Solar activity analysis using publicly available SDO/AIA data

Bart Buelens

Abstract

Data from the Atmospheric Imaging Assembly (AIA) onboard the Solar Dynamics Observatory (SDO) are used to analyze solar activity over a two year period. Image processing techniques are used to derive time series of characteristics of the distribution of intensities of solar images taken at 211 angstrom. Days with exceptionally high activity levels are identified in the time series. A comparison between the obtained series and the officially published sunspot numbers shows a high correlation.

This analysis is a case study demonstrating possibilities for amateur astronomers and astrostatisticians. The AIA/SDO data server is just one of many astronomical data sources publicly available. This offers new opportunities to the non-professional community to access data collected by high-end, professional instruments. This working paper should inspire fellow hobbyists to experiment with such data, and to develop code to explore, visualize and analyze the data. The domain of astrostatistics, which has been rediscovered by the professional community in recent years, is easily accessible to hobby scientists.

22 March 2013

astrostatistics.org

Working paper 2013BB01

1 Introduction

This working paper illustrates by means of a simple example the possibilities for hobby astronomers to experiment with astronomical data collected by professional instruments. Data collected by the Atmospheric Imaging Assembly (AIA)[1] are used. The AIA is mounted onboard the Solar Dynamics Observatory (SDO)[2] and collects high resolution images of the sun at multiple wavelengths, at a sub-minute cadence. A wide variety of data products is made available for public use. All programming in the present example is accomplished with the open source statistical software package R (R Core Team, 2012) and some additional packages.

A good general starting point for information about astrostatistics and astroinformatics is the Astrostatistics and Astroinformatics Portal hosted at Pennsylvania State University (Feigelson and Hilbe, 2014). In its section *Resources*, this website lists many relevant books[3]. Of particular value is Feigelson and Babu (2012), an excellent reference text providing a comprehensive overview of possible statistical applications within astronomy, including introductory material both on statistics and on the R software package.

Additional material on the case study discussed in the present paper, and other examples, are available online at the astrostatistics.org website[4] (Buelens, 2014).

2 AIA data

Data files collected by the AIA can be downloaded from the Joint Science Operations Center[5]. We use the synoptic data, which are 1k x 1k images in FITS format, available at http://jsoc.stanford.edu/data/aia/synoptic.

To mass download files we write a script in R. In this example we fetch one image at one wavelength for each available date. The chosen wavelength is 211 Angstrom. The script below resulted in 760 files being downloaded (on July 3rd, 2012 - note that new files are added every day). Package RCurl (Lang, 2011) is used to enable downloading of files through HTTP.

```
library(RCurl)
years = c("2010","2011","2012")
months = as.character(1:12)
```

[1] http://aia.lmsal.com
[2] http://sdo.gsfc.nasa.gov
[3] https://asaip.psu.edu/resources/recent-books
[4] http://www.astrostatistics.org
[5] http://jsoc.stanford.edu

```
months[nchar(months)==1] = paste("0",months[nchar(months)==1],sep="")
days = as.character(1:31)
days[nchar(days)==1] = paste("0",days[nchar(days)==1],sep="")

urlBase = "http://jsoc.stanford.edu/data/aia/synoptic"
locPath = "D:/data/SDOAIA"

for (yr in years) {
    for (mo in months) {
        for (da in days) {
            thisFile = paste("AIA",yr,mo,da,"_0000_0211.fits",sep="")
            thisPath = paste(yr,mo,da,"H0000",sep="/")
            print(thisFile)
            flush.console()
            myUrl = paste(urlBase, thisPath, thisFile, sep = "/")
            myDest = paste(locPath, thisFile, sep = "/")
            myBits = getBinaryURL(myUrl)
            if (length(myBits) > 500) { # if files don't exist we
                                        # get a short reply from the
                                        # server: skip these
                myFile = file(myDest, "wb")
                writeBin(myBits,myFile)
                close(myFile)
            }
        }
    }
}
```

The downloaded files are in compressed FITS format. The Fitsio package (Harris, 2012) for R cannot handle these. We use the funpack utility[6] to decompress the files (Pence, 1999). Again, this can be done in batch using following R script. Make sure to set the directory with the FITS files as the working directory.

```
sdoFiles = dir()
sdoFilesFz = paste(sdoFiles,".fz",sep="") # rename the original files
                                          # to have file extension .fz,
                                          # as required by funpack
file.rename(sdoFiles,sdoFilesFz)
for (thisFile in sdoFilesFz) {
```

[6] http://heasarc.nasa.gov/fitsio/fpack

```
    system(paste("d://fpack//funpack",thisFile))
}
```

This will generate a lot of data, approx. 3 GB, make sure to have enough disk space available.

Information on the data use policy is given here: http://sdo.gsfc.nasa.gov/data/rules.php. In line with these guidelines we say the following about the data files used in this example: Courtesy of NASA/SDO and the AIA, EVE, and HMI science teams.

3 Preprocessing

We select one file - arbitrarily - to analyze a little further, prior to batch processing all files. We convert its header to a list for convenience.

```
library(FITSio)
sdo = readFITS("AIA20100513_0000_0211.fits")
Hdr2List = function(h) {
    myL = list()
    i = 1
    while (i < length(h)) {
        myL[[h[i]]] = h[i+1]
        i = i + 2
    }
    return(myL)
}
hdrlst = Hdr2List(sdo$hdr)
```

Information on the keywords in the header (hdrlst) can be found in Nightingale (2009). Next, we get the actual image into a matrix, and obtain information on the coordinate axes. Then we plot the image.

```
x = sdo$imDat
ax1 = axVec(1,sdo$axDat)
ax2 = axVec(2,sdo$axDat)
image(ax1,ax2,x)
```

This results in Figure 1. The distribution of the pixel values is such that there are only few at the higher range. As is common in solar imagery, log-transforming may improve matters, see Figure 2.

```
xlog = x
xlog[x < 1] = 1 # avoid taking log of zero or negatives
xlog = log(xlog)
image(ax1,ax2,col=heat.colors(500))
```

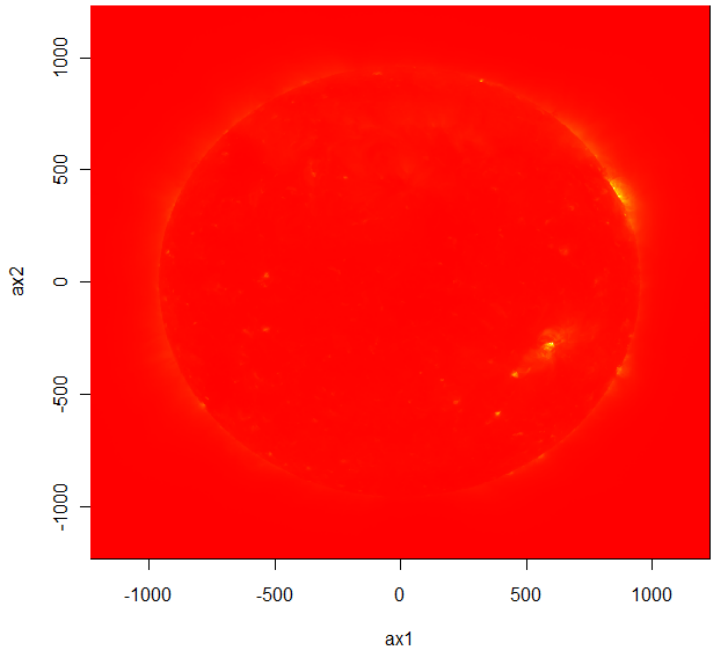

Figure 1: One of the downloaded images (this one was taken on 13 May 2010).

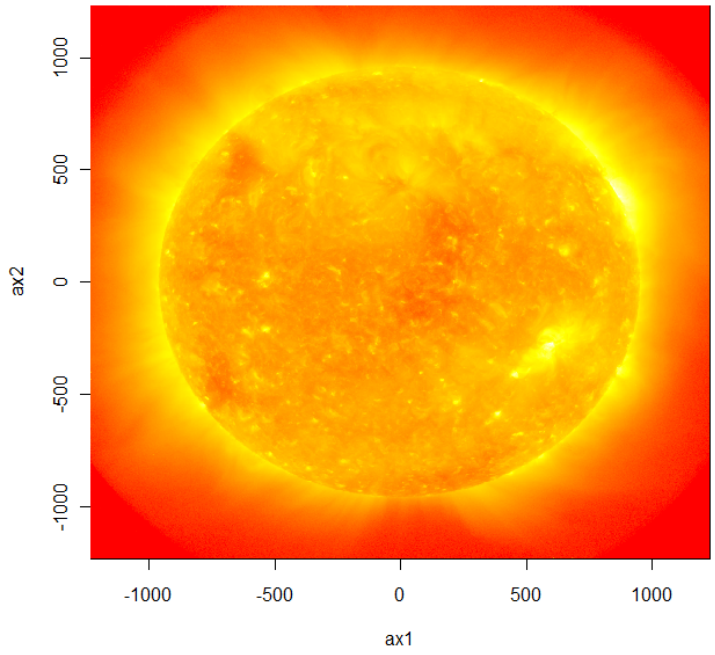

Figure 2: The same image as Figure 1, with intensities now color coded on the logarithmic scale.

Next we construct a mask to remove all pixels not on the solar disk. We retrieve the sun's radius in image coordinates from data in the header of the file.

```
sunR = as.numeric(hdrlst["IMSCL_MP"]) * as.numeric(hdrlst["R_SUN"])
                                                    # the sun's radius
M = outer(ax1^2,ax2^2,FUN="+") < sunR^2   # mask: TRUE if pixel in disk,
                                          # FALSE otherwise
xlogM = xlog * M
image(ax1,ax2,xlogM,col=heat.colors(500))
```

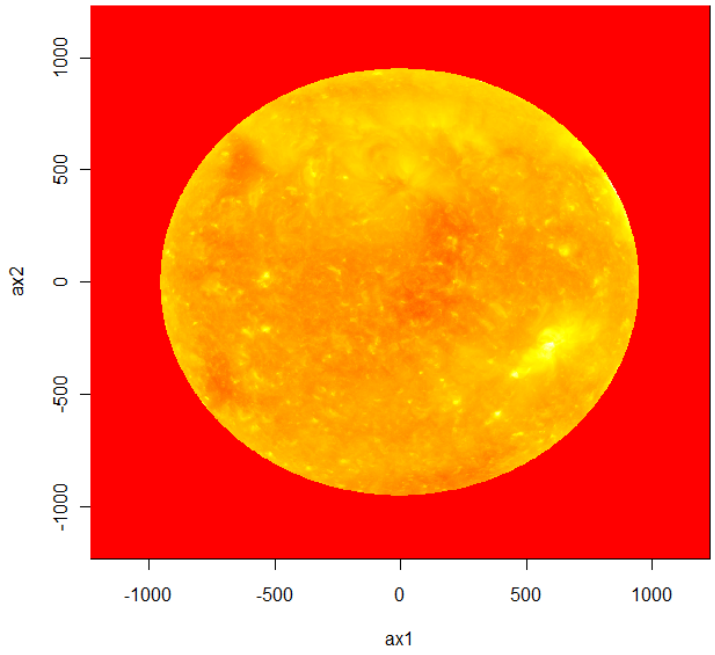

Figure 3: The same image as Figure 2, with a mask applied.

We do a simple analysis of the pixels representing the solar disk. We use the mask M to obtain the relevant data points. The functions summary() and quantile() give some distributional characteristics of the data.

```
xsun = x[M]
summary(xsun)
quantile(xsun, probs = seq(0, 1, 0.1))
```

The above can be done in batch, on all available data files. The results are accumulated in a data frame, which is written to disk. This data frame with results is analyzed in the next section (see link below). The following code implements the batch processing.

```
allFiles = dir("D:\\data\\SDOAIA\\") # replace with relevant directory!
A = data.frame(file = allFiles)
A$date = substr(A$file,4,11)

processFile = function(fileName) {
```

```
    sdo = readFITS(paste( "D:\\data\\SDOAIA\\" ,fileName,sep=""))
    hdrlst = Hdr2List(sdo$hdr)
    myR = as.numeric(hdrlst["IMSCL_MP"]) * as.numeric(hdrlst["R_SUN"])
    ax1 = axVec(1,sdo$axDat)
    ax2 = axVec(2,sdo$axDat)
    sunMask = outer(ax1^2,ax2^2,FUN="+") < myR^2
    xsun = sdo$imDat[sunMask]
    res = as.data.frame(t(quantile(xsun, probs = seq(0, 1, 0.1))))
    res$sum = sum(xsun)
    res$pixcount = length(xsun)
    res$mean = mean(xsun)
    return(res)
}

b = processFile(A$file[1])
A = cbind(A,b) # now all columns exist
resnames = names(b)

for (f in A$file) {
    A[A$file == f,resnames] = processFile(f)
    print(A[A$file == f,])
    flush.console()
}
names(A)[3:13] = paste("q",seq(0,100,10),sep="") # other names,
                                                 # for convenience later on
A$datechar = paste(substr(A$date,1,4),substr(A$date,5,6),
                   substr(A$date,7,8),sep="-")
A$date = as.Date(A$datechar,format="%Y-%m-%d") # make date of date class,
                                               # again for convenience later
save(A,file="AllResults.RData")
```

The result is a file containing the data frame A, ready for further analysis. Note that we encountered some corrupt files. The above routine failed on some files, and others resulted in odd statistics. Some manual investigation and comparisons with the input files learned that those files which had negative values in their 'xsun' pixels (see code above) indeed had corrupt images (noise, stripes, blank areas etc). Of the 760 files considered, 6 had issues. We removed them manually from the data frame. We have not investigated whether the files on the server are corrupt, or whether they got corrupted in the process of downloading. Code related to this issue is not shown.

4 Analysis results

4.1 Time series of solar activity

We load the results of the batch processing script. This is a data frame with for each image (or each day) some statistics. We can easily plot the variables in the data frame A as follows. For example, a plot of the mean value results in Figure 4.

```
load("AllResults.RData")
plot(A$date, A$mean, type="l")
```

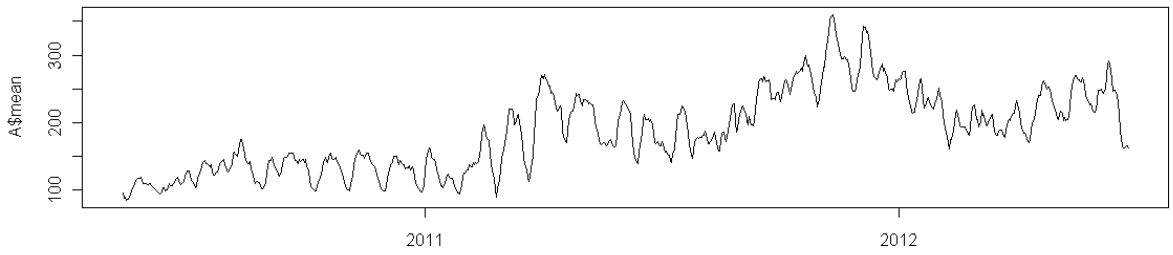

Figure 4: Time series of the mean intensity.

More fancy plotting is available through the ggplot2 package (Wickham, 2009). The code below constructs a plot which visualizes the distribution of pixel intensities over the two year period covered by our data set. The bottom and top quintile are omitted for clarity. This code first loads the required libraries, then constructs the plot. The result is Figure 5. The black line represents the median value. The darkest shade of blue ranges from the 40% to the 60% quantile, the lighter shade from 30% to 40% and 60% to 70%, and the lightest shade from 20% to 30% and 70% to 80%. Hence 60% of all values are in the shaded area. The tails of the distribution - the lower and upper quintiles - are not shown in this plot. The tick marks on the horizontal axis are put at the beginning of each month. For example '05-11' really is May 1st, 2011. We clearly see periodic changes, as well as prolonged periods of more intense activity.

```
library(ggplot2)
```

```
library(RColorBrewer)
myCol = brewer.pal(5,"Blues")
p = ggplot(A, aes(x=date))
p = p + geom_ribbon(aes(ymin=q20,ymax=q80),fill=myCol[2])
p = p + geom_ribbon(aes(ymin=q30,ymax=q70),fill=myCol[3])
p = p + geom_ribbon(aes(ymin=q40,ymax=q60),fill=myCol[4])
p = p + geom_line(aes(y=q50)) + xlab('date') + ylab('intensity')
p + scale_x_date(major = "months",format="%m-%y")
```

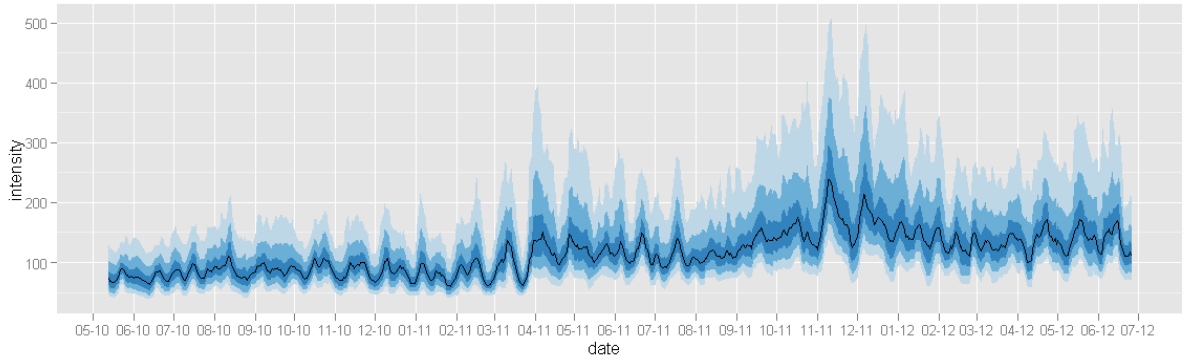

Figure 5: Time series of the median intensity (black line) and quantiles (blue ribbons).

4.2 Correlations

We have 11 quantiles for each day. The correlation between these is obtained as:

```
cor(A[,3:13])
```

The result is a 11-by-11 matrix (not shown). The q100 value, which is the maximum value, correlates the least with all others. We investigate this some further, by looking at how much greater the maximum is compared to the 90% quantile. The following code plots the distribution of this difference for the approx. two years worth of data that we have, see Figure 6.

```
hist(A$q100 - A$q90, breaks=40)
abline(v=11500,lwd=3,col="red")
```

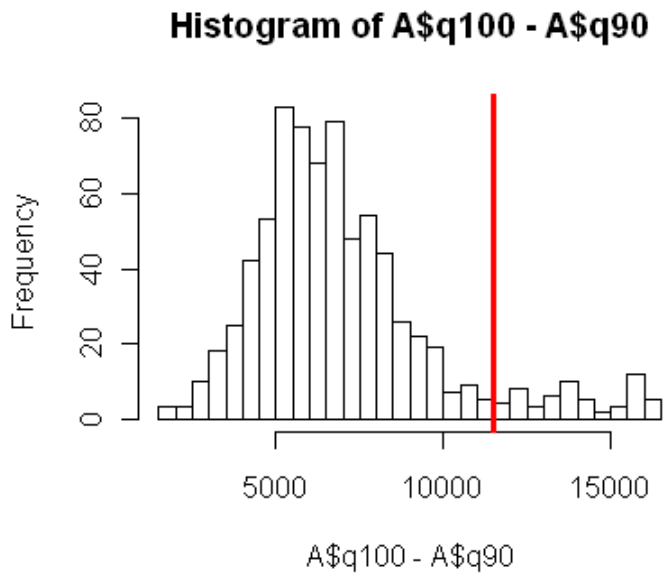

Figure 6: Histogram of the differences between the 100% and 90% quantiles.

The distribution looks like a normal distribution, with some additional unusually high numbers. Using methods like Gaussian mixture models we could try to identify some optimal decision boundary separating the normal distribution from the higher values, but here we pick a sensible value by eye, for simplicity: 11500. The red line in Figure 6 shows the location of this value. We write a little function that will generate a plot marking these unusual values in the time series.

```
myPlot = function(Asub) {
    Asub$index = 1:(dim(Asub)[1])
    unusualIndex = Asub[Asub$unusual,"index"]
    p = ggplot()
    p = p + geom_ribbon(data=Asub,
                mapping=aes(x=date,ymin=q90,ymax=q100),
                fill=myCol[2])
    for (i in unusualIndex) {
        p = p + geom_area(data=Asub[(i-1):(i+1),],
                mapping=aes(x = date, y = q100),
                fill="red")
    }
```

```
    p = p + geom_line(data=Asub,
                      mapping=aes(x=date,y=q100)) +
                      ylab('intensity')
    p = p + geom_line(data=Asub,mapping=aes(x=date,y=q90))
    p + scale_x_date(major = "months",format="%m-%y")
}
A$unusual = (A$q100 - A$q90)
myPlot(subset(A, date >= "2011-7-1" & date < "2011-12-31"))
```

This function allows us to easily restrict the date range covered in the plot. The example shows only a 6 month period, since that is nicer than all two years in one plot. The resulting plot is shown in Figure 7. The top black line is the maximum value, the bottom black line the 90% quantile, with the area in between shaded in light blue. In red the days are indicated with unusually high activity, in the sense that the difference between the maximum and the 90% quantile is unlikely to come from the normal distribution governing this quantity most of the time.

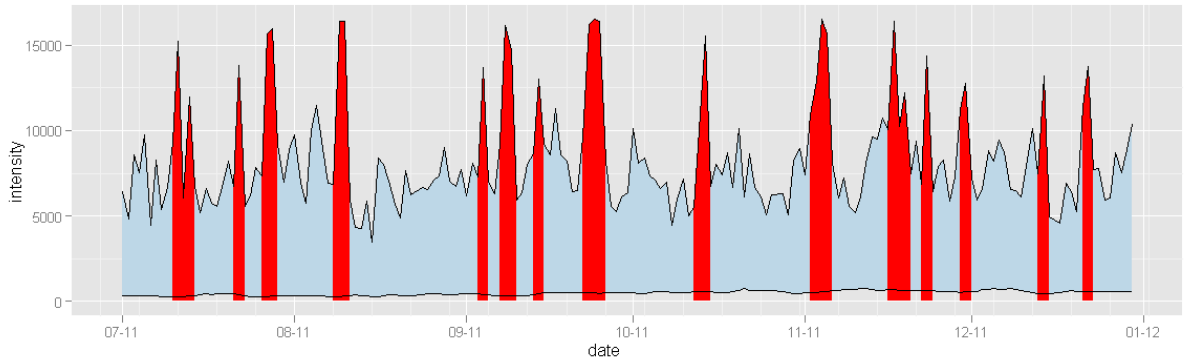

Figure 7: Intensities (90% to 100% quantile band) with unusually high values marked in red.

4.3 Comparing with daily sunspot numbers

We download the international daily sunspot number data for the year 2011. Information about sunspot indices can be found at the NOAA National Geophysical Data Center[7]. The international sunspot number is produced by the Solar Influences Data Analysis Center (SIDC). We downloaded data from the

[7]http://www.ngdc.noaa.gov/stp/solar/ssndata.html

NOAA FTP site[8]. Data credits: SIDC, RWC Belgium, World Data Center for the Sunspot Index, Royal Observatory of Belgium, 'year(s)-of-data'.

First we rearrange the data in a .csv text file for easy reading in R. We combine the sunspot number with our data frame A, thereby restricting our data to the year 2011. We put all this in data frame B:

```
S = read.csv2("2011daily.csv", stringsAsFactors = FALSE)
            # a simple csv file with 2 columns: date and number
m = match(A$datechar,S$date)
A$ssn = S[m,"number"]
B = subset(A,!is.na(ssn))
```

Consider some correlations between our data and the ssn (sunspot number).

```
cor(B$ssn, B$mean)
cor(B$ssn,B[,3:13])
```

The correlation with the mean is highest, 0.80. Second highest comes the 90% quantile with 0.77. We plot the mean and the ssn together, see Figure 8. The values on the horizontal axis now refer to the days in 2011. The black line are the sunspot numbers, the red is our daily mean.

```
plot(B$ssn,type="l",ylim=c(0,500))
lines(B$mean,col="red")
```

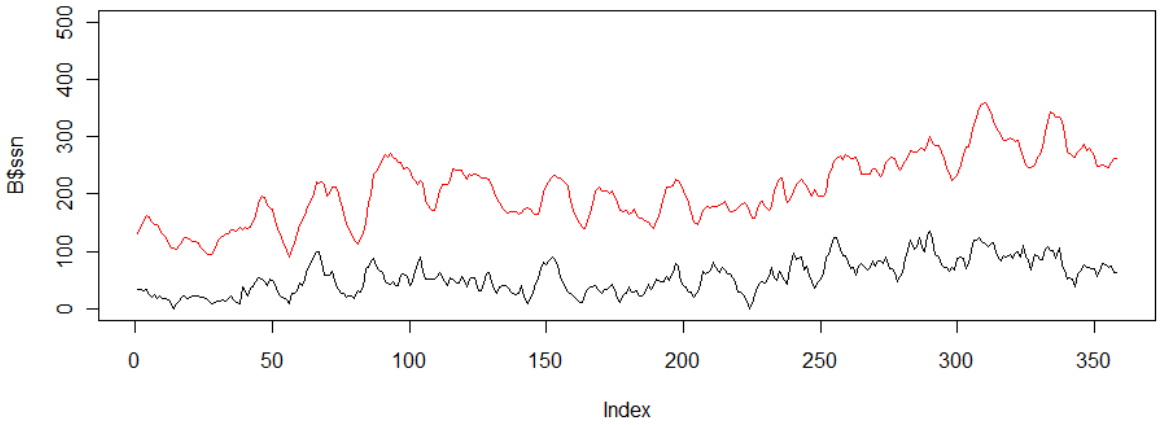

Figure 8: Mean intensity (red) and sunspot number (black).

[8]ftp://ftp.ngdc.noaa.gov/STP/SOLAR_DATA/SUNSPOT_NUMBERS/INTERNATIONAL/2011/2011

The combined correlation of the mean and q90 (the 90% quantile) with the ssn can be analyzed using linear modeling.

```
fit = lm(ssn ~ q90 + mean, B)
summary(fit) # gives some useful details
cor(B$ssn, fit$fitted.values)
plot(B$ssn,type="l")
lines(fit$fitted.values,col="red")
```

This fits a model with the ssn as dependent variable, and q90 and mean as predictors. The correlation between the predicted values and the ssn, 0.81, is only a little larger than between the mean alone and the ssn. In Figure 9, the black are again the sunspot numbers, and the red are our predictions under the model. Our predictions seem to be smoother than the original sunspot numbers, with fewer extremes. The predictions correspond fairly well to the sunspot numbers, although not all the time.

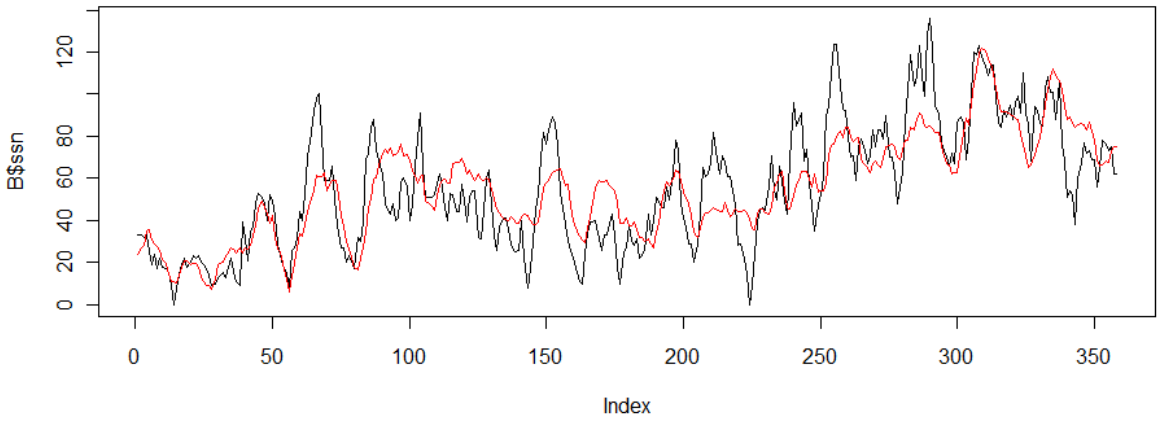

Figure 9: Predicted sunspot numbers (red) and true sunspot numbers (black).

5 Conclusions

The worked example in this paper should give interested amateur astronomers a taste of the possibilities available today: enormous amounts of astronomical data collected by professional instruments are publicly available through

the internet. The domain of astrostatistics, which has been rediscovered by the professional community in recent years, is easily accessible to hobby scientists. Powerful analytical software – such as R – is freely available under an open source license.

In this paper solar images collected by the AIA onboard the SDO are used to study activity levels of the sun. Data covering an almost two-year period are downloaded and processed. For each image several characteristics of the distribution of the intensities of the pixel values are derived. Days with exceptionally high activity levels are identified in the time series. A comparison between our obtained series and the officially published sunspot numbers shows a high correlation.

This example can be expanded upon, for example by using data collected at other wavelengths, deriving similar statistics, and doing a correlation analysis between wavelengths. More advanced time series analysis can be conducted, including forecasting of sunspot numbers.

References

Bart Buelens. Astrostatistics.org – Astronomy & Statistics, January 2014. URL `http://www.astrostatistics.org`.

Eric Feigelson and Joseph Hilbe. Astrostatistics and Astroinformatics Portal (ASAIP), January 2014. URL `https://asaip.psu.edu`.

Eric D. Feigelson and G. Jogesh Babu. *Modern Statistical Methods for Astronomy: With R Applications*. Cambridge University Press, New York, 2012.

Andrew Harris. *FITSio: FITS (Flexible Image Transport System) utilities*, 2012. URL `http://CRAN.R-project.org/package=FITSio`. R package version 1.2-0.

Duncan Temple Lang. *RCurl: General network (HTTP/FTP/...) client interface for R*, 2011. URL `http://CRAN.R-project.org/package=RCurl`. R package version 1.6-0.1.

R. W. Nightingale. *AIA/SDO FITS Keywords for Scientific Usage and Data Processing at Levels 0.1, 0.3, 0.5, 1.0q, 1.0, and 1.5*, 2009. AIA02840 Rev. E.

W. Pence. CFITSIO, v2.0: A New Full-Featured Data Interface. In D. M. Mehringer, R. L. Plante, and D. A. Roberts, editors, *Astronomical Data*

Analysis Software and Systems VIII, volume 172 of *Astronomical Society of the Pacific Conference Series*, page 487, 1999.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2012. URL `http://www.R-project.org/`. ISBN 3-900051-07-0.

Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer New York, 2009. ISBN 978-0-387-98140-6. URL `http://had.co.nz/ggplot2/book`.