ensure the code was correct, each section was tested individually using synthetic fields. The code to calculate the FSS was tested; using a small grid showed the same result as the same grid when calculated by hand.

The data sorting functions were tested using a 10 by 10 number grid. Using a list of 100 numbers, these were turned into a 2 by 2 array and cut to size. As the numbers were in order, it was easy to check the numbers were in the correct position.

Results were compared with results calculated by Roberts (2014) for different lead times over a period at the beginning of March 2014. As the results were very close, it is believed that the code was accurate and the small differences may be due to rounding errors.

## *3.5 Constraints and Limitations*

As discussed previously, edge effects can cause problems in the verification, especially if large accumulations of rain occur just on/off the grid. This is particularly problematic if the useful scale is close to the maximum grid size. The model also assumes constant grid size. The area chosen was decided specifically to allow for this and means that 1 grid square is approximately 25km. However, this also means that the code can be used with different analysis data from a different model on a different grid size with relative ease as the code works in terms of grid squares and only converts this into km at the very end. This is helpful as the new model currently being put in place by the Met Office has a grid size of approximately 17km (Met Office, 2014).

## *3.6 Worked Examples*

Two examples of the fields and how the FSS was used to give a spatial comparison are shown below. One may be considered a ‘good’ forecast whilst the other is a ‘poor’ forecast. This is to help understanding of the process that has previously been described in this section.

### *3.6.1 A ‘good’ forecast (Useful at grid scale, 25km)*

Taking a T+24 forecast from the 15th March 2014 for 12UTC 16th May. The rainfall rates of both the forecast and the ‘observational’ analysis grid can be seen in the thresholds have been converted to mm/hr Visual comparison of these plots shows clearly that the forecast looks very similar to the analysis. Therefore it would be expected for the skill as measured by the FSS to be high. As the first stage in computing the FSS was to convert the two fields to binary grids for the required threshold, Figure 17 shows the binary fields obtained for the 99th percentile (highest 1% of values). It is clear that the two binary images look very similar although the exact positioning of the shaded pixels is not identical, showing that the forecast has the heaviest rain in the correct area, but not exactly the right locations. The thresholds used are shown in Table 1.

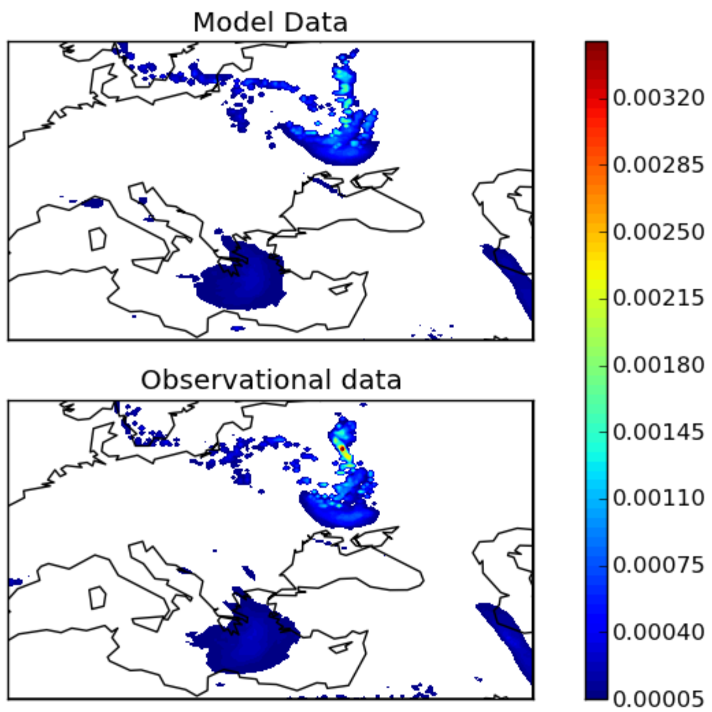



**Figure 16: T+24 forecast and analysis of rainfall at 12UTC on 16th May 2014. Rainfall rate is in kg/m$^2$/s**

The rainfall rate is in kg/m$^2$/s, which is the standard unit used in the Met Office and the unit in which the data is provided. This is equivalent to mm/s and can be converted into mm/hr by multiplying by 3600. To maintain consistency, the code keeps the data in the format it was originally in and therefore the range in the plots is in kg/m$^2$/s. The conversion of these into mm/hr, a unit that is more commonly used in meteorology, is discussed later in the results section where the thresholds have been converted to mm/hr. Visual comparison of these plots shows clearly that the forecast looks very similar to the analysis. Therefore it would be expected for the skill as measured by the FSS to be high. As the first stage in computing the FSS was to convert the two fields to binary grids for the required threshold- Figure 17 shows the binary fields obtained for the 99$^{th}$ percentile (highest 1% of values). It is clear that the two binary images look very similar although the exact positioning of the shaded pixels is not identical, showing that the forecast has the heaviest rain in the correct area, but not exactly the right locations. The thresholds used are shown in Table 1**.**

**Table 1: Precipitation rate threshold values for 16th May 2014**

| | **Model** | | **Analysis** | |
|---|---|---|---|---|
| **Percentage threshold** | 99% | 90% | 99% | 90% |
| **Actual threshold value (mm/hr)** | 2.17 | 0.22 | 2.62 | 0.25 |

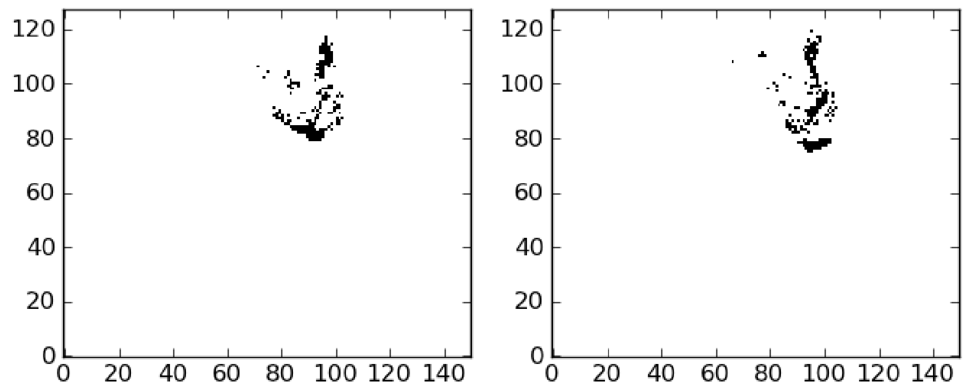



**Figure 17: 99% threshold binary plots for 12UTC 16th May. The plot on the left shows the forecast from 24 hours previously whilst the right is the analysis chart for the date in question. Plotted on a standard square grid rather than by latitude/longitude.**

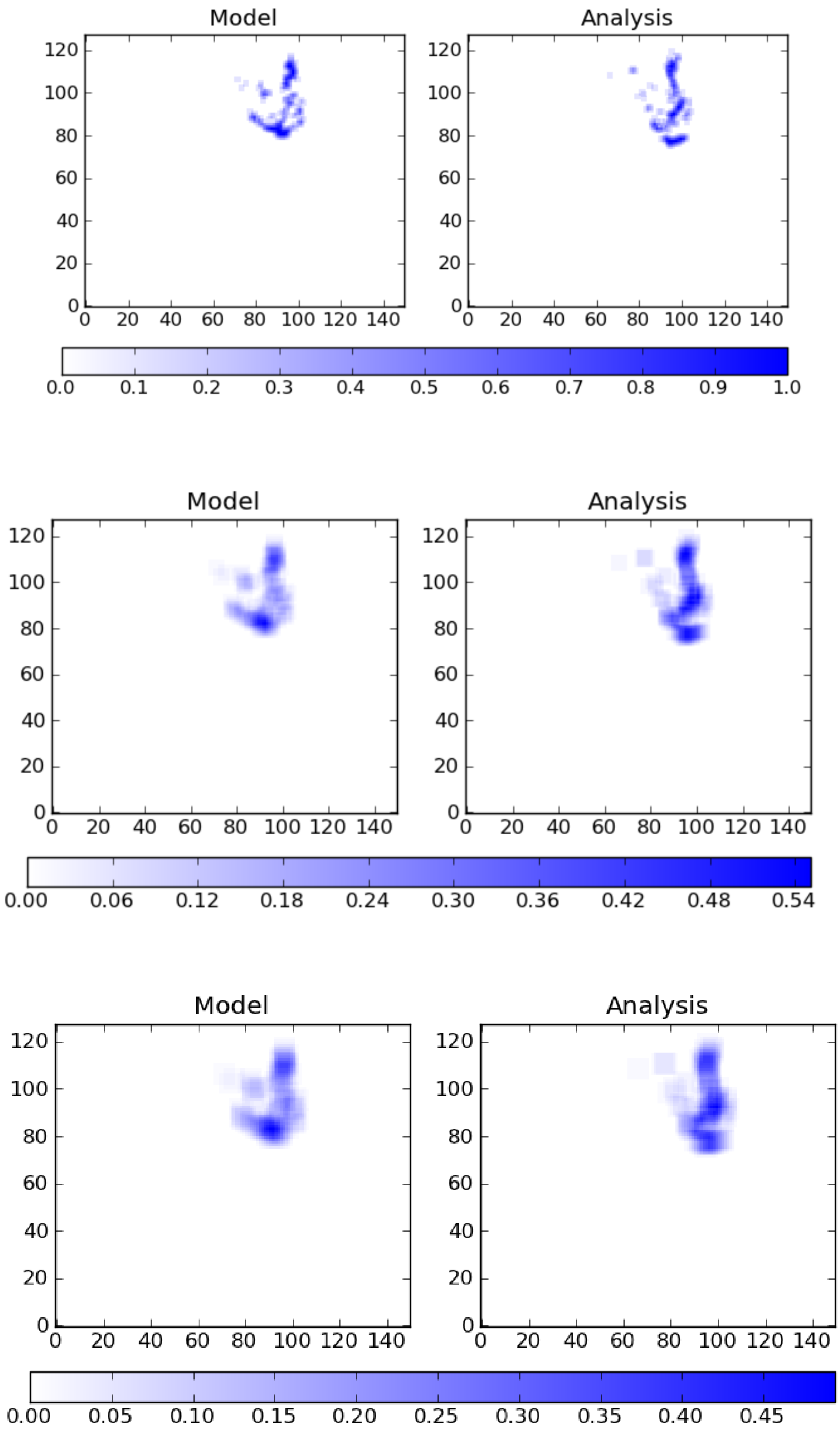



**Figure 18: Fractions grids at 3, 5 and 7 grid squares neighbourhoods for the T+24 forecast for 16th May 2014.**

**Table 2: FSS values at each scale for 99 and 90% thresholds for 16th May**

| Scale | 0.99 | 0.90 | Scale | 0.99 | 0.90 |
|---|---|---|---|---|---|
| **1** | 0.219 | 0.643 | **77** | 0.965 | 0.997 |
| **3** | 0.432 | 0.783 | **79** | 0.965 | 0.997 |
| **5** | 0.565 | 0.843 | **81** | 0.966 | 0.997 |
| **7** | 0.654 | 0.881 | **83** | 0.967 | 0.997 |
| **9** | 0.723 | 0.907 | **85** | 0.968 | 0.998 |
| **11** | 0.774 | 0.925 | **87** | 0.969 | 0.998 |
| **13** | 0.809 | 0.938 | **89** | 0.969 | 0.998 |
| **15** | 0.832 | 0.949 | **91** | 0.970 | 0.998 |
| **17** | 0.848 | 0.957 | **93** | 0.971 | 0.998 |
| **19** | 0.860 | 0.963 | **95** | 0.972 | 0.998 |
| **21** | 0.869 | 0.968 | **97** | 0.973 | 0.998 |
| **23** | 0.876 | 0.972 | **99** | 0.974 | 0.998 |
| **25** | 0.883 | 0.976 | **101** | 0.975 | 0.998 |
| **27** | 0.888 | 0.979 | **103** | 0.976 | 0.998 |
| **29** | 0.893 | 0.981 | **105** | 0.977 | 0.999 |
| **31** | 0.898 | 0.984 | **107** | 0.978 | 0.999 |
| **33** | 0.903 | 0.986 | **109** | 0.979 | 0.999 |
| **35** | 0.909 | 0.988 | **111** | 0.979 | 0.999 |
| **37** | 0.914 | 0.989 | **113** | 0.980 | 0.999 |
| **39** | 0.918 | 0.990 | **115** | 0.981 | 0.999 |
| **41** | 0.923 | 0.991 | **117** | 0.981 | 0.999 |
| **43** | 0.927 | 0.992 | **119** | 0.982 | 0.999 |
| **45** | 0.932 | 0.993 | **121** | 0.982 | 0.999 |
| **47** | 0.936 | 0.993 | **123** | 0.983 | 0.999 |
| **49** | 0.940 | 0.994 | **125** | 0.984 | 0.999 |
| **51** | 0.943 | 0.994 | **127** | 0.984 | 0.999 |
| **53** | 0.946 | 0.995 | **129** | 0.985 | 0.999 |
| **55** | 0.949 | 0.995 | **131** | 0.985 | 0.999 |
| **57** | 0.951 | 0.995 | **133** | 0.986 | 0.999 |
| **59** | 0.954 | 0.996 | **135** | 0.986 | 0.999 |
| **61** | 0.956 | 0.996 | **137** | 0.987 | 0.999 |
| **63** | 0.957 | 0.996 | **139** | 0.987 | 0.999 |
| **65** | 0.959 | 0.996 | **141** | 0.988 | 0.999 |
| **67** | 0.960 | 0.997 | **143** | 0.988 | 0.999 |
| **69** | 0.961 | 0.997 | **145** | 0.989 | 0.999 |
| **71** | 0.962 | 0.997 | **147** | 0.989 | 0.999 |
| **73** | 0.963 | 0.997 | **149** | 0.990 | 0.999 |
| **75** | 0.964 | 0.997 | | | |

Once each pixel was assigned a fraction between 0 and 1 (Figure 18) these were used to calculate the FSS. The FSS for each scale and threshold is shown in Table 2. In this example, the forecast is useful at grid scale as the FSS at neighbourhood size 1 is greater than 0.5.

### *3.6.2 A 'poor' forecast (Useful at 9 grid squares (225km))*

The T+48 forecast for January 6th (runtime 4thJanuary 12UTC) is shown in Figure 19. Just by a simple visual comparison, it can be seen this forecast is not quite as good a fit as the May 16th forecast seen in Section 3.5.4.1. The forecast rain is patchier and in some places, none-existent when compared with the observational analysis chart.

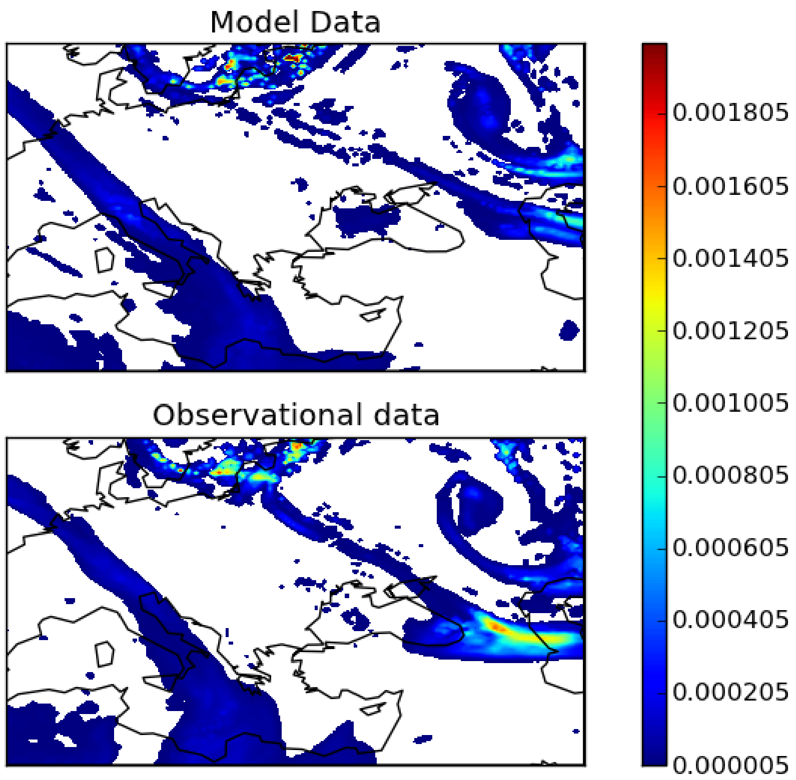



**Figure 19: Model and Analytical rainfall rate plots for January 6th 2014 at 12UTC. Rainfall rate is in kg/m$^2$/s.**

From this, the binary outputs for the 99% threshold would be expected to be slightly different, as can be seen in Figure 20 where the shapes are noticeably different. Therefore we would expect to reach a larger neighbourhood size before the forecast and analysis are close.

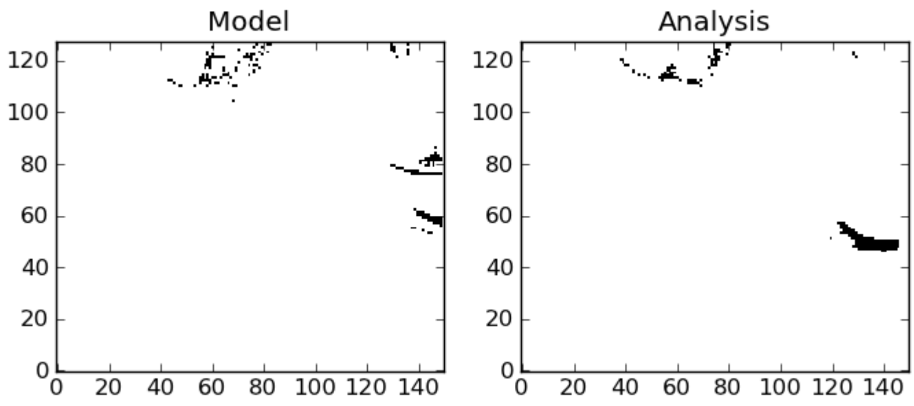



**Figure 20: Binary outputs for Model (left) and analytical (right) rainfall rates at 99% threshold.**

It is of interest to meteorologists to be able to say in advance the useful scale of forecasts. The surface pressure chart for January 6$^{th}$, Figure 21, shows an area of low pressure sitting over Italy. This is during the period of intense storms that passed over Europe during the winter of 13/14. Also around this time period was an unusually intense snow storm in North America, well outside the area being studied.

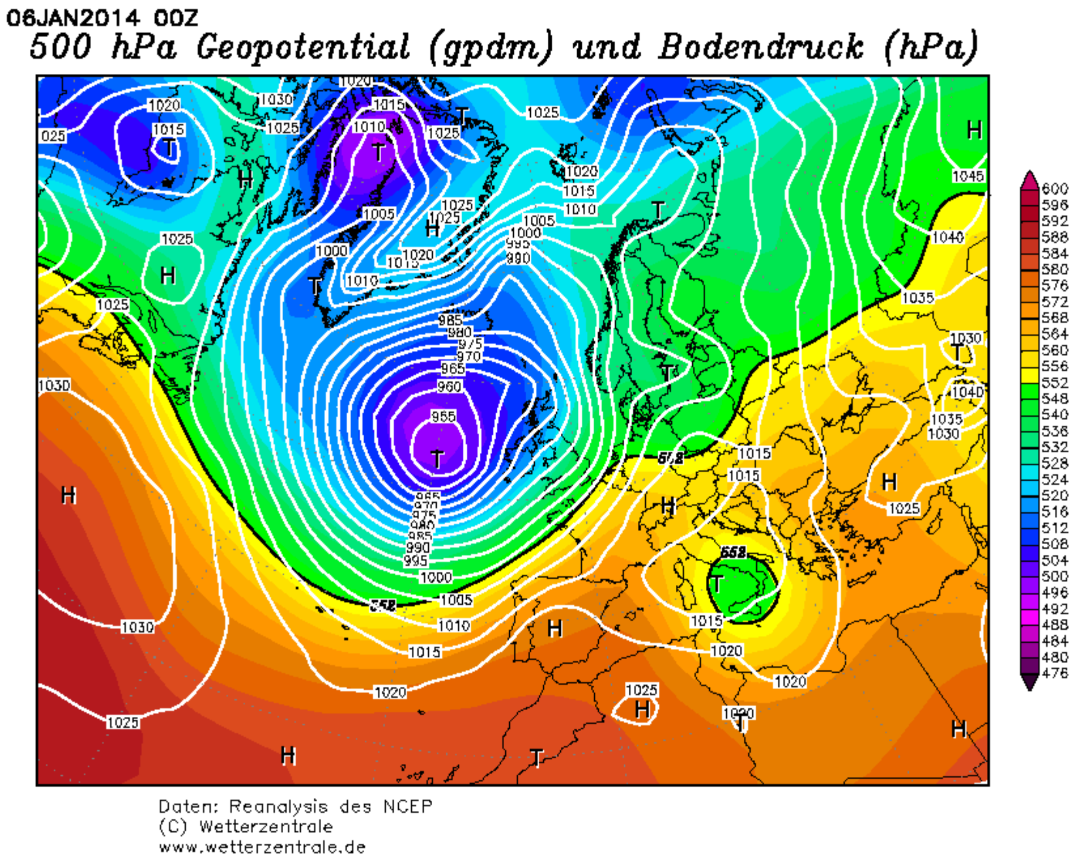



**Figure 21: Surface pressure chart for January 6th at 00UTC for Europe. (Source: Wetterzentrale, 2014)**

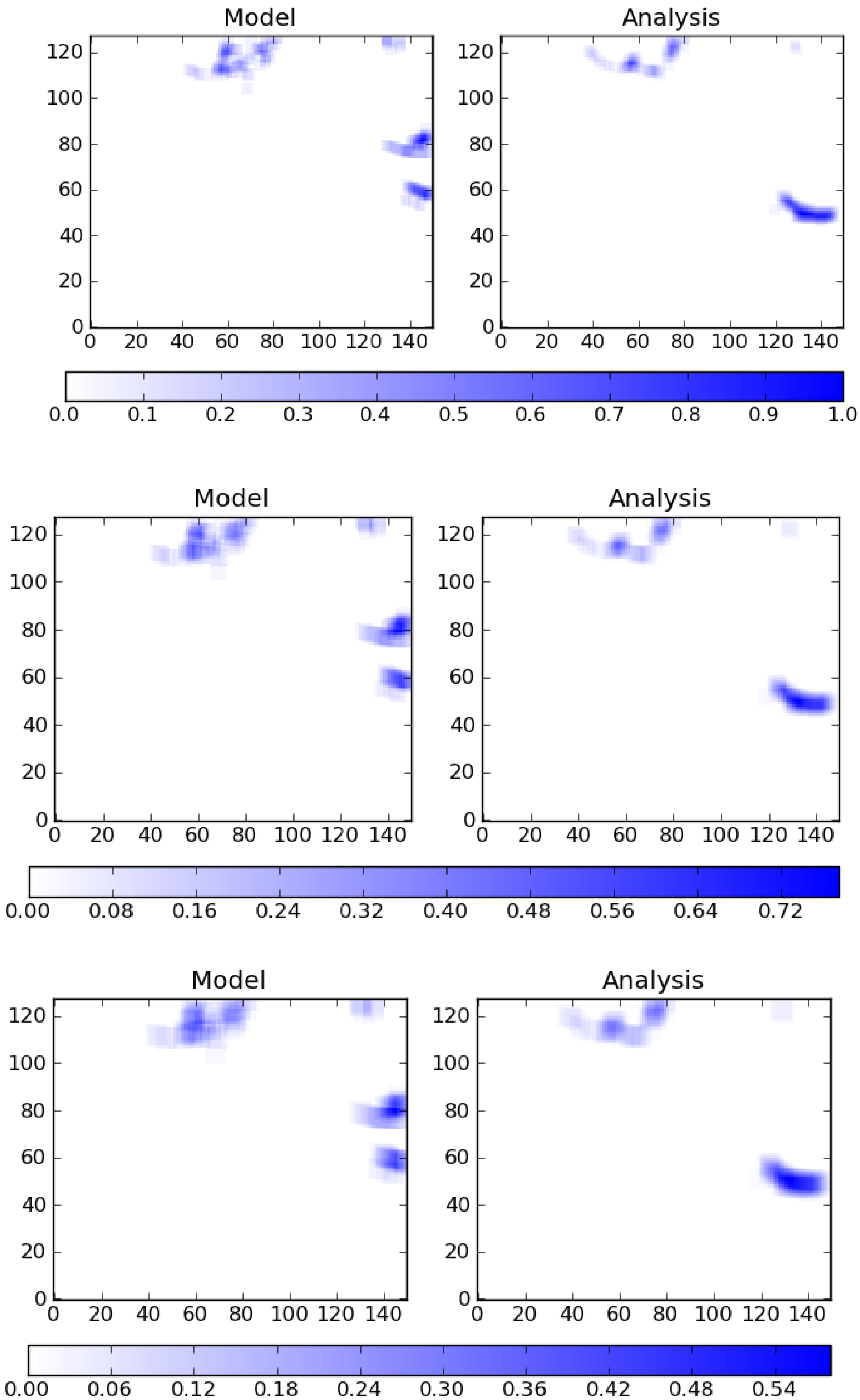



**Figure 22: Fractions grids at neighbourhood sizes of 1, 3, and 5 grid squares for T+48 forecast for January 6th.**

**Table 3: FSS values by scale at different thresholds for January 6th 2014**

| Scale | 0.99 | 0.95 | 0.90 | Scale | 0.99 | 0.95 | 0.90 |
|---|---|---|---|---|---|---|---|
| **1** | 0.099 | 0.190 | 0.310 | **77** | 0.822 | 0.809 | 0.918 |
| **3** | 0.183 | 0.266 | 0.390 | **79** | 0.829 | 0.816 | 0.922 |
| **5** | 0.231 | 0.310 | 0.444 | **81** | 0.835 | 0.822 | 0.926 |
| **7** | 0.263 | 0.349 | 0.493 | **83** | 0.842 | 0.828 | 0.930 |
| **9** | 0.291 | 0.387 | 0.540 | **85** | 0.848 | 0.835 | 0.934 |
| **11** | 0.323 | 0.428 | 0.585 | **87** | 0.853 | 0.841 | 0.937 |
| **13** | 0.360 | 0.469 | 0.626 | **89** | 0.858 | 0.847 | 0.941 |
| **15** | 0.396 | 0.509 | 0.663 | **91** | 0.864 | 0.854 | 0.944 |
| **17** | 0.429 | 0.543 | 0.695 | **93** | 0.868 | 0.860 | 0.947 |
| **19** | 0.456 | 0.572 | 0.721 | **95** | 0.873 | 0.866 | 0.950 |
| **21** | 0.479 | 0.595 | 0.743 | **97** | 0.878 | 0.872 | 0.952 |
| **23** | 0.498 | 0.612 | 0.759 | **99** | 0.883 | 0.878 | 0.955 |
| **25** | 0.515 | 0.625 | 0.772 | **101** | 0.888 | 0.884 | 0.958 |
| **27** | 0.532 | 0.636 | 0.782 | **103** | 0.893 | 0.890 | 0.960 |
| **29** | 0.550 | 0.645 | 0.790 | **105** | 0.899 | 0.896 | 0.963 |
| **31** | 0.571 | 0.654 | 0.798 | **107** | 0.904 | 0.902 | 0.965 |
| **33** | 0.593 | 0.663 | 0.805 | **109** | 0.910 | 0.907 | 0.967 |
| **35** | 0.614 | 0.672 | 0.812 | **111** | 0.916 | 0.912 | 0.969 |
| **37** | 0.635 | 0.681 | 0.818 | **113** | 0.921 | 0.917 | 0.971 |
| **39** | 0.654 | 0.690 | 0.824 | **115** | 0.926 | 0.921 | 0.973 |
| **41** | 0.671 | 0.698 | 0.830 | **117** | 0.931 | 0.925 | 0.974 |
| **43** | 0.687 | 0.705 | 0.835 | **119** | 0.936 | 0.929 | 0.976 |
| **45** | 0.701 | 0.712 | 0.840 | **121** | 0.940 | 0.932 | 0.977 |
| **47** | 0.713 | 0.718 | 0.844 | **123** | 0.944 | 0.935 | 0.978 |
| **49** | 0.724 | 0.723 | 0.849 | **125** | 0.947 | 0.939 | 0.980 |
| **51** | 0.734 | 0.729 | 0.854 | **127** | 0.950 | 0.942 | 0.981 |
| **53** | 0.743 | 0.735 | 0.859 | **129** | 0.953 | 0.944 | 0.982 |
| **55** | 0.751 | 0.741 | 0.864 | **131** | 0.956 | 0.947 | 0.983 |
| **57** | 0.759 | 0.747 | 0.869 | **133** | 0.959 | 0.950 | 0.984 |
| **59** | 0.767 | 0.753 | 0.875 | **135** | 0.962 | 0.952 | 0.984 |
| **61** | 0.774 | 0.759 | 0.880 | **137** | 0.964 | 0.955 | 0.985 |
| **63** | 0.780 | 0.765 | 0.886 | **139** | 0.967 | 0.957 | 0.986 |
| **65** | 0.786 | 0.772 | 0.891 | **141** | 0.970 | 0.959 | 0.987 |
| **67** | 0.792 | 0.778 | 0.896 | **143** | 0.973 | 0.961 | 0.988 |
| **69** | 0.798 | 0.784 | 0.901 | **145** | 0.976 | 0.963 | 0.988 |
| **71** | 0.803 | 0.790 | 0.905 | **147** | 0.978 | 0.965 | 0.989 |
| **73** | 0.809 | 0.797 | 0.910 | **149** | 0.980 | 0.967 | 0.990 |
| **75** | 0.816 | 0.803 | 0.914 | | | | |

# 4. Results

## 4.1 The year as a whole

The FSS skill score can be plotted against scale, for each individual comparison made. Figure 23 shows this for the T+24 forecasts over the whole grid for a 90% threshold. The 90$^{th}$ percentile threshold means that it is the top 10% of rainfall rates taken from all pixels in the domain that are compared. This threshold will pick out both frontal precipitation and convective areas of precipitation, but may miss some of the lightest precipitation. Each red line represents a comparison between a T+24 forecast and the analysis chart for the next day. The black line shows the mean FSS for the whole period. The blue dashed line is the line where FSS=0.5. If the skill is above this then the forecast is considered useful, as defined in the previous section. Therefore the point at which a line crosses this threshold is important. This number will be discussed later in this section. All the red lines without exception curve up towards high values of FSS with increasing spatial scale showing that forecast skill improves with increasing spatial scale – i.e. being less stringent about where the rain is positioned increases the skill. From this we can see that 92% are sufficiently skilful at grid scale for a 24 hour forecast.

The scale is given in number of grid squares. In the domain used, each grid square is taken to be 25km. Therefore a scale of 3 grid squares would be 75km. These results give the distinct impression that the T+24 global model forecasts for general areas of rain are very good. The precipitation is mostly in the correct place. It is helped by the fact that the analyses are not the actual noisy precipitation fields, but nevertheless it is showing that the 24-hour forecasts tend to put the precipitation very close to where the global model has it at the start of the forecasts (analyses) beginning 24-hours later.

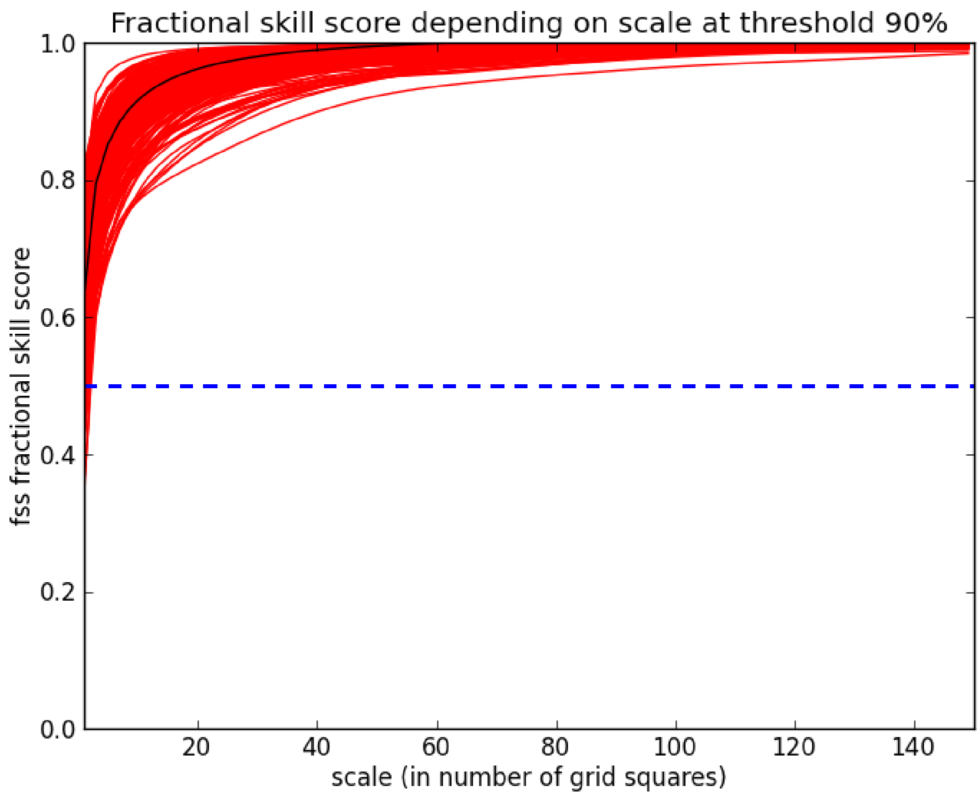



**Figure 23: FSS against Scale (km) for entire data set with a threshold of 90%**

Because the percentile thresholds have been used, it can be seen that the FSS tends towards 1 as the scale tends towards the size of the full grid. This is because, by definition, the same number of pixels are being compared and therefore there has to be perfect agreement over the whole domain. In the case seen in Figure 23 all the forecasts tend towards FSS=1 at scales much shorter than the full grid size (150 pixels). Therefore, plotting only the values up to a shorter grid size, such as 20, seen in Figure 24, can provide more detail about the scales at which the forecasts become useful. At the T+24 forecast range, all the forecasts are useful (FSS>0.5) at either 1 or 3 grid squares, which means the forecasts are consistently useful at scales of between 0-75km. The mean scale at which the forecasts become useful is calculated by taking the scale at which the FSS is first greater than 0.5 (starting at a scale of one grid square and increasing in size) for each comparison and then taking the mean of these values. It is not possible to tell at exactly what scale the FSS=0.5 between neighbourhood sizes without using a smaller grid size, the value used must be taken as the next neighbourhood size. Therefore, whilst the mean line indicates a mean FSS greater than 0.5 for a neighbourhood size of 1, the mean scale at which the forecast is considered useful is greater than 1. For the T+24 forecasts this value is 1.154 grid squares (28.25 km), suggesting that more forecasts are useful at grid scale than at a scale of 3 grid squares. This is discussed further later in this section.

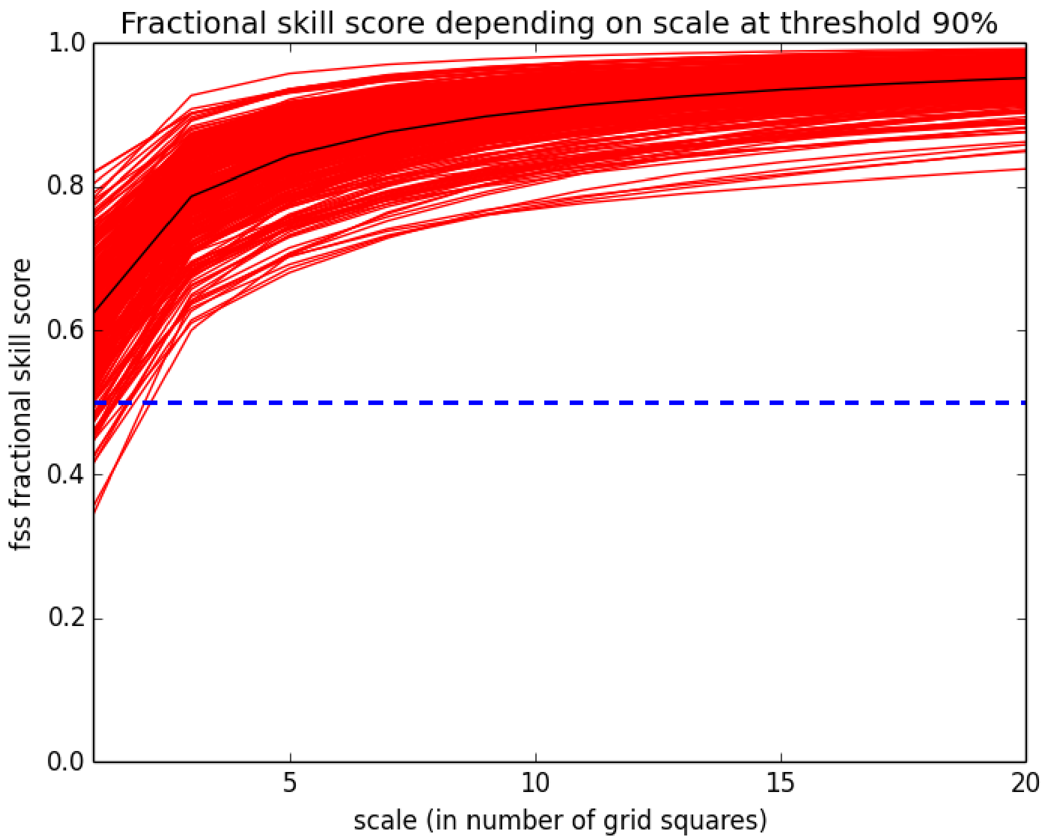



**Figure 24: FSS against scale as in Figure 1 but zoomed in to a smaller scale to show that all forecasts are considered useful by a scale of 3 grid squares.**

Figure 25 shows the T+48 forecasts for the same time period. It can be seen that there is a displacement to the right, showing a decrease in the skill at grid level. The mean line, again shown in black, shows that the FSS is much closer to 0.5 at grid scale, as can be seen more clearly in the zoomed in plot in Figure 26.

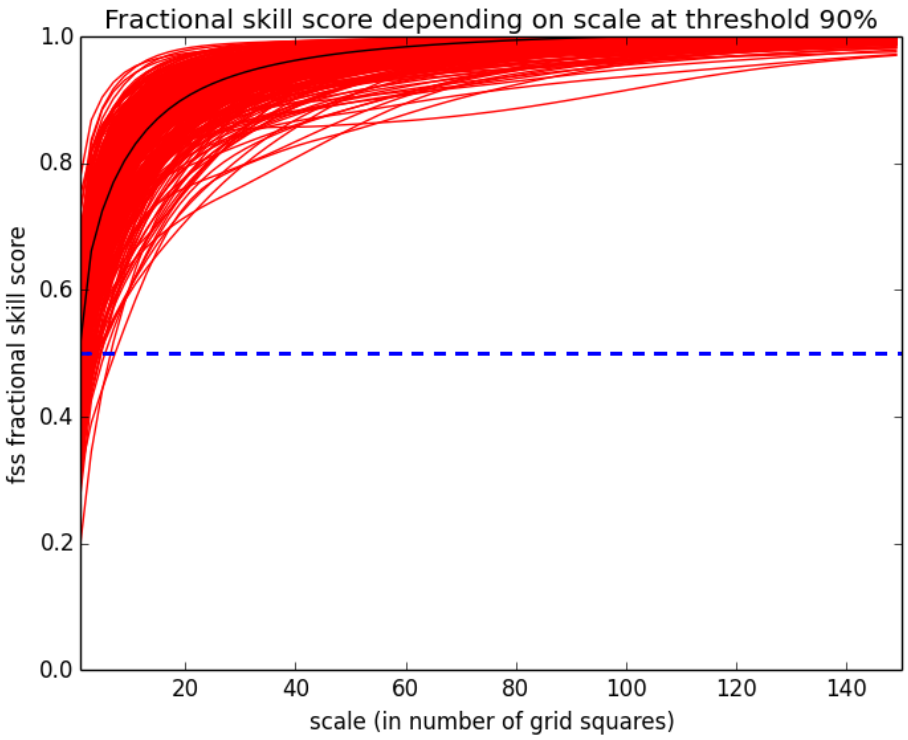



**Figure 25: FSS against scale for T+48 forecasts for the year period. Red lines show individual forecasts.**

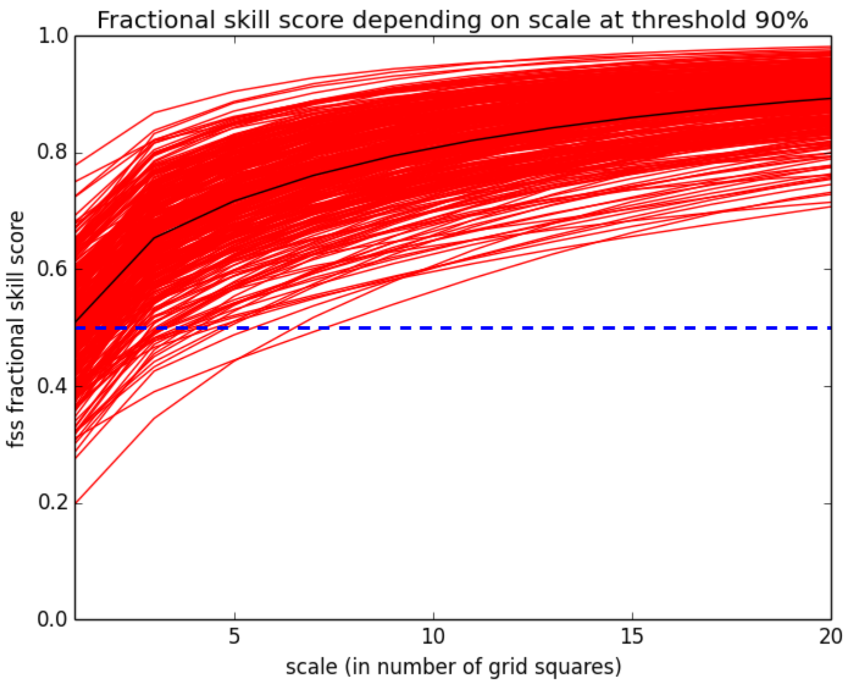



**Figure 26: FSS against scale for T+48 forecasts for whole year. Red lines are individual forecasts and the black line shows the mean of all forecasts.**

The same graphs for T+24 and T+48 can be plotted for the 99th percentile threshold. These can be seen below in Figure 27and Figure 28 respectively. Now only the heaviest 1% of rainfall is being compared, it is looking at more localised precipitation. This threshold would only pick out the heaviest parts of frontal bands or convective areas. For that reason it would be expected that the FSS would be slightly lower because the heavier more localised rain precipitation is expected to be more difficult to forecast. The graphs show that there is a much larger spread in the scale at which forecasts could be considered useful and lower skill overall than was the case for the 90th percentile. This suggests that whilst the global model is good at forecasting the location of the wider areas of precipitation, it has more difficulty accurately forecasting the position of the heaviest and more localised precipitation. The wide range of scales make it harder to say what scale the forecast is useful at on any given day, unlike at the 90% threshold. It appears that although most of the T+24 and T+48 forecasts are still good for the heaviest precipitation there are occasions when the forecast can go badly wrong.

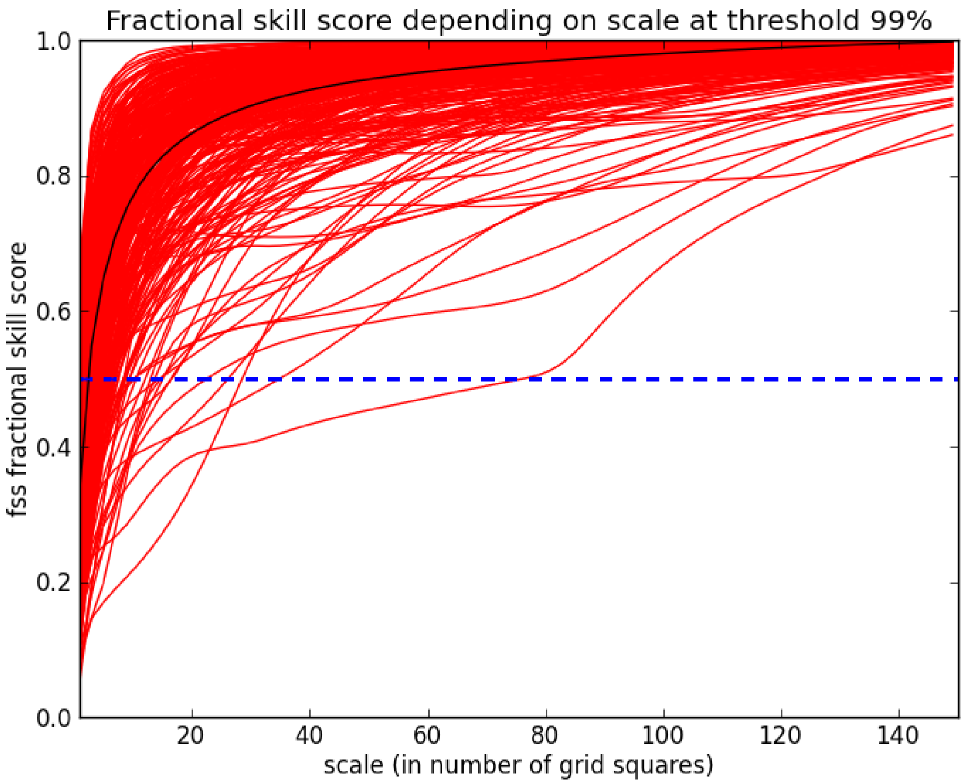



**Figure 27: T+24 forecast with a 99% threshold. Individual comparisons are shown in red with the mean in black.**

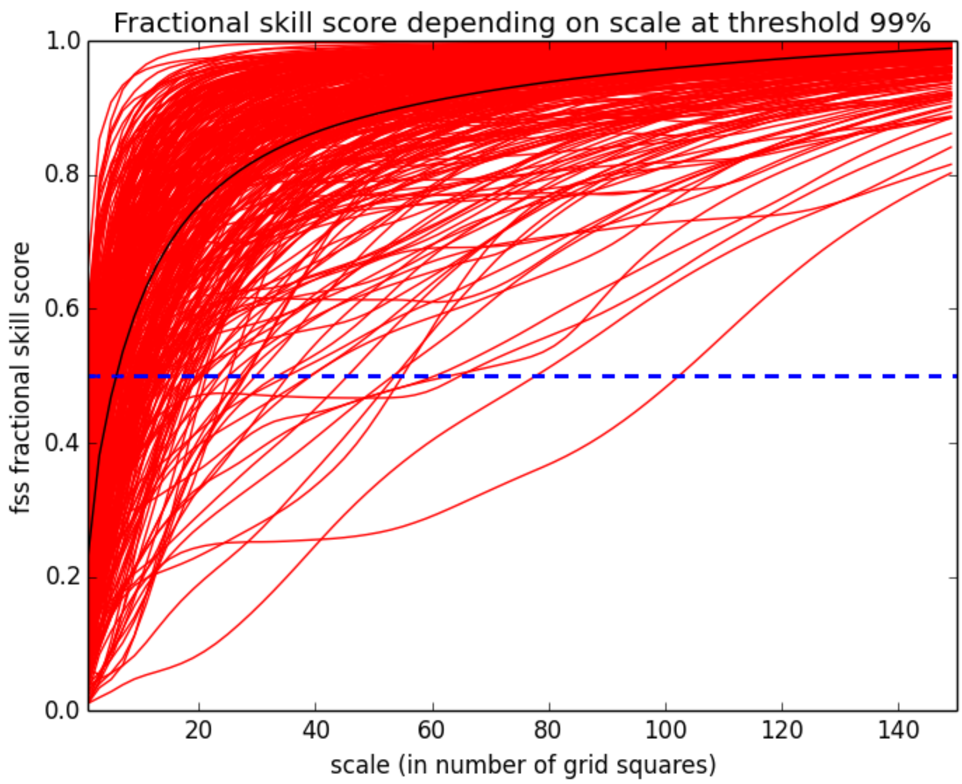



**Figure 28: FSS against scale for 99% threshold for T+24 (top) and T+48 (bottom) showing the wide range of scales at which the forecast would be considered useful.**

## *4.2 Seasonal Analysis*

In this section, the differences in the skill of forecasts for different seasons will be examined. Different weather conditions are experienced in different seasons, for example, during winter, the precipitation areas tend to be more frontal and move more quickly. There is little convection over land. During the summer, precipitation tends to come from convection, and is also usually more static. By looking at the seasons separately therefore, the effect of the different weather conditions on the skill can be seen. As discussed in Section 2, parameterised convective precipitation can be considered harder to forecast as there is less continuity. On the other hand, the faster moving systems in the winter may create problems as it is more difficult to forecast the position of the systems at a particular time.

### *4.2.1 Summer*

The FSS against the scale for the 3 summer months, June, July, August, at a 90% threshold can be seen in Figure 29. The mean scale at which the forecasts become useful is 1.198 grid squares or ~30km. This increases to 2.371 grid squares (~60km) for a T+48 forecast in the same season.

Logically, we would expect such an increase as increasing errors mean it is harder to forecast further in advance.

### *4.2.2 Autumn*

The autumnal plots, Figure 30, covering September, October and November show a mean useful scale of 1.143 grid squares, slightly lower than the summer. The reasons for this and the differences between the other seasons will be discussed later in this section. For the T+48 forecasts, the mean useful scale is 1.976 grid squares.

### *4.2.3 Winter*

For T+24, the mean useful scale is 1.022 grid squares and for T+48 1.841 grids squares (approximately, 26 and 40km respectively). These are the lowest from any season for both forecast lead times. The plots all show a sharp increase of skill with increasing scale at lower scales with the increase in skill reaching asymptotically to one as the scale approaches the full grid size, Figure 31.

### *4.2.4 Spring*

For T+24, the mean useful scale is 1.253 grid squares and for T+48 2.256 grid squares. This is the largest scale for T+24 but by T+48, the summer months mean useful scale is larger. Figure 32 shows a larger spread in the FSS with scale but still increases more rapidly at lower scales and reaches asymptotically for one at the full grid scale.

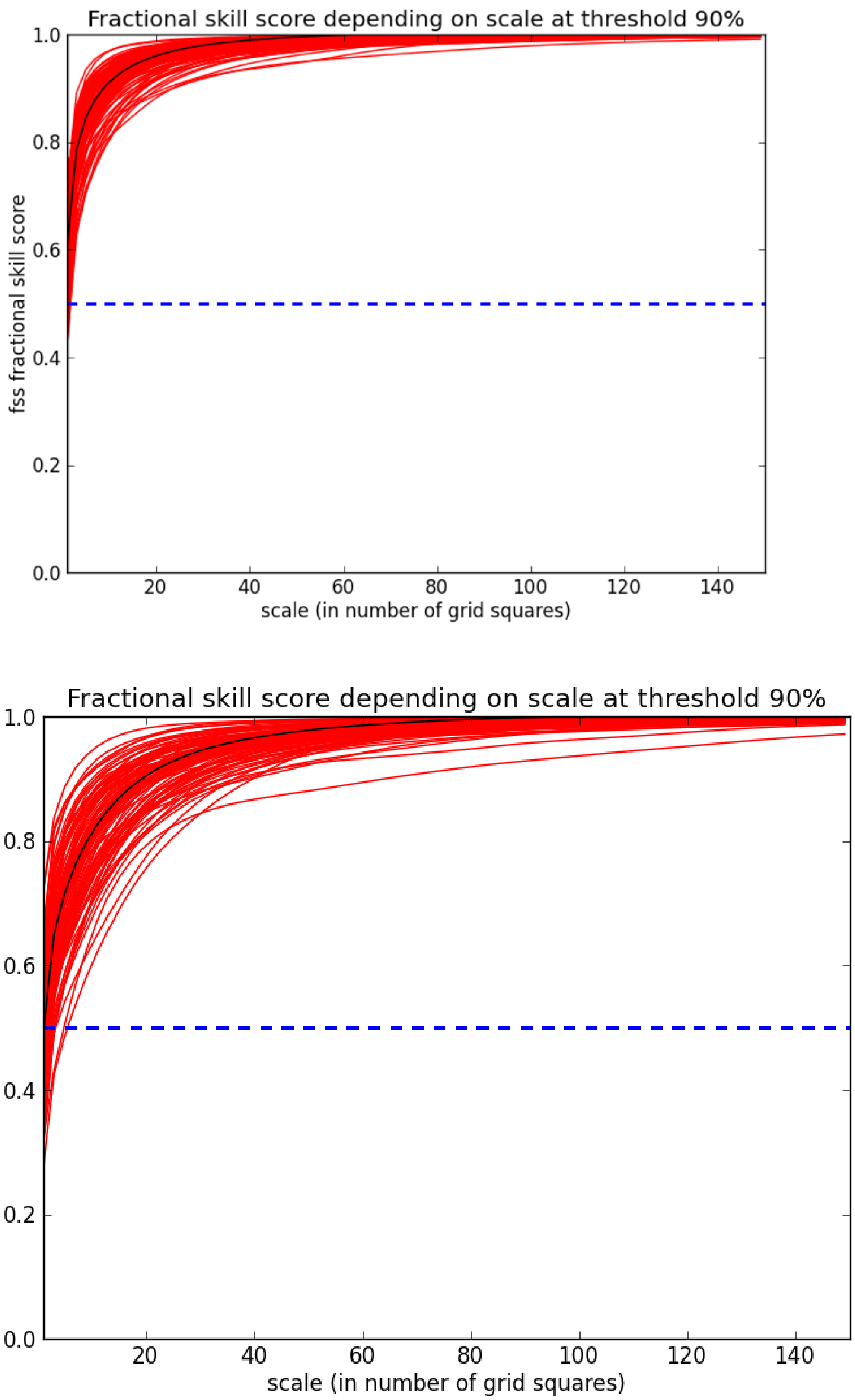
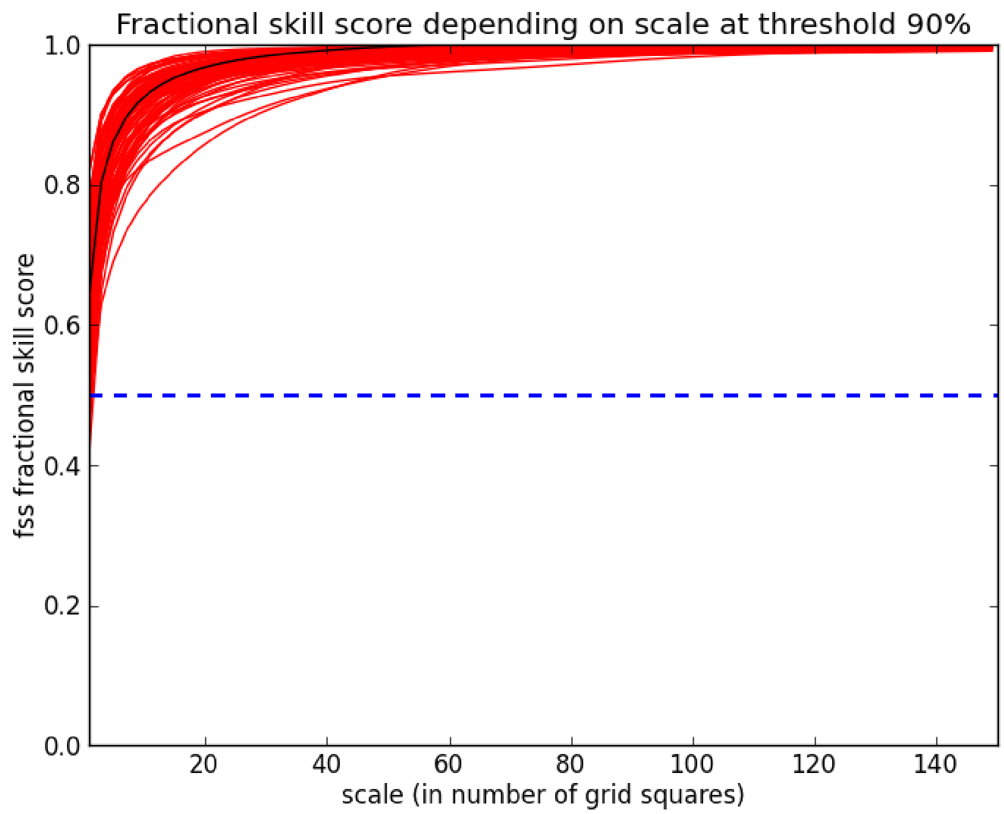



**Figure 29: FSS against scale for 90% thresholds during June, July, August for T+24 (top) and T+48(bottom) forecasts.**



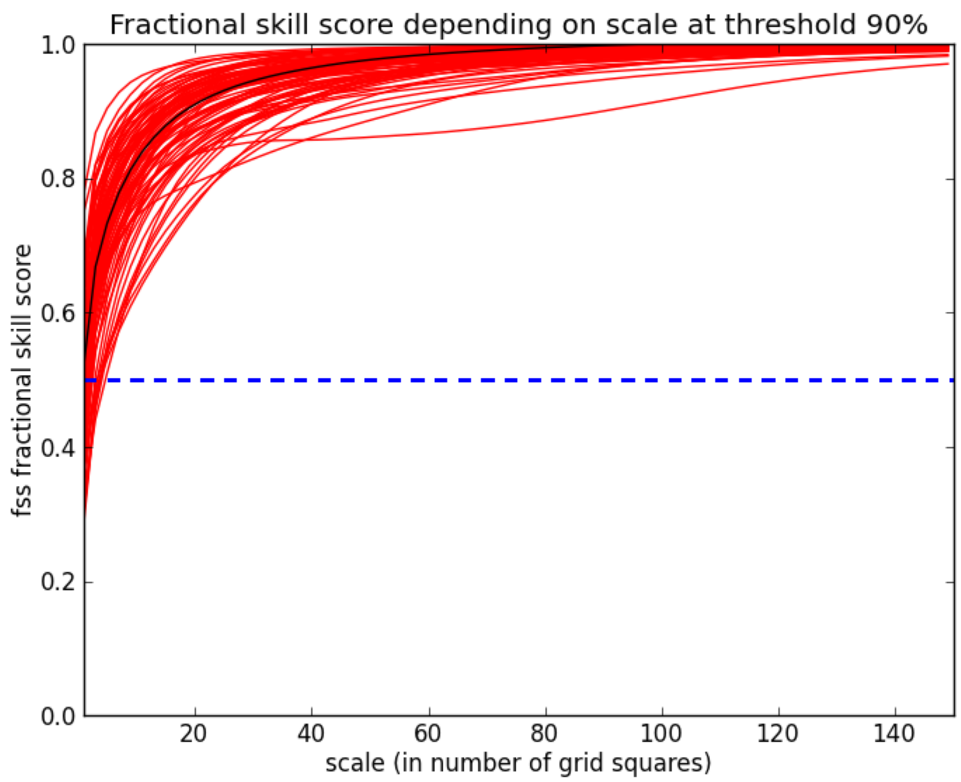
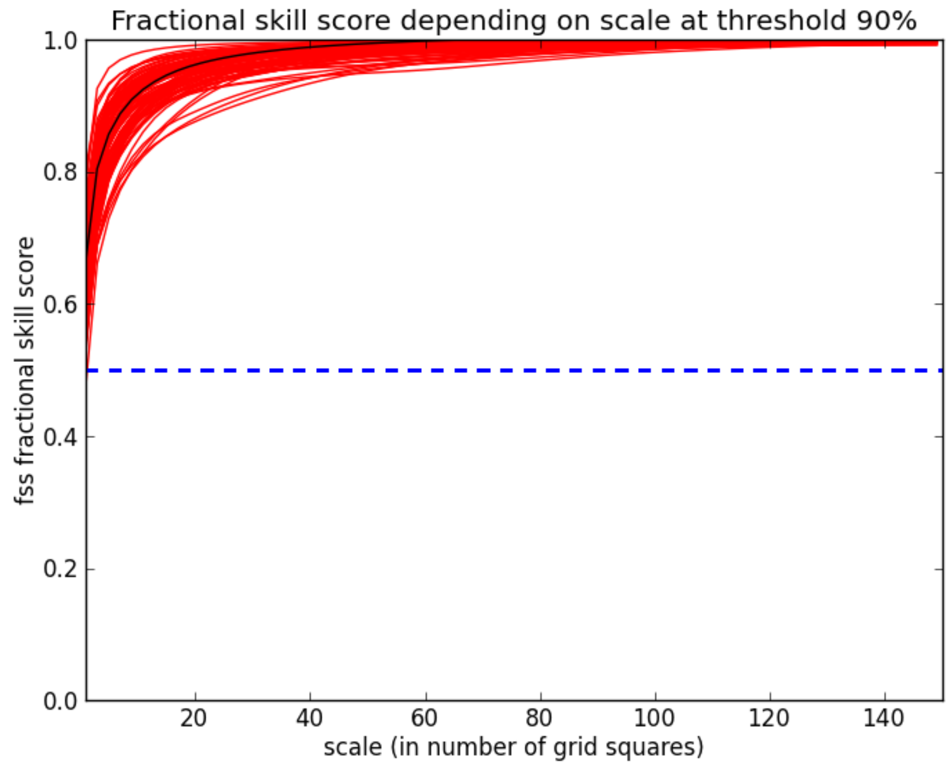




**Figure 30: FSS against scale for 90% thresholds during September, October, November for T+24 (top) and T+48(bottom) forecasts.**



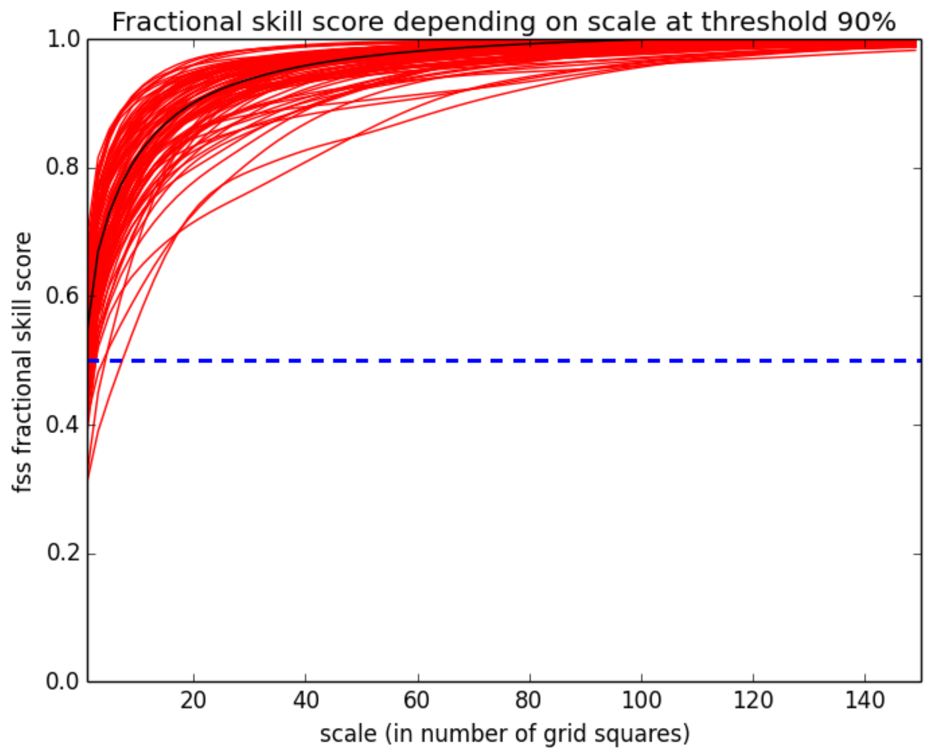




**Figure 31: FSS against scale for 90% thresholds during December, January, February for T+24 (top) and T+48(bottom) forecasts.**

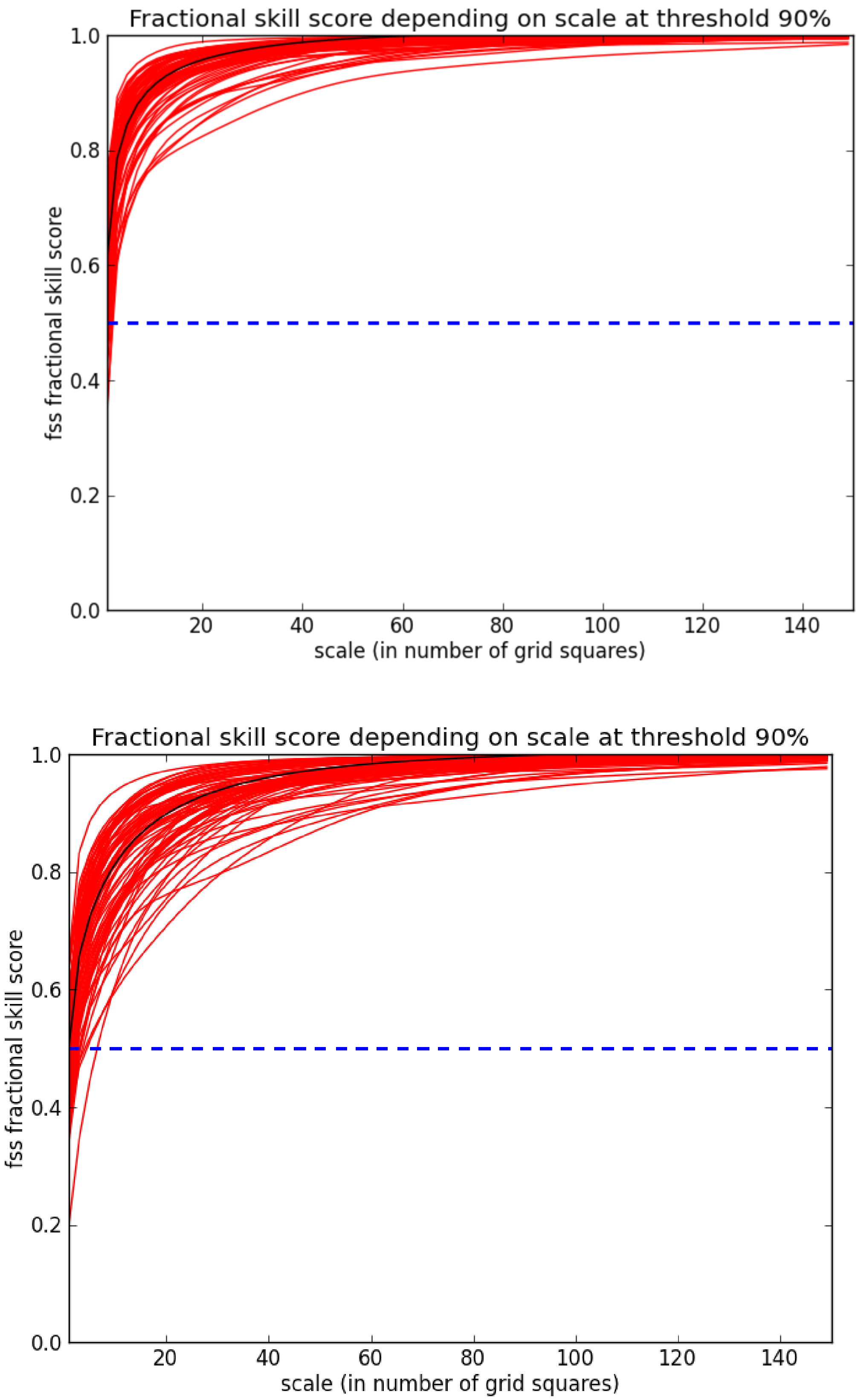



**Figure 32: FSS against scale for 90% thresholds during March, April, May for T+24 (top) and T+48(bottom) forecasts.**

### *4.2.5 Comparison*

The mean useful scale for each season is plotted in Figure 34 at the 90$^{th}$ percentile threshold for T+24 and T+48. This shows that during the spring and summer the useful scale is larger than during the autumn/ winter. This may be due to the smaller, more diurnally heated, spring 'showers' which are harder to predict than frontal or heavier convective storms that occur during the winter. The errors involved are more likely to grow quickly making forecasts worse at longer lead times. Winter is clearly the easiest season to forecast, despite being more mobile. This may be due to the bigger storms having larger precipitation structures that do not get displaced so quickly. Whilst there are no other studies to compare this against using the global model, Figure 33 shows the FSS against scale for selected case studies in May, July, August as calculated by Roberts and Lean (2008) for 4-hour accumulation periods on a 1km scale with a 6-hour forecast lead time. These graphs cannot be directly compared, but it can be seen that the plots show the same general trend, with FSS increasing as scale increases and tending towards one. The plot also shows variation in the useful scale during the spring/ summer.

For the purpose of this analysis, all types of precipitation have been classed together. Further work could be done to differentiate between different types of precipitation, either by looking at snow or rain or going further to look at the differences between convective (produced by the convection parameterisation scheme) and stratiform rain (represented on the grid).

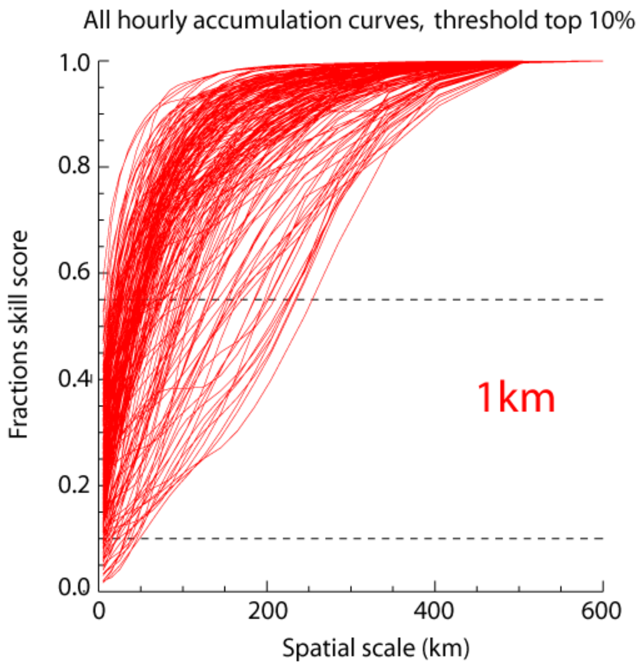



**Figure 33: FSS against scale for 4-hour accumulations with a grid scale of 1 km and a forecast lead time of 6 hours. Individual comparison from case studies in May, July, August (Source: Roberts and Lean, 2008)**

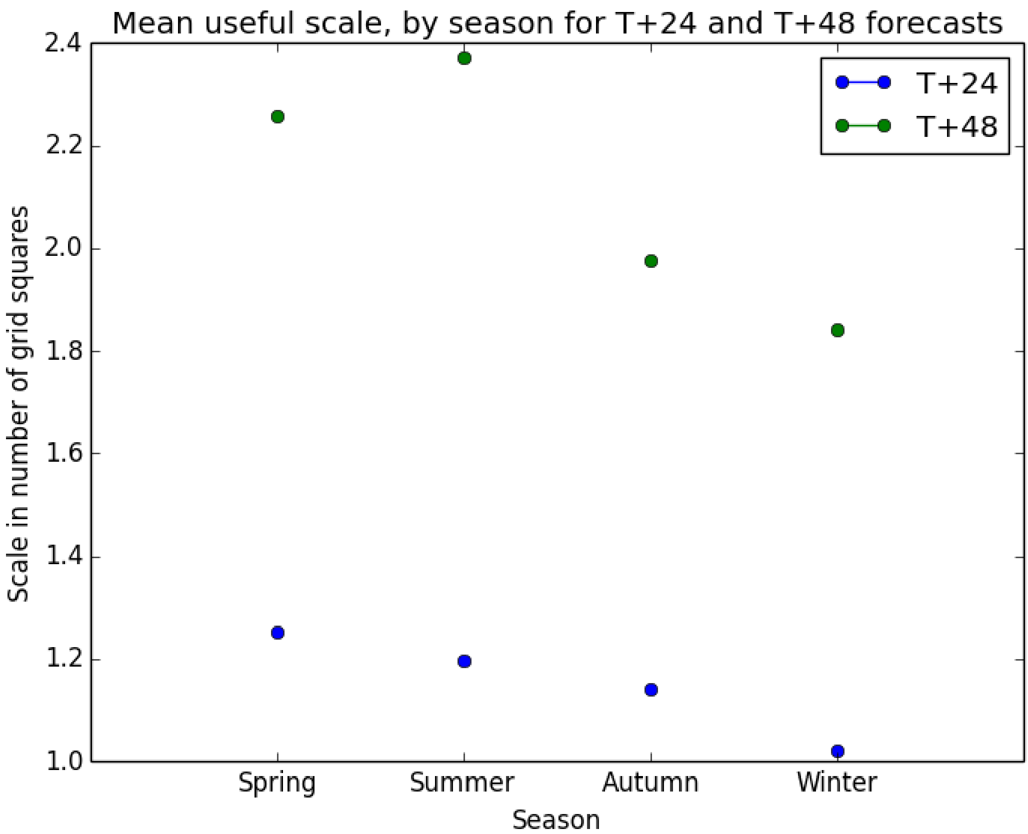



**Figure 34: The mean scale at which forecasts become useful (measured in grid squares) for each season for T+24 (blue) and T+48 (green) forecasts**

## *4.3 Long Range forecasts*

So far it has only been the 24 and 48-hour forecasts that have been examined. How the useful scale varies over longer forecast ranges is also important as it gives an indication of in which situations the forecasts may be considered reliable and when useful scale becomes too large. The series of plots in Figure 35 to Figure 39 shows the FSS against scale for the 90% threshold in March 2014 at T+24, T+48, T+72, T+96 and T+120. The mean scale at which the forecasts become useful are shown in Table 1 and plotted, along with the same results for the 99% threshold, in Figure 40.

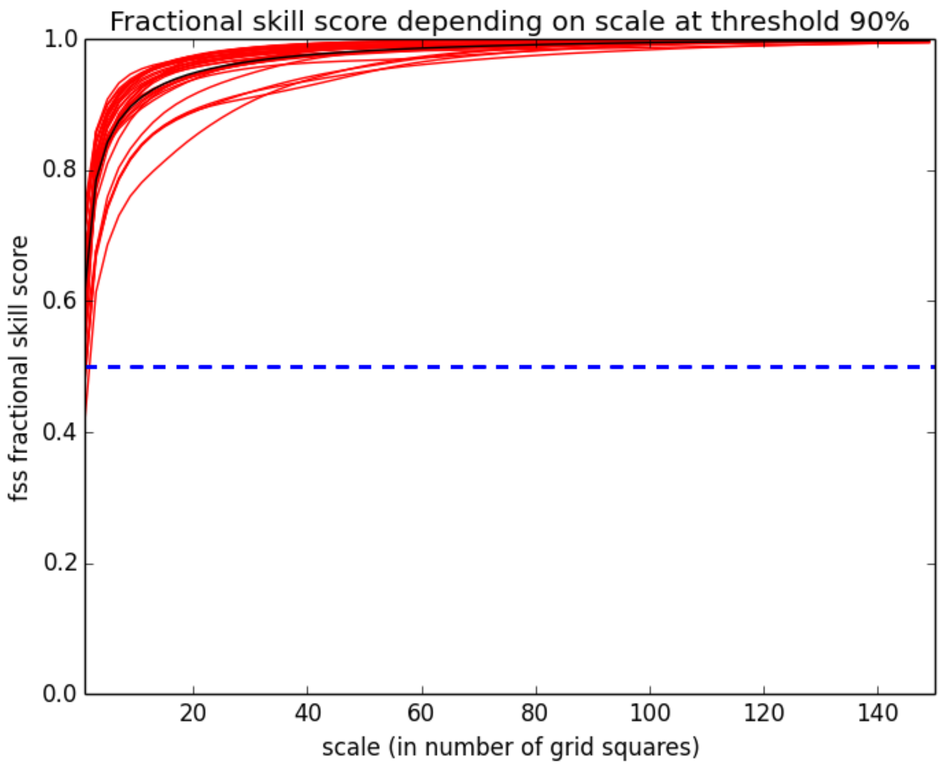



**Figure 35: FSS against scale for 90% thresholds during March for T+24 forecast.**

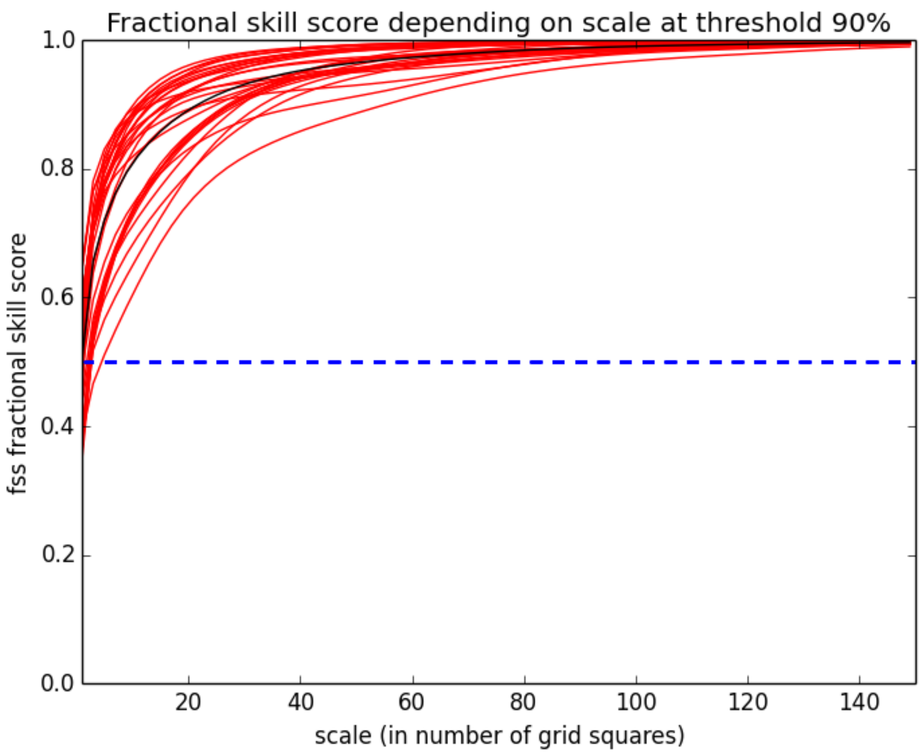



**Figure 36: FSS against scale for 90% thresholds during March for T+48 forecast.**

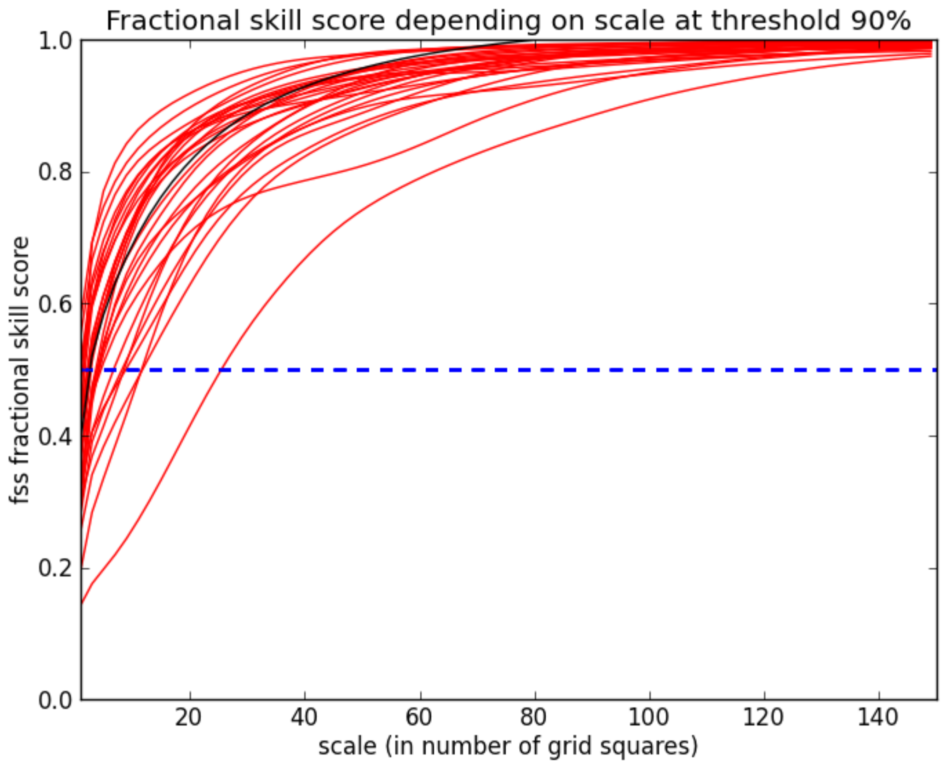



**Figure 37: FSS against scale for 90% thresholds during March for T+72 forecast.**

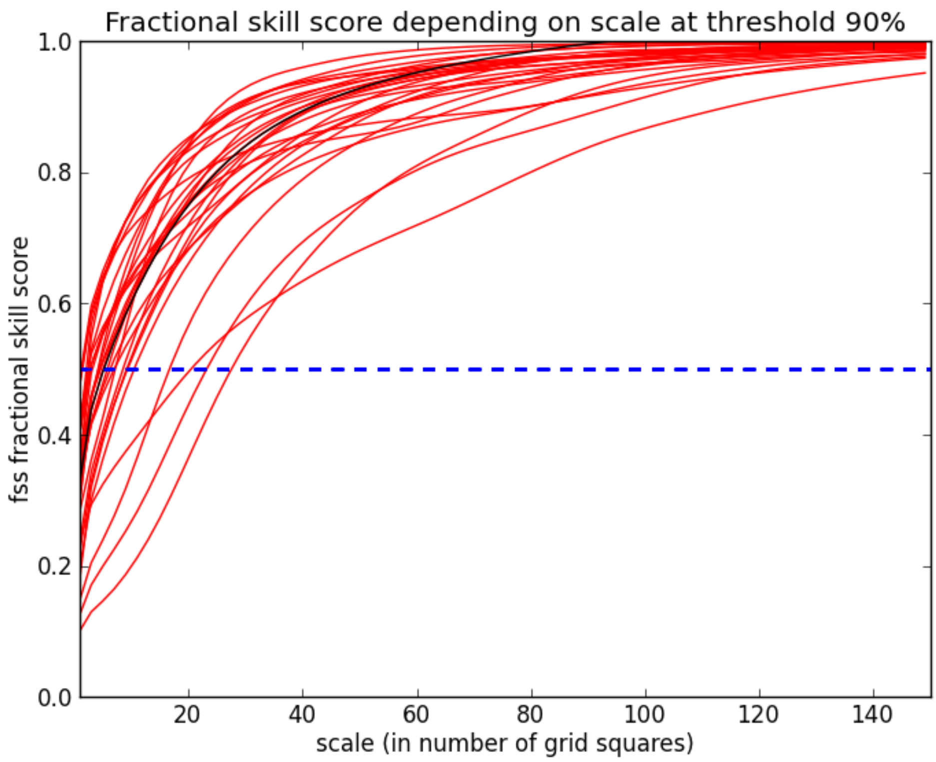



**Figure 38: FSS against scale for 90% thresholds during March for T+96 forecast.**

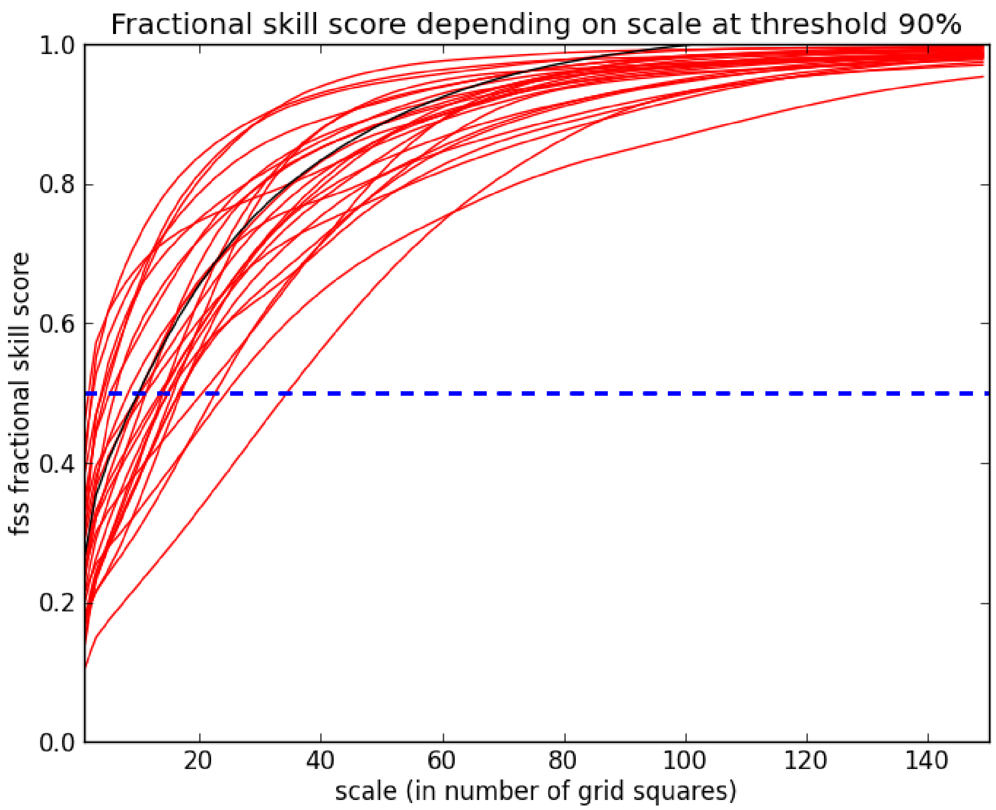



**Figure 39: FSS against scale for 90% thresholds during March for T+120 forecast.**

**Table 4: Mean Useful Scales for different forecast lead times at the 90 and 99% thresholds**

| Forecast Lead Time | Mean Useful Scale | | | |
|---|---|---|---|---|
| | 90% threshold | | 99% threshold | |
| | Grid Squares | Km | Grid Squares | km |
| **T+24** | 1.266 | 31.65 | 6.4 | 160 |
| **T+48** | 1.966 | 49.15 | 14.816 | 370.4 |
| **T+72** | 5.429 | 135.73 | 21.786 | 544.65 |
| **T+96** | 8.259 | 206.48 | 30.111 | 752.775 |
| **T+120** | 13.154 | 328.85 | 37.231 | 930.775 |

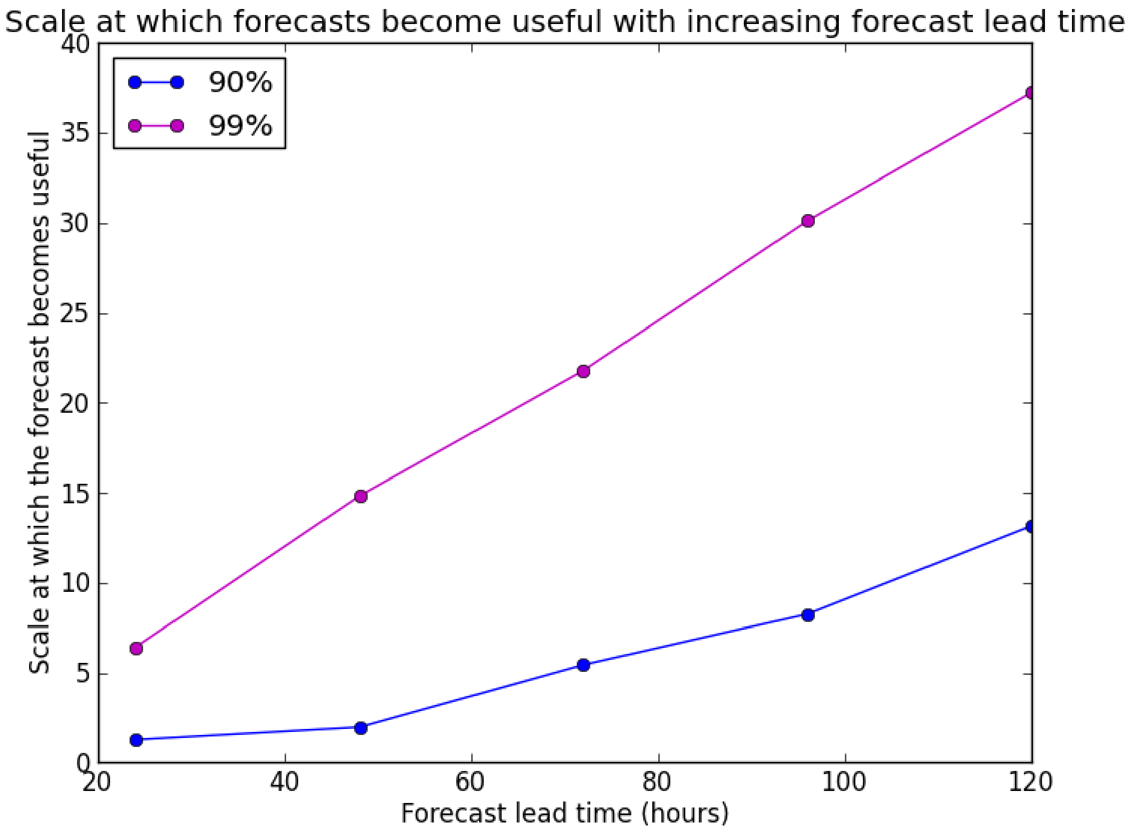



**Figure 40: Useful scale against forecast lead time for 99 and 90% thresholds for March 2014. Scale is shown in terms of grid squares (25km).**

From these plots, it is clear that the skill decreases with forecast lead time. It can also be seen that the spread in forecast error grows with forecast lead time. The 99$^{th}$ percentile is consistently more difficult to predict and the spatial error grows linearly, suggesting that the forecast error has not saturated (grown so poor it cannot get worse). The spatial error of the 90$^{th}$ percentile forecasts grows at a slower rate than for the 99$^{th}$ percentile forecasts and is not linear, with the increase being smaller at shorter lead times.

As Figure 40 shows, the useful scale increases with forecast lead time. With shorter lead times the useful scale is close to grid scale but by the T+130 forecast, the useful scale is closer to 10 grid squares (250km), an area approximately equal to that of a small country, such as Latvia.

For the forecast ranges shown, the curves show no sign of flattening, which would indicate saturation. It is expected that if further forecast ranges looking further ahead were to be examined, at some point the forecasts would all become equally poor and the increase in error would stop growing. However, for the forecast ranges shown (even out to T+120) the model can be considered sufficiently good at predicting the general areas of precipitation in the correct place and therefore it is locating frontal bands relatively close to where the model analysis locates them. In principle, this means that it should be reasonable to embed a regional or local convection-permitting model within the global model. This is because the convection-permitting model can more effectively represent convection cells provided that the global model places them close to the observed location. However, this would still need to be interpreted probabilistically as the showers may not be correctly located exactly by the model.

However, the drop in skill for the 99$^{th}$ percentile suggests that the use of the global model to predict the location of extremes would only be possible if a large margin of spatial error is given. If the model gives a precipitation amount that is much higher than its own climatology, meteorologists interpret this to mean that there is a chance of flood-producing rain, which may not be very effective if there is a large spatial displacement. Nevertheless, the output is still useful as it can be taken as an indication that precipitation is expected in the wider area and can be used in combination with output from regional and local models. It may be that looking at accumulations for the 99$^{th}$ percentile gives a different picture. Further research into this is required.

The flatness of the 90$^{th}$ percentile curves between T+24 and T+48 compared to later times could be because errors are slower to grow early on for the more widespread rain. However, it could also just be due to the sample size being too small as the plots show only one month.

The faster growth in the errors for the 99$^{th}$ percentile suggests faster error growth for the more localised precipitation. It may be that, if longer range forecasts were to be included, this curve would eventually flatten out whilst the 90$^{th}$ percentile curve increases to show similar forecast errors.

## *4.4 Predictability periods*

As discussed above, the scale at which a forecast becomes useful varies from one day to the next and between different meteorological conditions. The scale at which the forecast becomes useful can be plotted against day of the year to show periods where the forecasts are better/worse. Figure 41 shows such a graph for the T+24 forecasts with a 90% threshold. From this it can be seen that, at this forecast range most forecasts are useful at the grid scale. It should be noted that, due to the method used, the scale at which the forecast becomes useful is measured in units of grid squares and that increase in increments of 2. Therefore the useful scale, in km, will be 25, 75, 125, 175… km. In some instances there is a period of a few days where the useful scale increases, in other instances it is only one day where the useful scale increases. When it increases for longer than one day it suggests a period of lower predictability.

The gaps in the data are where either data was missing from the Met Office Archive or a comparison was not made either due to there not being enough grid cells containing precipitation for the 90th percentile to be taken, or due to problems with overlapping the start and end of the seasons.

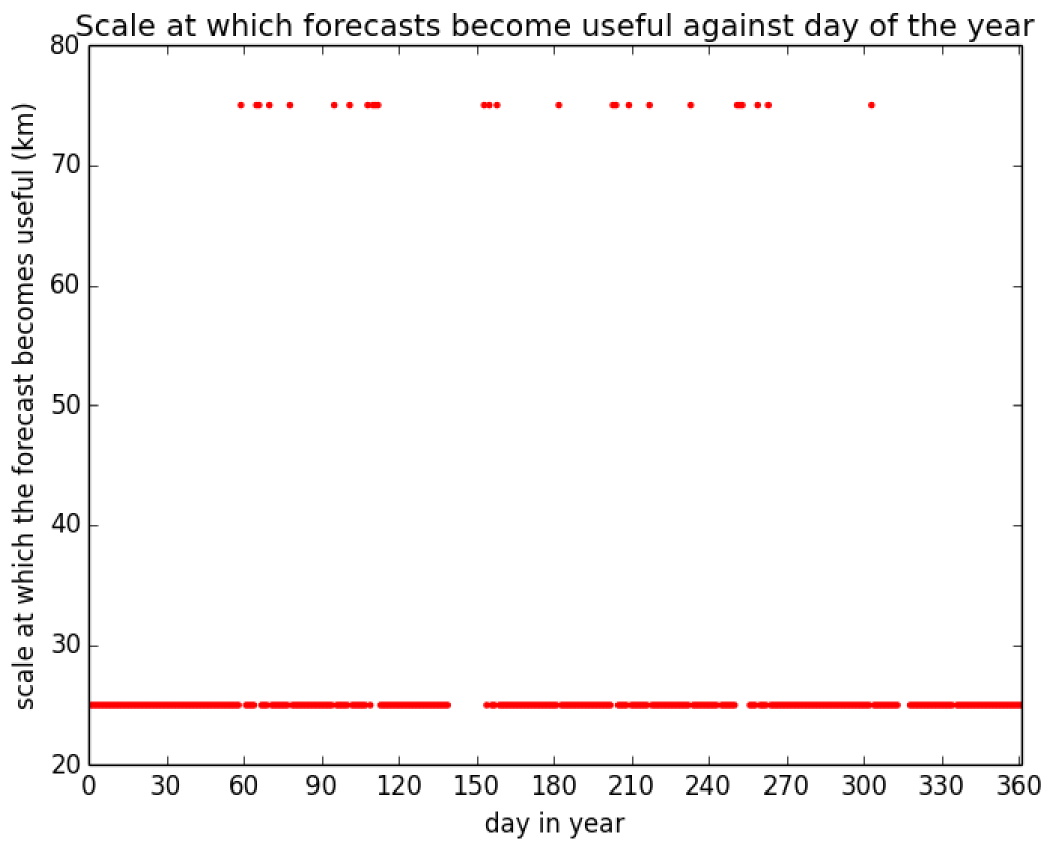



**Figure 41: Scale at which the forecast becomes useful against day of year for T+24 forecasts.**

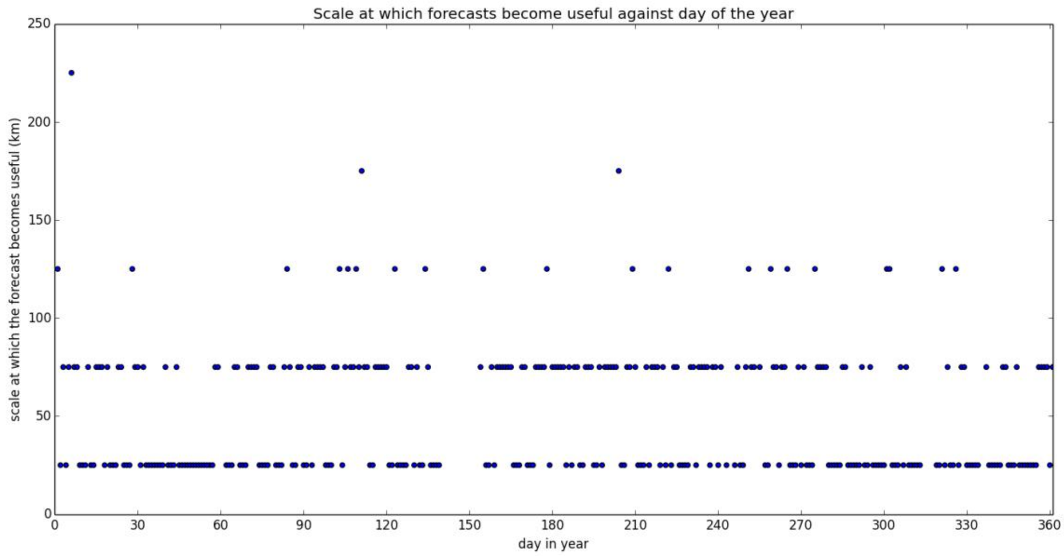



**Figure 42: Scale at which the forecast becomes useful against day of year for T+48 forecasts.**

Earlier it was seen that the forecasts for T+24 are all considered useful at either 1 or 3 grid squares. It can be seen from Figure 41 that most of the forecast are useful at grid scale. The mean useful scale is 28.25 km, just above grid scale. The mean has been used in this case, to take into account all the useful scales for all the forecasts however it is worth considering that the median and mode are both 25km (1 grid square). These techniques also give a middle value that gives a better representation of the scale on which we can calculate. The mean suggests that it is possible to interpolate between neighbourhood sizes and assumes a linear trend which is unknown.

At T+48 forecast range, there is more variability in the useful scale. More forecasts (93%) are useful at 3 grid squares with another 6% reaching 5 grid squares, 1% reaching 7 and <1% reaching 9 grid squares before being considered useful. This is important because it makes it more difficult for forecasters to say at what scale the forecast is useful.

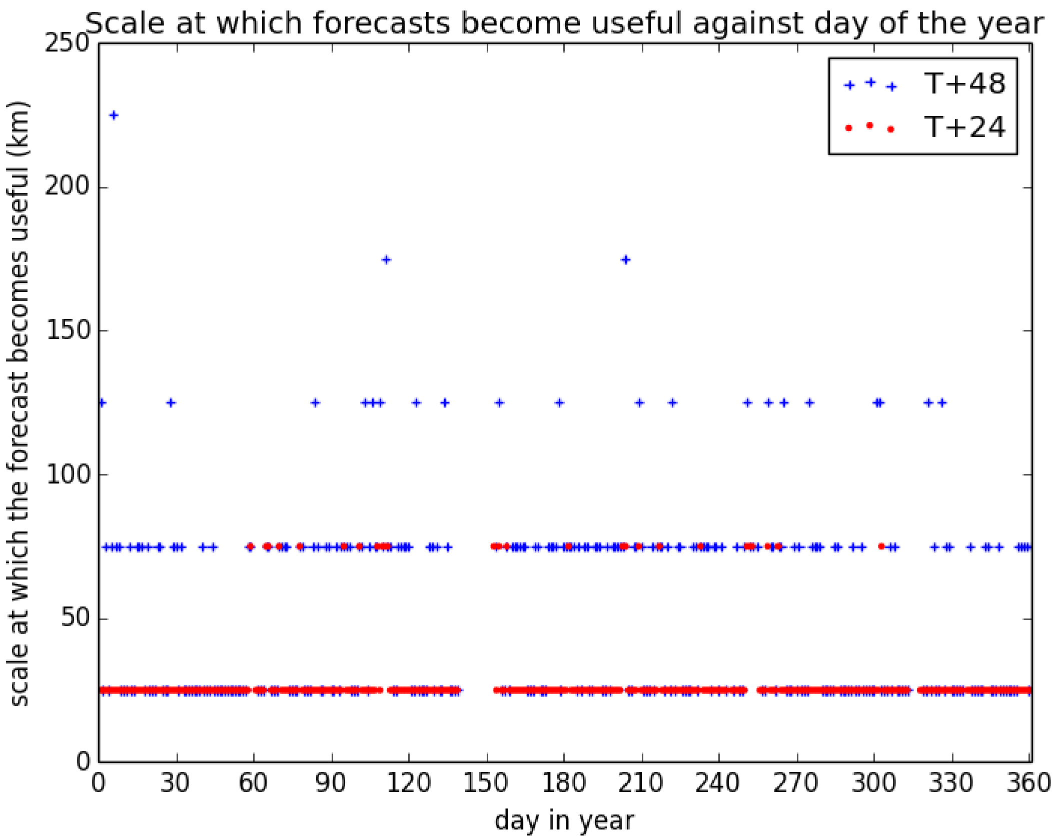



**Figure 43: Scale at which the forecast becomes useful against day of year for T+24 (red) and T+48 (blue) forecasts to allow for comparison**

It would be useful for meteorologists to be able to know, when a forecast is produced, what scale they expect it to be useful on. The useful scale on each day for the T+24 and T+48 forecasts has been plotted on the same Figure,

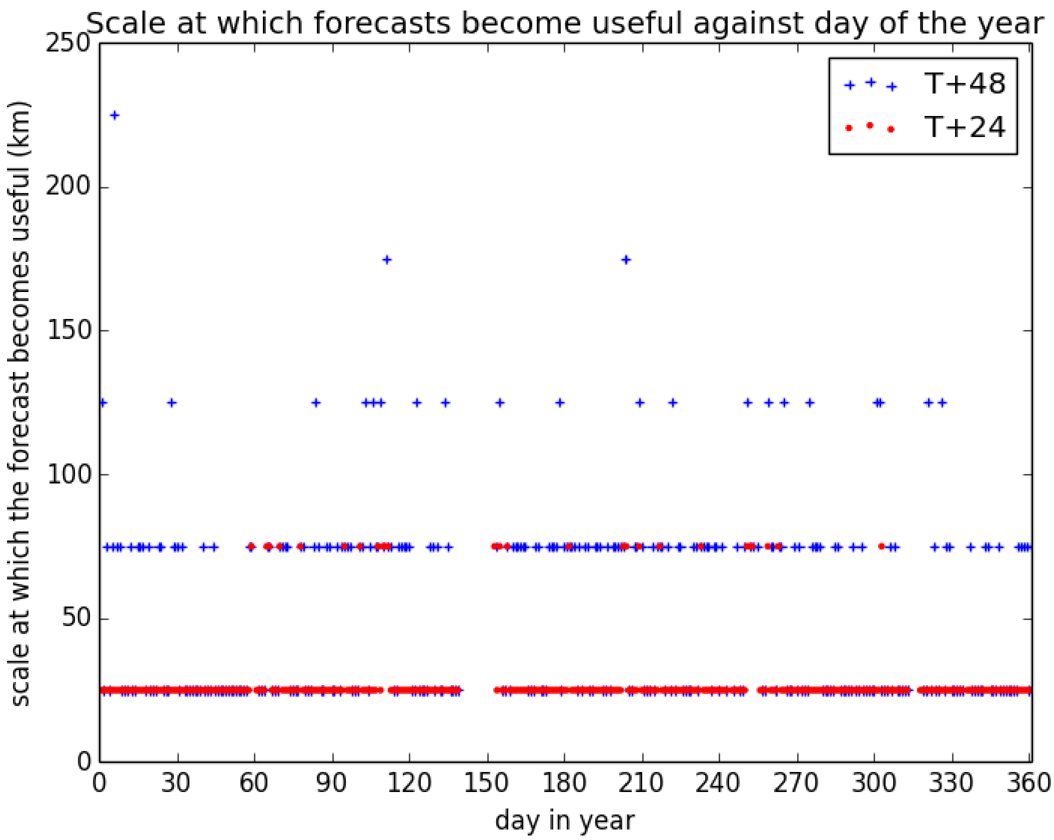



Figure 43, to allow a comparison to be made. It can be seen that, if the useful scale at T+24 is 3 grid squares, then the T+48 forecast is also not accurate at grid level. However there are times, such as early January when the T+48 is useful at the largest scale, the T+24 forecast is useful at grid scale. The largest values are one off occurrences and do not repeat regularly, but do tend to occur during periods when the forecast is not useful at grid scale. Further research is required to be able to say if these periods of less predictability are connected to certain weather events or precipitation. To be able to pinpoint the cause of this loss of predictability from grid scale would allow meteorologists using the global model data to predict the scale at which a new forecast would be useful as it is produced, meaning that probabilistic forecasts can be altered accordingly and that the positioning of fronts and bands of rain in nested regional and local models stepped down from the global model can be taken into account.

By separating the seasons, it can be seen more clearly how predictable each season is. Table 5 shows the number of days on which a forecast becomes useful at the given scale for the year combined as well as for each season individually for the T+24 forecasts. Whilst 92% of the forecasts are useful at grid scale over the whole year, remarkably, 99% of forecasts were useful at grid scale during the winter season. On the other end of the scale, during spring, 87% of forecasts were useful at grid scale.

**Table 5: Number of Forecasts becoming useful at each scale for whole year and by season for T+24 forecasts.**

| Scale | Year | Summer | Autumn | Winter | Spring |
|---|---|---|---|---|---|
| **1 (25km)** | 317 | 82 | 78 | 88 | 69 |
| **3 (75km)** | 26 | 9 | 6 | 1 | 10 |
| **Total** | **343** | **91** | **84** | **89** | **79** |

**Table 6: Number of forecasts becoming useful at each scale for whole year and by season for T+48 forecasts.**

| Scale | Year | Summer | Autumn | Winter | Spring |
|---|---|---|---|---|---|
| **1 (25km)** | 177 | 34 | 50 | 56 | 37 |
| **3 (75km)** | 137 | 50 | 24 | 29 | 34 |
| **5 (125km)** | 20 | 4 | 8 | 2 | 6 |
| **7 (175km)** | 2 | 1 | 0 | 0 | 1 |
| **9 (225km)** | 1 | 0 | 0 | 1 | 0 |
| **Total** | **337** | **89** | **82** | **88** | **78** |

Table 6 shows the same figures for the T+48 forecast. Over the year only 53% of forecasts can be considered useful at grid scale. However this varies from 38% in the summer to 64% in winter. It should be noted that, whilst the winter has more days useful at grid scale, it also has the largest useful scale over the whole year. This occurred on 6th January 2014 where the useful scale is 9 grid squares or 225 km. The conditions on this day are discussed in Section 3.5.2.

## *4.5 Thresholds*

Throughout the project, threshold precipitation rates have been used to allow only the spatial variation in forecasts examined and not precipitation-bias. The range of values, in mm/hr for the 90% threshold, as calculated and used in the code can be seen in Figure 44 showing a separate plot for each season. The distribution roughly follows a normal distribution, with values ranging between just above 0 to 0.6mm/hr. The spread is lowest in the winter and the largest in the summer. The combined result can be seen in Figure 45. The most common threshold is between 0.2 and 0.25mm/hr.

The different plots for the 99% thresholds are shown in Figure 46. There is a larger spread in thresholds with values up to 3 or 4 mm/hr. The distribution is similar to the 90% thresholds. These values are generally larger than the 90% thresholds.

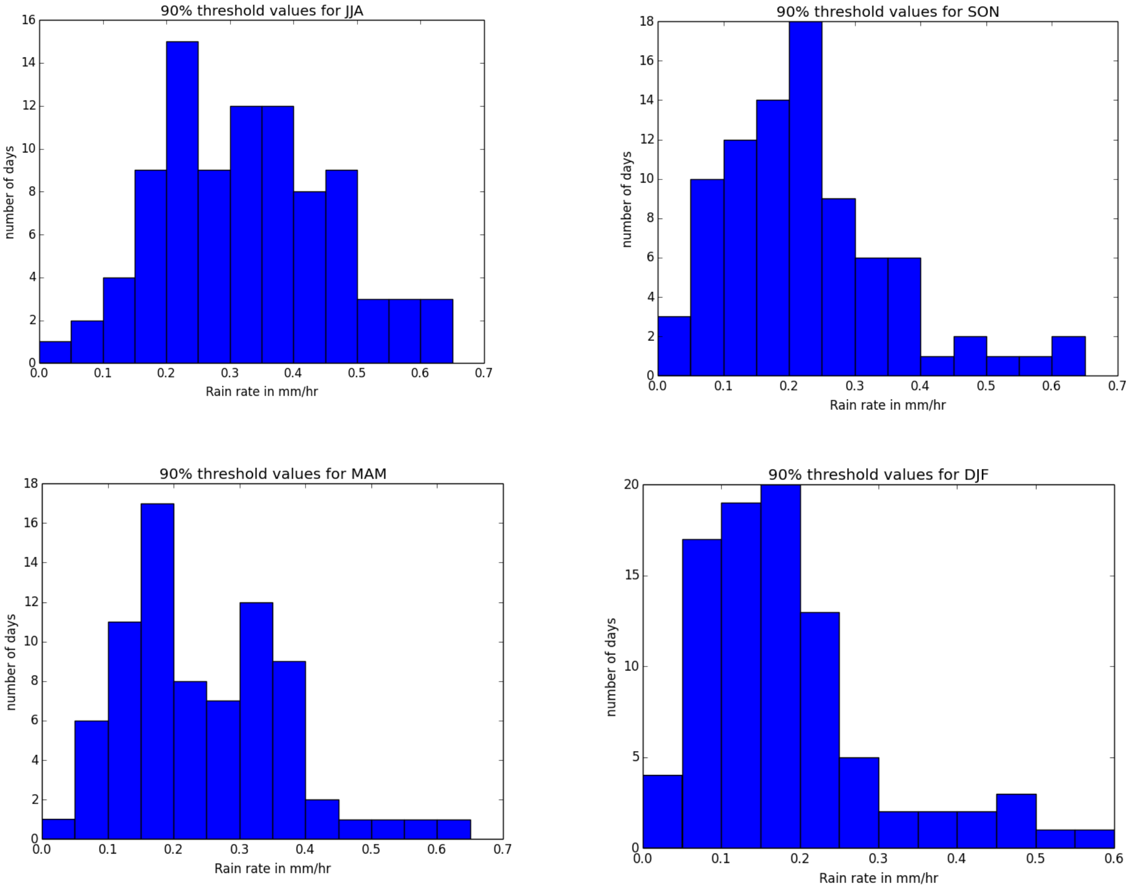



**Figure 44: 90% threshold rainfall rates in mm/hr. Clockwise from top left: JJA, SON, DJF MAM.**

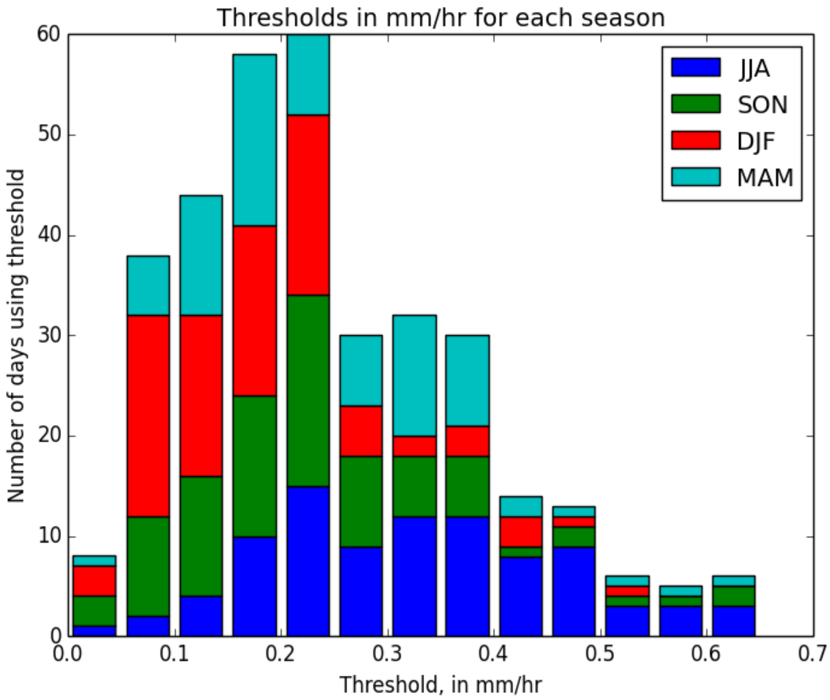



**Figure 45: 90% thresholds used combined for year. Values in mm/hr.**

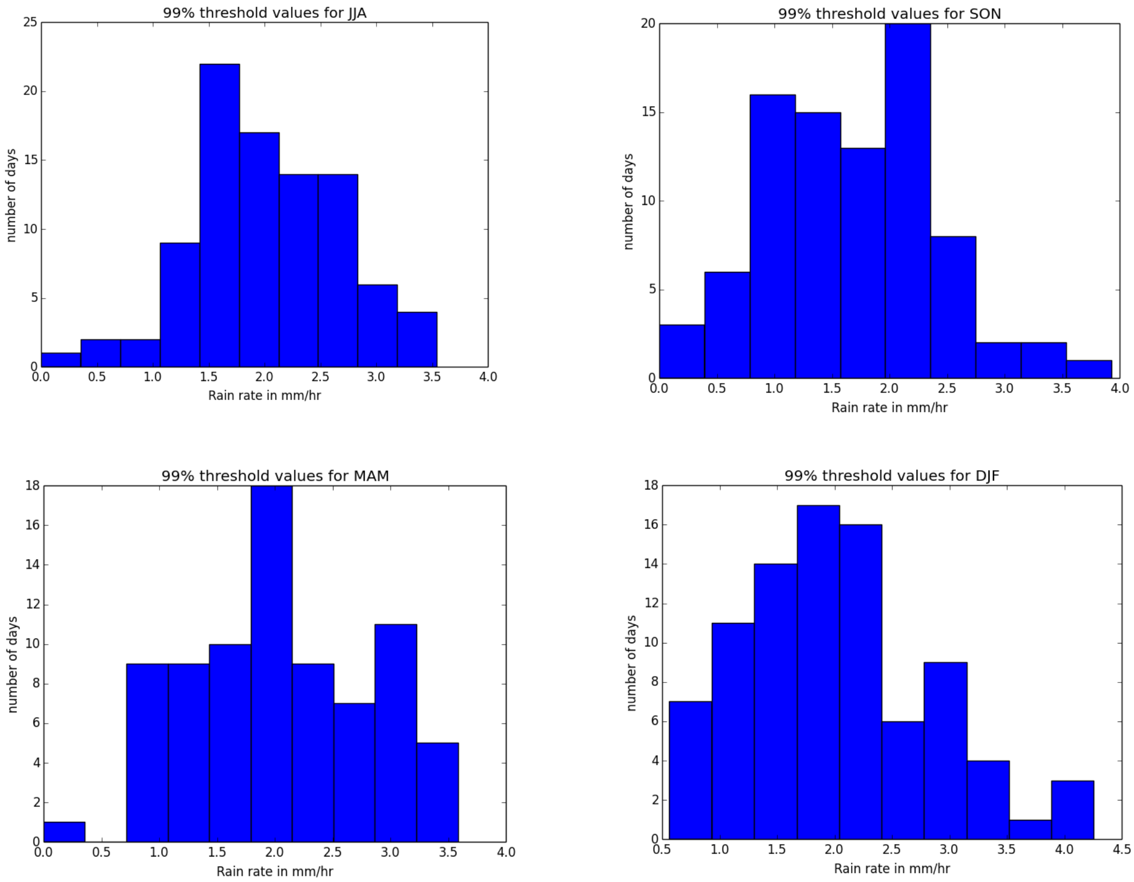



**Figure 46: 99% threshold rainfall rates in mm/hr. Clockwise from top left: JJA, SON, DJF MAM**

# 5. Conclusion

The aim of the project was to use the Fractions Skill Score (FSS) to examine the spatial scales over which forecasts from the Met Office global model can be regarded as having useful skill at different forecast lead times (forecast lengths). In this section, the method and the key results are summarised along with assumptions or problems encountered and ideas for future investigations.

## 5.1 The Method

The FSS was used to examine the spatial skill of a years' worth of global model precipitation forecasts (2013-2014). This method has not been used for a global model before because the technique was originally intended for fine-scale regional model comparison with radar and as such some problems were encountered and had to be resolved satisfactorily. A summary is given below.

### 5.1.1 Observational Data

In previous studies, focussed regional models, radar data has been used as the observational dataset against which comparisons are made. However, there is currently no global radar coverage so it would be impossible to satisfactorily verify a global model in this way. It may be possible, in a future study, to use a sub-section that is in an area where radar data gives complete coverage but this would be difficult to obtain and, as the resolution would be different, could be problematic.

Added to that, the global observation network, as seen in Section 1, does not provide sufficient observations for the whole globe and the observations are not representative of the precipitation produced by a global model grid square (~25x25km over UK). Observations are scarce over the ocean and in more remote areas such as the Sahara desert.

Therefore, it was decided that the best estimate of the observational state would be the model analyses produced at the start of each forecast run. The disadvantage of this approach is that the observational dataset is a combination of observations and model fields and is therefore not a true observed precipitation field. The benefit is that there is full spatial coverage and because the observed and forecast precipitation is on the same grid there are no problems with trying to verify precipitation over a model grid square against point observations. On balance, the benefits outweigh the disadvantage because the purpose is to examine the spatial accuracy of the forecasts and for this purpose a forecast that is very similar to the analysis can be deemed to be a very good forecast. For this project the T+0 charts from the 12UTC model run have been used for comparison. However, a further study might also look into the use of the T+6 forecasts from the 06UTC run as it is recognized that the analysis charts at T+0 may be biased in precipitation extent or amount compared to the forecasts because of the data assimilation process.

### 5.1.2 The grid size

As the Earth is a sphere, when a grid is projected onto it the grid squares vary in size and shape, with the grid at the poles being almost triangular. As the FSS assumes constant grid size, this means that a sub-section of the grid must be used instead. By careful selection, a sub-section can be chosen in which the grid squares are consistent enough that a standard grid size can be assumed. It is recommended that future work is conducted into potential methods to account for changing grid size as this would allow comparisons to be made on a larger scale and also at locations closer to the poles.

### 5.1.3 The Sub-section

It is not easy to apply the FSS to the full global model domain because the global model uses a latitude-longitude grid, which means that the distance between longitude points varies considerably between the equator and the poles, so there would have to be an additional latitude-dependent factor included when interpreting spatial distances. In addition, it is not straightforward to calculate the FSS when the grid wraps around (no edges). For those reasons, for this project, it was more sensible to extract a sub-domain covering most of Europe over which the assumption can be made that each model grid square is ~25km, and because the domain chosen is largely over land the analyses can be trusted more because there are more observations included in the data assimilation.

## 5.2 Annual Data

The data used has been from July 1st 2013-May 31st 2014. This should be a sufficiently large sample to draw some general conclusions but further work may take data from other years in comparison with this to see if the conclusions formed here are consistent for all years or if this year is unusual. There are a few days of missing data and some data at the start of a season may be missing if the data from the longer range forecasts at the end of the previous season are missing. This could be avoided by using the data set as a whole rather than each season individually and combining the results. However this involves large volumes of data that were unmanageable for this scale of project. The number of days missing is so small it is not expected to make a big difference.

It has been shown that forecast skill improves with spatial scale - it is easier to forecast rain somewhere within a 75km square than a 25km square. Although this result was expected, it is important to show. Forecast skill has been shown to drop off with forecast length and the spread of forecast skill increases with forecast length as poor forecasts occur more frequently. This was expected, but is important because it gives confidence in the more quantitative conclusions that follow. We also see that the more widespread precipitation ($90^{th}$ percentile threshold) is easier to forecast than the more localised precipitation ($99^{th}$ percentile).

### *5.2.1 T+24*

On average over the whole year, the T+24 forecasts are useful at grid scale over 90% of the time for a 90% threshold. For the rest of the time they are useful at 3 grid squares or 75km. The mean useful scale is 29km. This indicates that 24-hour forecasts are usually very good at putting the precipitation into the correct areas.

### *5.2.2 T+48*

At T+48, the scale at which the forecast become useful varies more. Only 50% of the forecasts are useful at grid scale. The largest scale that the forecasts are considered useful is 9 grid squares (225km) which occurs only once in the 337 days the forecasts have been compared. Further analysis of data from different years is required to be able to show if this is exceptional or not. The mean useful scale is 53km. This indicates that even 48-hour forecasts of precipitation areas have not degenerated much spatially, although an uncertainty of around 50 km should be considered when using these forecasts.

## *5.3 Seasonal differences*

The spatial skill of the forecasts varies with season. The mean scale at which useful skill is achieved is lower in the winter and autumn than in spring and summer. For a T+24 forecast, the mean useful scale is largest in the spring but for a T+48 forecast it is largest in the summer. It is true to for both 90% and 99% thresholds. This difference between seasons may be to do with the different nature of the precipitation in the different seasons. In spring and summer, more of the precipitation over land comes from convection which can tend to be more spatially localised and less temporally coherent than the frontal precipitation bands more evident in the winter and autumn.

## *5.4 Longer Range Forecasts*

The month of March has been analysed for forecasts up to 5 days ahead. The analysis for March can be taken as a reasonable guide for the other months, expecting the autumn and winter months to fare slightly better. However, this has yet to be tested and is an avenue for further work. Forecast spatial skill continues to decline out to 5 days. The spatial skill for the more localised precipitation drops more rapidly than for the more widespread precipitation and shows a linear decline in the scale at which useful skill is achieved. For a T+120(5 day), $90^{th}$ percentile forecast then the mean useful scale is 13 grid squares or 325km. This means that for those forecast lengths the forecasts can be used to forecast the occurrence of precipitation somewhere within larger areas such as larger countries or parts of continents rather than smaller countries or regions. The hope is that improvements in NWP models in coming years will reduce the size of this spatial error at all lead times. The use of this spatial approach is a good way of measuring the performance of global model precipitation forecasts and providing a way of detecting future improvements. Other traditional metrics, such as RMSE of surface pressure, are starting to become less useful because it is less relevant to the weather that people actually experience.

## *5.5 Thresholds*

Using the percentage thresholds works well for analysis of the spatial variation. Future studies could look at using set absolute thresholds to look at forecast bias. This would show any consistent over- or under-estimation of the rain rate. The thresholds used equate to around 0.1-0.5 mm/hr.

## *5.6 Further Work*

This project has completed a basic analysis of how the skill of precipitation forecasts depends on spatial scale, forecast length and season. There are many areas into which further work could expand. The suggestions below are based on the results of this project and comparison with current literature investigating different scales.

Throughout the project, the data has been split into four 'seasons' in order to compare between them. This showed that forecasts are easier to predict in the winter and hardest in the spring, at T+24 range and summer at T+48 forecast range. Further study may show whether variation between shorter periods such as months are in fact greater and are cancelled out by calculating over the season. Conversely, whilst the data set was too large to allow manipulation of the data set as a whole for the year, the results of a study into this may show whether certain periods (either in months or seasons) may be taken as a representation of the skill of the forecasts over the period of a whole year.

The area, over which the analysis takes place is mainly Eastern Europe. Whilst analysis charts are likely to be less accurate over the ocean, due to the lack of observations, the author would be interested to see whether the surface type affects the spatial scale at which the model becomes useful. By adapting the code to allow for the change in grid size due to the curvature of the earth, this could be developed further to include various surface types including desert, mountains, ocean and snow. However, this would also require research into the skill at different latitudes which may or may not be linked to the surface type. So far, only one sub-area (and therefore only one latitude) has been analysed and this may or may not represent the whole.

Throughout this project, all types of precipitation have been classed together. Further work could be done to differentiate between different types of precipitation, either by looking at snow or rain or going further to look at the differences between convective (produced by the convection parameterization scheme) and stratiform rain (represented on the grid).

The data from the model runs out to T+168 forecast. Analysis on these time scales has only been completed out to the time period T+120 and full analysis only for T+24 and T+48 forecasts. Further work must be done in the other seasons. Another option is to look at the skill of persistence forecasts by comparing the analysis data with the analysis data from the day before. 'Persistence' is often a measure against which models are compared to show the value they add so knowing the skill of persistence at different scales would be useful for verification scientists.

By using absolute value thresholds and plotting the FSS against scale, any bias in the model forecast would be seen as the FSS would not tend towards one. Further research into this is also required.

The analysis only looks at one domain size. Further research could look at the same area but with different domain sizes. Using a bigger or smaller grid may mean that more or less precipitation is included and may affect the skill score. Likewise, by using alternative methods for dealing with the domain edge, for example, including the precipitation in grid squares within the global model but just outside the sub-area in fractions calculations.

Analysis has been completed on the changing scale for FSS=0.5. Further analysis may look at how the FSS varies with forecast range at grid scale and other scales. This would be expected to show a decrease in skill with time.

# 6. References

Brandt, J, Skjøth , C.A., Christensen, J.H., Frohn, L.M., Geels, C., 2005: High-resolution weather forecasting using dynamical downscaling, *International conference on innovation advances and implementation of flood forecasting technology*.

Brier, G., 1950: Verification of Forecasts expressed in terms of Probability, *Mon. Wea. Rev.*, **78**, 1-3.

Brier, G. W., & Allen, R. A., 1951: Verification of weather forecasts. *Compendium of meteorology*, 841-848.

Casati, B, Ross, G, Stephenson, D.B, 2004: A new intensity-scale approach for the verification of spatial precipitation forecasts, *Meteorol. Appl.*, **11**, 141-154, doi: 10.1017/S1350482704001239.

Culverwell, I.D., Milton, S.F., 2000: An Evaluation of the Met. Office Global NWP Model by Comparison with ARM Data in the Tropical Western Pacific, *Tenth ARM Science Team Meeting Proceedings, San Antonio, Texas.*

Davies, T, Cullen, M.P.J, Malcom, A.J, Mazson, M.H, Staniforth, A, White, A.A., Wood, N, 2005: A new dynamical core for the Met Office's global and regional modelling of the atmosphere, *Q.J.R.Meteorol.Soc.*, **131**, 1759-1782, doi:10.1256/qj.04.101.

Ebert, E., 2008: Fuzzy verification of high-resolution gridded forecasts: a review and proposed framework, *Meteorol. Appl.*, **15**, 51-64, doi: 10.1002/met.25.

ECMWF, cited 2014: GEMS Architecture modelled on WMO's World Weather Watch. [Available at: http://gems.ecmwf.int/documents/workdescription/2_4_GEMS_Architecture_modelled_on_WMO_rsquo_s_World_Weather_Watch.html].

Gilleland, E., Ahiyevych, D., Brown, B.,G., Casati, B., Ebert, E., 2009: Intercomparison of Spatial Forecast Verification Methods, *Wea. Forecasting*, **24**, 1416-1430, doi:10.1175/2009WAF2222269.1.

Golding, B., 2012: Olympics 2012 – showcasing weather science, *MOSAC*, **17**, 137-144.

Gregory, D., Kershaw, R., Inness, P., 1990: Parametrization of momentum transport by convection. II**:** Tests in single-column and general circulation models, *Q. J. R. Meteorol. SOC.*, **123**, 1153-1183.

Gregory, D., Rowntree, P.R., 1990: A mass flux convection scheme with representation of cloud ensemble characteristics and stability dependent closure. *Mon. Wea. Rev.,* **118,** 1483-1506.

Harrison D.L., Driscoll, S.J., Kitchen, M., 2000: Improving precipitation estimates from weather radar using quality control and correction techniques. *Meteorol. Appl.,* **7**, 135-144.

Holloway, C.E., Woolnough ,S.J., Lister, G.M.S., 2012: Precipitation distributions for the explicit versus parameterised convection in a large-domain high-resolution tropical case study, *Q.J.R.Meteorol.Soc.*, **138**, 1692-1708, doi:10.1002/qj.1903

Inness, P., Dorling S., 2013: *Operational Weather Forecasting*, Wiley-Blackwell, 231pp.

Jolliffe, I.T., Stephenson, D.B., 2012: *Forecast Verification, A Practioner's Guide in Atmospheric Science,* Wiley- Blackwell

Kunkel, Kenneth E., Roger A. Pielke, Stanley A. Changnon, 1999: Temporal Fluctuations in Weather and Climate Extremes That Cause Economic and Human Health Impacts: A Review. *Bull. Amer. Meteor. Soc.*, **80**, 1077–1098.
doi: 10.1175/1520-0477(1999)080<1077:TFIWAC>2.0.CO;2

Landberg, L., 1999: Short-term prediction of the power production from wind farms, *Journal of Wind Engineering and Industrial Aerodynamics*, **80**, 207-220, doi: 10.1016/S0167-6105(98)00192-5

Lean, H., Clark, P.A., Dixon, M., Roberts, N., Fitch, A., Forbes, R., Halliwell, C., 2008: Characteristics of High-Resolution Versions of the Met Office Unified Model for Forecasting Convection over the United Kingdom, *Mon. Wea. Rev.*, **136**, 3408-3424, doi: 10.1175/2008MWR2332.1

Mesinger, F., 2008: Bias Adjusted Precipitation Threat Scores, *Adv. Geosci.*, **16**, 137–142.

Met Office, 2009: National Meteorological Library and Archive Fact Sheet 15- Weather radar

Met Office, 2011: Science and Technology Committee Report.  [Available online at: http://www.publications.parliament.uk/pa/cm201012/cmselect/cmsctech/1538/1538we02.html

Met Office, cited 2014: Operational Numerical Weather Prediction [Available online at: [http://research.metoffice.gov.uk/research/nwp/numerical/operational/]

Mittermaier, M., 2014: A Strategy for Verifying Near-Convection-Resolving Model Forecasts at Observing Sites, *Wea. Forecasting*, **29**, 185-204, doi: 10.1175/WAF-D-12-00075.1

Mittermaier, M., Roberts, N., 2010: Intercomparison of Spatial Forecast Verification Methods: Identifying Skilful Spatial Scales Using the Fractions Skill Score, *Wea. Forecasting,* 343-354, doi:10.1175/2009WAF2222260.1

Mittermaier, M., Roberts, N., Thompson, S.A., 2013: A long-term assessment of precipitation forecast skill using the Fractions Skill Score, *Meteorol. Appl.*, **20**, 176-186, doi: 10.1002/met.296

Murray, K.B., Di Muro, F., Finn, A., Leszczyc, P.P., 2010: The effect of weather on consumer spending, *Journal of Retailing and Consumer Services*, **17**, 512-520, doi:10.1016/j.jretconser.2010.08.006

Roberts, N. M., 2003: The impact of a change to the use of the convection scheme to high resolution simulations of convective events. *Met Office Forecasting Research Tech. Rep. 407*, 31.

Roberts, N., (2005): An investigation of the ability of a storm scale configuration of the Met Office NWP model to predict flood producing rainfall. *UK Met Office Tech. Rep*, *455*, 80.

Roberts, N., 2008: Assessing the spatial and temporal variation in the skill of precipitation forecasts from an NWP model, *Meteorol. Appl.*, **15**, 163-169, doi: 10.1002.met57

Roberts, N., 2014: Personal Correspondence

Roberts, N. and Lean, H., 2008: Scale-Selective Verification of Rainfall Accumulations from High-Resolution Forecasts of Convective Events, *Mon. Wea. Rev,* **136**, 78-97, doi:10.1175/2007MWR2123.1.

Roemmich, D., G.C. Johnson, S. Riser, R. Davis, J. Gilson, W.B. Owens, S.L. Garzoli, C. Schmid, and M. Ignaszewski, 2009: The Argo Program: Observing the global ocean with profiling floats, *Oceanography,* **22(2),** 34–43, doi:10.5670/oceanog.2009.36.

Russ, J.C., 2007: *The Image Processing Handbook, Fifth Edition*, CRC Press

Simonsohn, U., 2009: Weather to go to College, *The Economic Journal*, **120**, 170-180, doi:10.1111/j.1468-0297.2009.02296.x.

Theis *et al*, S.E. Hense, A., Damrath, U., 2005: Probabilistic precipitation forecasts from a deterministic model: a pragmatic approach, *Meteorol. Appl.*, **12**, 257-268, doi:10.1017/S1350482705001763.

Thorne, J.E., 1977: The effect of weather on sport. *Weather*, **32**, 258-263, doi:10.1002/j.14778696.1977.tb04568

Wang, P., Smeaton, A, Lao, S., O'Connor, E., Ling, Y., O'Connor, N., 2008: Short-Term Rainfall Nowcasting: Using Rainfall Radar Imaging, *XYZ Symposium*.

Weusthoff, T., Ament, F., Arpagaus, M., Rotach, M.W., 2010: Assessing the Benefits of Convection-Permitting Models by Neighborhood Verification: Examples from MAP D-PHASE. *Mon. Wea. Rev.*, **138**, 3418–3433. doi: 10.1175/2010MWR3380.1.

WMO, 2008: Measurement of Precipitation, *Guide to Meteorological Instruments and Methods of Observation*, 1.6-1 – 1.6-23

Zacharov, P., Rezacova, D., 2009: Using the fractions skill score to assess the relationship between an ensemble QPF spread and skill, *Atmospheric Research*, **94**, 684–693.

## *Appendix A- Python Code*

## *Code 1- input*

```
"""Input.py
@Author: Bethan Jones
"""
import numpy as np
import fsstools
import random
import matplotlib.pyplot as plt
import time
from datetime import date, timedelta
import plotting2 as plotting
import sorting
import matplotlib.mlab as mlab
import os
import glob

#set lat/lon grid
lat0=90.2340702210663199
lon0=-0.3515625
lat=np.zeros(770)
lon=np.zeros(1025)
for i in range(0,770):
   lat[i]= lat0+((i)*-0.2340702210663199)
for j in range (0,1025):
   lon[j]=lon0+((j)*0.3515625)

#reduce lat/lons to sub-area
cutLat=lat[range(128,256)]
cutLon=lon[range(1,150)]
points=129*1025
newfilepath='/home/cg009107/Dissertation/Data/Year/EndOf/DJF.txt' #saves output here
f1=open(newfilepath, 'a+')
f1.write(u'\uFEFF'.encode('UTF-8'))
path='/home/cg009107/Dissertation/Data/Year/DJFdat/'  #imports from here
listing=os.listdir(path)
list=[]  #creates empty list
steps=8  #number of forecasts produced each day
percentage=0
arr=np.zeros([steps*len(listing),(129)*(151)+4])
b=0
for infile in listing:  #for each file in the folder
   string=path+infile
   print string, percentage
   t=time.strptime(infile[:8],'%Y%m%d')
   yr=t.tm_year
   mon=t.tm_mon
   day=t.tm_mday
   percentage +=1
   for i in range(0,steps):
      try:
         dat, forecast=sorting.ImportDatFile(string,i,lat, lon)
```

```
            mod=sorting.cutToSize(dat)
            xlen=len(mod[:,0])
            ylen=len(mod[0,:])
            print 'forecast time', forecast
            m=0
            oneD=np.zeros((xlen+1)*(ylen+1)+5)
            infile=infile.replace('.dat','')

            #plotting.plotGrids(mod, mod, cutLat, cutLon)  #plots the model grid so can check
inputted correctly
            arr[b,0]=yr
            arr[b,1]=mon
            arr[b,2]=day
            arr[b,3]=forecast  #checks the date and lead time of forecast
            count=4
          for j in range(0,xlen):
              for k in range(0,ylen):  #puts data into array
                x=mod[j,k]
                arr[b,count]=x
                m=m+1
                count +=1

            b=b+1
          except IndexError:  #if problems then pass over
            pass
        print arr
          #plotting.plotGrids(mod, check, cutLat, cutLon)
      np.savez(newfilepath, arr)  #saves data in arrays

      f1.close()
```

## *Code 2- multi*

```
"""
Created on Wed Jul 02 12:01:48 2014
@author: Bethan Jones
"""

import numpy as np
import fsstools
import random
import matplotlib.pyplot as plt
import plotting2 as plotting
import sorting
import matplotlib.mlab as mlab
import play
import time
import datetime
import utils

xmod=128
ymod=150

fssarray=[]
#---------------------------------------------------------------------
#Define latitude and longitude
lat0=90.2340702210663199
lon0=-0.3515625
lat=np.zeros(770)
lon=np.zeros(1025)
for i in range(0,770):
   lat[i]= lat0+((i)*-0.2340702210663199)
for j in range (0,1025):
   lon[j]=lon0+((j)*0.3515625)

GLat=lat[range(128,256)]
GLon=lon[range(0,150)]

#---------------------------------------------------------------------
#Import Data
npzfile=np.load('/home/cg009107/Dissertation/Data/Year/EndOf/SON.txt.npz')
npzfile.files
array=npzfile['arr_0']
days=len(array)
dateArr=np.zeros([days,19484])
dateArr[:,1:]=array

#Create new column with date string for future lookup
for i in range (days):
   year=str(int(array[i,0]))
   if array[i,1]<10:
      month='0'+str(int(array[i,1]))
   else:
```

```
        month=str(int(array[i,1]))
    if array[i,2]<10:
        day='0'+str(int(array[i,2]))
    else:
        day=str((int(array[i,2])))
    #print year, month, day
    dateArr[i,0]=time.strftime(year+month+day)
    #print dateArr[i,0]
norain=[]

zero=[row for row in dateArr if row[4]==0]    #creates array of observational data
one=[row for row in dateArr if row[4]==24]    #creates array of +24h forecasts
two=[row for row in dateArr if row[4]==48]
three=[row for row in dateArr if row[4]==72]
four=[row for row in dateArr if row[4]==96]
five=[row for row in dateArr if row[4]==120]
six=[row for row in dateArr if row[4]==144]
seven=[row for row in dateArr if row[4]==168]

a=int(len(zero))
thresholds=np.zeros([len(zero),8])
threshcount=0
#-----------------------------------------------------------------------------------------------------------------------
timesteps=[zero, one, two, three, four, five, six, seven]
k=0
for cloud in range(1,2):  #loops once, could increase to look at more than one timestep at once
    for q in range(0, a):   #0 to 'a' for full range of days
        step=2
        row=zero[q]
        today=row[0]
        theDate=datetime.date(int(row[1]),int(row[2]),int(row[3]))
        obsv=row[range(5,19484)]
        modDate=theDate-datetime.timedelta(days=step)
        print theDate , modDate, 'the date, mod date'
        yr=str(modDate.year)
        mon=(modDate.month)
        day=(modDate.day)
        print obsv
        if mon<10:
            month='0'+str(mon)
        else:
            month=str(mon)
        if day<10:
            day='0'+str(day)
        else:
            day=str(day)
        strModDate=yr+month+day   #sets date of forecast
        modRow=0
        for k in range(0,len(timesteps[step])):  #searches for forecast day
            kyear=str(int(timesteps[step][k][1]))
            kmon=timesteps[step][k][2]
```

```
            kday=timesteps[step][k][3]
            if (kyear==yr):   #is year the same?
               print 'year good'
               #print kmon, mon
               if int(kmon)==mon:    # is month the same?
                  print 'mon good'
                  if int(kday)==int(day): #is it the right day?
                     modRow=k    #sets the index number to find that row
                     print 'modrow set at k=', k
                     break
                  else:
                     print 'not day', kday, 'need day', day

         model=timesteps[step][modRow][range(5,19484)]  #locates data in array
         print 'model +', timesteps[step][modRow][4], 'obs time', row[4]
         print strModDate, timesteps[step][modRow][0], today, row[0]
         if strModDate != str(int(timesteps[step][modRow][0])):  #checks dates compatible
            print 'dates not compatible'
            pass
         else:

            mod=sorting.dataIntoGrid(model, GLat, GLon)  #puts data into grid
            obs=sorting.dataIntoGrid(obsv, GLat, GLon)
            #plotting.plotGrids(mod, obs, GLat, GLon)  #plot data
            countM, countO= play.countRainyDays(mod,obs)   #check there is enough precip.
            threshMod, threshObs,thresholds, percent, threshcount, norain=play.setThresholds(mod,
obs, xmod, ymod, thresholds, threshcount, today, norain)

            print threshMod, threshObs
            FSS=play.compFSS(mod , obs, countM, countO, threshMod, threshObs, percent, today)
         #print FSS
            k+=1
            fssarray.append(FSS)
   print thresholds

   #save FSS arrays
   np.savez('/home/cg009107/Dissertation/Data/Year/EndOf/SON48thresholds',thresholds)
   np.savez('/home/cg009107/Dissertation/Data/Year/EndOf/SON48FSS',fssarray)
```

## *Code 3- play*

```
"""
Created on Mon May 12 12:16:39 2014
@author: Bethan Jones
"""

import numpy as np
import fsstools
import random
import matplotlib.pyplot as plt
import plotting2 as plotting
import sorting
import matplotlib.mlab as mlab


plt.close('all')   #close all open figures
#------------------------------------------------------------------------------
#create lat lon grid
lat0=90.2340702210663199
lon0=-0.3515625
lat=np.zeros(770)
lon=np.zeros(1025)
for i in range(0,770):
   lat[i]= lat0+((i)*-0.2340702210663199)
for j in range (0,1025):
   lon[j]=lon0+((j)*0.3515625)

GLat=lat[range(128,256)]
GLon=lon[range(0,150)]

#-----------------------------------------------------------------------------
xlen=128
ylen=150
#------------------------------------------------------------------------------
#Set values

def countRainyDays(mod, obs):
   """Counts how many pixels in grid show rain."""
   xmod=len(mod[:,0])    #grid size
   ymod=len(mod[0,:])
   countM=0
   countO=0
   for a in range(0,xmod):
      for b in range(0,ymod):  #loops over whole grid
         if mod[a,b]>=0.00000000001:
            countM=countM+1
         if obs[a,b]>=0.00000000001:
            countO=countO+1
   print countM, 'rainy points in Model'
   print countO, 'rainy points in Obs'
   return countM, countO
```

```
#-------------------------------------------------------------------------------
#Set thresholds to check data against

def setThresholds(mod,obs, xmod, ymod, thresholds, threshcount, date, norain):
    """docstring"""
    maxi=obs.max()  #finds maximum value
    percent=[0.99, 0.95, 0.90]   #set percentages as thresholds
    threshObs=np.zeros(len(percent))   #create empty arrays to enter thresholds in.
    threshMod=np.zeros(len(percent))
#finds thresholds as percentages of grid
    for t in range(0,len(percent)):
        threshMod[t]= np.sort(np.ravel(mod))[xmod*ymod*percent[t]]  #sorts min to max and finds
threshold percentile
        threshObs[t]= np.sort(np.ravel(obs))[xmod*ymod*percent[t]]
        thresholds[threshcount,2+t]=np.sort(np.ravel(mod))[xmod*ymod*percent[t]]
        thresholds[threshcount,5+t]=np.sort(np.ravel(obs))[xmod*ymod*percent[t]]
        thresholds[threshcount,1]=percent[t]
        thresholds[threshcount,0]=date
        if threshMod[t]==0 or threshObs[t]==0:
            norain.append(date)
    print threshMod, threshObs
    threshcount +=1
    return threshMod, threshObs, thresholds, percent, threshcount, norain

#-------------------------------------------------------------------------------
#calculate FSS using grid at step size below.
#Step should be multiple of 2

def compFSS(mod, obs, countM, countO, threshMod, threshObs, percent, today):
    """ calculates FSS of two grids for thresholds provided"""
    step1=2
    rows=int(150/step1) #+ int(50/step2) + int(100/step3)-6 # + (int(xmod-200)/step4)-2

    FSS2=np.zeros([rows+1,len(percent)+1])
    FSS2[0,0]=today
    for i in range(len(percent)):    #sets header row with percentages
        FSS2[0,i+1]=percent[i]
    i=1
    print threshMod, threshObs
    for val in range(0,len(percent)):    #for all thresholds
        if (float(countM)/(len(GLat)*len(GLon)))>=(percent[val]):  #if no precip do nothing
            print ('Not enough rainy points for', percent[val], 'percent threshold in Model data on',
today)
            print percent[val]

        elif float(countO)/(len(GLat)*len(GLon))>=(percent[val]):
            print ('Not enough rainy points for', percent[val], 'percent threshold in Observational data
on', today )

        else: #if there is enough precip then
```

```
        #create binary grids
        Bo, Bm= fsstools.convertToBinary(mod,obs,threshMod[val], threshObs[val])
        #plotting.plotGrids(Bm,Bo, GLat,Glon)  #plot binary grids
        SummedO, SummedM=fsstools.sumBinary(Bo,Bm) #create summed grids
        #plotting.plotGrids(SummedM, SummedO, GLat, GLon)
        FSS2[1,0]=1
        i=len(Bo[:,0])   #number of rows
        j=len(Bo[0,:])   #number of columns
        fracMod=np.zeros([i,j])
        fracObs=np.zeros([i,j])

        for a in range (0,i):
          for b in range (0, j):     # creates fractions grid for neighbourhood 1 (same as binary grid)
            fracMod[a,b]=Bm[a,b]
            fracObs[a,b]=Bo[a,b]
        FSS2[1,val+1] =fsstools.fractionSkillScore(fracMod,fracObs) #calc. FSS
        print '1'
        num=2
        i=i+1

        for scale in range(3,150,step1):   #sets scales to calculate increase by 2 from 1 to max grid
          FSS2[num,0]=scale #sets first column as scale
          frac2Mod,frac2Obs=fsstools.genFracMeth2(SummedO, SummedM,scale)
          #plotting.plotFractions(frac2Mod, frac2Obs, percent[val], i, scale )
          FSS2[num,val+1] =fsstools.fractionSkillScore(frac2Mod,frac2Obs)
          print num, val+1, scale, FSS2[num,val+1]
          num=num+1
          i=i+1
          print scale   #so can tell how far it is along and that it is doing stuff!
    return FSS2    #returns table of FSS values by scale and threshold
```

## *Code 4- fsstools*

```
"""
Spyder Editor

@Author: Bethan Jones
"""
import numpy as np
import random
from sklearn import metrics

#-----------------------------------------------------------------------------
def convertToBinary(model, observed, threshMod, threshObs):
    """takes model and observational data (in grid squares) and a threshold value
    for precipitation and returns a grid with 1 for precipitation over that value
    and 0 for precipitation below that value. Assumes  equal sized grids"""
    i=len(model[:,0])   #number of rows
    j=len(model[0,:])   #number of columns
    Bo=np.zeros([i,j])    #create binary grid for observations
    Bm=np.zeros([i,j])    #create binary grid for forecast model
    for a in range (0,i-1):
        for b in range(0,j-1):    #for each individual grid square
            if model[a,b]>threshMod:   #if value is greater than threshold
                Bm[a,b]=1   #set corresponding binary grid square to 1

            if observed[a,b]>threshObs:  #as above but for observed values
                Bo[a,b]=1
    return Bo, Bm


#-----------------------------------------------------------------------------
def sumBinary(Bo, Bm):
    """Takes binary grids and sums from bottom left corner to top right corner"""
    i=len(Bo[:,0])
    j=len(Bo[0,:])
    SummedM=np.zeros([i,j])  #creates array of zeros
    SummedO=np.zeros([i,j])
    SummedM[0,0]=Bm[0,0]  #set top left value
    SummedO[0,0]=Bo[0,0]
    for b in range (1,j):  #sets first row
        SummedM[0,b]=SummedM[0,b-1]+Bm[0,b]
        SummedO[0,b]=SummedO[0,b-1]+Bo[0,b]
    for a in range(1,i):  #sets first column
        SummedM[a,0]=SummedM[a-1,0]+Bm[a,0]
        SummedO[a,0]=SummedO[a-1,0]+Bo[a,0]
        for b in range(1,j): #sum the rest of array

            SummedM[a,b]=SummedM[a-1,b]+SummedM[a,b-1]-SummedM[a-1,b-1]+Bm[a,b]
            SummedO[a,b]=SummedO[a-1,b]+SummedO[a,b-1]-SummedO[a-1,b-1]+Bo[a,b]
    return SummedO, SummedM
```

```
#------------------------------------------------------------------------------
def generateFractions(Bo,Bm,N):    #The inefficient way
    """Takes binary grids and spatial scale to return fraction of squares with
    rain around the square. Returns two arrays."""
    n=(N-1)/2
    i=len(Bo[:,0])   #number of rows
    j=len(Bo[0,:])   #number of columns
    Fm= np.zeros([i,j])
    Fo= np.zeros([i,j])
    denMod=np.zeros([i,j])   #creates masking grid of the number of squares available to divide by
    denObs=np.zeros([i,j])
    denMod[:]=N*N  #sets initial size of masking array, maximum number of grid squares.
    denObs[:]=N*N
    if N<=10:  #If large array, do less calculations
        loop=1
    elif N>10 and N<=50:
        loop=4
    elif N>50:
        loop=int(N/5)+1

    for a in range (0,i,loop):
        for b in range(0,j,loop):   #for every pixel
            for c in range (-n,n+1):
                for d in range (-n,n+1):
                    if (((a+c <0)or(b+d<0))or ((a+c or b+d)>=i)):
                        denMod[a,b]=denMod[a,b]-1     #if grid square doesn't exist- discount it from calculations
                        denObs[a,b]=denObs[a,b]-1
                    else:
                        try:
                            Fm[a,b]=Fm[a,b]+Bm[a+c,b+d]   #sums the values of grid squares around each grid square for model
                            Fo[a,b]=Fo[a,b]+Bo[a+c,b+d]   #as above for observations
                        except IndexError:
                            denMod[a,b]=denMod[a,b]-1    #if above doesn't work discount it.
                            denObs[a,b]=denObs[a,b]-1
                            pass
    for a in range (i):
        for b in range(j):    #over whole array
            Fm[a,b]=Fm[a,b]/N*N   # divide sum by the number of grid squares counted to create fraction
            Fo[a,b]=Fo[a,b]/N*N
            print Fm[a,b], Fo[a,b]
    return Fm, Fo
```

```
#---------------------------------------------------------------------------
def genFracMeth2(SummedO, SummedM, N):  #The more efficient way
   """ Calculates fractions grids for neighbourhood N using Summed grids"""
   n=(N-1)/2
   i=len(SummedO[:,0])   #number of rows
   j=len(SummedO[0,:])   #number of columns
   Fm2= np.zeros([i,j])
   Fo2= np.zeros([i,j])

   for a in range (0,i,1):
      for b in range(0,j,1):  #for every pixel
         if (a+n)>=(i-1):   #check if off grid set to 0
            aplusn=i-2
         else:
            aplusn=a+n
         if (a-n)<=0:
            aminusn=0
            #print 'a-n <0', n, a, b
         else:
            aminusn=a-n-1
         if (b+n)>=(j-1):
            bplusn=j-2
         else:
            bplusn=b+n
         if (b-n)<=0:
            bminusn=0
         else:
            bminusn= (b-n)-1
         Fm2[a,b]=(SummedM[aplusn,bplusn]-SummedM[aminusn,bplusn]-
SummedM[aplusn,bminusn]+SummedM[aminusn,bminusn])/(N*N)
         Fo2[a,b]=(SummedO[aplusn,bplusn]-SummedO[aminusn,bplusn]-
SummedO[aplusn,bminusn]+SummedO[aminusn,bminusn])/(N*N)

   return Fm2, Fo2

#---------------------------------------------------------------------------
def fractionSkillScore(Fm,Fo):
   """Calculates fraction skill score when given fractions grid. Fed two grids
   (of equal size) the first being the fractional grid of the model the second
   being the observed data fractional grid. Returns single float number"""
   i=len(Fo[:,0])   #number of rows
   j=len(Fo[0,:])   #number of columns

   arrMSE=np.zeros([i,j])  #to create grid for MSE
   sqObs=np.zeros([i,j])   #blank arrays to fill for reference MSE
   sqMod=np.zeros([i,j])   #blank arrays to fill for reference MSE

   for a in range (0,i):
      for b in range(0,j):    #over whole array
         arrMSE[a,b]=(Fo[a,b]-Fm[a,b])*(Fo[a,b]-Fm[a,b])  #fills in MSE
```

```
        sqObs[a,b]=Fo[a,b]*Fo[a,b]   #For reference MSE
        sqMod[a,b]=Fm[a,b]*Fm[a,b]   #For reference MSE
MSE=sum(sum(arrMSE))/(i*j)  #Calculate MSE
refMSE=(sum(sum(sqObs))+sum(sum(sqMod)))/(i*j)  #Calculate reference MSE

FSS=1-(MSE/refMSE)   #Calculate Fractional Skill Score

return FSS
```

## *Code 5- sorting*

```
"""
Created on Thu May 22 12:12:42 2014
@author: Bethan Jones
"""

import numpy as np

#-----------------------------------------------------------------------------
def ImportDatFile(location, row, lat, lon):
    """imports datafile extracts grid size and creates array"""
    data=np.genfromtxt(location)  #import data

    l=len(data[row,:])
    xlen=np.int(data[row,0])
    ylen=np.int(data[row,1])
    forecast=data[row,2]   #forecast range eg T+24 (not in right order)

    #extract precipitation rates
    val=data[row,range(3,l)]
    print val
    name=np.zeros((ylen+1,xlen+1))
    name[:,0]=lat[:]
    name[0,:]=lon[:]
    for  i in range(0, int(ylen)):
        for j in range(0, int(xlen) ):
            name[(i+1),j+1]=val[(i*xlen)+j]
    return name, forecast


#-----------------------------------------------------------------------------
def alignGrids(data1,data2):
    """Ensures grids are the same size for comparison. Assumes starts at same point"""

    x1=len(data1[0,:])
    y1=len(data1[:,0])

    x2=len(data2[0,:])
    y2=len(data2[:,0])
    if (x1==x2 and y1==y2):
        return data1, data2
    elif (x1==x2 and y1!=y2):
        if y1>y2:
            new=data1[:,range(0,y2)]
            return new, data2
        else:
            new=data2[:,range(0,y1)]
            return data1, new
    elif (x1!=x2 and y1==y2):
        if x1>x2:
            new=data1[range(0,y2),:]
```

```
            return new, data2
        else:
            new=data2[range(0,y1),:]
            return data1, new
    else:
        if x1>x2:
            new=data1[range(0,x2),:]
            if y1>y2:
                new2=new[:,range(0,y2)]
                return new2, data2
            else:
                new2=data2[:,range(0,y1)]
                return new, new2#return new, data2
        else:
            new=data2[range(0,x1),:]
            #return data1, new
            if y1>y2:
                new2=data1[:,range(0,y2)]
                return new2, new
            else:
                new2=new[:,range(0,y1)]
                return data1, new2

def cutToSize(model):
    """ Cuts size of grid if required """
    a=int(len(model)/2)
    b=50
    c=52
    mod1=model[:,range(1,151)]
    mod=mod1[range(128,256),:]
    return mod

#-----------------------------------------------------------------------------------------------
def dataIntoGrid(listData, lat, lon):
    """imports datafile extracts grid size and creates array"""
    grid=np.zeros([len(lat), len(lon)])
    count=0
    for i in range(0,len(lat)):
        for j in range(0,len(lon)):
            grid[i,j]=listData[count]
            count +=1

    return grid
```

## *Code 6- test*

```
"""
Created on Tue May 27 11:30:20 2014
@author: Bethan Jones
"""

import numpy as np
import utils
import plotting2 as plotting
import matplotlib.pyplot as plt
import math

#import data arrays
npzfile=np.load('N:/Dissertation/Data/Year/EndOf/SON24thresholds.npz')
FSSfiles=np.load('N:/Dissertation/Data/Year/EndOf/SON24FSS.npz')

#seperate arrays
thresholds=npzfile['arr_0']*3600
FSSall=FSSfiles['arr_0']

#Set empty arrays for scale becomes useful
useful99=np.zeros(len(FSSall))
useful90=np.zeros(len(FSSall))

#Create empty arrays for FSS at each scale
scale1=np.zeros(len(FSSall))
scale3=np.zeros(len(FSSall))
scale5=np.zeros(len(FSSall))
scale7=np.zeros(len(FSSall))

for i in range(0,len(FSSall)):  #for all days
   FSS=FSSall[i]
   date=int(FSS[0,0])
   scale1[i]=FSS[1,3]
   scale3[i]=FSS[2,3]
   scale5[i]=FSS[3,3]
   scale7[i]=FSS[4,3]
   for j in range(1,len(FSS[:,0])):
      if FSS[j,1]>=0.5:
         useful99[i]=FSS[j,0]
         break
      else:
         useful99[i]=FSS[j,0]
   for j in range(1,len(FSS[:,0])):
         if FSS[j,3]>=0.5:
            useful90[i]=FSS[j,0]
            break
         else:
            useful90[i]=FSS[j,0]

   plotting.plotFSS(FSS,150,2)
```

```
#plot histogram of thresholds on different days
plt.figure()
plt.hist(thresholds[1:,2],bins=14)
plt.xlabel('Rain rate in mm/hr')
plt.ylabel('number of days')
plt.title('99% threshold values for period June, July, August')
plt.show()

plt.figure()
plt.hist(thresholds[1:,4],bins=[0,0.05,0.1,0.15,0.2,0.25,0.3,0.35,0.4,0.45,0.5,0.55,0.6,0.65])
plt.xlabel('Rain rate in mm/hr')
plt.ylabel('number of days')
plt.title('90% threshold values for March')
plt.show()

# print important values to screen
print 'means', np.mean(useful99), np.mean(useful95), np.mean(useful90)
print 'max', np.max(useful99), np.max(useful90)
print 'min', np.min(useful99), np.min(useful90)
print '25th, 75th percentile', np.percentile(useful99,[25,75]), np.percentile(useful90,[25,75])
```

## *Code 7- plotting*

```
    ""
Created on Mon May 19 11:22:28 2014
@author: Bethan Jones
"""

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as colours
from mpl_toolkits.basemap import Basemap

#-----------------------------------------------------------------------------
def plotGrids(Mod, Obs, Glat, Glon):
    """plots grids in colour scale to allow for easy visulisation"""

    plt.figure()  #makes sure not plotting over something


    cmap2 =plt.cm.jet     #('Blue',256)    #colours.Colormap('PuBu')  #set colour map
    fig, axes = plt.subplots(nrows=1, ncols=2)
    llclat=30  #Glat[len(Glat)-1]
    llclon=0   #Glon[0]
    urclat=60  #Glat[0]
    urclon=150*0.3515625   #Glon[len(Glon)-1]   #set area

    plt.subplot(2,1,1)   #2,1,1 for one above other, 1,2,1 for side by side

m=Basemap(projection='cyl',llcrnrlat=llclat,urcrnrlat=urclat,llcrnrlon=llclon,urcrnrlon=urclon,resoluti
on='c')
    m.drawcoastlines()
    ny= Mod.shape[0]; nx=Mod.shape[1]
    lons, lats= m.makegrid(nx,ny)
    x,y= m(lons,lats)
    maxi=np.max(Mod)
    maxio=np.max(Obs)
    print maxi, maxio

    a=np.zeros(50)
    for i in range(1,len(a)+1):
        a[i-1]=0.000+(i*0.00005)   #sets values for colour map
    img2 = m.contourf(x,y,Mod,a,cmap=cmap2)
    plt.gca().invert_yaxis()
    plt.title('Model Data')

    plt.subplot(2,1,2)   #2,1,2 for one above other, 1,2,2 for side by side

m=Basemap(projection='cyl',llcrnrlat=llclat,urcrnrlat=urclat,llcrnrlon=llclon,urcrnrlon=urclon,resoluti
on='c')
    m.drawcoastlines()
    ny= Obs.shape[0]; nx=Obs.shape[1]
    lons, lats= m.makegrid(nx,ny)
```

```
    x,y= m(lons,lats)
    img3 = m.contourf(x,y,Obs,a,cmap=cmap2)
    plt.gca().invert_yaxis()
    plt.title('Observational data')

    cbar_ax = fig.add_axes([0.90, 0.1, 0.02, 0.8])

    plt.show()

#--------------------------------------------------------------------------------
def plotBinary(Bm, Bo,thresh,count):
    """plots black/white binary grid"""
    plt.figure(count)

    cmap = colours.ListedColormap(['white','black'])
    bounds=[0,0.5,0.5,1]
    norm = colours.BoundaryNorm(bounds, cmap.N)
    plt.subplot(1,2,1)
    # tell imshow about color map so that only set colors are used
    img = plt.imshow(Bm,interpolation='nearest', cmap = cmap,norm=norm)
    plt.title('Model and Obserbvational output in binary for a threshold of %s percent of maximum
amount' % thresh)
    plt.subplot(1,2,2)
    # tell imshow about color map so that only set colors are used
    img2 = plt.imshow(Bo,interpolation='nearest', cmap = cmap,norm=norm)
    # make a color bar
    plt.colorbar(img,cmap=cmap,norm=norm,boundaries=bounds,ticks=[0,1])

    plt.show()

#-------------------------------------------------------------------------------------
def plotFSS(FSS,grid, legend):
    """PLots FSS against grid size for different thresholds."""

    plt.plot(FSS[1:,0],FSS[1:,legend+1],label=FSS[0,0])
    percent=[99,95,90]
    plt.xlabel('scale (in number of grid squares)')
    plt.ylabel('fss fractional skill score')
    title=str('Fractional skill score depending on scale at threshold'+ str(percent[legend])+ '%')
    plt.title(title)

    plt.xlim(1,grid)
    plt.ylim(0,1)
    plt.axhline(y=0.5, ls='dashed')

#------------------------------------------------------------------------------
def plotFractions(Fm, Fo,thresh,count,scale):
    """plots black/white binary grid"""
    plt.figure(count)
    cmap2 = colours.LinearSegmentedColormap.from_list('my_colormap', ['white','blue'],256)
    plt.subplot(1,2,1)
```

```
img2 = plt.imshow(Fm,interpolation='nearest', cmap = cmap2, origin='lower')

# make a color bar
plt.colorbar(img2,cmap=cmap2)
plt.title('Model fractions for a threshold of %s percent of maximum amount in a
neighbourhood of %s grid squares' % (thresh, scale))

plt.subplot(1,2,2)

cmap3 = colours.LinearSegmentedColormap.from_list('my_colormap', ['white','blue'],256)

img3 = plt.imshow(Fo,interpolation='nearest', cmap = cmap3, origin='lower')
plt.show()
```