UCL students reports

Overview

1) Methods like CUSUM charts can indeed be used to visualize individual station series and to identify time periods during which there were problems related to observers. The reports from I. and H. show this, but they also show that the approach needs further work. In particular, integrated or smoothed charts have to be combined with individual value charts to facilitate viewing, there have to be tools to zoom in on time periods, make annotations, flag values as known departure from the standard procedure which should be excluded from the calculation of the sun spot number. As an interesting observation, I. points out that the site FRI has very high availability but low performance. It may be useful to train/harmonize the observers and observation methods at this site.

2) The study of the sources of variability confirms the observations made by Dudok de Wit. The observed variability is a mix of two main sources, one related to the sun, the other to the observers. The character of the sun-related variability is more 'Poisson' the other more 'relative'. The students point out that a model for the time domain after suitable transformation is more of a ARIMA(3,1,1) type than of an AR(8) type. Since the time-domain variability is related to the sun itself, the same model may be used for all (station) series. One should check to which extent it is capable of filling in limited amounts of missing data and of producing residuals which could be used for station characterization and validation (CUSUM on the residuals, or EWMA)

3) The Haar-Fisz transform should be looked at in more detail, in particular to see whether it can be extended without too much difficulty to general numbers of observations and to moderate amounts of missing data. Its peculiarity that it is not simply transforming the current value but takes the time neighbourhood into account leads to attractive results.

4) With respect to algorithms, the works of C. and A. are interesting in several ways. C. finds that more components in the SVD would lead to an improved scheme. On the other hand, she does not show what the improvement in MSE means in practice. We don't know either (yet) what the reconstruction scheme does in the case of periods when there is a shift in a station. C. created Python code and I'm still checking it, but for that, I have to learn more Python.

5) A. also looked at the set of methods for the reconstruction of missing values. He also observes that taking 4 singular value components is not yet fully exhausting the data. He also makes a very interesting remark about newer work on soft imputation. We should definitely look at that.

In the latter context, the recent talk by Stephen Boyd deserves attention. Advances in the practical implementation of convex optimization algorithms make it possible to solve problems like ours 'directly'. That is, if we are able to formulate our problem or an approximation to it as a convex optimization with rather simple bounds, we can use efficient algorithms and a standardized formulation language to solve them. I tried out the cvx algorithm available in the Julia programming language and I was able to get it to run on a toy example. So far, this suite of methods is readily available in Matlab and Julia. I haven't seen a Python package yet.

What might we be able to do:
Formulate the problem like this:
We would like to find a positive vector S (the sunspot index) such that
ai#*#S+bi is close to Xi
where Xi is the vector of (available) observations at station Xi
and ai and bi are either simply constants or again vectors which satisfy constraints on variation or patterns such that some appropriately chosen objective in S, a, and b is minimized

A few additional constraints may be needed.

Reports

UNIVERSITE CATHOLIQUE DE LOUVAIN

FACULTE DES SCIENCES

ECOLE DE STATISTIQUE, BIOSTATISTIQUE

ET SCIENCES ACTUARIELLES

LSTAT2390 – Statistical Consulting

The sunspot number and the solar cycle : Variability Analysis

Executive Summary

International Sunspot number is a composite index based on a large number of observations from different stations. It is therefore a time series that is subject to many constraints and source of uncertainties. Extracting information from this time serires is not easy and often requires preliminary treatments. This paper will discuss three different methods for stabilizing de variance of the series and their effects on the estimation of two main types of errors : The prediction error within the stations and the dispersion error between the stations. The step that follows the estimation of errors is the definition of a link between the errors and the international sunspot number. This link varies according to the transformations and between the stations, nevertheless the common components specific to the estimated errors can be extracted and studied.

Louvain-La-Neuve

May 2017

LSTAT2390 – Statistical Consulting
The sunspot number and the solar cycle : Variability Analysis

1 Introduction

This project is interested in sunspots, these are dark spots appearing in groups on the solar surface. They are characterized by a lower temperature and intense magnetic activity of the sun. The sunspot number is a direct observation of solar activity that have been counted since the invention of the telescope in the early 17th century. Today, a lot of stations collect the number of sunspots every day. All this information have merging in a single composite index : Sunspot Index and Long-term Solar Observations[1] (SILSO).

Sunspot numbers are counted by the following equation : $N_w = k(N_g + 10N_s)$. This equation defines the solar activity N_w as a function of the number N_s of spots, the number N_g of groups of spots and a coefficient k correcting the result according to the observers. This formula, introduced by *Rudolf Wolf*[2], makes it possible to quantify solar activity.

This report focuses on estimating the errors in the sunspot numbers obtained and stabilizing the variance of the time series. Le purpose is a better understanding of the model and source of sunspot variability. The interest in analysing this variability is to identify the different sources of possible errors when calculating the composite index of Silso such as missing values and how to manage them, the weather or the impact of the observer on the count of sunpots. Understanding the uncertainty structure is important when aggregating the various observed data into a global index that provides a perpetual approximation of solar activity. The analysis made in this document has as its starting point the article of *T.Dudok de Wit, L.Lefèvre and F.Clette [1]* where the authors' proposition is to distinguish two types of errors:

- **Time-domain errors** : Errors over time for each station. Enables to consider the variability within the stations.
- **Dispersion errors** : Errors between stations. Enables to consider the variability between the stations.

This paper will begin with the explanation and application of several methods for variance stabilization and the effect on the data representation. It will follow the estimate of the two differents types of error for each transformation. This report will conclude with a comparison of the estimated erros with with the composite index of sunspot number from Silso and a short conclusion.

[1] Data are available on `http://www.sidc.be/silso`

[2] Johann Rudolph Wolf (1816 -1893), a Swiss astronomer who studied the Sun

2 Database

The dataset includes sunspot data from 52 observations stations from 1981 to 2015. The original database contains a large number of missing values and there may be several reasons for that such as the weather, the lack of observers available or simply the lack of financial means. There are numerous methods to fill these missing values, usually by interpolation but it's a complex discussion and it will not be discussed in this report. Nonetheless, the data that will be used will already have had a preliminary treatment in order to fill these missing values from the approach proposed by *T. Dudok de Wit* [2]. This method is based on an iterative singular value decomposition to interpolate missing value of the original dataset by gaussian smoothing.

3 Variance stabilisation

The times series of the sunspot numbers are caracterized by heteroskedasticity in the series, the variance are not constant over time. Furthermore, a non-decreasing relationship between the mean and the variance can be observed. This relationship follows the solar cycle which is a period during which the activity of the Sun varies by reproducing the same phenomena as during the previous period. A period usually lasts between 8 to 14 years. Generally speaking, if the mean of the sunspot number is large, the solar cycle is at the top and the variance is also large. This is partly due to the fact that the increase in solar spot raises the uncertainty associated with counting them. The possibility of errors linked to observers, the material used by one of them, even the weather or any other reason makes a raise in the probability of errors.

Futhermore, for most statistical analysis, variance stabilisation is recommanded. Variance stabilisation is a procedure which consists of turning the sunspot number into a new random variable whose distribution is a Gaussian of fixed variance and mean, whatever the level of solar activity is (Bartlett, 1947). In this document, three variance stabilisation methods were compared :

- **Square root transform**: $\sqrt{SSN}$
- **Simple Anscombe transform**: $2\sqrt{(SSN + 3/8)}$
- **Haar-Fisz transform**: Data driven transformed by wavelet (Refer to the *appendix A.1* for more methodological information).

The *Figure 1* shows orginal times series of sunspot number in station *UC2* and its three transformations. On the left graph, we can see that, for the orignal data, the sunspots number follows the solar cycle and there is a significant non decreasing relationship between mean and variance. If mean of the time series increases, the variance also increases and we are in the middle of a solar cylce. Furthermore, this relation seems to disappear with the three other transformations and the variance of the sunspot number seems more stable. The graph on the right represents the stationnary data obtained by considering the difference with the first lag. Stationnary data gives a better representation of the effect of variance stabilization but it's also an underlying assumption to Box-Jenkins analysis which will be used later to estimate Time domain errors.

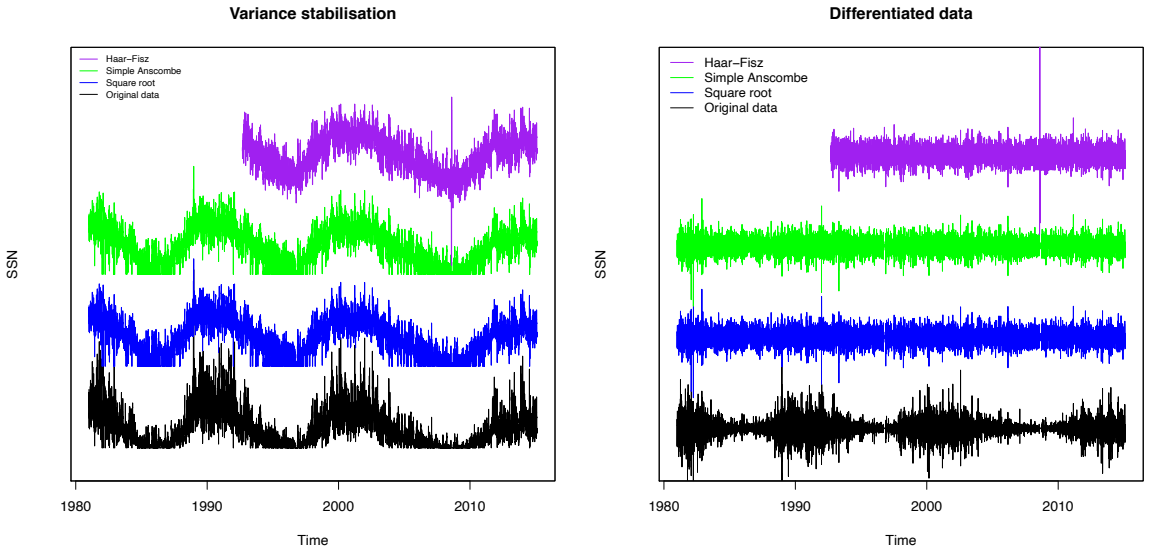


Figure 1: Variance stabilisation methods of the Non-stationary UC2 sunspots numbers (Left) and for stationary equivalent data (Right)

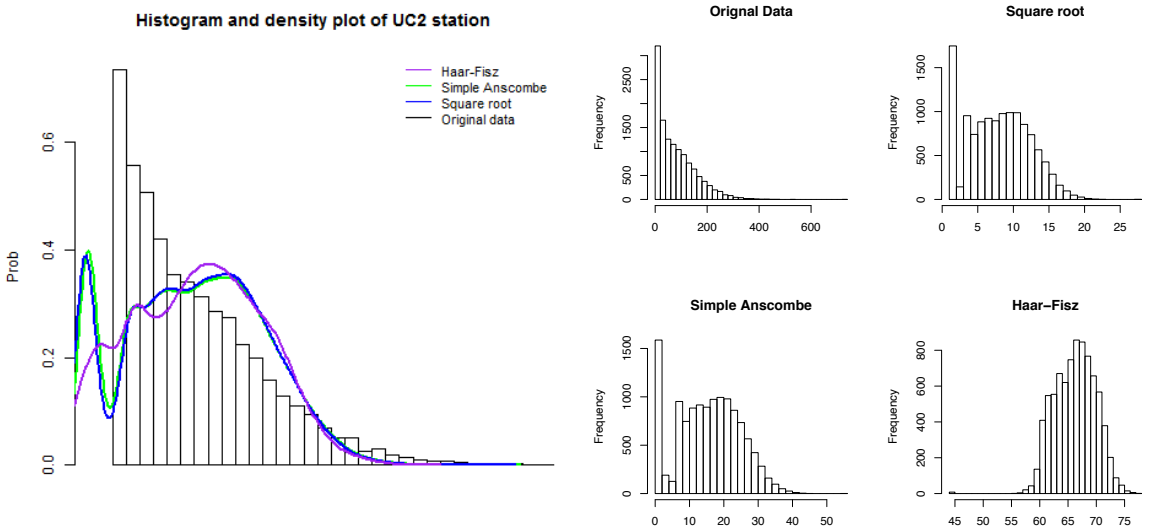


Figure 2: Histogram of tranformed data (Right) and with density of tranformed data (Left)

In the Haar-Fisz transformed data chart (*figure 1*), a pike can be observed around august 2008. There are no logical explanation that can be found in the original data to explain that. A rough explanation is probably that it is due to computation errors with the transformed data. See *appendix A.2.1* for more information. Therefore, visually, for each of the transformations, there is a clear improvement in terms of stabilized variance. We can conclude that the transformations clearly help stablize the variance. On the basis of the histogramme, *figure 2*, Haar-Fisz offers a better transformation, it is look more like Gaussianne distribution. Square

root and simple Anscombe transformation charts show a first pike in the histogramme that is not gaussian. This pike is explained by the number of observations equal to zero in the orignal data.

To prevent the variance of the sunspot number from rising when the solar cycle is at the top, simple transformations by square root already offer good results. However, despite the important underlaying assumptions[3] to Haar fisz implementation, this transformation is more similar to a gaussian distribution.

4 Time Domain errors

The main purpose of this report is to better understand the underlying uncertainties around the sunspot numbers and how variance stabilization methods can help support the analysis. Thereby these different transformations (Square root, Simple Anscombe and Haar-Fisz) need to be further compared to when fitting models and estimating time domain errors and dispersion errors.

4.1 Estimation

In order to estimate time domain errors, the autoregressive model of order $p = 8$ (AR8) is replicated as per the reference article. It compares with a more standard autoregressive integrated moving average[4] (ARIMA) approach. After transforming the data to stabilize the variance and differentiate it to reach weak stationarity, data are also scaled to provide consistent and comparable measures.

Station	**Orignal data**	**Square root**	**Anscombe**	**Haar-Fisz**
LO	(5,1,3)	(5,1,5)	(3,1,5)	(3,1,2)
UC2	(5,1,2)	(3,1,1)	(3,1,1)	(1,1,1)
CA	(4,1,5)	(4,1,3)	(4,1,3)	(1,1,5)
KS2	(3,1,5)	(4,1,2)	(4,1,2)	(3,1,4)
KZ	(5,1,2)	(4,1,5)	(4,1,5)	(1,1,5)

Table 1: Best ARIMA(p,d,q) model obtained for 5 examples stations.

The analysis of the ACF and PACF, *appendix A.2.3*, leads to include Moving Average (MA) parameters in the model. Pure autoregressive model isn't the most parsimonious model. All estimated ARIMA models also have a differencing parameter that is used to reach stationarity. The *table 1* shows that adding MA parameters often leads to a smaller number of AR parameters overall.

In order to evaluate the fitting quality of the model, the Akaike Information Criterion (AIC) is used. As the following equation shows, this criterion take into account the goodness of fit of the model and the complexity of the model by penalizing the number of parameters.

[3]Non decreasing Mean-variance relationship and the length of the observed data must be of power 2

[4]Notation : ARIMA(p,d,q) with number of autoregressive parameters p, degree of differencing d and number of moving average parameters q

The best model is the one that minimizes this criterion.

$$AIC = 2k - 2\ln L \qquad \begin{aligned} k &= \text{number of parameter} \\ L &= \text{ maximized value of the likelihood} \end{aligned}$$

Table 2 represents the AIC for our five example stations. Several remarks can be made. First, ARIMA models always have a lower AIC than their AR equivalents. Square root transform and simple anscombe transform are very similar and are lower AIC than the fit on the original data. This shows the positive effect of variance stabilization on the models. Haar-Fisz transform is generally better than the original data but shows more ambiguous results compared to other transformations. The last remark is that there is still a relatively large difference in result between each station.

	Orignal data		**Square root**		**Anscombe**		**Haar-Fisz**	
Station	AR(8)	ARIMA	AR(8)	ARIMA	AR(8)	ARIMA	AR(8)	ARIMA
LO	1742	1055	944	215	1224	565	2368	1929
UC2	13130	12533	9659	9088	9718	9148	8151	7745
CA	6513	5874	3699	3053	3849	3207	4081	3670
KS2	8525	7852	5726	5075	5871	5225	4013	3568
KZ	8145	7468	5427	4770	5593	5012	5541	5158

Table 2: Comparaison of models by Akaike Information Criterion

4.2 Validation

When comparing AR and ARIMA models for all three transforms, two main conclusions can be made and can be see on *figure 3*.[5]

- The residuals are generally not Gaussian and fail the white noise test for the AR(8) models. They are generally closer to normality for ARIMA models, and can be considered white noise for the first 20 to 30 lags for the ARIMA models based on the square root and Anscomb transforms.
- When the data is only centred, the residuals are generally not Gaussian, have a non-stable variance and cannot be considered white noise, regardless of the model. However transforming the data leads to residuals with a stable variance.

The QQplot in the *appendix A.3.2* shows that the residuals are generally closer to normality for ARIMA models. The Box-Pierce statique also shows that they can be considered white noise for the first 20 to 30 lags for the ARIMA models based on the square root and Anscomb transforms. Since we work in daily data, the number of significant lag could have a link with the time of rotation of the sun (averaging over 27 days).

[5]Other diagnostic plot are display in *appendix A.3.1*

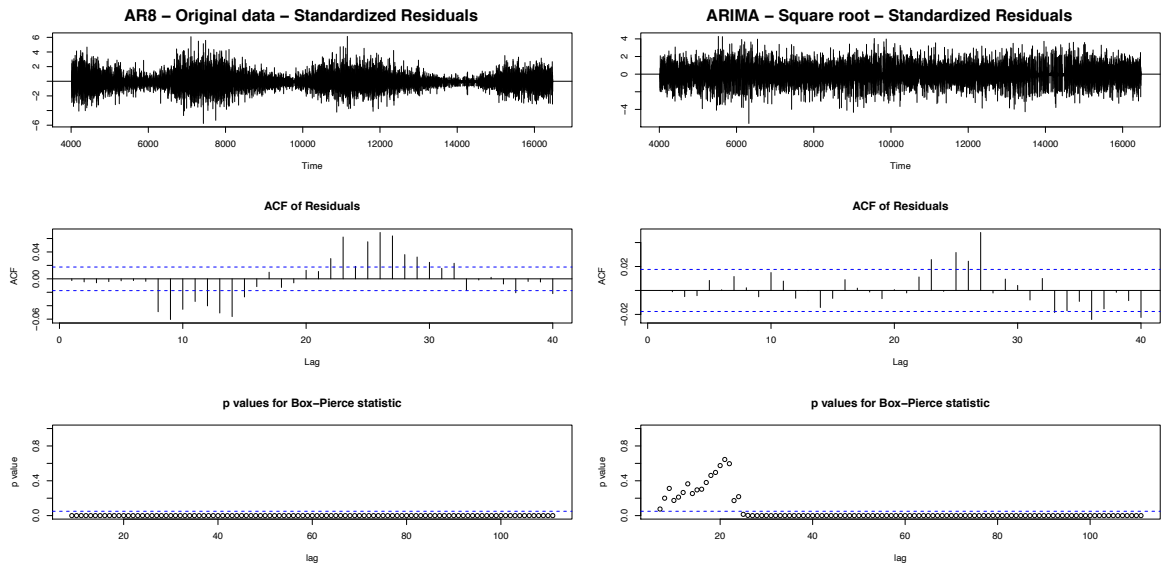


Figure 3: Diagnostic plot for AR(8) model on original data (Left) and ARIMA(3,1,1) on square root transform for UC2 station (Right)

4.3 Prediction

The Root Mean Square Error of Prediction (RMSEP) can be calculated to compare the quality of prediction of each model. It is then calculated recursivly by one ahead prediction approach[6] on the final t observations of the series. For our 5 example stations, the choice of $t = 6000$ (roughly 50% of observations) was made. It's a good compromise between model robustness, number of predictions and computing time. RMSEP is also calculated only for predictions with the non-missing values in the original serie. Data are always centered and scaled after transformation with the intention to obtain comparable RMSE values for each of the estimated models.

Figure 4 shows the evolution of the RMSEP over the final t observations for ARIMA model on *UC2* station. The RMSEP of the original data seems to follow the solar cycle : The variance of the data and de RMSEP are smaller when the solar cycle is at a low point. The minimum of the original data curve (index = 3000) corresponds to the end of the last solar cycle in the year 2009. On the other hand, Square root and Anscombe transform stabilize the RMSEP more quickly and Haar-Fisz transform has greater RMSEP but is less subject to solar cycles than original data

Table 3 displays RMSEP for AR and ARIMA models on all considered transformations for 5 stations. Although the ARIMA models are a better fit for the data, particularly on the transformed data, the RMSEP is only marginally smaller. In terms of transformation, the Haar-Fisz transform leads to a larger RMSEP, therefore, the square root or simple Anscombe transforms seem to be a more suitable transform to estimate the time domain errors. Moreover, even if the ARIMA model is more justifiable in terms of residuals' diagnostic. The residuals look more like a withe nose. In terms of predictive capacity, ARIMA model is only marginally lower than the AR model. The advantage of choosing AR is that a single model

[6] At every time m, prediction is estimated in time $m + 1$ and compared with his true value for the last t observations

$$RMSEP = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}}$$

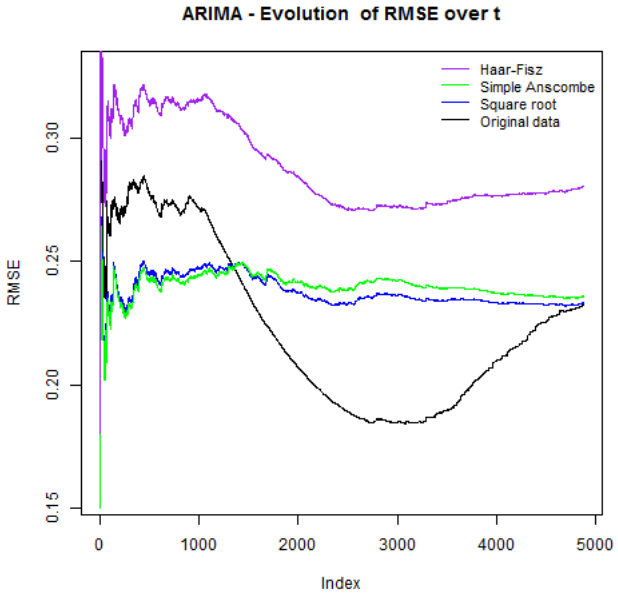


Figure 4: RMSEP estimated recursively for each t

	Orignal data		**Square root**		**Anscombe**		**Haar-Fisz**	
Station	AR(8)	ARIMA	AR(8)	ARIMA	AR(8)	ARIMA	AR(8)	ARIMA
LO	0.239	0.232	0.240	0.233	0.243	0.236	0.280	0.280
UC2	0.362	0.353	0.328	0.319	0.329	0.320	0.384	0.371
CA	0.314	0.306	0.283	0.275	0.284	0.275	0.316	0.304
KS2	0.310	0.302	0.287	0.278	0.288	0.280	0.280	0,300
KZ	0.308	0.301	0.299	0.292	0.301	0.294	0.347	0.338

Table 3: RMSE of AR and ARIMA model for 5 example stations

estimates the errors in each station, while the ARIMA model needs to adjust these parameters for each station.

Figure 5 provides a visual representation of errors estimated by the ARIMA model for each of the transformations and for the UC2 station. As with the sunspot number, there is a positive effect of the transformations. The first scaterplot shows a visually stable series for the square root, simple anscombe and Haar-Fisz transformation. On the other hand, the histograms are less clear. The effect of the three transformations on the distribution of errors is less marked and look more like a Poisson distribution

4.4 Best common ARMA model

In the previous section, it was shown the ARIMA models are a better fit for the data. It may also be possible to define a fixed-parameter ARIMA model with a good prediction capability for each of the stations. To find the best common model, all combination for autoregressive and moving average parameters ranging from 1 to 5 were tested. The summary table for UC2 is shown in *appendix A.3.3*. The models are then compared on the basis of

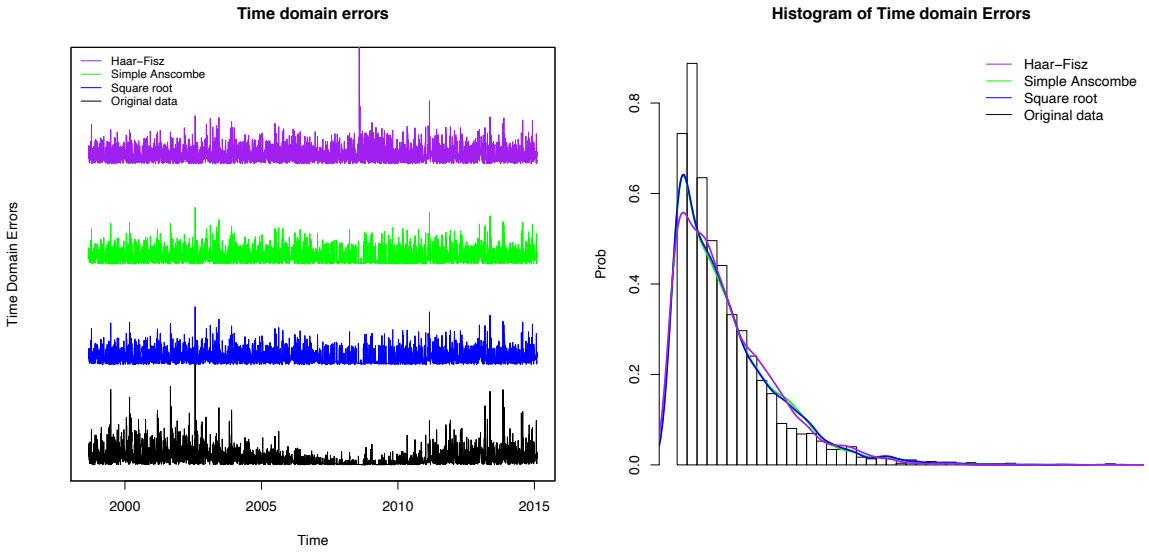


Figure 5: Scaterplot and histogram of Time domain errors for original data and transformed data

their respective RMSE and AIC. There is no model that is particularly better than the others for the 5 stations. However, the difference in terms of RMSE and AIC between the different ARIMAs is marginal. Therefore, we could consider arbitrarily a model ARIMA(3,1,1) with very few parameters and good goodness of fit. This model could be fit for each station.

5 Dispersion Errors

The second type of error that can be calculated for sunspot numbers is the dispersion error between stations. In this section, two points will be studied : Comparaison of dispersion errors between transformations and comparaison between the original dataset with and without the missing values. We are working here on data that are not prefiltered and therefore contain missing values. The Haar-Fisz transformation will no longer be considered since it does not apply to this type of data.

Dispestion errors are estimated in a two step procedure on an subsample of de 15 best stations with the least number of missing values, as the article of reference of *T.Dudok de Wit, L.Lefèvre and F.Clette [1].*

1. Scaling[7] sunspot numbers for station i, $N_{W,i}(t)$, by the international sunspot number S_N from SILSO and estimate the residual errors for each station by :

$$N_{W,i}(t) = \gamma_i N_{W,i}(t) \qquad \textit{where} \quad \gamma_i = \frac{S_N}{N_{W,i}} \quad \textit{And}$$

$$\epsilon_i(t) = N_{W,i}(t) - \frac{1}{N}\sum_{i=1}^{N} N_{W,i}(t) \qquad \sum_{i=1}^{N} N_{W,i}(t) = \text{ means of the N stations}$$

[7]The scaling factor γ_i is estimated by by weighted total least squares.

2. Dispertion Errors is then calculated for each date based on residual of the 15 best stations.

$$\sigma(t) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\epsilon_i^2(t)}$$

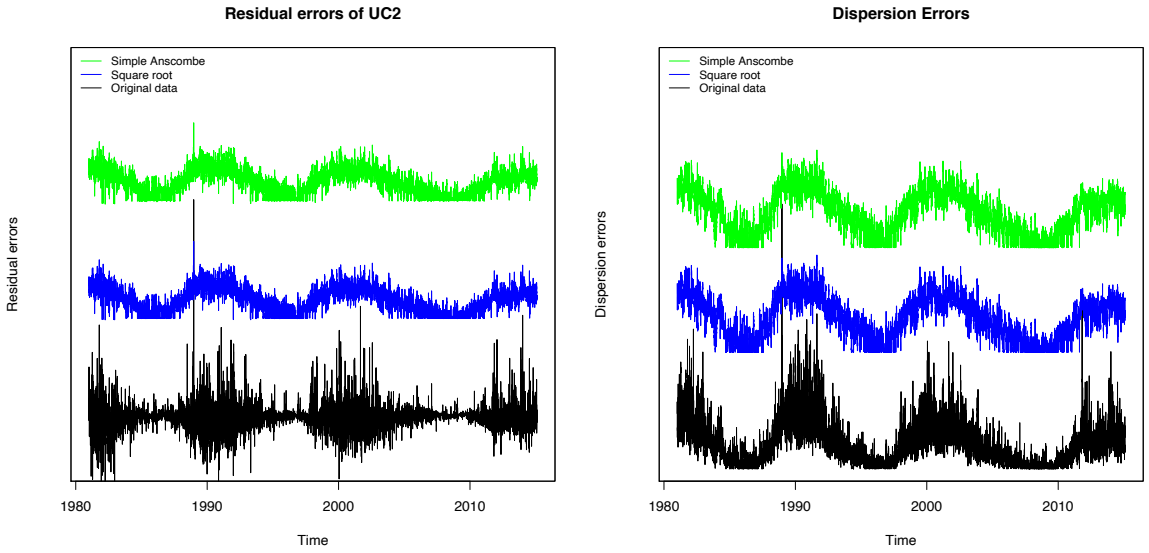


Figure 6: Residual errors for UC2 (Left) and Dispersion errors (Right) for orginal data and transformed data

On the one hand, *figure 6* shows the residual errors estimated by the first step for the station UC2 and, on the other hand, the overall dispersion between the stations to be estimated by the second step. Dispersion errors is time-dependent and increases with the sunspot number and the solar cycle. If the number of sunspots is large, it is more difficult to count them and the variability between the stations increases. While square root and simple Anscombe transform seems more stable for de residual errors of UC2 and the overal Dispersion errors between station. In *appendix A.4.1* histogram of these errors and stationnary representation of dispertion errors confirme the positive effect of square root transformation.

The *appendix A.4.2* also shows the effect of filling the gaps on the dispersion errors. As can be expected, estimating the missing values increases the variability of the data and de Dispersion Errors between stations

6 Link between error et Sunspot number

Now that Time domain errors and dispersion errors are estimated, we seek to define what relationship they have with the Sunspot Index and Long-term Solar Observations (SILSO), the international sunspot number. We would also like to observe the effect of variance stabilisation on this relationship.

6.1 Time domain errors and SILSO number

For the time domain errors, only the predictions on the last $t = 6000$ observations are considered and compared to sunspot numbers from SILSO (SSN). That represents the observations between 1998-09-06 and 2015-02-06. The relationship between time domain errors and Sunspot numbers is difficult to perceive on the raw data. However, when data are aggregated over 27 days, which corresponds on average to a rotation of the sun, the data is then smoothed and a relationship arises. By aggregating the data, the smallest variation is no longer automatically perceived[8]. Only the longern term variations (compared to the daily terme duration) become visible.

When we take the data of origin without transformation and aggregated to 27 days, a relationship close to the square root can be observed (figure *18*) but this differs according to the station. The graph also shows a funnel shape showing an increase in the variability of the error when the number of sunspots increases.

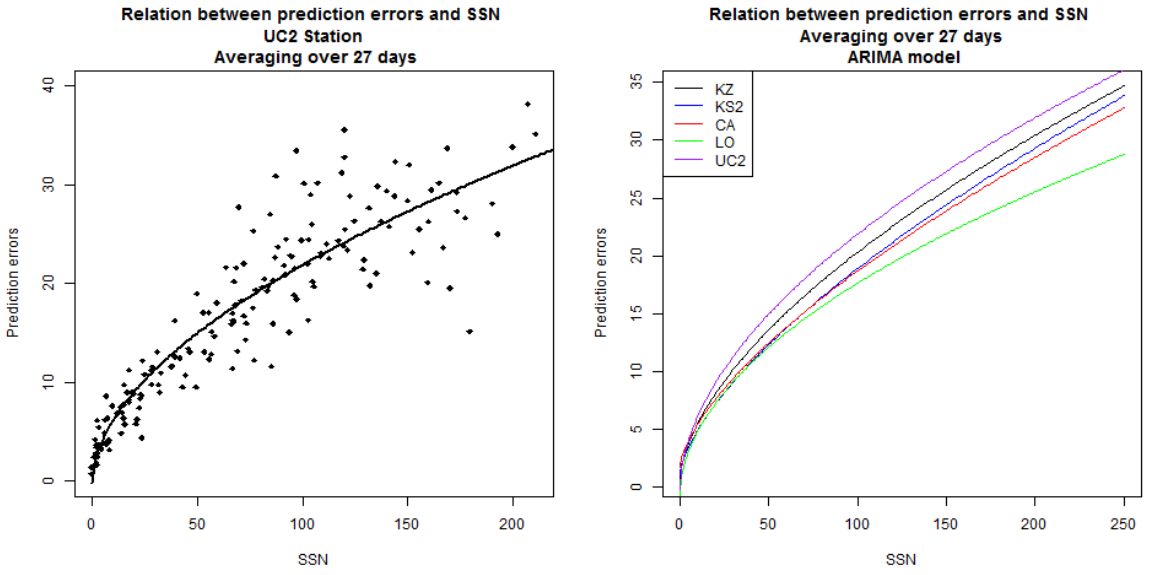


Figure 7: Relationship between time domain errors and SSN, averaging over 27 days (Left) and for 5 example stations (Right)

$$Error \sim \alpha + \beta \times SSN^{\gamma}$$

	Lo	CA	UC2	KS2	KZ
α	-0.71	1.75	0.18	0.54	1.09
β	1.69	0.81	1.82	1.27	0.84
γ	0.52	0.66	0.54	0.60	0.66

Table 4: Result of estimation of relationship between time domain errors and sunspot number

In *appendix A.5.1*, a representation of the relationship between aggregate Times domain errors and SSN for each of the transformations is available. The conclusions are similar to

[8]For example: weather, change of personnel, ... etc

what was said at the beginning of this document. The square root relationship with a form of funnel disappears and gives way to a more stable relationship where the variability of the error does not seem to be increased with the sunspot number

6.2 Dispersion error and silso number

Now the dispersion error $\sigma(t)$ will be put in relation with the sunspot number from Silso. From now on, we only consider the square root transformation. Simple anscombe is very similar and will have identical conclusions whereas the Haar-Fisz transformation adapts less easily in the case of dispersion errors since it does not manage the problem of the missing values.

The *figure 8* shows this link for the original data and the square root transformation. Similar to the Time domain error, the graph has a form of funnel, however, this time the relationship is more linear. When we observe the square root transform, the relationship is then clearly square root. Transforming data thus modifies the link between the errors and the sunspot numbers but does not do so consistently. In the *appendix A.5.2*, the graphs representing the average over 27 days confirm this observation by showing narrow curves. Nevertheless the transformed data curve no longer has this form of funnel, the variability does not seem to increase with the SSN when the data is transformed into square root. Stabilizing the variance by the square root transformation for each station improves the variability of the dispersion errors.

Figure 8: Relationship between Dispersion errors and Sunspot number of Silso for original data and square root transform

	Original data $\sigma(t) = \alpha + \beta \times SSN$	**Square root transform** $\sigma(t) = \alpha + \beta \times SSN^{1/2}$
α	1.9771	-2.50
β	0.1863	17.55

7 Conclusion

In this report, we first tested different transformations to stabilize the variance. The three tested methods lead to more homoskedasticity data but the square root and Simple Anscombe transforms seem to give better results in term of predictive capacity (RMSEP), while Haar-Fisz transform returns more gaussian data. It is therefore not necessary to use complex transformations to obtain interesting results. Nevertheless, the estimation of the relationship between the mean and the variance of the Haar-fizs method makes it possible to obtain a more suitable distribution.

For the estimation of time domain errors, prediction errors in each station, we have tested a general AR8 model and more specific ARIMA. The last one was more statically justifiable and offered better diagnostics than AR(8) models but lead to similar RMSE and confirm the magnitude of the time domain errors found in the original article.

There is also a clear improvement in the estimation of this error with stabilized variance of transformed data. Square root transforms offers very interesting results. The estimated errors are more constant and do not fluctuate with the solar cycle. Simple Anscombe is as always very similar to square root. Haar-fisz is more ambiguous, the transformation seems to give good and stable Time domain errors over time but its RMSE is significantly higher. Finally, for the Time domain errors part, it was possible to define an ARIMA model (3.1.1) by comparing the AIC and RMSE criterion which could potentially adapt to all the stations.

Dispersion errors (Errors betwen the stations) are time-dependent. Like time domain erros, transformation of the data has positive effect on dispersion errors.

In the next section, we made the link between the errors and the sunspot number from silso. The link raw data base is not easily observed, however, when aggregating the data (i.e. averaging over 27 days - sun rotation time), the result is smoothed and the daily noise around the data disappears. For the time domain error, we notice a square root link with the international sunspot number. The shape of the funnel also shows that when the SSN increases, the variance of the data also increases. Nevertheless, the effect of the transformations shows a more random and stable link between errors and SSN. The errors no longer increase with the number of sunspots and the square root link disappears.

Contrary to the time domain errors, the link between the dispersion errors and the international sunspot number is linear for the original data. The form of funnel remains present when SSN increases. At the level of the square root transformation, the relationship changes to a square root link. This change is probably due to the transformation but, as before, the link is more limited and the variability of the data is lower.

A Appendix

A.1 Haar-Fisz Transform

DDHFT is an automatic method to stabilize the variance from an estimate of the relationship between variance and mean. This method has shown good results data that follow Poisson distribution (more information are available on *P.Fryzlewics and al.(2007)* [3]). Data Driven Haar fisz transform is an orthogonal transformation that uses the coefficients obtained from the decomposition by wavelet of the series. The coefficients then contain the information on the local behavior of the data.

This method is subject to 2 main conditions: The variance must be a non-decreasing function of the mean $\sigma^2 = h(\mu)$ and the length of the observed data must be of power $J = \log_2(n)$

The to transform data follow these step (R function, $ddhft.np.2$ in variance stabilization by Data-Driven Haar-Fisz package)

1. Wavelet transforms the time series by the Haar wavelet.
2. Estimation of mean-variance relationship between finest level smoothing and detail wavelet coefficients using isotonic regression

$$\sigma_i^2 = h(\mu_i) + \epsilon_i$$

3. Divide wavelet coefficients by smooth ones subject to the estimated mean-variance relationship
4. Perform the inverse Haar wavelet transform of the modified coefficients.

A.2 Variance stabilisation

A.2.1 Haar-Fisz Transform Extreme value

This appendix focuses on a portion of the data containing extreme values of the the Haar-Fisz transformation. A priori these values as of august 2008 do not derive their origins from the original data. Figure 9 compares transformed data by the Haar-Fisz method and the original data for the *UC2* station. However, values of this amplitude do not appear on each station and seem be specific to UC2. By hypothesis, it's assumed that these extreme values occur during wavelet transformation from the Haar-Fisz method, it's a computational problem.

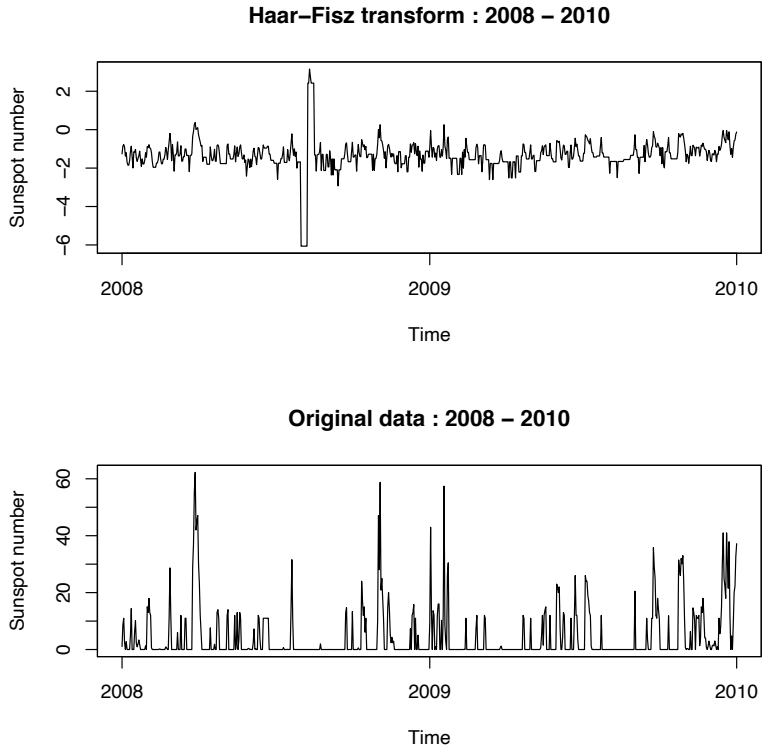


Figure 9: Comparison of the Haar-Fisz transformation and the original data between 2008 and 2010

A.2.2 Short term Variance stabilization

The low term standard deviation, such as taken every 20 days, is also subject to fluctuation of the solar cycles. The *figure 10* represents the evolution of standard deviation by a 20 days interval and shows the positive impact of the three variance stabilization methods. Conclusions are similar to the data taken as a whole. There is an important relationship between the mean and the variance for the original data. This relationship disappears after transformation and data and the data has a local standard deviation that is more random and homoscedasticity

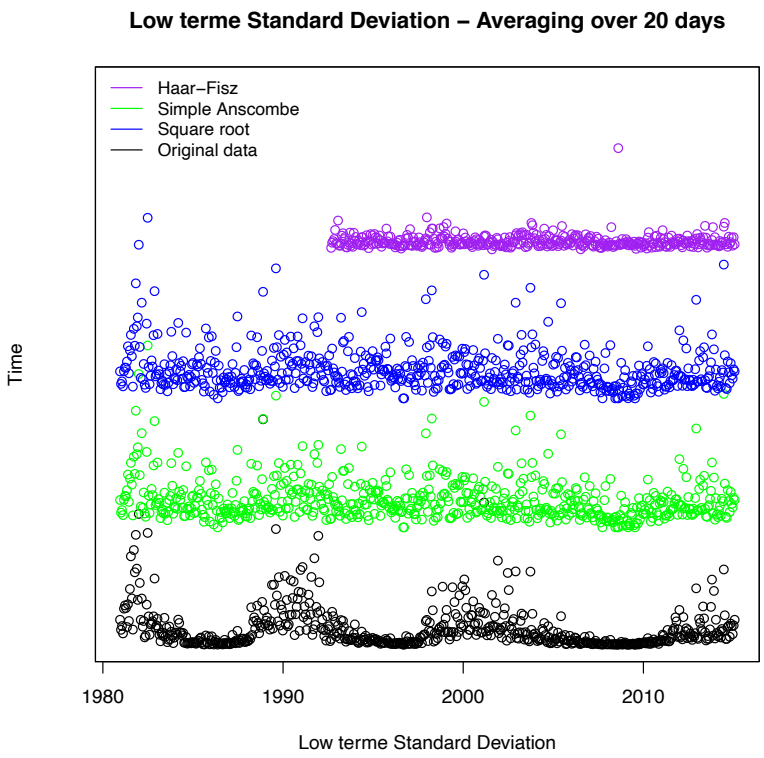


Figure 10: Effect of stabilization method on low terme Standard Deviation - Averaging over 20 days

A.2.3 Autocorrelation functions

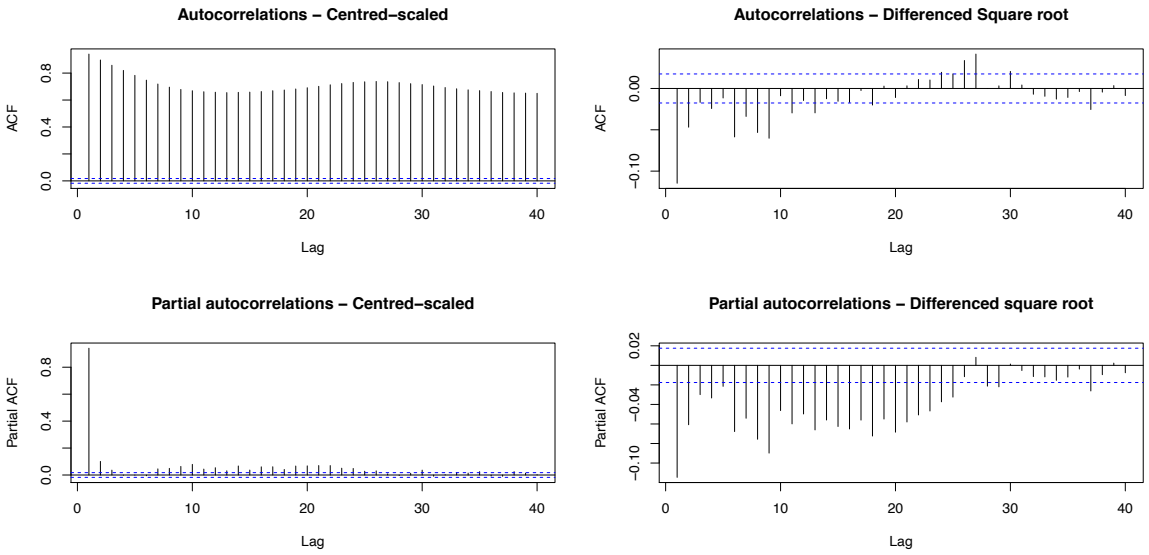


Figure 11: Autocorrelations function and partial autocorelation function for UC2 - Original data (Right) and UC2 - Square root transformation (Left)

A.3 Time domain errors

A.3.1 Dianostic plot

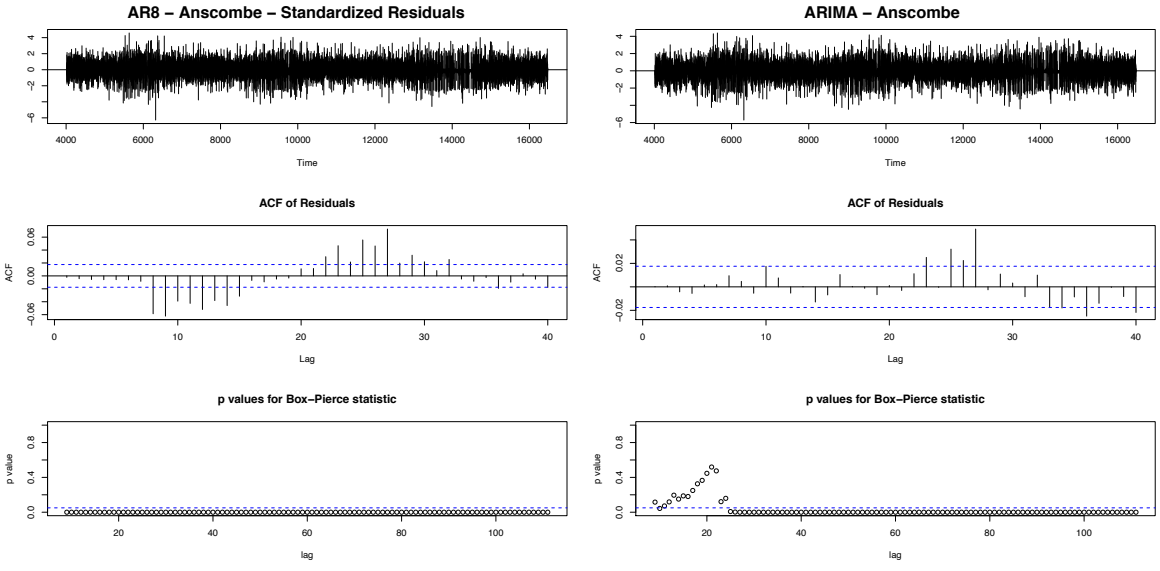


Figure 12: Diagnostic plot for AR(8) model (Left) and ARIMA(3,1,1) with simple Anscombe transform for UC2 station (Right)

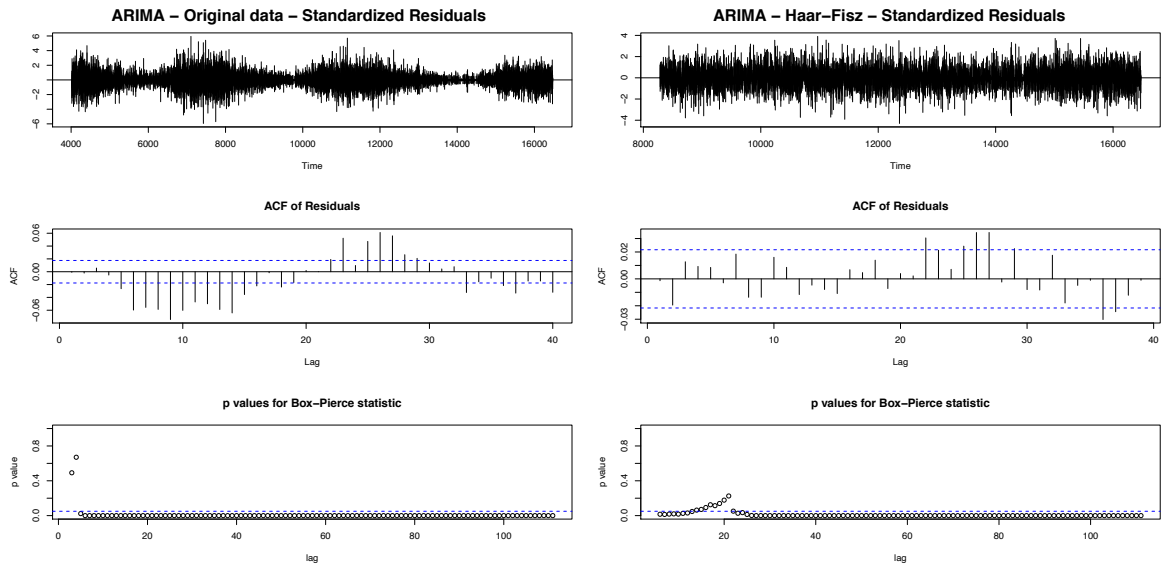


Figure 13: Diagnostic plot for ARIMA model on original data (Left) and ARIMA on Haar-Fisz transform for UC2 station (Right)

A.3.2 QQplot of residuals ARMA model

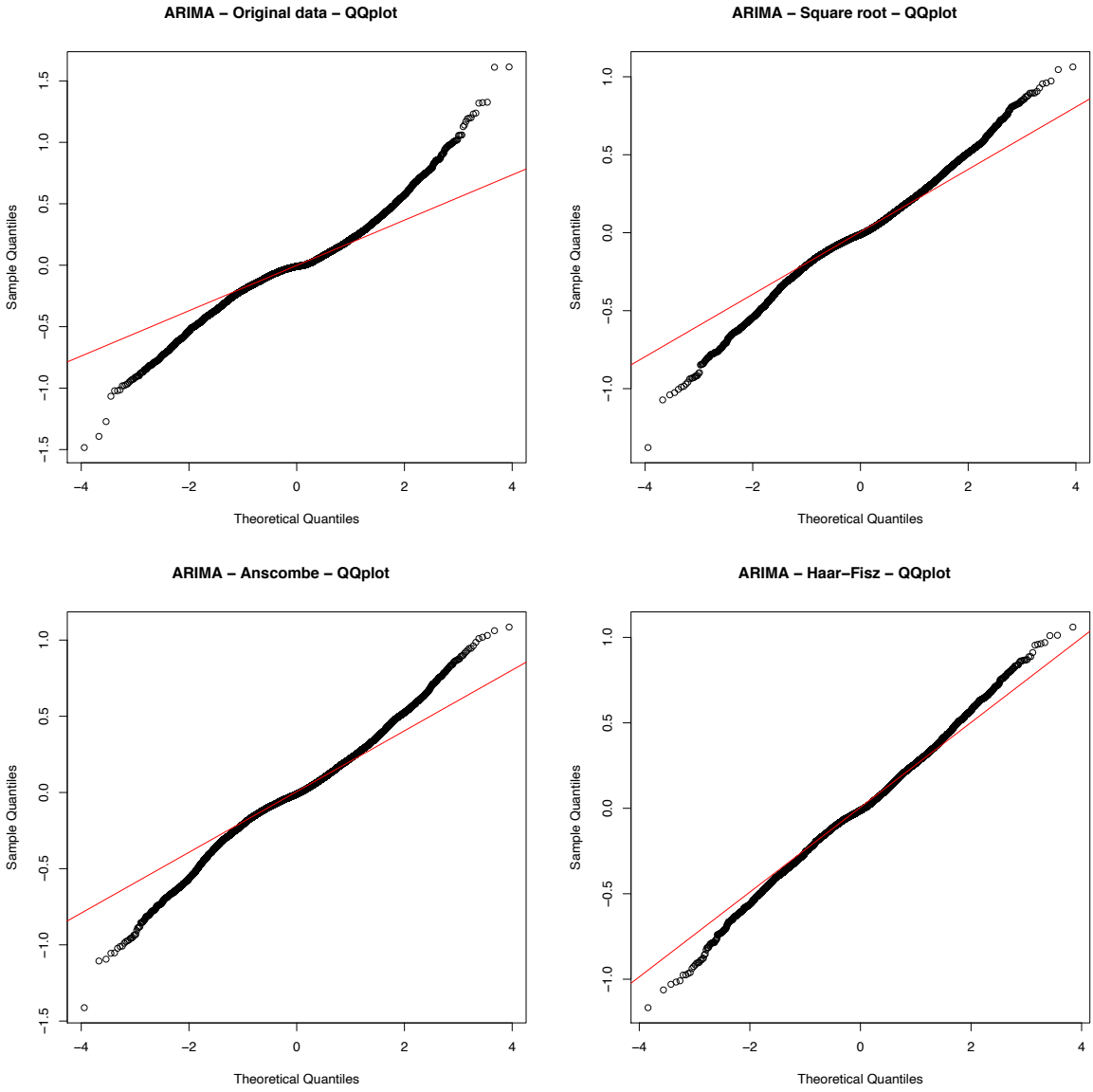


Figure 14: QQ plot of residuals of ARIMA model on original data, square root, simple Anscombe and Haar-Fisz transform

A.3.3 Best common ARMA model

RMSEP of ARIMA(p,1,q) for original data

	(p,1,1)	(p,1,2)	(p,1,3)	(p,1,4)	(p,1,5)
(1,1,q)	0.415	0.413	0.411	0.411	0.411
(2,1,q)	0.413	0.412	0.411	0.411	0.411
(3,1,q)	0.411	0.411	0.412	0.411	0.411
(4,1,q)	0.411	0.411	0.411	0.411	0.411
(5,1,q)	0.411	0.410	0.411	0.411	0.411

RMSEP of ARIMA(p,1,q) for Square root transform

	(p,1,1)	(p,1,2)	(p,1,3)	(p,1,4)	(p,1,5)
(1,1,q)	0.338	0.337	0.336	0.336	0.336
(2,1,q)	0.337	0.336	0.336	0.336	0.336
(3,1,q)	0.336	0.336	0.336	0.336	0.336
(4,1,q)	0.336	0.336	0.336	0.336	0.335
(5,1,q)	0.336	0.336	0.336	0.335	0.336

AIC of ARIMA(p,1,q) for original data

	(p,1,1)	(p,1,2)	(p,1,3)	(p,1,4)	(p,1,5)
(1,1,q)	12,647	12,625	12,574	12,575	12,569
(2,1,q)	12,630	12,598	12,576	12,578	12,574
(3,1,q)	12,571	12,573	12,573	12,570	12,520
(4,1,q)	12,572	12,566	12,566	12,576	12,503
(5,1,q)	12,571	12,532	12,577	12,579	12,572

AIC of ARIMA(p,1,q) for original data

	(p,1,1)	(p,1,2)	(p,1,3)	(p,1,4)	(p,1,5)
(1,1,q)	9,135	9,129	9,090	9,090	9,082
(2,1,q)	9,130	9,110	9,091	9,094	9,093
(3,1,q)	9,088	9,089	9,087	9,024	9,098
(4,1,q)	9,089	9,092	9,093	9,083	9,087
(5,1,q)	9,084	9,092	9,092	9,031	9,086

A.4 Dispersion errors

A.4.1 Transformation Effect

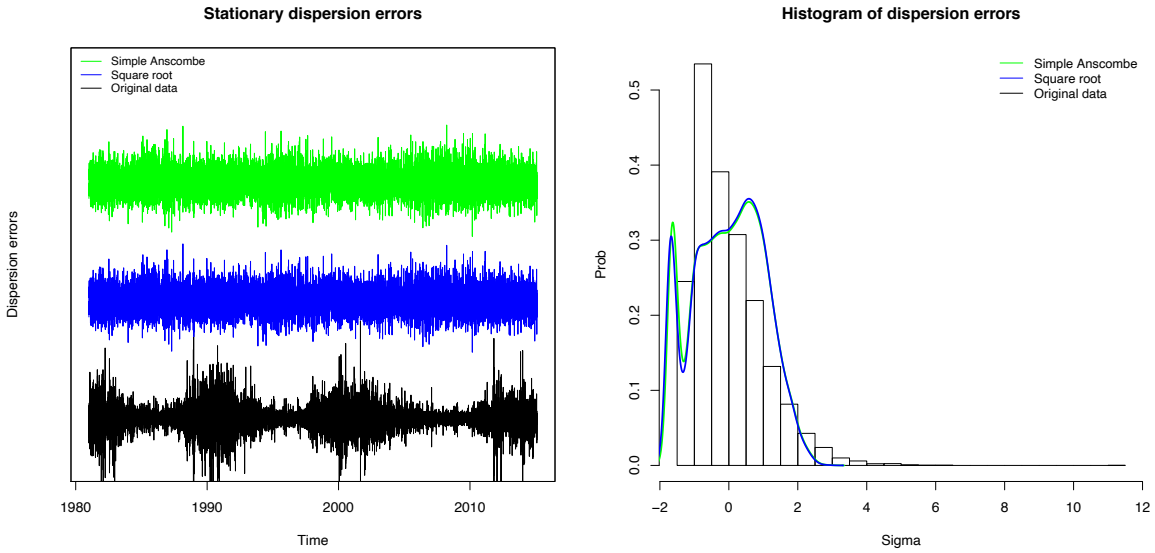


Figure 15: Dispersion Error on stationnary data (Left) and histogram of dispersion Error for orignal data and transformed data (Rigth)

A.4.2 Effect of filling in the gaps

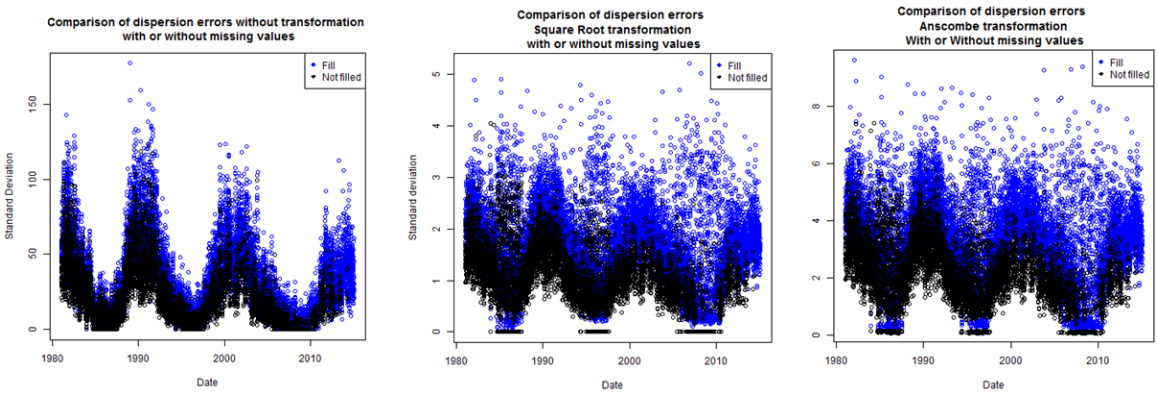


Figure 16: Effect of filling in the gaps for original data, square root and Anscombe transform

A.5 Comparaison errors

A.5.1 Time domain errors and Silso number

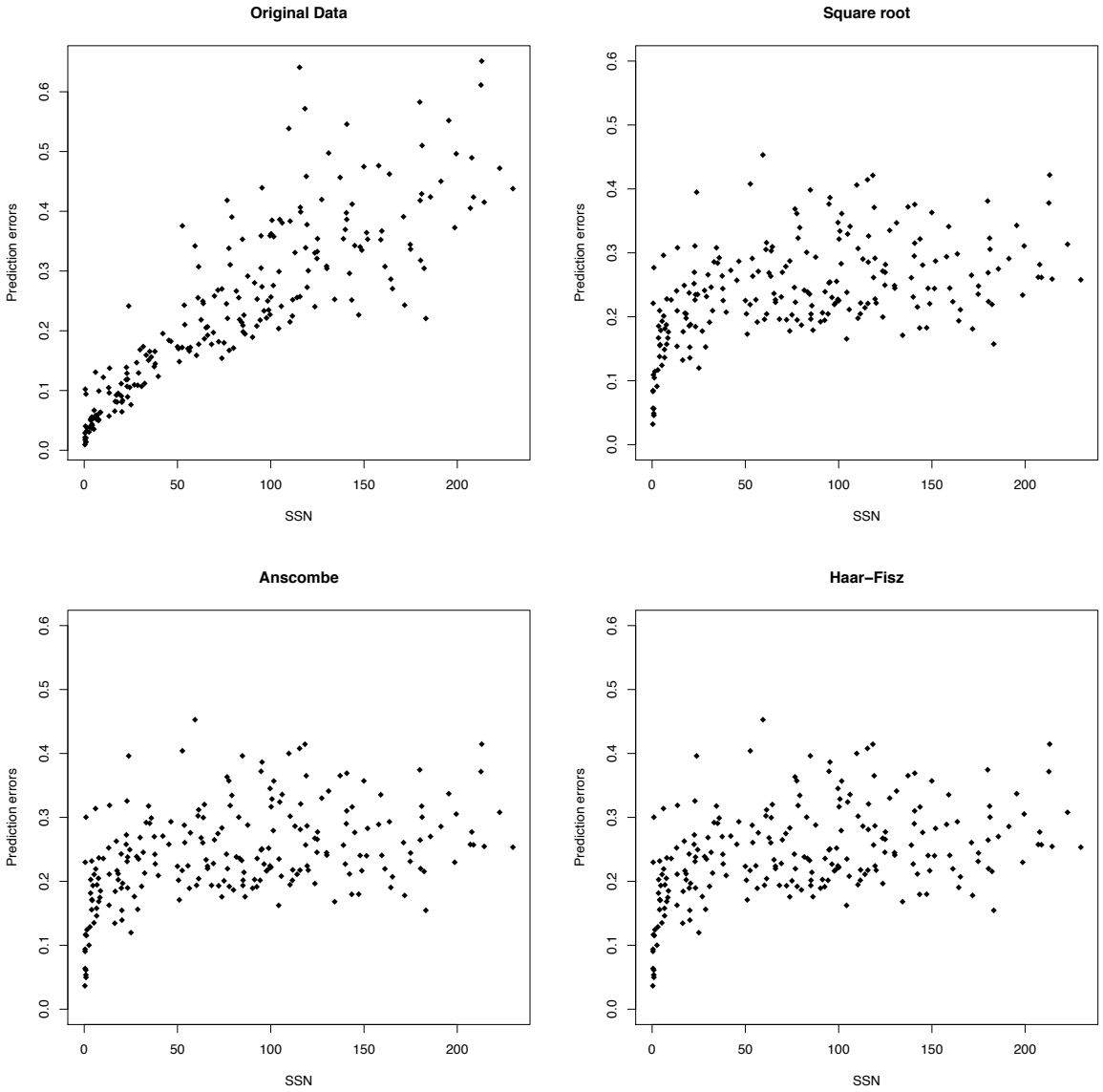


Figure 17: QQ plot of residuals of ARIMA model on original data, square root, simple Anscombe and Haar-Fisz transform

A.5.2 Dispersion errors and Silso number

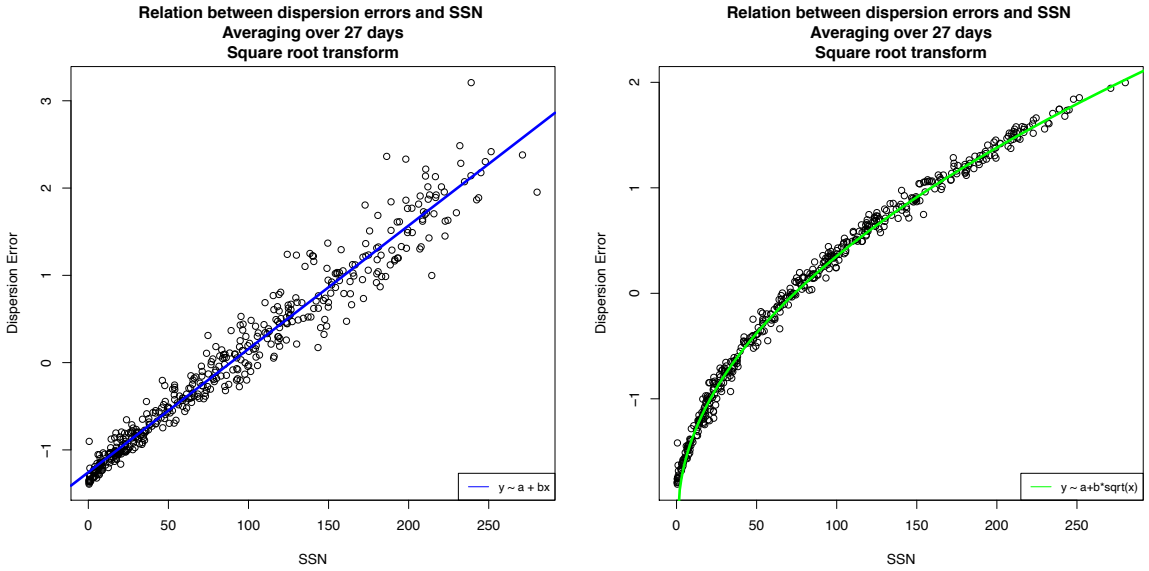


Figure 18: relationship between Dispersion Error and Sunspot number of silso for original data and square root transform, averaging over 27 days

References

[1] T. Dudok de Wit, L. Lefèvre, and F. Clette, "Uncertainties in the Sunspot Numbers: Estimation and Implications," *Solar Physics*, vol. 291, pp. 2709–2731, Nov. 2016.

[2] T. Dudok de Wit, "A method for filling gaps in solar irradiance and solar proxy data," *Astronomy & Astrophysics*, vol. 533, p. A29, Sept. 2011.

[3] P. Fryzlewicz, V. Delouille, and G. P. Nason, "GOES-8 X-ray sensor variance stabilization using the multiscale data-driven Haar–Fisz transform," *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, vol. 56, no. 1, pp. 99–116, 2007.

[4] P. Fryzlewicz and V. Delouille, "A data-driven Haar-Fisz transform for multiscale variance stabilization," in *Statistical Signal Processing, 2005 IEEE/SP 13th Workshop on*, pp. 539–544, IEEE, 2005.

[5] F. Clette, D. Berghmans, P. Vanlommel, R. A. Van der Linden, A. Koeckelenbergh, and L. Wauters, "From the Wolf number to the International Sunspot Index: 25 years of SIDC," *Advances in Space Research*, vol. 40, pp. 919–928, Jan. 2007.

LSAT2390: Statistical Consulting
VALUSUN Project
Filling Missing Values

7 mai 2017

Executive Summary The SunSpot Number (SSN) serves multiple purposes. It is the only direct multi-secular tracer of the solar cycle and a quantitative reference of both the solar irradiance and sun driven processes. The number of sunspots reflects the solar activity. It is measured by observers around the world looking everyday at the sun. Due to either cloud cover or 'human related' reasons, the signal coming from each station contains gaps.

Missing values are a general problem in data analysis and we consider here a data-adaptive and non-parametric method developed by T. Dudock de Wit [1]. The idea is that what an observer see at time t is correlated to what he observed in time t-1, this is the time correlation. On top of it, since all observers are looking at the same sun, their observeations are also correlated. The method uses both the time correlation and the between stations correlation to fill the gaps by means of low-rank approximations based on the singular value decomposition.

The goal of this project was first to translate the original code written by the author in MATLAB to both Python and R, and ensure that the obtained results are equal to those obtained via the original MATLAB code. The current report is mainly about the Python part. Secondly, we analyze the time complexity. The Python code is slower than the MATLAB code but with the same asymptotic behavior. Thirdly, as the method uses parameters, we design a cross-validation procedure to adjust one of the parameters, the number of components taken into account in the singular value decomposition, which should be set equal to 10 according to the results.

---

1. Dudock de Wit T., A method for filling gaps in solar irradiance and solar proxy data. Astronomy & AstroPhysics, 533, A29.

1 Introduction

The SunSpot Number (SSN) serves multiple purposes. It is the only direct multi-secular tracer of the solar cycle and a quantitative reference of both the solar irradiance and sun driven processes. The number of sunspots reflects the solar activity. It is measured by observers around the world looking everyday at the sun. Due to either cloud cover or 'human related' reasons, the signal coming from each station contains gaps.

Missing values are a general problem in data analysis and we consider here a data-adaptive and non-parametric method developed by T. Dudock de Wit [2]. The idea is that what an observer see at time t is correlated to what he observed in time t-1, this is the time correlation. On top of it, since all observers are looking at the same sun, their observeations are also correlated. The method uses both the time correlation and the between stations correlation to fill the gaps by means of low-rank approximations based on the singular value decomposition.

This report shortly introduce the method developed by T. Dudock de Wit to fill missing values before analyzing the time complexity of the method. More precisely, we compare the time complexity of the Python code versus MATLAB code.

The python code, available in appendix A, is composed of 2 main files :
— `interpolsvd_em.py` : contains the main method as well as helper methods.
— `script.py` : is divided in three parts. Firstly, it contains code to read the data from a file `Alldata.csv` containing missing values, apply the method and export the results (data without missing values) in another file `filledData.csv`. Secondly, it contains the code used to obtain the time complexity results presented in section 3. Thirdly, it presents the code used to perform the cross-validation procedure described in section 4.

2 Method

The method developed by T. Dudock de Wit is based on an iterative singular value decomposition. We start by filling the gaps with a basic method and keep track of the gaps position. For example, we can fill all the gaps of a given station by the time average of this station. Once the matrix is bascially filled, we can compute its singular value decompostion :

$$M = U\Sigma V^T$$

We then compute the rank 1 approximation of the matrix :

$$M^* = U_1\Sigma_1 V_1^T$$

Afterwards, we update the position of the orginal gaps with the values obtained with this low rank approximation. All the other values remain unchanged. We compute SE as the sum of squarred difference between the old gaps replacement and the new. As long as this value E is above a predefined threshold, we iterate this computation : compute the low rank approximation of rank one, replace the old gaps, compute E...

2. Dudock de Wit T., A method for filling gaps in solar irradiance and solar proxy data. Astronomy & AstroPhysics, 533, A29.

Once the process does not lead to significant variation anymore, the data are said consistent, and we execute the same procedure but with a rank two approximation and so on. The maximal considered rank, K, is determined by the user when launching the method. In practice, T. Dudock de Wit set it to 4 in his MATLAB code.

Let us see more precisely on which data matrix we will work. The original data matrix, $I(t, S)$, is composed of S columns, one columns per station and each line of the matrix corresponds to a given time of measurement. Remember that we want to exploit the between stations correlation as well as the time correlation. The columns give the information about between stations but not yet about the time correlaiton. To obtain both informations in the columns, we will not represent a station by one column but rather by D columns, such that if S_t^A is the signal coming from station A, the D columns representing this stations are shifted replicates of the original signal :

$$[S_t^A, S_{t+1}^A, ..., S_{t+D-1}^A]$$

The number D of replicates used is also determined by the user.
The signature of the Python method is given by :

```
interpolsvd_em(I,D,K)
```

In practice, it relies on some more practical parameters that are not detailed here.

To ensure that the translated version gives the same results as the original MATLAB version, we run both codes on the data available in the `Alldata.csv` file on moodle. We then check whether the obtained results are the same. Since it is indeed the case, both codes are well performing the same computations. The overall computing time on file `Alldata.csv` is around 2 minutes.

3 Time Complexity

In order to measure the execution time of the method, we run it for an input data matrix of increasing size. In practice, we increase here the number of considered time steps (or increasing number of lines in the matrix. This approach can be justify by the fact that every day the number of time steps increases while the number of stations remains more or less constant over time. Moreover, the number of lines in the matrix is far more important than the number of columns (33 333 verus 59).

Since the method is only valid for stations having at least 5% of non missing values, we need to select the stations with the most complete values to perform our analyzis. We selected here the five stations having the smallest number of missing values.

As we can see on figure 1, the Python code is slower than the MATLAB one but seems to have the same asymptotic behavior. It is to note that the current version of the Python code is not optimized for the python language, it is a simple translation of the MATLAB code.

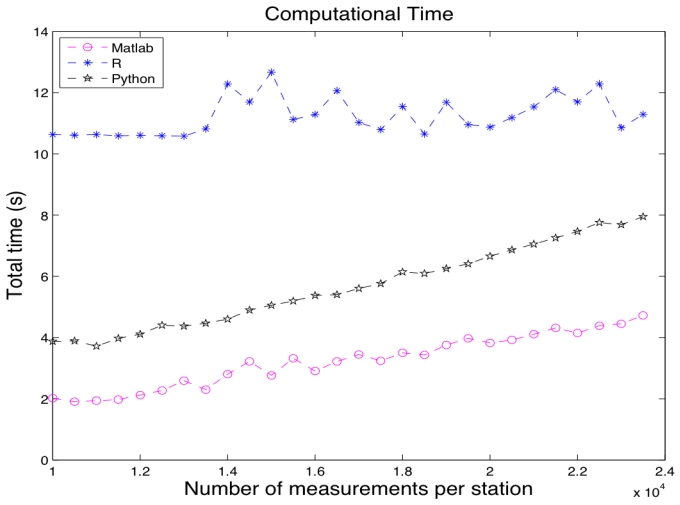


FIGURE 1 – Time Complexity

4 Parameters adjustment

We focus in this section on how to tune the parameter K which is the maximal rank taken into account in the iterative singular value decomposition procedure. Althought it is set to 4 by the author, we can wonder if this value is well fitted. In this section, we describe a cross-validation procedure allowing to select a right value for K. The intuitive idea is that a high value will keep more information but also more noise and increase the overfitting risk. We start by introducing the K-fold cross-validation procedure before presenting the results.

4.1 K-fold Cross-Validation

Cross-validation is a well-known procedure in machine learning to adjust the parameters of a method. The idea is to test the different values for the parameters and assess the performance of the model build on each value. To assess the performance of a model we need to test it on data that has not been involved in the computation procedure.

To this aim, we here randomly select some data, having a non missing value, and remove it. We then apply, for each value of the parameter, the method and compute the error between the filled value and the real value that we know in that case. We can compute the sum of squared errors on removed data. The lower this value, the better the value of the parameter. The problem is that we could be unlucky, the selected data to removed might be the easiest or the worst data of the dataset to predict. To avoid this problem and ensure that each single data has been part of an assessment procedure, we randomly split the original dataset in let's say 10 groups. We repeat the procedure for each single group and compute the Mean Squared Error (MSE) over all the 10 groups. This gives a more objective way of assessing the different values of the parameter.

The major disadvantage of this 10-fold cross-validation procedure is the computational time. Indeed, for 10 groups and 10 different values of the parameter, we will execute the method $10 \times 10 = 100$ times. Moreover, in our case, the execution time grows with the maximal considered K. It means

that assessing larger and larger values of K will lead to longer and longer computational time.

The pseudo-code of the procedure is presented below :

Algorithm 1 Cross-Validation

for each Validation Fold, j **do**
    replace the data of Fold j by NaN
    for each value of K, k **do**
        Fill missing value, using the method with K=k
        Compute SE_j^k
Compute $MSE_k = \frac{1}{10}\sum_{j=1}^{10} SE_j^k$

Althought this procedure allows to select the most appropriate value of K, it gives no clue about the overall accuracy of the method. What is the error we should expect on filled data ?

An estimation of the accuracy can be obtained by performing what we can call a cross-test procedure. A naive idea would be to select the best value for K via the cross-validation procedure, then remove some random data and apply the method with the selected K. The error could be estimated by the error commited on these values. However, we need to remember that all values where used for the cross-validation procedure, even the one that we later remove for assessing the accuracy. Therefore, even if the random data selected seem neutral, they are not. To assess the accuracy of a method we need data that have never been seen in any way by the method. This can be done by a cross-test procedure described in pseudo-code .This is again a 10-Fold procedure.

Algorithm 2 Performance evaluation

for each Test Fold, i **do**
    replace the data of Fold i by NaN
    for each Validation Fold, j **do**
        replace the data of Fold j by NaN
        for each value of K, k **do**
            Fill missing value, using the method with K=k
            Compute SE_j^k
    Compute the mean over all j of the MSE_k^j and deduce the optimal value of K, k
    Fill the gaps using the method with K=k
    Compute SE_i
Generalized Error = $\frac{1}{10}\sum_{i=1}^{10} SE_i$

As we can see, for each test fold, after removing the corresponding data, we select the most appropriate K based on the remaining data. It is possible to obtain different optimal values for K on different test folds. This cross-test procedure assesses the performance of the whole adjustement process : select the best K and fill the gaps.

Eventually, the computational time is drastically increased when performing a cross-test procedure. We have to fill the gaps $10 \times 10 \times 10$ times. Other testing procedures could be developed. For example, we could run it only one time on randomly selected values. However, the estimator would

be of higher variance.

4.2 Results

The cross-validation procedure described above has been run for K going from 1 to 15. The results are presented on figure 2. The computational time to obtain those results is around 15 hours, showing the heaviness of the computation. The cross-test procedure was not performed due to too smalll computing capacity.

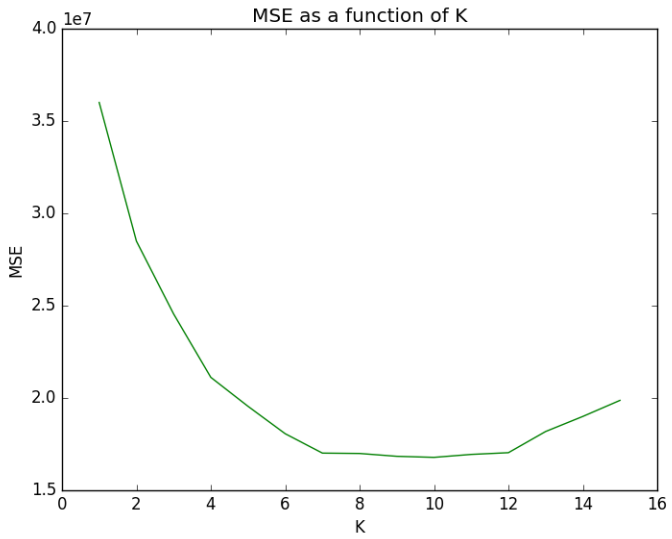


FIGURE 2 – Cross-Validation : K

As we can see on figure 2, the most appropriate value for K, based on the data seems to be 10.

5 Conclusion

Firstly, the main goal of this project was to translate the MATLAB code to Python, which has been done.

Secondly, we analyzed the time complexity of the code in both languages and showed that even if the Python code is not optimized for the Python language, it follows the same asymptotic time complexity than the MATLAB code.

Thirdly, we introduced a cross-validation procedure to select the most appropriate value for the maximal considered rank in the method, K. We found that considering low rank approximation up to rank 10 allow to get the smaller reconstruction error on our data. On top of it, we proposed a cross-test procedure to assess the performance of the method and have an idea of the error committed on filled data. However, due to the high computing load of the procedure, we were not able to perform it in reasonable time on our machine.

A Code

A.1 interpolsvd_em.py

```
#########################################################################
# Statistical Consulting Course - LSBA - UCL
#########################################################################
# VALUSUN Project
# Tranlation of the MATLAB code written by T. Dudock de Wit
#########################################################################
# Author: Caroline Sautelet
# Date: 3 May 2017
#########################################################################

# Import modules
import numpy as np
from numpy import linalg as LA
from scipy.signal import lfilter
import math
import scipy
import pandas as pd
import time
import numpy as np

#########################################################################
# interpolsvd_em
# fills gaps in monovariate or multivariate data
# by SVD-imputation (closely related to expectation-maximization)
# The method decomposes the data into two time scales, which are processed
# separately and then merged at the end. The cutoff time scale (nsmo) is
# expressed in number of samples. A gaussian filter is used for filtering.
# Monovariate data must be embedded first (nembed>1).
# In the initial data set, gaps are supposed to be filled in with NaNs
# Example for daily solar data with a cutoff at 81 days
# yf = interpolsvd_em(y,3,81,0,20,1);
#########################################################################
# INPUT:

# - y: array or matrix of data with gaps, gaps must be filled by NaN values.
# - nembed: embedding dimension  Default(0) is 1
#           must be > 1 for a monovariate data set
# - nsmo: cutoff time scale scale (in nr of samples). Set nsmo=0 if only one
#         single time scale is desired. Default is 0
# - ncomp: max number of iterations  Default (0) is 30
# - niter: number of significant components, to be specified for running
#          in automatic mode. Default (0) leads to manual selection
```

```
# OUTPUT:
# - yk: same data set as y, but with gaps filled
# - Ak: weight distrtibution of the SVD

##########################################################################
def interpolsvd_em(y,nembed = 1,nsmo = 0,ncomp = 0,niter=30):

    # Ensure inputs are ok or modify them
    if niter <1:
        niter = 30
    if nembed < 1:
        nembed = 1
    if nsmo < 1:
        nsmo = 0

    yshape = y.shape
    nrow_y = yshape[0]
    ncol_y = yshape[1]

    swap = 0
    # max cumulative energy (%) for selecting nr of significant components
    Emax = 95
    # iterations stop after relative energy change is < threshold
    threshold1 = 0.00001

    # Detect shape of input array and transpose if necessary in order
    # to have more rows than columns
    if ncol_y > 2*nrow_y:
        y = np.transpose(y)
        swap = 1
        tmp = nrow_y
        nrow_y = ncol_y
        ncol_y = tmp

    # Estimate mean and standard deviation and standardise
    ave_y = np.zeros((1,ncol_y))
    std_y = np.zeros((1,ncol_y))

    for i in range(0,ncol_y):
        w = np.nonzero(~np.isnan(y[:,i]))[0]

        if len(w)>1:
            ave_y[0][i] = y[w,i].mean()
            std_y[0][i] = np.std(y[w,i],ddof=1)

            y[:,i] = (y[:,i] - ave_y[0][i])/std_y[0][i]
```

```
    else:
        ave_y[0][i] = 0
        std_y[0][i] = 1

# perform some tests
for i in range(0,ncol_y):
    assert(sum(np.isnan(y[:,i])) < len(y[:,i])),"each column should have
assert(ncol_y >= 2 or nembed >=2),'embedding dimension must be >1 for mo
assert(ncomp<=ncol_y*nembed),"number of components cannot exceed: "+str(

# Embed record if necessary
if nembed > 1:
    x = embedy(y,nembed,1,0)
else:
    x = y.copy()

tmp = x.shape
nrowx = tmp[0]
ncolx = tmp[1]

# Weight each record according to the number of NaNs
# larger weight is given to record with fewer gaps
weight = np.zeros((1,ncolx))
for i in range(0,ncolx):
    n = sum(np.isnan(x[:,i]))
    weight[0][i] = (nrowx - n)/nrowx
weight = weight/max(max(weight))
weight = weight*weight
for i in range(0,ncolx):
    x[:,i] = x[:,i]*weight[0][i]



# first iteration: start by filling in gaps by linear interpolation
xi = np.zeros((nrowx,ncolx))
ave_x = np.zeros((1,ncolx))
t = x.shape

for i in range(0,ncolx):
    w = list(np.nonzero(~np.isnan(x[:,i]))[0])
    dummy = list(x[w,i])
    ave_x[0][i] = x[w,i].mean()
    dummy = [0] + dummy + [0]
    w = [-1] + w + [t[0]]
    f = scipy.interpolate.interp1d(w,dummy)
    xi[:,i] = f(range(0,nrowx))
```

```
xnew = x.copy()
xi = np.reshape(xi,(1,ncolx*nrowx))
xtmp = np.reshape(x,(1,ncolx*nrowx))
ind_NaN = np.nonzero(np.isnan(xtmp[0,:]))[0]
nNaN = len(ind_NaN)
for i in ind_NaN:
    xtmp[0][i] = xi[0][i]
xnew = np.reshape(xtmp,(nrowx,ncolx))

# subtract again the mean
xnew = xnew - (np.zeros((nrowx,1))+1)*xnew.mean(axis = 0)

# first estimate dominant mode nr 1
print('ncomp = 1')
err = [0]*niter
nNan = len(ind_NaN)
for i in range(0,niter):

    xfit = rank_reduce(xnew,1)
    xold = xnew.copy()
    xtmp = np.reshape(xnew,(1,ncolx*nrowx))
    xtmpfit = np.reshape(xfit,(1,ncolx*nrowx))
    xtmp[0][ind_NaN] = xtmpfit[0][ind_NaN]
    xnew = np.reshape(xtmp,(nrowx,ncolx))
    tmp = np.zeros((1,ncolx))
    tmp[0] = np.mean(xnew, axis=0)
    xnew = xnew - np.dot((np.zeros((nrowx,1))+1),tmp)
    xtmp = np.reshape(xnew,(1,ncolx*nrowx))
    xtmpold = np.reshape(xold,(1,ncolx*nrowx))

    e = xtmp[0][ind_NaN] - xtmpold[0][ind_NaN]

    err[i] = np.sqrt(sum(e*e)/nNan)


    print('ncomp = 1 iteration ' + str(i) + "rel.error " + str(err[i]))

    if err[i] < threshold1:
        break



# ask for number of components
if ncomp <1:
    U, S, Vh = LA.svd(xnew,full_matrices = False) # do the SVD
    Ak = np.diag(S) # singuar values of the SVD
    E = Ak*Ak
```

```
    E = 100*E/sum(E) # fraction amount of energy for each SVD mode
    nE = len(E)
    print("using "+str(ncomp2)+ " components out of " + str(nE))
else:
    ncomp2 = ncomp
    U, S, Vh = LA.svd(xnew,full_matrices = False)
    Ak = np.diag(S)

# now start the main loop
if nsmo > 1:
    for k in range(2,ncomp2+1):
        print("ncomp = " + str(k))
        for i in range(0,niter):
            xlp = smooth_gauss(xnew,nsmo)
            xhp = xnew - xlp

            xlp = rank_reduce(xlp,k)
            xhp = rank_reduce(xhp,k)

            xold = xnew.copy()
            xtmpnew = np.reshape(xnew,(1,nrowx*ncolx))
            xtmplp = np.reshape(xlp,(1,nrowx*ncolx))
            xtmphp = np.reshape(xhp,(1,nrowx*ncolx))

            xtmpnew[0][ind_NaN] = xtmplp[0][ind_NaN] + xtmphp[0][ind_NaN

            xnew = np.reshape(xtmpnew,(nrowx,ncolx))

            xnew = xnew - (np.zeros((nrowx,1))+1)*xnew.mean(axis = 0)
            xtmpnew = np.reshape(xnew,(1,nrowx*ncolx))
            xtmpold = np.reshape(xold,(1,nrowx*ncolx))

            e = xtmpnew[0][ind_NaN] - xtmpold[0][ind_NaN]
            err[i] = np.sqrt(sum(e*e)/nNaN)

            if err[i] < threshold1:
                print("error threshold reached")
                print("iteration :   " + str(i))
                break

else:
    for k in range(1,ncomp2+1):
        for i in range(0,niter):
            xhp = xnew.copy()
            xhp = rank_reduce(xhp,k)
            xold = xnew.copy()
            xtmpnew = reshape(xnew,(1,ncolx*nrowx))
```

```
                xtmpold = reshape(xold,(1,ncolx*nrowx))
                xtmphp = reshape(xhp,(1,ncolx*nrowx))

                xtmpnew[ind_NaN] = xhp[ind_NaN]

                xnew = np.reshape(ntmpnew,(nrowx,ncolx))

                xnew = xnew - (np.zeros((nrowx,1))+1)*xnew.mean(axis = 0)
                xtmpnew = reshape(xnew,(1,ncolx*nrowx))
                e = xtmpnew[ind_NaN] - xtmpold[ind_NaN]
                err[i] = np.sqrt(sum(e*e)/nNaN)

                if err[i] < threshold1:
                    break

    # recompose the data by adding the mean
    for i in range(0,ncolx):

        w = np.nonzero(~np.isnan(ORIGINAL[:,i]))[0]

        xnew[:,i] = xnew[:,i]/weight[0][i]
        xnew[:,i] = xnew[:,i] - xnew[w,i].mean() + ave_x[0][i]
    # de-embed the data
    if nembed > 1:
        yf = deembedy(xnew,ncol_y,1,0)
    else:
        yf = xnew

    # restore mean and stdev
    for i in range(0,ncol_y):
        yf[:,i] = yf[:,i]*std_y[0][i] + ave_y[0][i]

    if swap:
        yf = np.transpose(yf)

    return yf

##########################################################################
# embedy
# embeds a set of time series [x1 x2 ... xm] into an D-dimensional
# space by taking as state vectors the consecutive sequences
# z(k,:) = [x1(k) x1(k+T) ... x1(k+D*T-1) x2(k) x2(k+T) ...  xm(k+D*T-1)]
# for k=1 to n-D*T+1,  where n is the length of x and T is the embedding
# delay
##########################################################################
# INPUT:
# - x: input matrix (records are columns)                [n,m]
```

```
# - D: embedding dimension                                    [1,1]
# - T: embedding delay (integer>0), default is 1              [1,1]
# -displ: set displ=0 to prevent size from being displayed
#           default is with display                             [1,1]
# OUTPUT:
# - y: embedded matrix                                        [n-D+1, D*m]
#################################################################################
def embedy(x,D,T = 1,displ = 1):
    T = int(T)
    assert( T > 0),"embedding delay must be >0"
    xshape = x.shape
    nrow = xshape[0]
    ncol = xshape[1]
    nrowy = nrow - (D-1)*T

    assert(nrowy >= 2),"embedding dimension D must be < " + str((nrow-1)/T)
    if displ:
        print(["size of embedded matrix is :" + str(nrowy) + 'x' + str(D*nco
    y = np.zeros((nrowy,D*ncol))
    for j in range(0,ncol):
        for i in range(0,D):
            y[:,i+j*D] = x[np.array(range(0,nrowy)) + i*T,j]
    return y

#################################################################################
# deembedy
# deembeds an m-dimensional space which has been created
# by EMBEDY. It returns the averaged state vector x
#################################################################################
# INPUT:
# - y: embedded matrix                                          [n,m]
# - nset: # of data sets used to build y, default is 1
# - T: embedding delay (integer), default is 1
# - displ: set displ=0 to prevent matrix size from being displayed
#              default is with display
# OUTPUT:
# - x: averaged state vector                                    [n+(m-1)T,1]
#################################################################################
def deembedy(y,nset=1,T=1,displ=1):

    yshape = y.shape
    n1 = yshape[0]
    n2 = yshape[1]
    m = nset
    M = n2/nset
    n = n1 + (M-1)*T
    assert((M-int(M))== 0)," nset does not match the size of the data matrix
```

```
    M = int(M)
    n = int(n)
    if displ:
        print(['embedding dimension : '+ str(M)])
        print(['length of array     : '+str(n)])
    x = np.zeros((n,m))
    dx = x
    xx = np.zeros((n,M))
    for j in range(0,m):
        for i in range(0,M):
            a = np.zeros((i*T,1)).flatten()
            b = y[:,i+j*M]
            c = np.zeros(((M-i-1)*T,1)).flatten()
            xx[:,i] =np.concatenate([a,b,c])

        nor = np.array([int(i) for i in np.array(range(1,M*T+1))/T - 0.5]) -
        tmp = xx.mean(axis = 1)
        tmp2 = np.concatenate([nor,M*(np.zeros((1,n-2*M*T)).flatten()+1),nor
        x[:,j] = tmp/np.transpose(tmp2)
        x[:,j] *= M
    return x

##########################################################################
# rank_reduce
# generates a low-rank version version of matrix X by computing
# its SVD, and then then reconstructing X by keeping only its ncomp most
# important singular values. This function bypasses the (slower) full
# computation of the SVD by estimating the singular vectirs through
# diagonalisation of the covariance matrix of X
##########################################################################
# INPUT:
# - X : input matrix
# - ncomp: number of significant components

# OUTPUT:
# - Xfit: reduced rank version of X
##########################################################################
def rank_reduce(X,ncomp):
    xshape = X.shape
    nrow = xshape[0]
    ncol = xshape[1]

    if ncol>nrow:
        swap = 1
        X = np.transpose(X)
    else:
        swap = 0
```

```
    M = np.dot(np.transpose(X),X)
    D,V = LA.eig(M)


    order = np.argsort(D)
    order = order[::-1]
    D = D[order]
    V = V[:,order]
    K = V.copy()
    Vshape = V.shape
    col = Vshape[1]
    for i in range(0,col):
        K[:,i] = V[:,order[i]]
    V = K
    US = np.dot(X,V)
    xfit = np.dot(US[:,range(0,ncomp)],np.transpose(V[:,range(0,ncomp)]))

    if swap:
        xfit = np.transpose(xfit)
    return xfit

##########################################################################
# smooth_gauss
# smooths data in x using either a gaussian window
# which has a total width given by 2*ns+1 (each side contains ns data points
##########################################################################
# INPUT:
# - x: data matrix to smooth (smoothing is done columnwise)
# - ns: smoothing width

# OUTPUT:
# - y: smoothed data
##########################################################################
def smooth_gauss(x,ns):
    ns = math.ceil(ns)
    xshape = x.shape
    n = xshape[0]
    m = xshape[1]

    # ns must not exceed n/2
    ns = int(min(math.floor(float(n)/2 - 1),ns))

    if ns <= 1:
        return x
    w = np.transpose(np.array(range(-ns,ns+1))/ns)
    w = np.exp(-3*w*w)
    w = w/sum(w)
```

```
    z = np.zeros((1,m))

    z[0,:] = x[range(0,ns),:].mean(axis = 0)
    a = np.dot(np.zeros((ns,1))+1,z)
    z = np.zeros((1,m))
    z[0,:] = x[range(n-ns-1,n),:].mean(axis = 0)
    b = np.dot(np.zeros((ns,1))+1,z)
    x = np.concatenate([a,x],axis = 0)
    x = np.concatenate([x,b],axis = 0)
    y = np.zeros((n+2*ns,m))

    # do the convolution
    for i in range(0,m):
        y[:,i] = lfilter(w,1,x[:,i])
    return y[np.array(range(0,n))+2*ns,:]
```

A.2 script.py

```
##############################################################################
# Statistical Consulting Course - LSBA - UCL
##############################################################################
# VALUSUN Project
##############################################################################
# Author: Caroline Sautelet
# Date: 3 May 2017
##############################################################################
import interpolsvd_em as Method
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
import random
import time
##############################################################################
# Import data
##############################################################################

df=pd.read_csv('Alldata.csv', sep=',',header=0)

##############################################################################
# DATA PREPROCESSING
##############################################################################
I = np.where(df.SSN == -1)[0]
df.SSN[I] = float('NaN')
station_name = df['station']
SSN = df['SSN']
```

```
decdate = list(df.decdate)
res = df.pivot(columns='station', index='decdate',values='SSN')

EnoughData = dict()
T = float(res.shape[0])
rejected = 0
print("-----------------------------------------------")
print("Remove Stations having to few complete values")
for station in set(station_name):
    I = res[station].isnull().sum()
    if I >= 0.95*T:
        EnoughData[station] = False
        res = res.drop(station, 1)
        print("reject station: "  +station)
    else:
        EnoughData[station] = True

SSN = res.as_matrix()
nstations = SSN.shape[1]
nrec = SSN.shape[0]
##############################################################################
# Launch the method
##############################################################################
print("-----------------------------------------------")
print(" Start the method")
t = time.time()
z = Method.interpolsvd_em(np.sqrt(SSN+1),2,81,4)
z = z*z - 1
z[np.where(z<0)] = 0
elapsed = time.time()-t
print("-----------------------------------------------")
print("Done")
print("Elapsed time:  " + str(elapsed))
print("-----------------------------------------------")
np.savetxt("filledData.csv", z, delimiter=",",fmt='%.4f')

##############################################################################
#Time Complexity Analysis
##############################################################################
#select the 5 stations having the smallest # of gaps
tmp = SSN[9408:nrec,[12,15,2,19,47]]
timeForTrial = [0]*28

current = 10000 # initial number of time step used

for i in range(0,28):
```

```
    currentData = tmp[0:current,:]
    t = time.time()
    z = Method.interpolsvd_em(np.sqrt(currentData+1),2,81,4)
    z = z*z - 1
    timeForTrial[i] = time.time()-t
    current += 500 # add data for next trial

# Save results in text file
np.savetxt("pythonTimeComplexity.txt", np.array(timeForTrial), newline=" ")

##########################################################################
#   Cross-Validation Procedure
##########################################################################
# total nbr of values:
V = nrec*nstations
print("nrec = " + str(nrec))
for i in range(0,nstations):
    print(i)
    print(SSN[:,i].shape)
    tmp = nrec - sum(np.isnan(SSN[:,i]))
    print(tmp)
    print("Percentage = " + str(tmp/nrec))

tmp = SSN
shapeSSN = tmp.shape
Nrec = shapeSSN[0]
Nstations = shapeSSN[1]
# find indices of non NaN values
longLine = np.reshape(tmp,(1,Nrec*Nstations))
all_indices = list(range(0,Nrec*Nstations))
indices_non_NaN = np.where(~np.isnan(longLine[0]))
indices_non_NaN = list(indices_non_NaN[0])
nbrValues = len(indices_non_NaN)
n = math.floor(nbrValues*0.1)
permuted = indices_non_NaN
random.shuffle(permuted)
Error_K = np.zeros((5,10))
for K in range(11,16):
    print("K equal to " + str(K))
    for val in range(0,10):
        current_start = val*n
        if val < 9:
            current_end = (val+1)*n
        else:
            current_end = nbrValues
        print("Starting at " + str(current_start))
        print("Ending at " + str(current_end))
```

```
        Trial_SSN = longLine.copy()
        for i in range(current_start,current_end):
            Trial_SSN[0][permuted[i]] = float('NaN')
        Trial_matrix = np.reshape(Trial_SSN,(Nrec,Nstations))

        res = Method.interpolsvd_em(np.sqrt(Trial_matrix+1),nembed = 2,nsmo
        res = res*res - 1
        for i in range(0,Nrec):
            for j in range(0,Nstations):
                if res[i,j]<0:
                    res[i,j] = 0
        res = np.reshape(res,(1,Nrec*Nstations))
        # Compute Squared Error
        E = 0
        for i in range(current_start,current_end):
            E += (longLine[0][permuted[i]] - res[0][permuted[i]])*(longLine
        Error_K[K-11][val] = E
print(Error_K)
np.savetxt('Error_K.csv', Error_K, fmt='%1.3f')
```

OBSERVING SUNSPOT NUMBER

CUSUM charts as a tool for evaluation of stations performance

Abstract

This paper discusses the application of CUSUM charts for the sunspots data. The aim of this application is to help in quality control of particular observing stations. The technique is described and required modifications as data transformation and calculation of deviation are explained. The use of different parameters, as target, allowance value or control limits is discussed and illustrated with examples. Some indications for interpretation of CUSUM charts are given to facilitate the future work and an example of interpretation for Uccle, Locarno and Kanzelhöhe are given. Finally, R code is provided with the paper to enable quick visualisation of CUSUM charts for sunspots data.

Hanna Pawelec

Table of contents

Introduction

Sunspots are the darker areas on the photosphere of the Sun. Because of the contrast, they seem black when observed from Earth. Since 17th century and the invention of telescope, humans have been counting their number. The huge amount of data let to discover cycles with high and null number of spots and to understand the link between the Sun activity and different Earth phenomenon, for example, the climate change.

270 stations, situated mostly in Europe, have contributed to the solar data collection and 80-100 do it on a regular basis. Among those, the majority of observers are amateurs. Since 1981, the collected data are centralized by the Sunspot Index Data Center, located in The Royal Observatory of Belgium in Uccle (Brussels). The Uccle Observatory collects the everyday data on the sunspots number from around the world and watch over its quality.

Indeed, high variations in the number of observed sunspots exist between stations. On the one hand, these differences are due to the equipment quality (optical resolution). They are supposed to be relatively stable in the time, a station may, for example, all the time measure a few percent less spots than others. On the other hand, some differences are a consequence of human factor: the interpretation of definition of pore, sunspot or sunspots group may differ between observers, it may especially be true for new or substituting observers; optical capacities or distractions may also influence the final daily number observed.

Control the quality of data of each station is thus crucial to assure the quality of the estimated international sunspot number that is communicated to scientists working on the sunspot number. CUSUM charts are a common solution in the industrial world to track the quality control on a period of time. This paper investigates the application of this method to the observations of sunspots and whether a CUSUM chart could be a practical tool to control the stations performance.

Firstly, the use and the purpose of CUSUM charts will be explained. The data transformation, necessary for the further analyses, will be highlighted. Next, three methods of calculating stations bias and the choice of the chart parameters will be discussed. Finally, a few examples of the application will be presented and interpreted. R code to generate a CUSUM chart adapted to the sunspots observations is provided in the annex of this paper.

1) What is a CUSUM chart?

The CUSUM (from cumulative sum) chart is a statistical quality control method developed to detect changes in industrial processes. The aim was to measure deviations from a target (certain size of a piece in a factory, a chemical density etc.) corresponding to the mean of the series. As the deviations are accumulated, even minor changes can be detected.

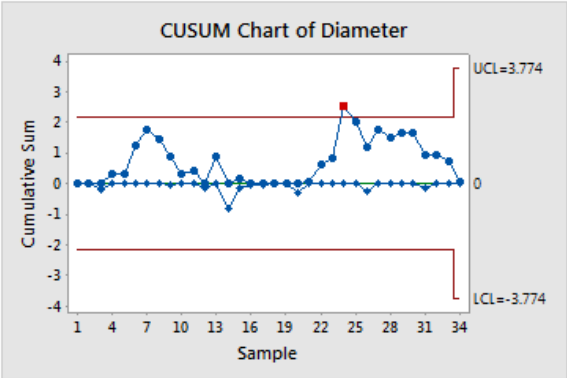
There are two types of CUSUM charts: one sided and two-sided. To plot the first one, an upper CUSUM value, measuring deviations above the target, and a lower CUSUM value, measuring deviations below the target, need to be calculated. The control limits indicate if some observations are out of control. In the two-sided CUSUM, both upper and lower values are combined and plotted as one. A V-mask method was designed for two-sided CUSUM to detect if process is out of control.

The basic formula to calculate CUSUM values is as follows:

For the upper values: $C^{+}_{i} = \max[0, x_i - (T + K) + C^{+}_{i-1}]$
For the lower values: $C^{-}_{i} = \min[0, x_i - (T - K) - C^{-}_{i-1}]$

Where T is the target (often expressed as μ_o) and K is the allowance value or reference value. Deviations from the target are accumulated only when their values exceed the reference value. It is generally equal to the half of the shift (in standard deviations) that is supposed to be detected (so often one standard deviation divided by two). The control limits, noted with the letter H, are generally fixed as 5 standard deviations.

Once, the values are calculated, the lower line and the upper line as well as the control limits can be plotted. The observations that are outside those two limit lines are out of control. An example of a CUSUM chart is shown below.



Source: http://support.minitab.com/en-us/minitab/17/topic-library/quality-tools/control-charts/understanding-time-weighted-control-charts/what-is-a-cusum-chart/

2) Plotting CUSUM charts with R

In this paper, the R programming language is used to calculate and to plot CUSUM charts. Although different packages exist to plot CUSUM charts, here, the qcc package for quality control charting and statistical process control has be chosen[1]. This package offers "cusum" function calculating CUSUM values and generating one-sided CUSUM chart with the possibility to choose different parameters.

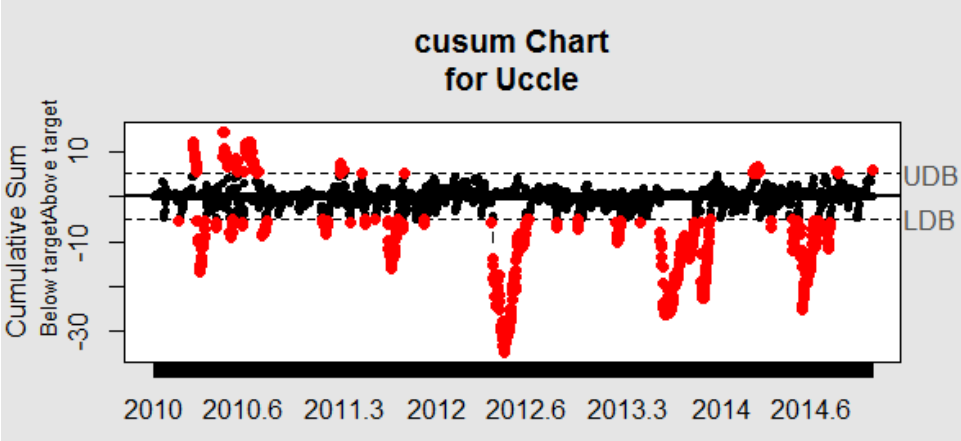
The formula used to calculate CUSUM values is slightly different from the one described above. Indeed, the R function calculates the standardized CUSUM. This is obtained by dividing the deviations by the process standard deviation. The formula could then be write as follows:

For the upper values: $C^{+}_i = \max(0, C^{+}_{i-1} + \frac{(x_i - T)}{sd} - K)$

For the lower values: $C^{-}_i = \min(0, C^{-}_{i-1} + \frac{(x_i - T)}{sd} + K)$

The example below illustrates the result that can be obtained using R "cusum" function applied to the Uccle observations between 2010 and 2015. Observations out of control are marked in red. The exact values of lower and upper CUSUM can be obtained using $pos and $neg call.

[1] Scrucca, L. (2004). qcc: an R package for quality control charting and statistical process control. R News 4/1, 11-17.

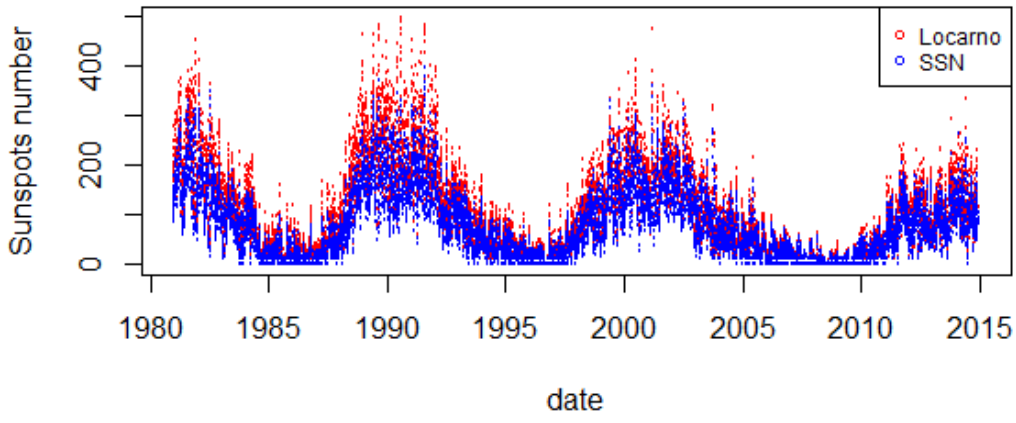


3) Applying CUSUM charts to sunspot number

The sunspots observations are not a typical example of industrial quality control problem. Thus, some modifications and adaptations need to be made to apply CUSUM charts in order to analyse a particular station's performance.

a) Data preparation and transformation

The aim of applying CUSUM chart to a particular station observing sunspots is to detect irregularities in observations. It is done by comparing the station's observations with the estimated sunspots number (called further SSN), supposing that this one is true. The simplest way to obtain this estimation is to take the average or median of daily observations from every station but more complex methods exist. The basic visualisation of sunspots observations from Locarno and from the estimated SSN is shown below.



Contrary to the industrial process where target is generally constant, the sunspot number has its intrinsic variation following the solar activity. The number of sunspots is sometimes equal to zero while sometimes it is counted in hundreds. Therefore, it is not possible to choose a constant as a target. The solution is to take the deviation of the station's observation from the estimated sunspot number. Thus, the target should be equal to zero (no deviation) but this is not so evident (the question will be discussed further).

Moreover, the variations of those deviations are heteroscedastic. When there are no sunspots (low cycle), the deviation is very low or null and when there are hundreds of sunspots (high cycle), the deviation is very high (daily differences between stations can sometimes be counted in hundreds). To reduce the effect of the heteroscedasticity, a data transformation should be applied. Variance stabilization can be obtained through Anscombe transform as the data follow kind of Poisson distribution. In the following of this paper, the simplified transformation will be applied: the square root transformation.

Another problem when working with sunspot data are missing values. Indeed, the cloudy weather sometimes makes it impossible to observe the number of sunspots. Thus, the percentage of missing values varies according to the stations geographical localisation but no station has complete data. As CUSUM values are calculated based on the successive values, it should be used with data that have no missing values. The missing observations from data used in this paper were replaced by the general trend scaled by the individual station factor and an intercept with the fit done on the Anscombe scale.

b) Calculating the deviation of one station's observations from the estimated SSN

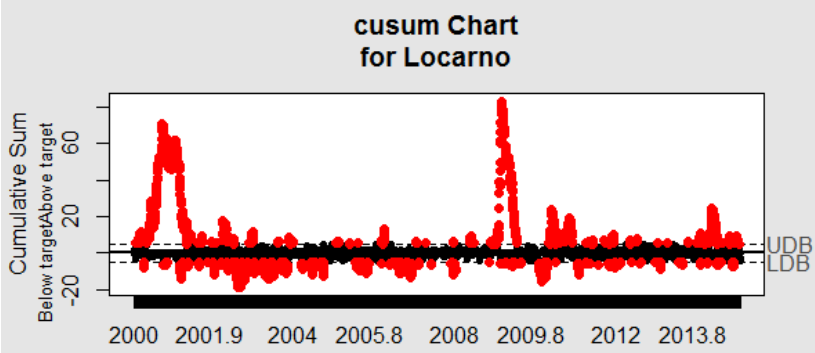
There are different methods to calculate the deviation of one station's observations from the estimated SSN. Here, they are all applied after square root transformation as described before. Three different methods will be discussed but this list does not pretend to be exhaustive.

The main aim of those methods is to distinguish two sources of station's deviation. The first one is a permanent deviation that might be due to the quality of equipment. Some stations are measuring constantly 20% more or 20% less sunspots than there are in the estimated SSN while others are very close to the estimated SSN. The application of CUSUM chart is not

aimed here to explain this kind of variation. The second type of deviation is more temporary or starts only in some point in the time. It might be, for example, a temporary change of observer for a less qualified one that would observe for some time more or less sunspots than this station does usually. The utility of a CUSUM chart for sunspot observation is more linked to detecting irregularities regarding station usual performance than comparing it with the estimated SSN. To extract the constant deviation, the estimated SSN is scaled by a certain factor whose value depends on method chosen.

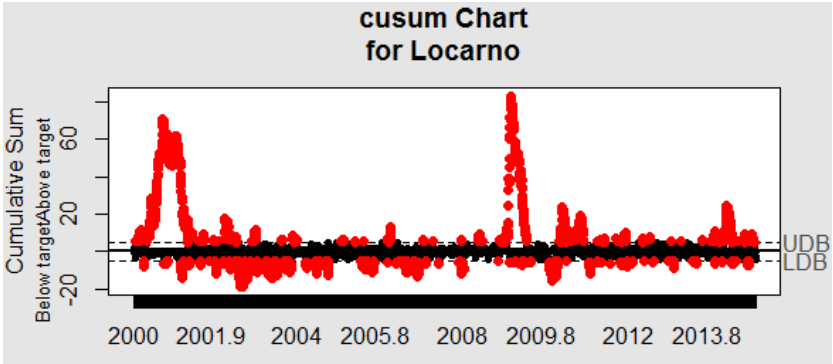
Method 1: regression residuals

In the first method, a simple linear regression without intercept is run with the station's observations as dependent variable and the estimated SSN as independent variable. The residuals from regression are then used to plot the CUSUM chart with the target either equal to 0 or equal to the mean of residuals.



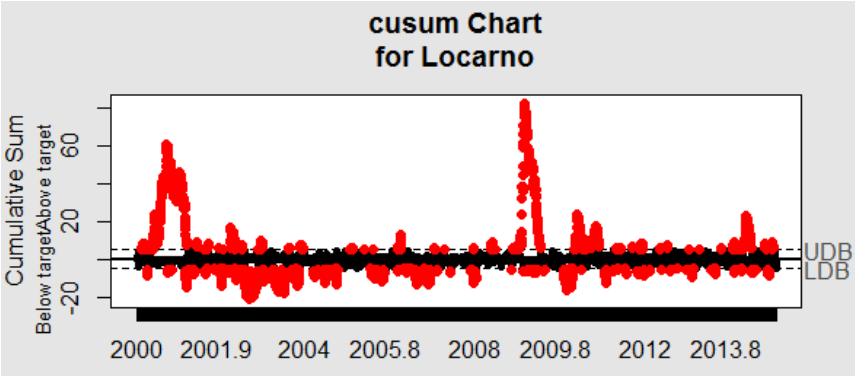
Method 2: regression coefficient

The second method is very similar to the first one, a simple linear regression without intercept is also run with those two variables, but instead of taking the residuals, the regression coefficient is used to scale the estimated SSN. Then, the difference between the station's observations and the scaled SSN is calculated and used to plot the CUSUM chart.



Method 3: sums ratio

In the third method, a sum of all station's observations and sum of all estimated SSN (in both cases, after the root square transformation) are calculated. The global factor is calculated as the ratio of those two sums (sum of station divided by the sum of SSN) and used to scale the estimated SSN. Again, the difference between the station's observations and the scaled SSN is calculated and used to plot the CUSUM chart. This method is less sensible to the choice of target (between 0 or the mean). Moreover, the global factor indicates the proportion of station's observations regarding the estimated SSN (constant deviation) but it can also be taken from the regression coefficient in previous methods.



c) Choice of parameters

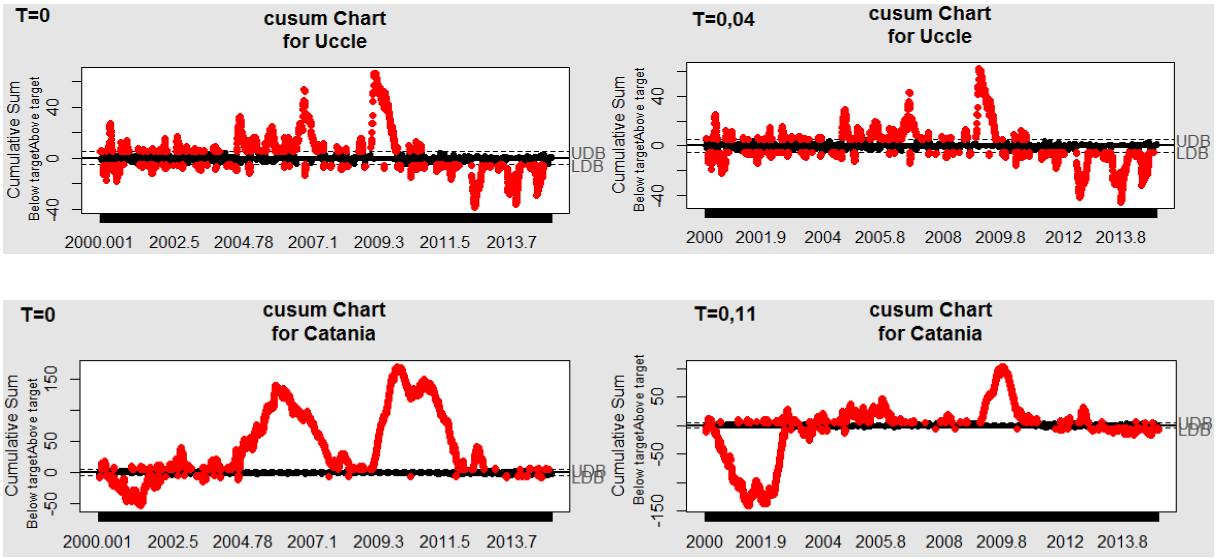
There are different parameters to choose when plotting CUSUM charts. The choice alters the appearance of the charts.

Target

The choice of the target may sometimes have an impact on the appearance of the CUSUM chart. Choosing zero as the target may seem obvious. Indeed, when studying a deviation, it is generally expected to be null if model is perfect. However, behind the choice of target equal to zero there is a strong hypothesis that the estimated SSN is true and that station's observations should be equal to those estimated ones. The choice of the mean seems then more neutral.

The importance of this choice is illustrated by the following examples. First two represent CUSUM charts for Uccle for 2000-2015 (residuals method). On the left, the target was forced to be zero while on the right the target was equal to the mean of deviations. As the mean is

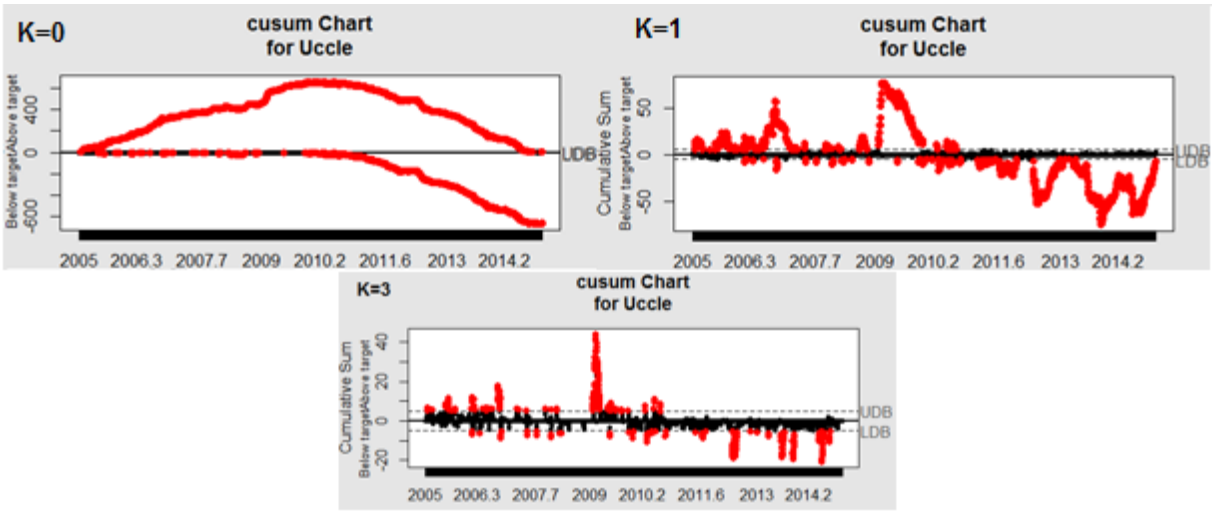
rather close to zero, both graphs are very similar and the only slight difference is visible around 2006-2007 when the peak is higher when target is equal to zero. However, two following charts are quite different. They represent observations from 2000-2015 from Catania (residuals method). While the same drifts are visible, their scale differs. There is no such difference when using sums ratio method and both charts look like the one on the left.



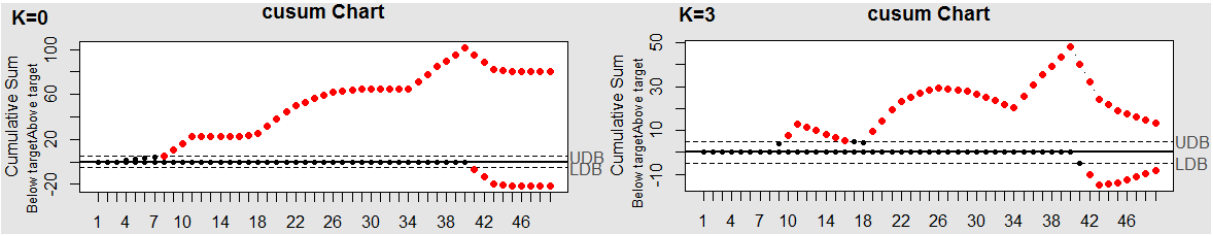
Allowance value

As explained before, allowance value corresponds to the half of the shift that is expected to be detected. In the examples shown before, the shift was fixed at 1 (default value) so from every standardized deviation 0,5 (=0,5 standard deviation) was extracted. This method is commonly used as it emphasizes important drifts.

In the example below, 3 different shift values are compared. Regarding the high number of observations out of control (in red), a higher allowance value than 1 could be considered but it could also be corrected with the control limits (see further).



The advantage of using no shift is an easier interpretation what is illustrated below. When no allowance value is used, a positive slope indicates deviation superior to 0, a negative slope indicates deviation below 0 and horizontal slope indicates no deviation. When a shift is added, the slope is negative when there is no deviation at all. Thus, when using an allowance value, the positive slope should only be observed in upper line and the negative slope in lower line. On the other side, the value of CUSUM (number of standard deviations above/below the target) makes more sense when using an allowance value because it only describes a single drift and does not refer to the whole cumulated deviation. Thus, both charts, with and without an allowance value, could be complementary. The first one (K=0) illustrated better the evolution in time while the second one (K>0) helps better to detect shifts and drifts.



Control limits

By default, the control limits are fixed as 5 standard deviations below and above the target. As numerous observations are out of control, a higher threshold could be considered.

Time period

Shorter time periods are more visible on CUSUM charts. Scaling and global factor are calculated based on the period chosen so the choice may change the appearance of the chart. A longer period permits to observe slow shifts.

Expressing CUSUM values as percentage

Expressing the CUSUM values as percentage has been considered. However, it would only make sense during the high cycles (lots of sunspots) so it is not the best option for a general model. Moreover, the CUSUM charts presented here are plotted based on standardized values so they are comparable. Thus, the use of percentage would not only be complex but seems also unnecessary.

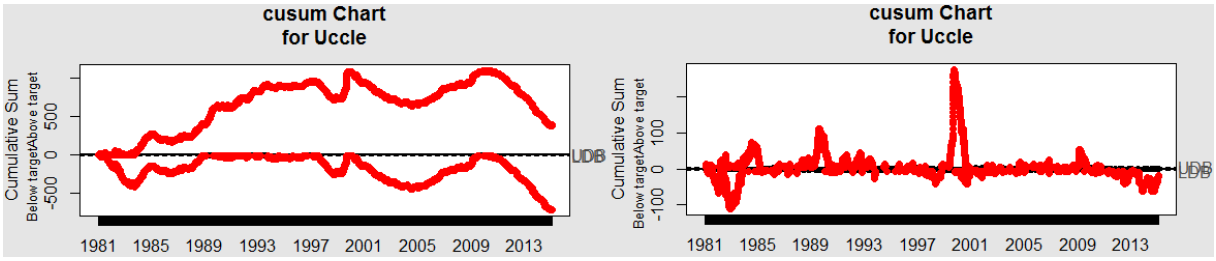
d) Interpreting a CUSUM chart

Because of the scaling of the estimated SSN, the interpretation of CUSUM chart is not obvious. It might not be true that when a deviation is positive, the station's observations are above the estimated SSN. Thus, the interpretation refers more to the station's usual performance. When a deviation is positive, the station is observing more than usually and when a deviation is negative, the station is observing less than usually.

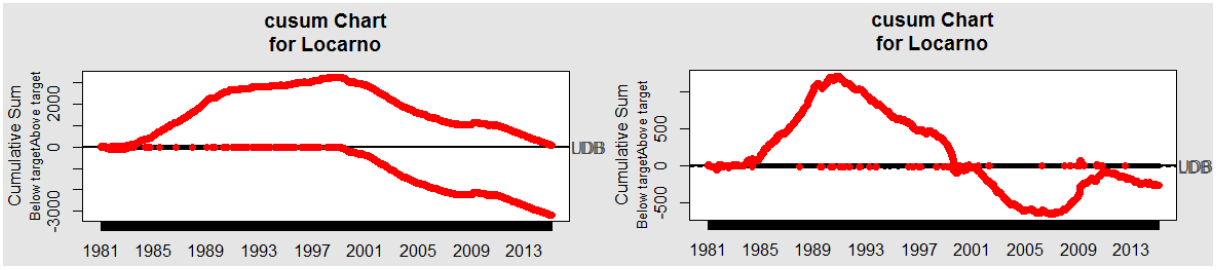
The strength of the slope indicates the magnitude of the deviation. A strong slope indicates an important deviation while weaker slope indicates a small deviation. As the CUSUM chart represents a cumulative sum of deviations, small deviations that are constant for a certain period of time may become very visible on the chart.

4) CUSUM charts for Uccle, Locarno and Kanzelhöhe

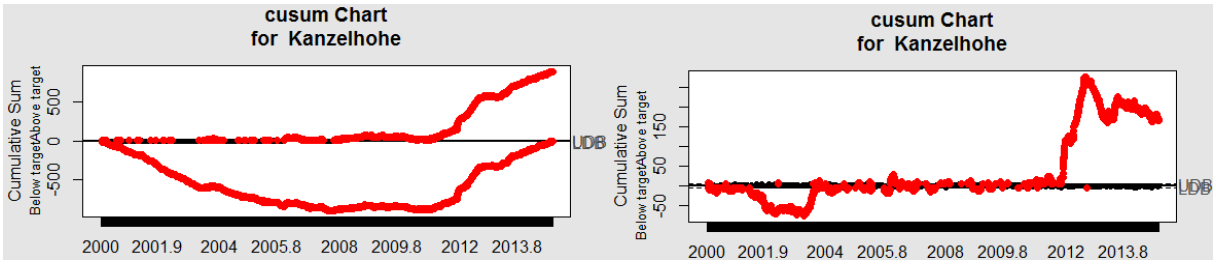
The CUSUM charts below were calculated based on the sums ratio method with target equal to 0 and K equal respectively to 0 and 1. In the beginning of 80s, observations were underestimated regarding the later observations. Since the end on 80s a progressive growth in estimations can be seen on the left chart followed by a period of stable observations. The slope is rather weak what explains why it is less visible on the second chart with an allowance value (0,5 standard deviation). End on 90s and the beginning of 200 is marked by an important shift that started by a slight underestimation followed by a very strong positive peak. It is then recovered progressively to relapse importantly since 2011, a trend that is continuing.



Deviations from Uccle observations are rather variable with one very important shift. However, when comparing the CUSUM values with other stations, they are quite low. The highest peak in Uccle is a little bit above 200 standard deviations while the highest peak in Locarno is above 1000. On the other hand, Locarno station after overestimations (regarding its own performance) has known a long period of stable observations. However, since 2000, their observations are progressively underestimated. Regarding the constant deviation, the Uccle global factor is 0,99 (positive deviations correspond thus to the observations above the estimated SSN) while the Locarno global factor is 1,15.



Choosing a shorter period of time enables to see more detail. Below two charts for Kanzelhöhe are presented. They show a period of underestimation followed by a long period of stable observations. Since 2011 an important upward drift is observed which is now slowly progressing. Unsurprisingly, around 2011 a new high solar cycle starts. Even though a root square transformation is applied, deviations are generally observed during high cycles. The Kanzelhöhe factor is 1,05.



Conclusion

This paper explains the applicability of CUSUM charts to the sunspots observations in order to detect shifts and drifts in observations of particular stations. Modifications required, as data transformation or calculations of deviation, were explained and different parameters of CUSUM charts were discussed. However, the final choice of methods and parameters depends on the needs of each user. The R code provided enables to visualize quickly different charts and to evaluate the rapidly the effect of parameters and method chosen. An example and indications for interpretation have been given but it remains open as the choice of parameters is large.

The CUSUM chart enables to perceive the periods when a particular station recedes from its usual performance. The question remains what should be done with the observations out of control and whether they should be removed, corrected or left intact.

Annexe: R code

Advice regarding the use of the code:

This code provides a function adapted to the sunspots observations. It requires the installation of the package “qcc”. 4 arguments need to be indicated:

data : the database with the sunspots observations

SSNVariable : name of the variable (between “”) containing the estimated SSN

stationVariable : name of the variable containing the station’s observations

timeVariable : name of the variable containing date information.

Other arguments are optional:

startTime : starting year of the observations to plot (default is 1981)

endTime : end year of the observations to plot (default is 2015)

target0 : if TRUE then target is equal to zero, if false then target is equal to the mean (default is T)

method : method for calculating deviations (choose one between "residuals", "sumRatio", "coefficient"

shift : shift that is expected to be detected in standard deviation (the allowance value will be equal to the half of shift chosen, default value is 1)

interval : the measure of control limits in standard deviations (default is 5).

statistics : if TRUE, statistics are printed below the chart (default is FALSE)

stationName: name of the station to be plotted above the chart (between “”)

```
# work directory  to fill in
setwd("")

#data importation
Sunspots <-
  read.table("SSNfilledChris.csv", header=TRUE, sep=",",
na.strings="NA", dec=".", strip.white=TRUE)

#Library
#for the first use: install.packages("qcc")
library(qcc)

#function

SSNcusum=function(data, SSNVariable, stationVariable, timeVariable,
startTime=1981, endTime=2015, target0=T, method=c("residuals",
"sumRatio", "coefficient"), shift=1, interval=5, statistics=F,
stationName){

  #choice of the period of time and key variables
  time=as.name(timeVariable)
  dataB=data[which(data$time>=startTime&data$time<endTime),]
  date=data[which(data$time>=startTime&data$time<endTime),timeVariable]
```

```
station=data[which(data$time>=startTime&data$time<endTime),stationVaria
ble]
  SSN=data[which(data$time>=startTime&data$time<endTime),SSNVariable]
  SSN[which(SSN<0)]=0


  #residuals method
  if(method=="residuals"){
  regression=lm(sqrt(station)~sqrt(SSN)- 1) #calculating résiduals
    if(target0==T) {
    cusumChart=cusum(regression$residuals, center=0,
std.dev=sd.xbar.one(regression$residuals), se.shift=shift,
decision.interval=interval, label=round(date, 1), add.stats=statistics,
data.name=stationName) #cusum chart
    summary(cusumChart)
    return(paste("method", method, ';', "Factor:",
round(regression$coefficiens, 2)))}
    if(target0==F)  {
    cusumChart=cusum(regression$residuals, se.shift=shift,
decision.interval=interval, label=round(date, 1), add.stats=statistics,
data.name=stationName) #cusum chart
    summary(cusumChart)
    return(paste("method:", method, ';', "Factor:",
round(regression$coefficiens, 2)))}
  }

  #sumRatio method
  if(method=="sumRatio"){
    factor = sum(sqrt(station))/sum(sqrt(SSN)) #calculating the global
factor
    bias=sqrt(station)-sqrt(SSN)*factor #rescaling and calculating the
bias

    if(target0==T) {
      cusumChart=cusum(bias, center=0, std.dev=sd.xbar.one(bias),
se.shift=shift, decision.interval=interval, label=round(date, 1),
add.stats=statistics, data.name=stationName) #cusum chart
      summary(cusumChart)
      return(paste("Method:", method, ';', "Factor:", round(factor,
2)))}

    if(target0==F)  {
      cusumChart=cusum(bias, se.shift=shift,
decision.interval=interval, label=round(date, 1), add.stats=statistics,
data.name=stationName) #cusum chart
      summary(cusumChart)
      return(paste("Method:", method, ';', "Factor:", round(factor,
2)))}
  }

  #coefficient method

  if(method=="coefficient"){
    regression=lm(sqrt(station)~sqrt(SSN)- 1) #calculating the
coefficient
    bias = sqrt(station) - sqrt(SSN)*regression$coefficients #rescaling
and calculating the bias
```

```
    if(target0==T) {
      cusumChart=cusum(bias, center=0, std.dev=sd.xbar.one(bias),
se.shift=shift, decision.interval=interval, label=round(date, 1),
add.stats=statistics, data.name=stationName) #cusum chart
      summary(cusumChart)
      return(paste("Method:", method, ';', "Factor:",
round(regression$coefficients, 2)))}
    if(target0==F)  {
      cusumChart=cusum(bias, se.shift=shift,
decision.interval=interval, label=round(date, 1), add.stats=statistics,
data.name=stationName) #cusum chart
      summary(cusumChart)
      return(paste("Method:", method, ';', "Factor:",
round(regression$coefficients, 2)))}
  }
  if(method!="residuals" & method!="coefficient" & method!="sumRatio")
{
    print("Incorrect method, choose between 'residuals', 'coefficient'
and 'sumRatio'")
  }
}

#example of calling of the function
SSNcusum(data=Sunspots, SSNVariable="SSN",
stationVariable="wnUC_d.txt", timeVariable="times", startTime=2000,
endTime=2015,target0=T, method="sumRatio", shift=0, interval=5,
statistics=F, stationName = " Uccle")
```

RAPPORT

The Sunspot numbers

Royal Observatory of Belgium

Ilham Sebban 15/05/2017

Table des matières

Résumé

La production du nombre de tâches solaires international (ISN) est passée de l'Observatoire de Zurich à l'Observatoire Royal de Belgique en 1981, en construisant une série ISN à partir des observations fournies par des stations individuelles du réseau mondial SILSO.

Ce travail se veut mettre en lumière la qualité des données communiquées par le réseau des observateurs, et ce, en tenant compte de l'influence du facteur humain sur les données observées et plus précisément de l'erreur de l'observateur. Pour cette fin, une exploration des données sous l'angle des données manquantes et des variances a été conjuguée à la constitution des cartes de contrôle chronologiques CUSUM.

Les cartes de contrôle ont été construites sur la base de trois approches d'estimation de l'erreur de l'observateur, et ce, au profit de la station de référence Locarno et de la station Uccle. Il en a résulté que l'allure des cartes de contrôle demeure la même pour les trois scénarii d'estimation. Aussi, on a constaté que les deux stations ont connu des périodes d'instabilité différentes, sauf que Locarno se caractérise par la présence de drifts qui s'étalent sur des périodes de temps assez longues, et Uccle par des shifts accentués qui présentent des écarts brusques des limites de contrôle en de courtes périodes de temps.

Il est à noter que sur les 40 stations étudiées, juste 18 sont actives depuis 2010, avec des variances hétérogènes, mais particulièrement faible en 2010. Par ailleurs, les cartes CUSUM afférentes aux stations actives ayant moins de 50% de données manquantes en 2010-2014 ont révélé que toutes les stations en question étaient instables en 2010 excepté les stations HU et HE.

Finalement, la conjugaison des trois critères d'appréciation des stations a permis d'identifier des stations actives qu'on peut qualifier de crédibles, tel est le cas pour la station EB marquée par une stabilité des données depuis début 2012. Et vice versa, des stations qui n'ont pas réussi à maintenir sous contrôle leurs observations durant toute la période 2010-2014, tels que les stations FRI et QU.

1. Contexte

Depuis 1981, l'Observatoire royal de Belgique se charge de mettre à disposition le nombre de tâches solaires international (ISN). Ce dernier est calculé en fonction de données provenant de 30 pays, principalement européens. 270 stations ont contribué à l'indice et 80 contribuent régulièrement. La majorité des observateurs sont des amateurs. Ceci étant, le facteur humain influence fortement les observations et constitue une source d'erreur.

Le ISN est largement utilisé comme principal indice solaire de référence sur lequel se fondent des recherches scientifiques et des centaines d'études publiées dans divers domaines de la science. Ainsi, il est important de contrôler en permanence la qualité des données, en analysant la stabilité des stations individuelles dans le temps.

Ce travail se veut pour objectif de déterminer un ensemble de critères pour l'évaluation de la stabilité des stations et d'identifier les points d'arrêt éventuels dans le temps.

2. Méthodologie adoptée

L'analyse a porté sur une base de données comportant le ISN et les observations par station, il s'agit de 59 observations provenant de 40 stations.

De prime abord, pour apprécier la stabilité des données à travers le temps et identifier les points d'arrêt, on a fait le choix d'appliquer une carte de contrôle CUSUM[1].

Le principe de la carte CUSUM se définit par la construction d'une carte capable de détecter des petites dérives en utilisant des informations antérieures pour la décision et pas uniquement le dernier point. Il s'agit de représenter la somme des différences à la cible depuis la dernière alarme : les sommes cumulées.

Voulant établir une carte de contrôle mettant en exergue l'erreur de l'observateur, on a procédé à une estimation des résidus selon trois méthodes.

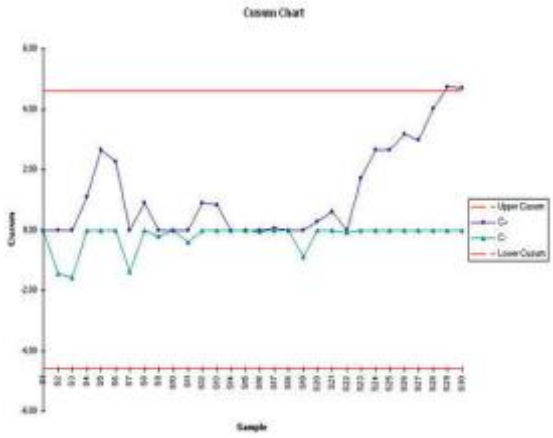
- Méthode1 : établir une régression basée sur les données transformées via la transformation Anscomb et en déduire les résidus, à savoir :
  - $2/\alpha\sqrt{Yt} +3/ 8*\alpha^2$ (pour les stations)
  - $2/\alpha\sqrt{Xt} +3/ 8*\alpha^2$ (pour le ISN)
- Méthode 2 : réaliser une régression basée sur la racine carrée des données brutes et en déduire les résidus ;
- Méthode 3 : calcul des résidus selon la formule suivante :
  - Résidu= $\sqrt{Yt} - \alpha *\sqrt{SSN}$
  - α = médiane ($\sqrt{Yt} / \sqrt{SSN}$)

Il importe de signaler qu'avant de faire la régression un lien positif entre le ISN et les observations par station a été vérifié et confirmé.

[1] Logiciel utilisé : QI Macros SPC Software pour Excel

Le diagramme CUSUM se compose des éléments suivants :

- Une valeur cible
- Le paramètre h : représente les limites inférieure et supérieure de contrôle : Il s'agit de 4*Ecart-type ; en effet, la carte CUSUM déclare que le processus devient instable une fois qu'il est au-delà 4 fois l'écart type de la moyenne ;
- Les lignes C+ et C- qui représentent la somme cumulative des écarts des valeurs successives à partir de la cible[2].



Graphe 1 : exemple d'une carte CUSUM

Les formules de calcul se présentent comme suit :

- C+=Xvar+(valeur cible+k)+cumul
- C-=Xvar-(valeur cible+k)+cumul
- K=Ecart-type/2

Quant à la définition de la valeur cible, on a fait le choix de maintenir la moyenne. Une réflexion logique qui peut se justifier pourrait nous inciter à fixer la valeur cible à 0, voulant avoir une erreur de l'observateur nulle. Néanmoins, cette valeur influence fortement l'allure de la carte CUSUM, et cette option ne nous permettra pas d'avoir une carte neutre et balancée.

Le processus ci-dessus a été exécuté au profit de la station Uccle et la station Locarno[3] pour la période allant de 1981 à 2014, et ce, sur la base des trois estimations de l'erreur en vue de constater des différences éventuelles.

Par la suite, et dans le but d'avoir une visibilité actualisée sur une période récente, une appréciation de la fiabilité du processus sur la phase réduite de 2010-2014 a été réalisée.

L'objectif de ce travail étant de mettre en lumière la fiabilité des stations, on a procédé dans un deuxième temps à une exploration globale des données brutes observées. Ladite exploration a été faite sous l'angle des données manquantes et de la variance pour la période allant de janvier 1981 à décembre 2014.

Il en a résulté l'identification des stations qui ont été actives durant les 5 dernières années, et une appréciation de la stabilité des données y afférentes a été mise en exergue via la carte CUSUM sur la période 2010-2014, et ce, sur la base des résidus calculés à travers la méthode 3.

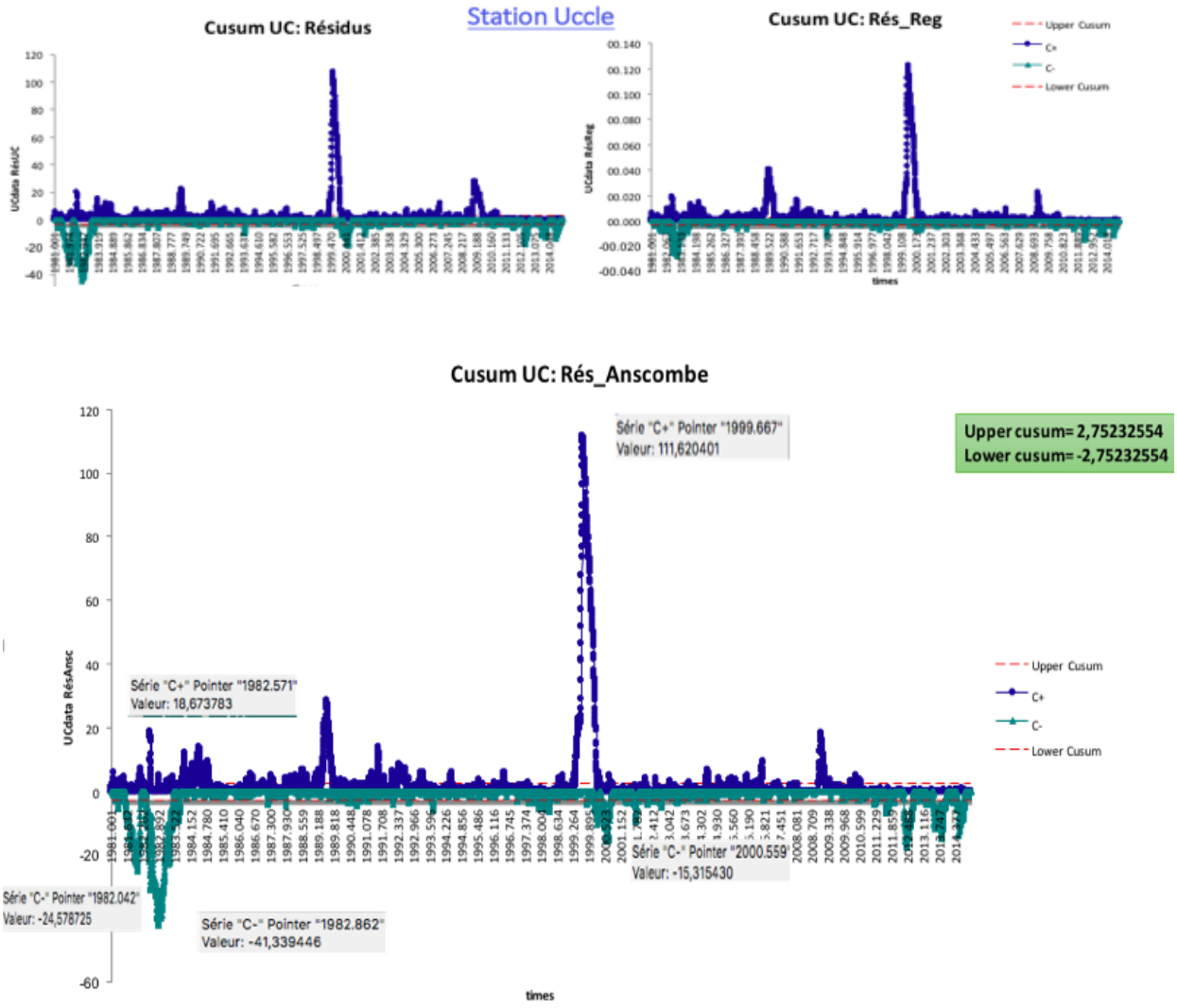
3. Carte CUSUM : Uccle

a. Période 1981-2014

Ci-dessous les cartes CUSUM établies sur les trois versions de résidus de la station Uccle pour la période 1981-2014, on voit clairement que les tendances sont les mêmes, ainsi que les

[2] Les résultats de la carte CUSUM sont en valeur et non pas en pourcentage pour des raisons de simplicité d'interprétation
[3] La station de référence pour assurer la continuité avec les observations précédentes de Zurich

périodicités qu'on pourrait qualifier comme hors contrôle. Egalement, ces tendances correspondent aux fluctuations observées au niveau de la représentation graphique des données brutes transformées en racine carré.



Graphe 2 : Cartes CUSUM de la station Uccle (Basées sur les trois méthodes de calcul de résidus) 1981-2014

La carte ci-dessus révèle généralement la présence de certaines périodes de stabilité mais aussi d'une pointe particulièrement élevée d'une valeur de 111, 62, après laquelle le signal a rechuté pour passer à -15,31en septembre 1999.

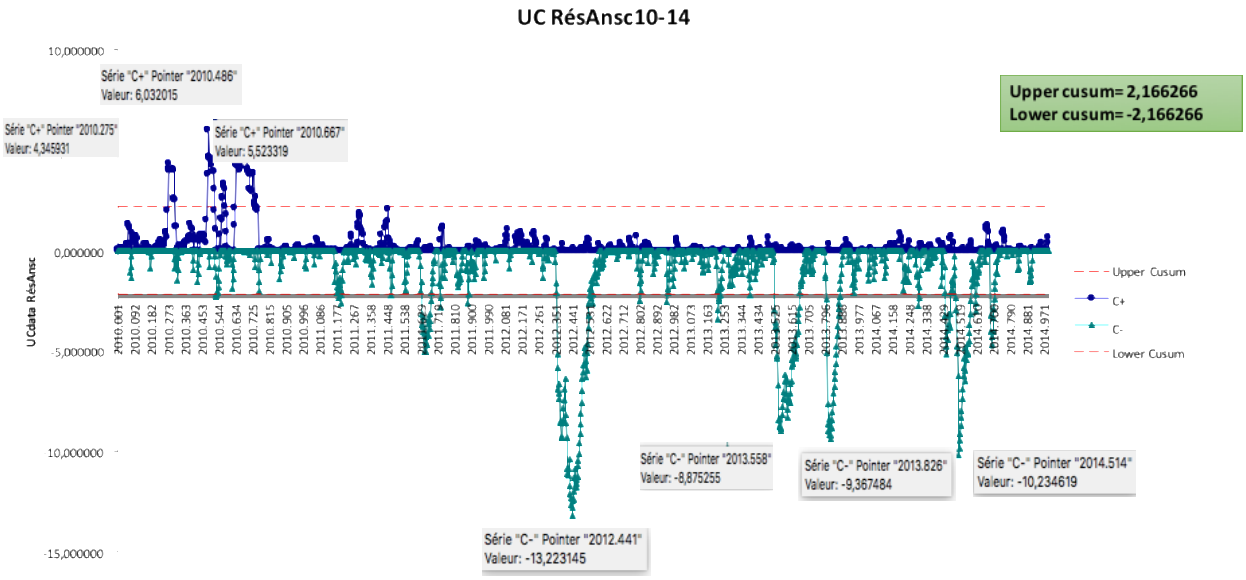
Un autre shift tout aussi important est détecté en 1982 et précisément entre la fin du mois de juillet et le mois de novembre où la valeur Cusum est passée de -24,57 à 18,67 pour chuter par la suite à -41,33.

Par ailleurs, il existe aussi quelques dérives pas aussi importantes que celles précitées après 2010, qu'on va examiner dans la prochaine sous-section.

b. ***Période 2010-2014***

Sur cette sous-section, on fait le focus sur la période 2010-2014, période durant laquelle la station Uccle se caractérisait par la présence d'un nombre de dérives relativement réduit mais qui sont surtout négatives, ceci dit, elles ne sont certes pas nombreuses, mais demeurent très importantes. En effet, Les shifts constatés ont atteint une valeur C- de -13,22 en juin 2012 et -10, 23 en juillet 2014, sachant que la limite inférieure est de -2,166, seuil au-delà duquel la station peut être qualifiée comme instable.

Station UCCLE: 2010-2014



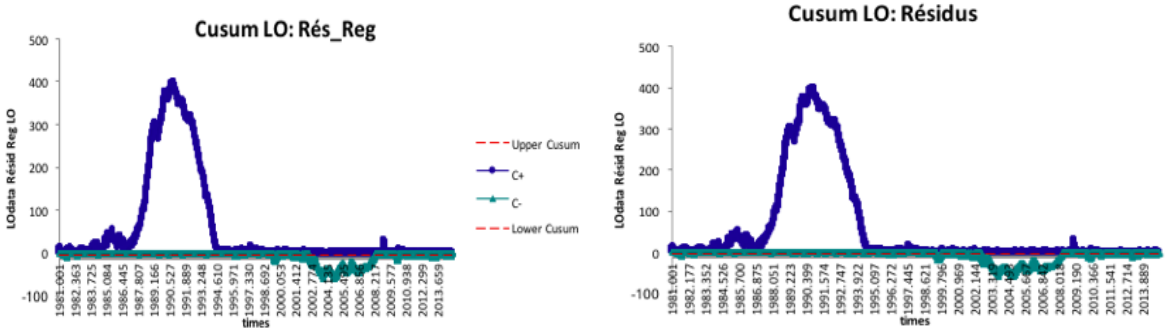
Graphe 3 : Carte CUSUM de la station Uccle 2010-2014

Ceci étant, la station Uccle se caractérise relativement par une certaine stabilité, toujours est-il, il y est des shifts importants avec des écarts brusques et accentués.

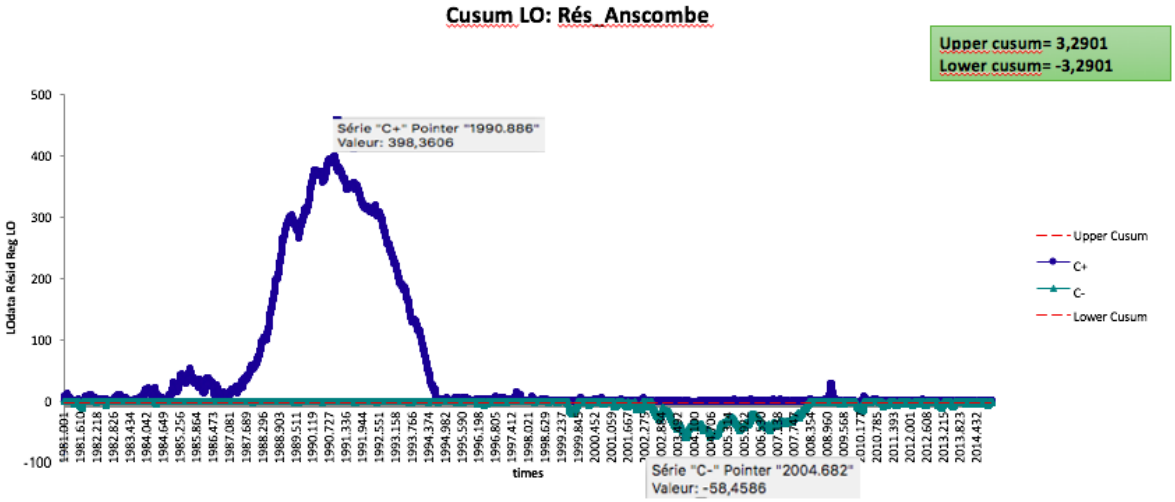
4. Carte CUSUM : Locarno

a. Période 1981-2014

Les résultats des 3 scénarii pour Locarno sont aussi similaires, et s'inscrivent bien dans les tendances des données originaux, ceci conforte ce qui a été avancé pour la station Uccle.



Station LOCARNO



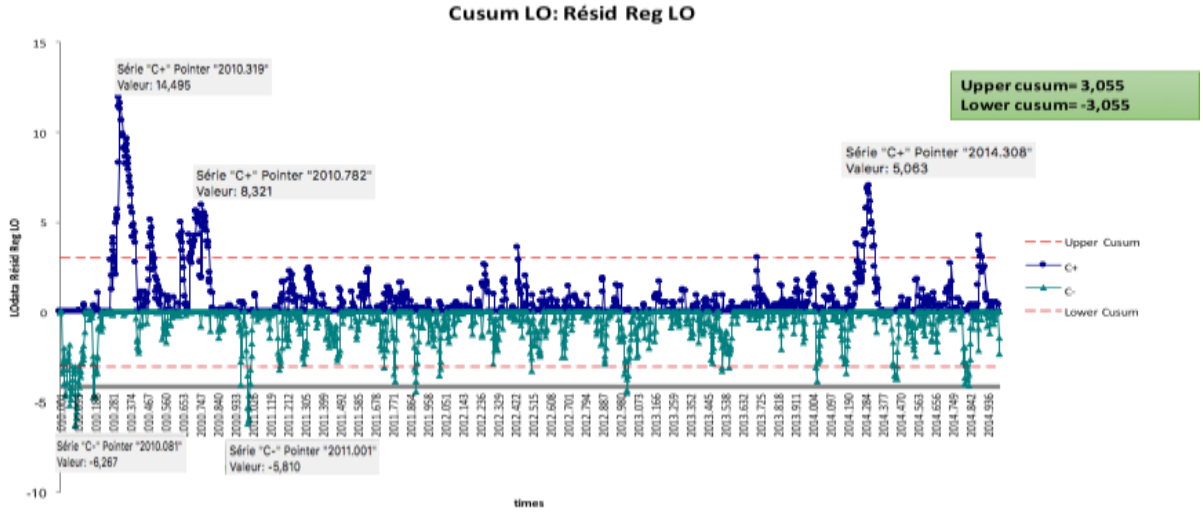
Graphe 4 : Cartes CUSUM de la station Locarno (Basées sur les trois méthodes de calcul de résidus) 1981-2014

Pour la présente station, on constate 2 importants drifts, un premier qui a atteint une valeur C+ de 398 en novembre 1990 et un second d'une valeur C- de -58,45 en septembre 2004, néanmoins, Locarno reste relativement stable et n'inscrit pas des écarts prononcés.

b. ***Période 2010-2014***

Le graphe ci-après retrace l'état du processus en 2010-2014, il met en lumière une vingtaine de dérives, dont les plus importantes qui ont cumulé un écart relativement important ont été constaté en 2010 et précisément en janvier avec une valeur C- de -6,267 et en avril avec un C+ positif de 8,321, sinon, il y a eu par la suite des dérives mais qui ne sont pas très prononcées et restent moins importantes que celles enregistrées en 2010.

Station LOCARNO: 2010-2014



Graphe 5 : Carte CUSUM de la station Locarno 2010-2014

Finalement, on remarque que la carte CUSUM de la station Locarno sur toute la période 1981-2014, n'a pas mis en exergue les différentes dérives constatées sur la carte portant sur la période 2010-2014.

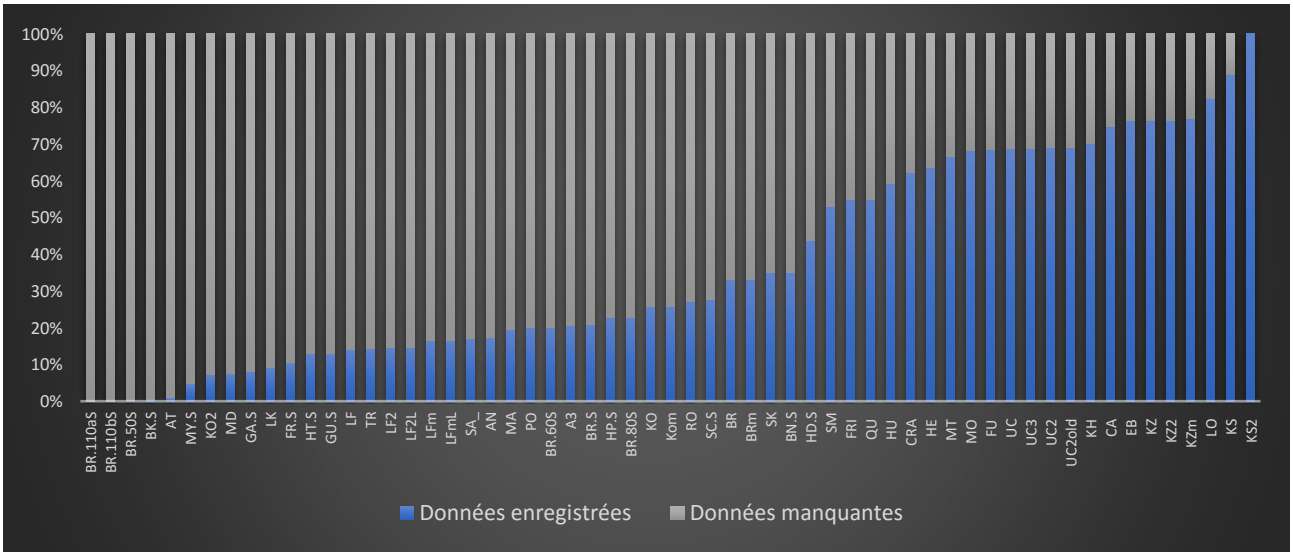
Cela nous apprend qu'une carte établie sur une longue période couplée à la présence des dérives relativement moins importantes ou qui ne s'écartent pas trop des limites inférieure ou supérieure, ne permet pas de détecter facilement les petits décalages pouvant exister.

A cet égard, On peut faire le constat que la station Uccle, contrairement à la station Locarno se caractérise par la présence des écarts accentués et plus importants constituant des shifts considérables.

5. Exploration des données

a. Données manquantes

Une exploration des données nous a permis de faire un premier constat relatif à la disponibilité des stations, en effet, un bon nombre de stations ne fournissent pas ou peu de données depuis les années 80. En effet, 36 observations sur 59 proviennent de stations dont plus de 50% de données sont manquantes depuis 1981.



Graphe 6 : Etat des données manquantes 1981-2014.

Le nombre d'observations provenant de stations ayant arrêté de fournir des données depuis 1989 ou moins sont au nombre de **15**, il s'agit notamment de : AT ; BK.S ; FR.S ; BR.110aS ; BR.110bS ; MY.S ; LF ; LF2 ; LF2L ; LFm ; LFmL ;LK ; MA ; MD ; KO2.

D'autres stations ont arrêté d'observer le soleil depuis les années 90 (entre 1990 et 1999), il s'agit exactement de **14** observateurs : BR.50S ; BR.60S ; BR.80S ; BR.S ; BR ; BRm ; A3 ; AN ; GU.S ; TR ; HP.S ; HT.S ;KO ; KOm.

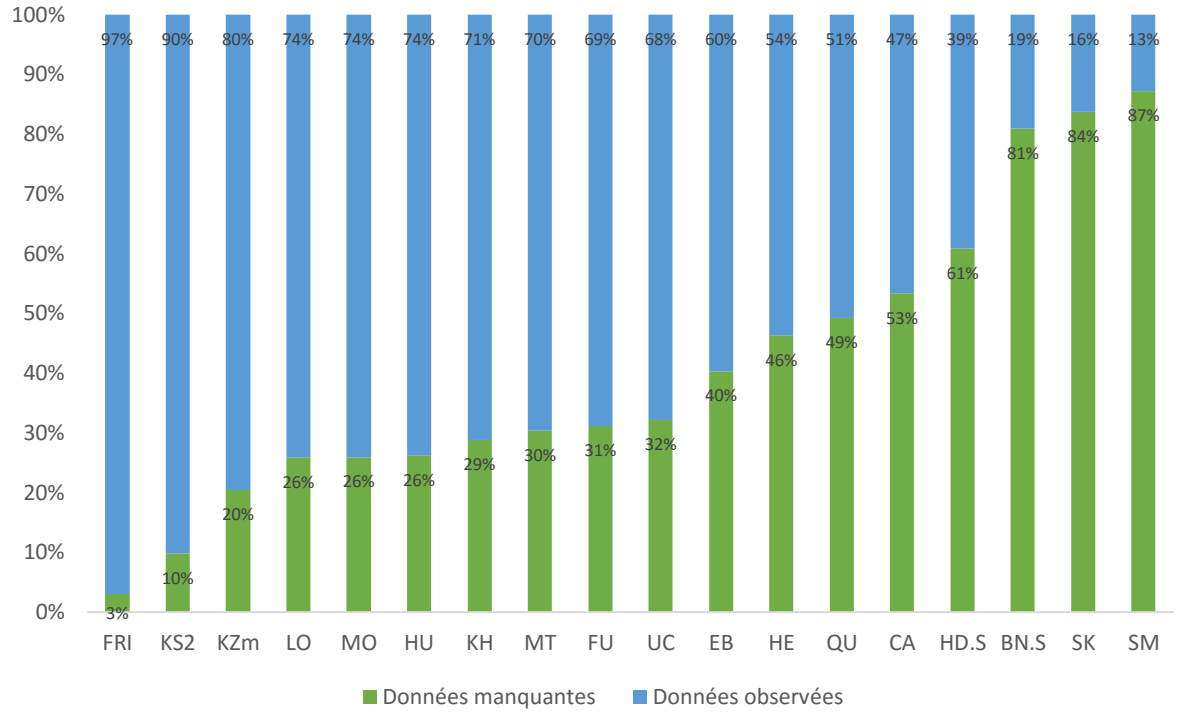
Depuis les années 2000 (entre 2000 et 2010), **6** stations ont cessé de fournir des mesures SSN, à savoir : SC.S ; GA.S ; RO ; SA ; PO ; CRA.

Ceci étant, 50% des mesures sont estimées depuis les années 90, vu qu'elles émanent de stations inactives.

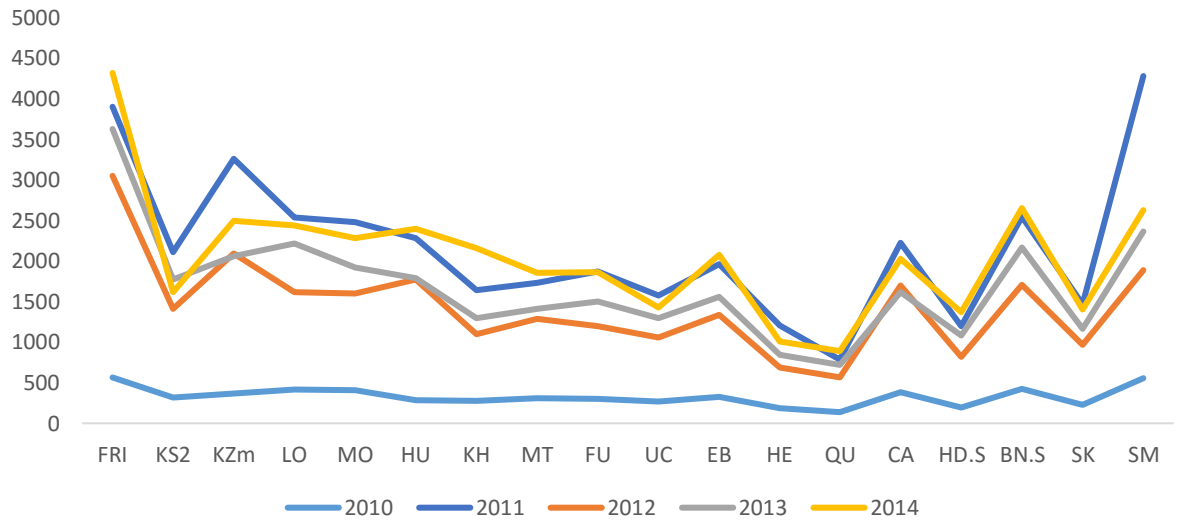
b. ***Stations actives***

Sur l'ensemble des stations, 18 ont été actives durant les 5 dernières années (2010-2014). Par active on entend qu'elles ont fourni en moins une observation pendant cette période.

Les graphes ci-dessous présentent lesdites stations, et illustrent respectivement l'état des données manquantes et observées par station, et la variance enregistrée par année et par station.



Graphe 7 : Etat des données manquantes chez les stations actives 2010-2014



Graphe 8 : Variances chez les stations actives 2010-2014.

Il résulte du classement des stations selon la proportion des données manquantes illustré par le graphe 7 que la station FRI est la plus assidue avec juste 3% de données manquantes sur toute la période, suivi de KS2 avec 10%, la station Locarno enregistre 26% des données manquantes, quant à la station Uccle elle occupe la 10ème position (32%).

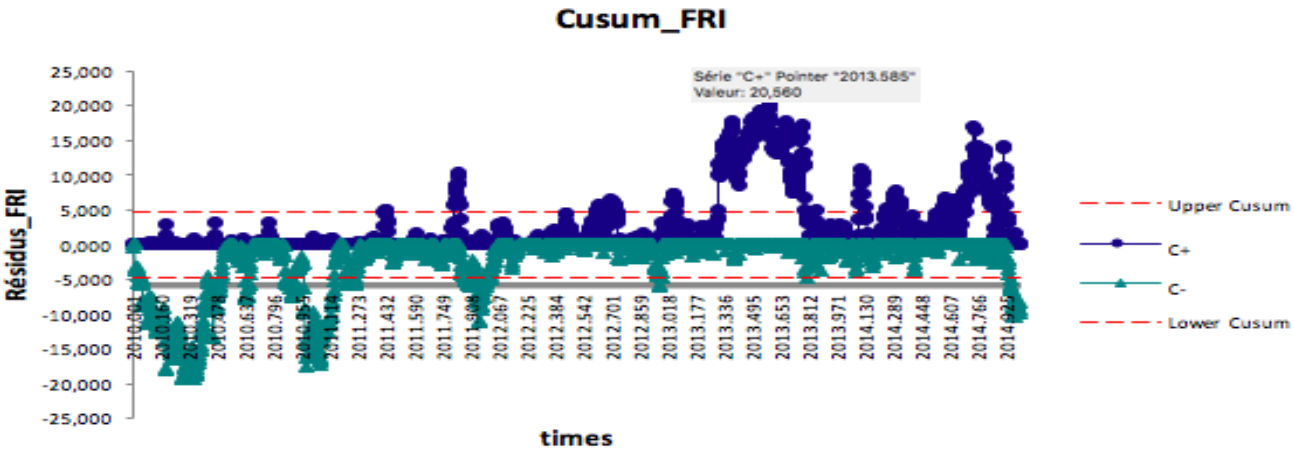
Le graphe 4 illustre la variance observée par année et montre globalement que les variances ont les mêmes tendances par station, avec une variance en 2010 particulièrement basse relativement aux autres années. (Annexe1 : un tableau détaillé des variances par station).

Les deux graphes ci-dessus nous renseignent sur l'assiduité des stations mais ne mettent pas en évidence la qualité des observations fournies par ces dernières. qu'il n'y a pas de liens évidents entre la proportion des données manquantes et les variances enregistrées, De ce fait, il serait approprié de renforcer notre analyse via des cartes Cusum en vue d'éclairer notre lanterne par rapport à la qualité des données par station.

Ayant le souci d'avoir une visibilité actualisée qui tient compte même des petits écarts au niveau des stations actives, l'exécution de ce procédé va porter les stations ayant moins de 50% de données manquantes, et ce, sur la période 2010-2014.

c. ***Cartes CUSUM des stations actives***

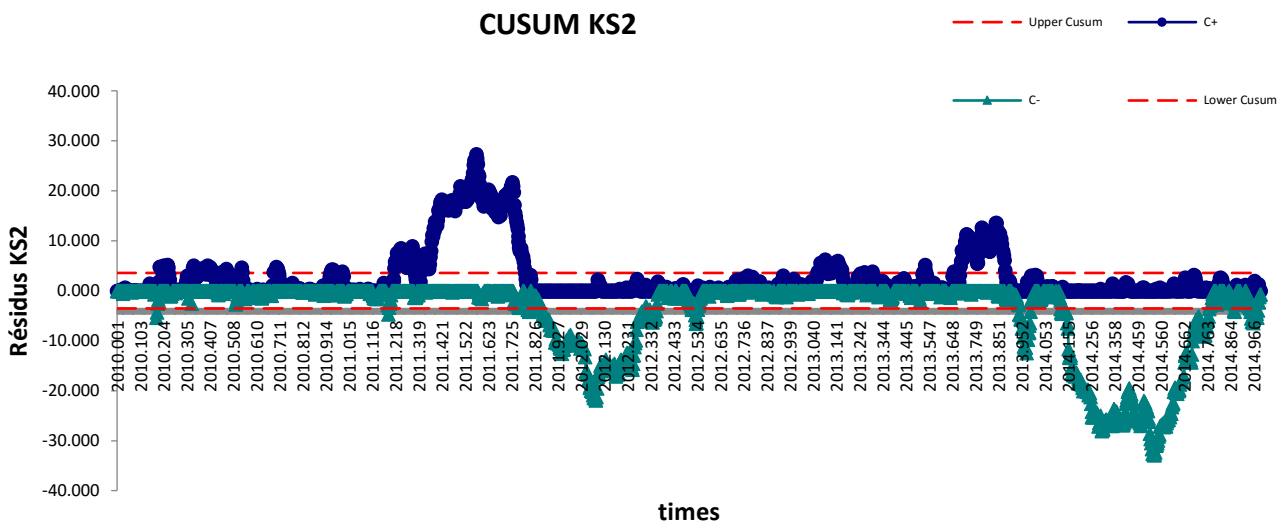
Certes, la station FRI est la plus assidue, toutefois, sa variance est la plus élevée, sauf en 2010 où elle a occupé la deuxième position après la station SM. De surcroît, sa carte CUSUM illustre un processus instable marqué par la présence de plusieurs dérives sans aucune ou de très courtes périodes de stabilité.



Graphe 9 : Carte CUSUM de la station FRI (2010-2014)

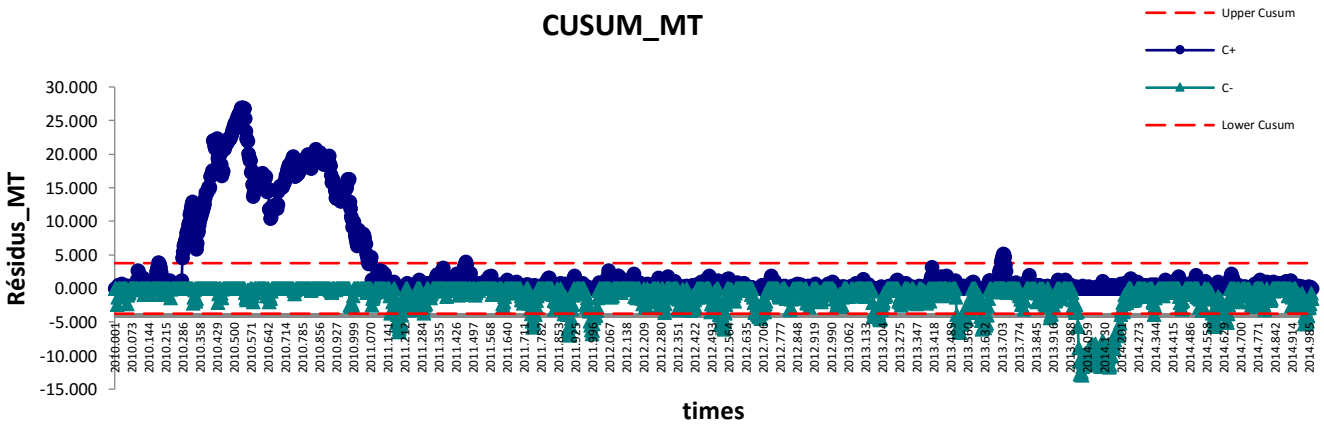
La station KS2 combine une variance relativement dans la moyenne des autres stations ainsi qu'une très bonne disponibilité. Néanmoins, son processus est marqué par la présence de quatre dérives importantes. Les deux premières se sont succédées et se sont étalées sur une période allant d'avril 2011 à janvier 2012.

Egalement, un drift considérable a marqué l'année 2014 avec une valeur C- de -31, 717 atteinte en mois de juillet.



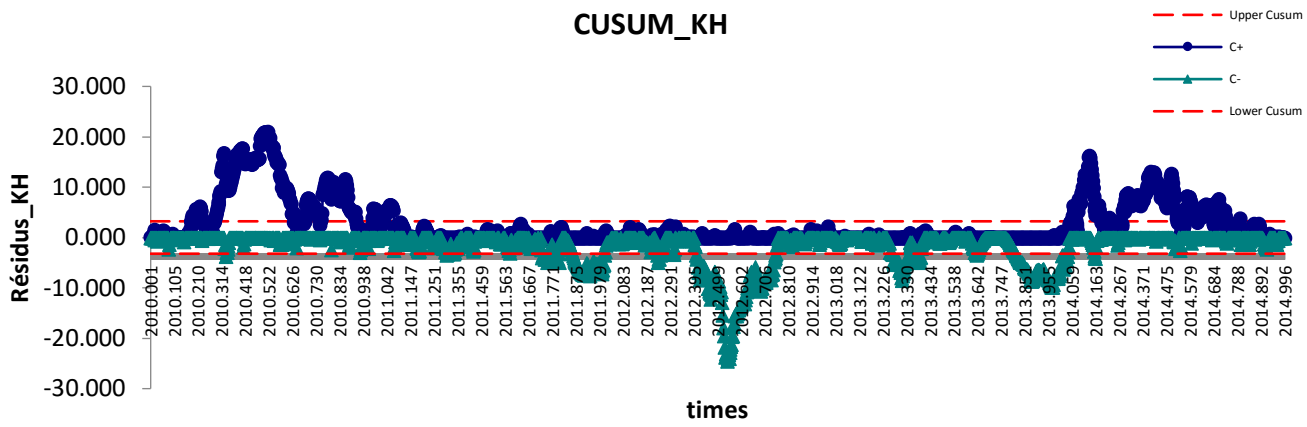
Graphe 10 : Carte CUSUM de la station KS2 (2010-2014)

La station MT a enregistré durant la période 2010- 2014 une variance un peu en dessous de la moyenne, elle est assez disponible avec juste 30% de données manquantes, et son processus est relativement stable avec un drift important entre juillet et octobre 2010, et un deuxième moins accentué en janvier 2014.



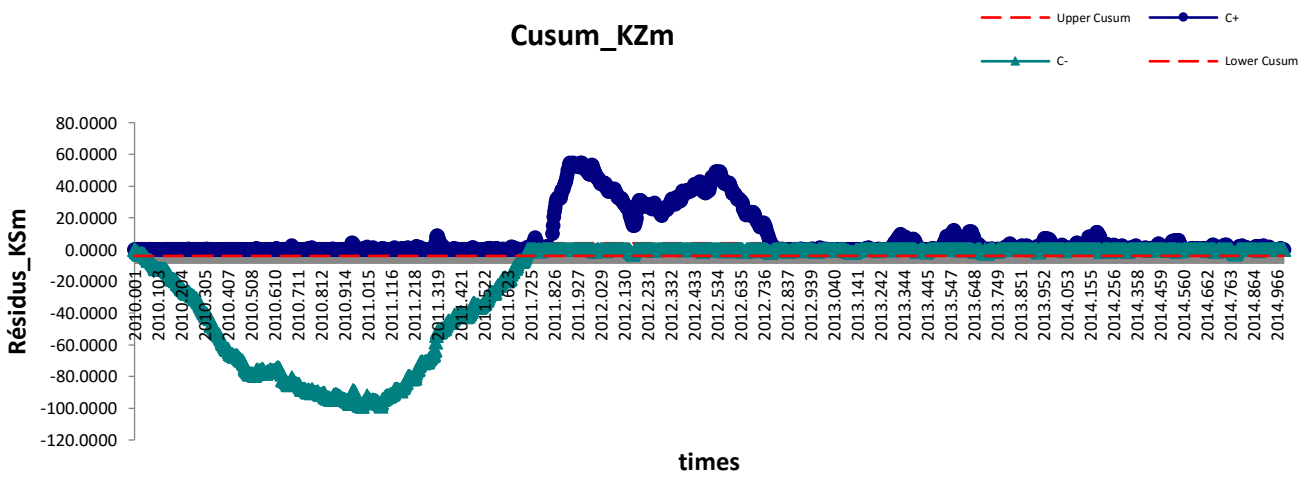
Graphe 11 : Carte CUSUM de la station MT (2010-2014)

La carte de contrôle de la station KH illustre à son tour un processus instable avec quelques périodes de stabilité en 2011, et des points d'arrêt dont les plus importants sont constatés en juillet 2010 et juillet 2012. Par ailleurs, les données sont complètement instables depuis novembre 2013.



Graphe 12 : Carte CUSUM de la station KH (2010-2014)

La station KZm se caractérise par une bonne disponibilité, et un niveau de variance relativement dans la moyenne, de surcroit, son processus durant les cinq dernières années a connu une certaine stabilité à partir de septembre 2012, après deux importants drifts enregistrés entre 2010 et juillet 2012.

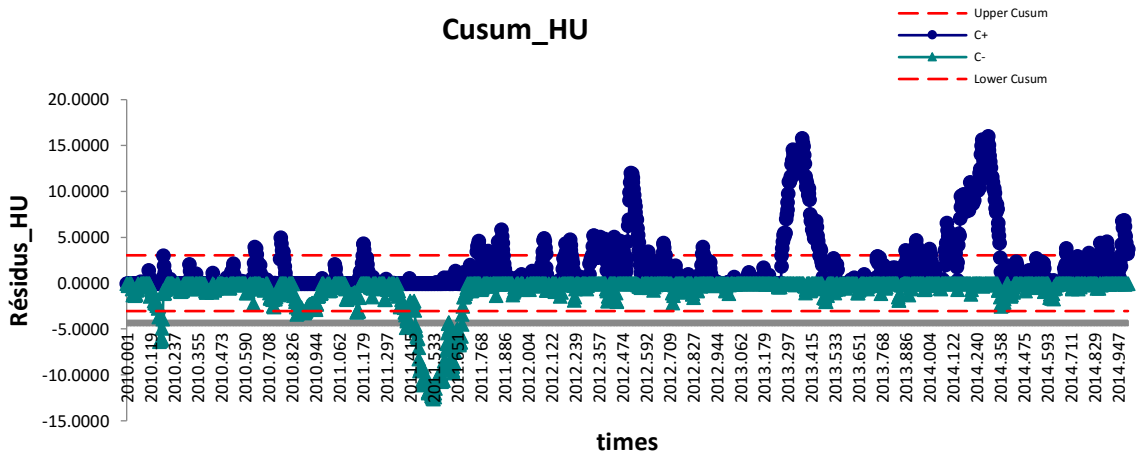


Graphe 13 : Carte CUSUM de la station KZm (2010-2014)

Les dérives de la station HU sont principalement des shifts importants, les données passent d'un dépassement de la limite supérieure à un écartement de la limite inférieure de manière brusque et en une courte période de temps.

Les principaux shifts sont au nombre de 4, et ont été constaté respectivement en juillet 2011, juillet 2012, mai 2013 et avril 2014

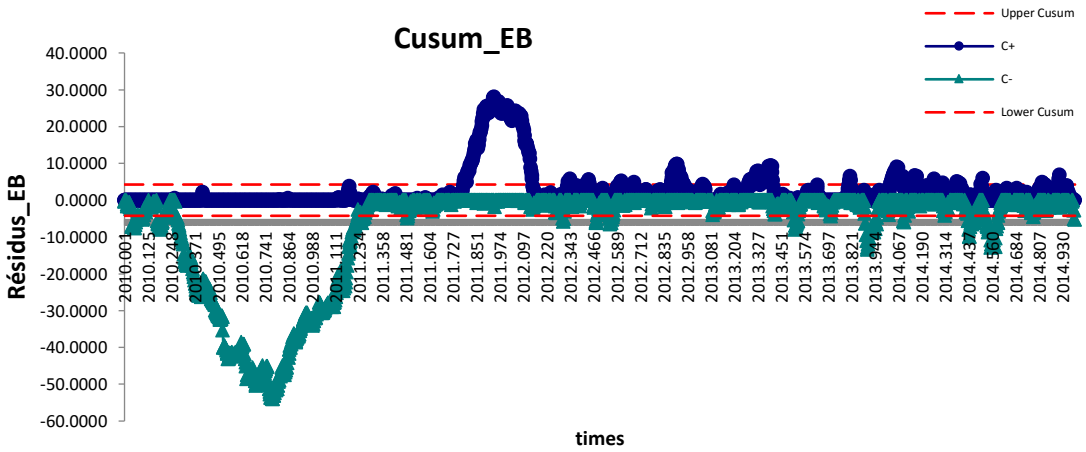
.



Graphe 14 : Carte CUSUM de la station HU (2010-2014)

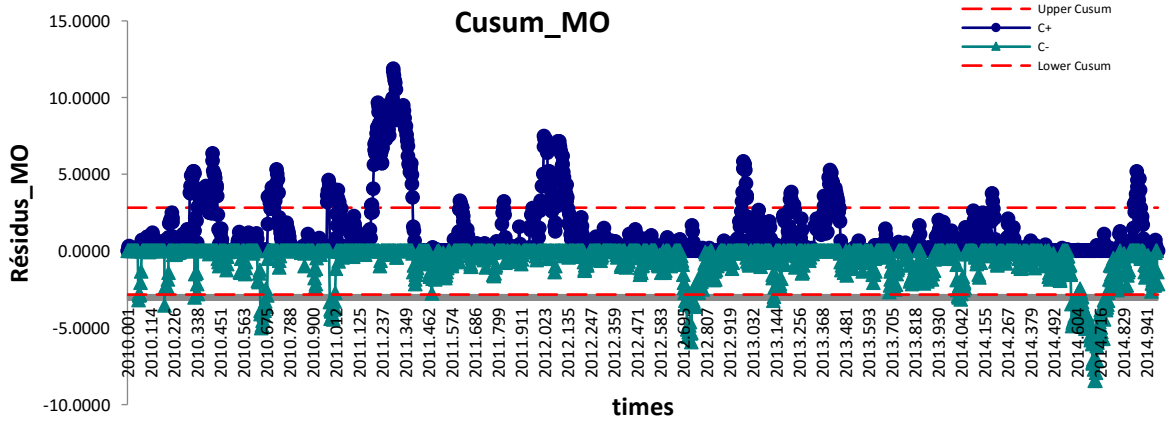
A l'image d'un bon nombre d'observateurs, la station EB s'est écartée de la limite inférieure de contrôle (-4,24) avec une valeur de -53,51en 2010, et s'en est suivi un drift pas aussi important que le premier en décembre 2011.

A partir de 2012, une certaine stabilité est constatée avec quelques petites dérives.



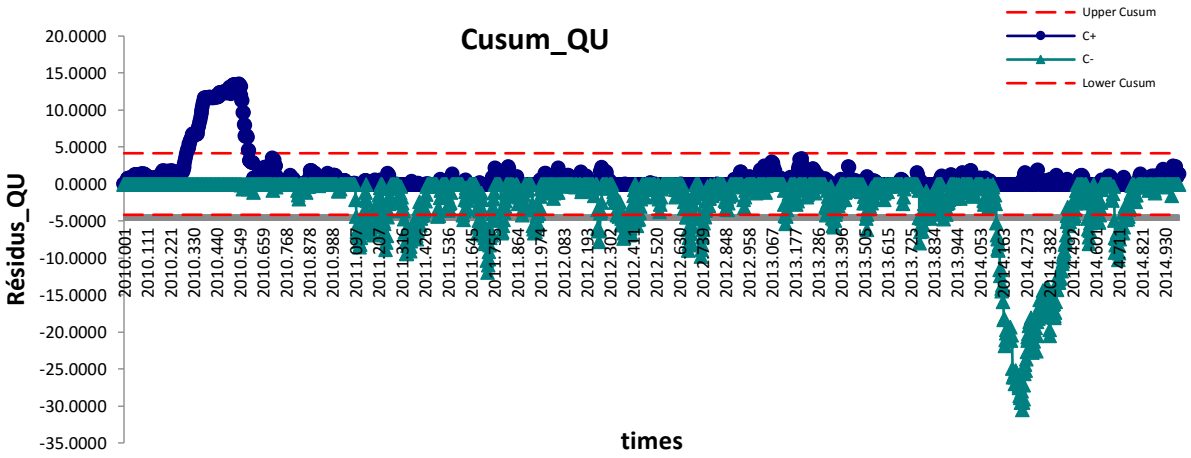
Graphe 15 : Carte CUSUM de la station EB (2010-2014)

La station MO présente également des shifts tout au long de la période 2010-2014 avec certaines périodes de stabilité, notamment entre janvier et juillet 2012 et entre juin 2013 et juillet 2014



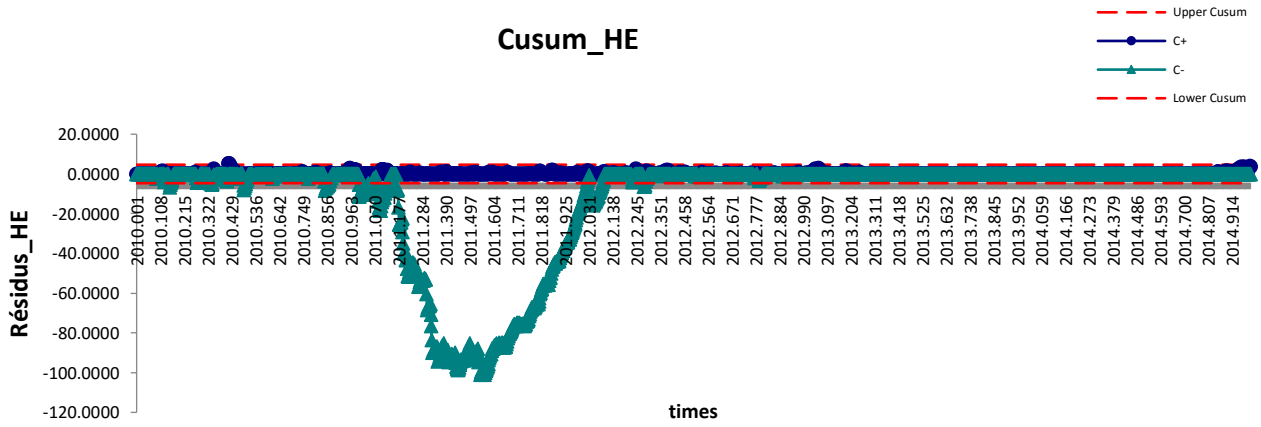
Graphe 16 : Carte CUSUM de la station MO (2010-2014)

Contrairement à la station FRI, la station QU présente la variance la moins élevée durant les cinq dernières années. Son processus est entièrement instable, avec 2 dérives importantes enregistrées respectivement en juillet 2010 et en Avril 2014,



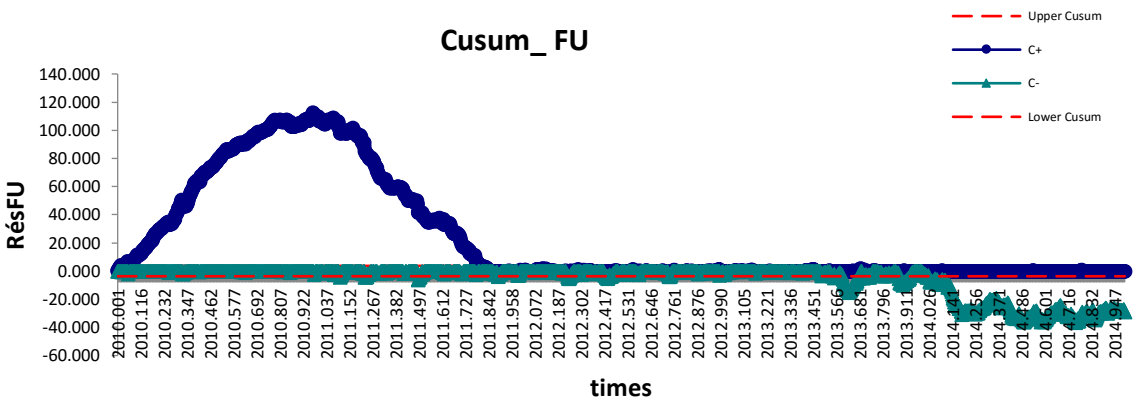
Graphe 17 : Carte CUSUM de la station QU (2010-2014)

Alors que presque l'ensemble des stations ont connu une grande instabilité en 2010, la station HE se distingue par une bonne qualité de donnée en cette année.
Par ailleurs, elle a connu un drift important en 2011, après lequel, une stabilité est constatée, et elle est due probablement à la proportion de données manquantes qui avoisine 86% en 2013-2014.



Graphe 18 : Carte CUSUM de la station HE (2010-2014)

La station FU se caractérise par une bonne stabilité des données en 2012-2013 avec juste 30% de données manquantes dans cette période.
Sinon, une grande phase d'instabilité a couvert la phase 2010-2011, également, le processus s'est éloignée de la limite inférieure en 2014.



Graphe 19 : Carte CUSUM de la station EB (2010-2014)

Ci-après un tableau récapitulatif qui résume la situation par station active avec moins de 50% de données manquantes, et ce, en termes de niveau variance et de la qualité des données issues de leurs cartes CUSUM respectives.

Stations	% Données manquantes	Variance	Périodicité des dérives				
			2010	2011	2012	2013	2014
FRI	3%	↑	Instable				
LO	26%	↑	Instable	Octobre	Stable	Stable	Janvier et Avril
KS2	10%	___		De Juillet à septembre	Janvier	Octobre	De Avril à juillet
KZm	20%	↑	Instable	Instable	De janvier à Juillet	Stable	Stable
UC	32%	↓	Instable	Août	Juin	Octobre et Juillet	Juillet
KH	↓	↓	Juillet	Décembre	Juillet	Avril et décembre	Janvier et Juillet
MO	26%	↑	Instable avec de courtes périodes de stabilité				
HU	26%	___	Stable	Juillet	Juillet	Mai	Avril
MT	30%	___	De juillet à octobre	Assez Stable			Janvier
QU	49%	↓	Instable				
EB	40%	___	Instable	Décembre	Stable	Stable	Stable
HE	46%	↓	Stable	Juillet	Stabilité due probablement au manque de données		
FU	31%	___	Instable jusqu'à octobre 2011		Stable	Stable	Instable

Tableau 1 : Récapitulatif de l'état des stations actives 2010-2014

Conclusion

En conclusion, on peut dire que le suivi des données manquantes et des variances par station permet de faire, respectivement, une appréciation de l'assiduité des stations et du niveau de dispersion des observations. En outre, ce travail a affirmé l'importance de prendre en ligne de compte la qualité des observations fournies par chaque station, étant donné la contribution de celles-ci dans la construction de l'ISN.

L'exécution de ce processus sur la période 2010-2014 nous a permis d'identifier des stations assidues et relativement stables durant des périodes assez longues tels que les stations EB, KZm et FU ; et également des stations instables avec de très courtes périodes de stabilité, tels que les stations FRI, MO et QU.

Il est à noter que des cartes de contrôle sur de courtes périodes sont plus sensibles même aux petits décalages et permettent d'identifier facilement les petites dérives. D'ailleurs, ceci a été illustré par les cartes établies sur la période 2010-2014 au profit de la station Locarno. En effet, on a vu clairement que les petits écarts des limites sur cette dernière période n'ont pas été fortement perceptible sur la carte réalisée sur la phase 1981-2014.

Eu égard aux éléments avancés, une appréciation de la qualité des données sur de courtes périodes, de manière automatique et régulière aura certainement une valeur ajoutée considérable. De ce fait, il serait intéressant d'automatiser un processus d'appréciation de la fiabilité des données de manière ponctuelle.

Liste des graphes et tableaux

Annexe : Variances par station active 2010-2014

Année	2010	2011	2012	2013	2014
QU	137,419	784,962	566,232	720,119	890,213
HE	188,733	1201,586	690,512	847,894	1004,171
HD.S	193,926	1199,604	820,9	1084,494	1371,162
SK	224,259	1470,093	967,198	1164,579	1402,857
UC	270,342	1578,248	1059,76	1295,108	1427,927
KH	272,656	1640,744	1099,227	1293,079	2155,643
HU	285,982	2280,936	1773,684	1788,119	2394,997
FU	302,264	1873,122	1199,926	1503,007	1860,118
MT	306,275	1732,52	1291,449	1411,568	1856,466
KS2	313,728	2108,025	1408,632	1775,791	1614,463
EB	322,177	1961,823	1335,848	1555,619	2075,355
KZm	366,543	3263,928	2090,719	2057,564	2495,363
CA	382,619	2221,741	1700,647	1612,923	2024,954
MO	409,189	2481,609	1596,38	1923,191	2284,483
LO	415,708	2538,463	1617,486	2218,882	2440,574
BN.S	421,442	2544,028	1711,037	2168,23	2652,52
SM	557,112	4281,325	1885,296	2368,487	2630,002
FRI	564,918	3903,607	3051,467	3628,37	4319,702

Val-U-Sun Project : Computational Aspects to Handle Missing Values

LSTAT2390 Statistical consulting (C.Ritter)

Antoine Pissoort

Friday 5th May, 2017

Executive Summary

This project will give a first emphasis on the analysis of solar activity through the Sunspot Number. We have first put our efforts on the code translation and adaptation from Matlab® to R with focus on vectorization issues and parallel computing for efficiency. Aware of the in-depth work that the client will have to do into this project in the future, we put emphasis on giving ideas together with reusable and easily tunable features (tools) to make relevant future analysis. We have then created a R package to gather functions and form a first basis for future work. We developed visualization tools such as shiny applications. A refined cross-validation procedure has led us to reconsider the results of Dudok de Wit (2011) for parameters tuning. This cross-validation scheme has been made repeated and K-fold where we found the importance of other parameters on the tuning of SVD number of components. Interesting results have been provided with new computational ideas allowing to expand our vision and to think further in the area.

Contents

Introduction

This project (Val-U-Sun) will study the solar activity through a quantity called the *Sunspot Number*. This quantity is widely used as the main reference solar index on which hundreds of published studies are based (see Clette et al. (2007) for more details). The *Sunspot Number* (SSN) is a combination of the number N_g of sunspot groups that are present on the Sun and the total number of sunspots N_S, (see de Wit et al. (2016)). This quantity is subject to various uncertainties, which behave as Poisson-like random variables. More information on the Sunspot Number can be found on the references cited above.

This project will more focus on the computational aspect, trying to *fill the gaps* that occur for quite different reasons. We will rely on a method originated from Dudok de Wit (2011) to fill data gaps in multi-wavelength or in multichannel records using a data-adaptive and nonparametric method.

This report have the following structure : in section **0** we will present the tools that have been developed during this project (i.e., the R package and the Shiny applications). In section **1** we will present the method of Dudok de Wit (2011) to fill the gaps and our reimplementation. In section **2** we will compare 2 languages (Python and R) that have been used to translate this method. In section **3** we will try to do better with other methods such as the one from C.Ritter or by replacing the gaussian smoothing by splines. In section **4** we will present a big cross-validation scheme before drawing a 5.

As we will do computational time analysis (profiling, benchmarking, . . .), we must notice that the following computations are run on a *i7-2820QM CPU 2.30GHz, 24GB RAM*.

0 Development Tools

We would like to start by introducing the tools we developed. We put this section here as we will mention this package during the project for better comprehension[1].

A R package to ease the use of the project four ourself but also especially for future user (e.g., the reader). And a shiny application to summarize smoothly the huge amount of results and outputs we obtained.

0.1 R Package (Github)

We have chosen to locate the package on github as it is the easiest way for the users to use it. You can directly fin it in the following address.

`https://github.com/proto4426/ValUSunSSN`

It only requires the `devtools` package, so make sure to have it installed on your machine ! To install it, you just have to type

```
devtools::install_github("proto4426/ValUSunSSN")
```

in the R console. Or you can also look (e.g. if any error) and follow instructions from the README.md in the github repository.

In this repository, you can find the whole content of the package, that is the following **folders** :

[1] But if you prefer, you can go directly in the next section to introduce the subject.

- **/R** : contains all `.R` files containing the functions we have build and used during the project. These are the functions you are allowed to use when you have loaded the package. Note that a single file, e.g. `UsedFunc.R`, can contain several functions. See appendix **A.5** for more details on the `.R` files.

- **/Scripts** (?) : contains some scripts realized during the project. We included them for smoother understanding of the project. It is not part of the package, but as their size is limited, it will not slower the package's installations. See appendix **A.3** for more details on these files.

- **/data** : gathers all the data we have computed to use it more conveniently, for example inside the Shiny applications. From that, these are in `.Rdata` but they can be easily transformed in other formats such as `.csv`. These can be used inside a R session simply by typing e.g. `data("ssn_splines81.RData")` if you want the filled data computed with the splines method (see section **3.2**) starting at 1981. See appendix **A.2** for more details of each datasets available.

- **/inst** : contains the Shiny applications for use directly through the R package. See next subsection for more information on these applications.

- **/man** : Contains the documentation of the functions builded by the package `roxygen2`.

- **/vignettes** (*not updated*) : have been created for better understanding of the usage of the package's functions and some results, and to make links with the projects. We leave them in the repository because it is a convenient way to explain a package. These are old versions based on the non-transformed data . We thought afterwards that it would be more smooth to summarize all information in the Shiny applications (see next subsection) and hence we did not update it.

0.2 Visualization

For the sake of this project, we have developed Shiny applications to improve visualization abilities directly located inside the package. This comes from the fact that we did not want to discard arbitrarily some stations. We wanted any users to be able to represent the observed values of every stations together with the filling method of his choice, to compare it with other stations and other filling methods.

To run these application (1 at a time), be sure to have the package loaded with `library(ValUSunSSN)` to enable the use of the data inside the application and then run either

```
runExample('stations') # To obtain plots of the stations. Or

runExample('residuals') # To obtain plots of the residuals.
```

This would open two different applications which are each doing the following :

- The first one only compares the observed value with the filled value, for the different filling methods.

- The second one compares the '**residuals**' of two stations obtained by our method with another selected method. We allow the user to express it in % scale. To do so, we normalized the residuals

with the *SILSO* to obtain more meaningful results. The *SILSO* method will be explained in section 3. This normalization is probably not the best method and this could be easily improved. Moreover, note that when comparing with the SILSO itself, this gives weird/positive(?) results.

We are aware that the methods already provided in these applications are not exhaustive. By making this application, we aimed at allowing the interested user to be able to easily modify the code to introduce any other methods he would want.

A important **difficulty** we have faced was to translate stations with different names across the datasets we were provided to build the applications. We put efforts to make *individual corrections*, but this is possible that there remains a very few errors in this part especially for stations whose name finish by "-S". That should be corrected yet. However, *individual correction* is not preferable and the goal would be to look at an automated method which could match any pair of station's names. This would be an easy *text recognition* case but this should have 100% accuracy. Finally note that this is important for

Note that a **more accurate** method (statistically) can be implemented relying on the variability analysis which could give us estimates of the standard errors. For example, we could assess if there is a significant difference between the methods we considered, or between the stations themselves, for a given time period, and then take better decisions.

We give an example of the outputs provided by the "residuals" application in figure 3 in appendix A.1. Actually, we remark by exploring other stations the huge difference with the "Chris method" around January 2009 arise for most stations, negative or positive (...) This is an example of what things we can detect with this tool, but we will not inspect this further.

Improvements : An implementation with the well known Javascript library `plotly` would be preferable to obtain better visualization with dynamic graphs, i.e. by allowing to better select areas of interests, etc[2]... It would be quite easy to translate it as the syntax of `plotly` in R is very similar to `ggplot2` which has been used to build this application so far. The old "vignette" available in the **/vignettes** folder of the github repository makes use of `plotly`.

Note that an implementation of Highcharts[3] with the R package `highcharter` would be also an interesting idea and not too difficult to realize.

1 Code Translation : "Interpolating SVD-EM" method

The method is *nonparametric* and *data-adaptive* based on *singular value decomposition* (SVD) with close relationship to Expectation-Maximization (EM) method (originated from Dempster et al. (1977)).

The method is based on Dudok de Wit (2011) where it is explained in details. Let $y \in \mathbb{R}^{n \times m}$ denotes the matrix representing the time index (in days) in rows and the stations as columns. The main assumption for this method is the linear correlation between the time records.

As the explanations are rich in the above cited article, we will try to present a more specific explanation based on our specific case and comprehension with the data at hand and the reimplemented code.

[2] see `https://plot.ly/r` for the official R reference
[3] see `https://www.highcharts.com`

We have put efforts to design the most efficient code we can do so far, e.g. by minimizing the `for` loops, i.e. by vectorizing the code.

1.1 Explanation of the Function : `interpolsvd_em()`

We first note that an advantage of building a R package is that you can have directly access to the help page of each function, say `interpolsvd_em()`, by doing `help(interpolsvd_em)` or `?interpolsvd_em()` in the console. Hence, we will not repeat ourself[4] and we only present a brief summary of the function's inputs :

- *y* : Matrix containing the numeric data of interest. Here, it will be typically time units in rows and station(s) in column(s).
- *nembed* (=2) : number of dimensions to which the input matrix will be embedded. It Must be > 1 if only one single time series is given. Embedding involves a weighted averaging over time and is thus appropriate when there is a high correlation in time.
- *nsmo* (=81) : Cutoff time scale (in number of samples). Set it to 0 if only one single time scale is desired.
- *ncomp* (=4) : Maximum number of components retained from the SVD. **(check with cv)**
- *niter* (=30) and *threshold1* ($=10^{-5}$): control the "convergence" of the algorithm by specifying the number of iterations for each number of components during

The default values given in () will be used to develop the first algorithms. We study more deeply these values by cross-validation in section **5**.

Note also that relevant comments have been added to the R file where the function is implemented[5]. You are invited to refer to it. We now explain this function step by step :

1. After the definition of the function, it normalizes the input matrix. Indeed, as the SVD is scaling dependent, it is important to normalize the data. Then, it performs some tests to check if there are enough observed values per stations.
2. After applying the embedding (if asked), it will weigh each records according to the number of each station's missing values. Larger weights are given to records with fewer gaps which will improve the quality of the method since larger gaps are harder to interpolate. The inverse transformation is done at the end of the algorithm (see step 6).
3. A first approximation to fill the gaps is made by doing a simple linear interpolation. Improvements could be found by changing this simple interpolation method, even if the impact on the whole function is expected to be small. This will provide a new matrix on which the SVD will be applied. This step is needed as the SVD cannot work with missing values.

[4] And as the old adage goes in scientific computing : *Don't Repeat Yourself*
[5] you can retrieve it in **/R/interpolsvd_em.R** in the package's Github repository

4. The 1st mode of the SVD is then computed. It replaces the cases which had gaps with the values given by the SVD. Note that, same as with the other incoming modes, this will iterate `niter` times but it will stop at $iter = i$ if the mean squared difference between the values at $iter_{i-1}$ and $iter_i$ for cases which had gaps is below the `threshold1`.

5. The huge loop start and the other modes are computed iteratively (thus from 2 to `ncomp`) with the same iteration scheme as for the 1st mode. But the gaussian smoothing (see `?smooth_gauss`) has been applied at the beginning of each (nested) iterations on the new matrix given by step 4. We subtract the new average (over the stations) on each case for stability.

6. Finally, we recompose the data by doing the inverse transformations (see step 1 and 2) and we return a list with the new matrix with filled gaps (and also he distribution of the weights of the initial SVD in a vector as the other component of the named list).

Square Root transformation

The provided code had made a square root transformation of the initial data, i.e.

$$y_{\text{new}} = \sqrt{y + 1} \tag{1}$$

where y contains the initial data matrix. Then, we process the algorithm on y_{new} and we get the filled values y_{filled}. Finally, we transform back :

$$y_{\text{final}} = (y_{\text{filled}} \times y_{\text{filled}}) - 1, \tag{2}$$

where the "-1" term applies to all entries of the obtained matrix.

Following again Dudok de Wit (2011), this is desirable since this will increase the linear correlation between the records. Actually, we made this mistake of omitting this step, and we started writing a "vignette" based on these results. It can be retrieved in the **/vignettes/OLD/** folder of the Github repository. We let these results available as we think that indeed, it could be interesting to see how the results are different.

1.2 Example : Uccle2

We present an example of the output we can get from this function : see figure 4 that we leave in appendix as these kind of individual results are not much insightful. The information provided is not really effective and this is the reason why we developed the Shiny application discussed in section **0.2** to be able to maximize, not only the information we can have from the method, but also the access to this information and its readability. With our knowledge on the subject, we were not able to arbitrarily choose one (or a group of) station(s) to display and to centralize analysis from it.

The comments we can do from this graph on figure 4 are :

- The period from the start of the series (1924) and $\approx$ 1945 is quite unreliable because there are only a few stations that have observations at this time. This happens for most of stations and this is why at this period the filled SSN does not follow the `solar_cycle()`.

- The filled values do not necessarily follow perfectly the observed values (due in particular to the influence of other stations), and sometimes there happen to be a very large values when the solar activity is high (one is above 800 here around 1990 !).

2 Languages Comparisons

As Matlab® is a commercial product, we were not able to bring it to the comparison and hence, we will 'only' compare the two most used languages in data analysis, Python and R.

Based on the (slightly refined) script in Python given by my project's partner Caroline, we have constructed the plot in figure 1 showing the computation time complexity of `interpolsvd_em()` with its default values (see section **1.1**). We selected the 7 stations which have more than 17000 observed values and we took observations between 18000 and 22000 to balance between computation time, accuracy of the comparison and the fact there must be enough observed values for the algorithm to converge. The graph is depicted in the following figure 1 :

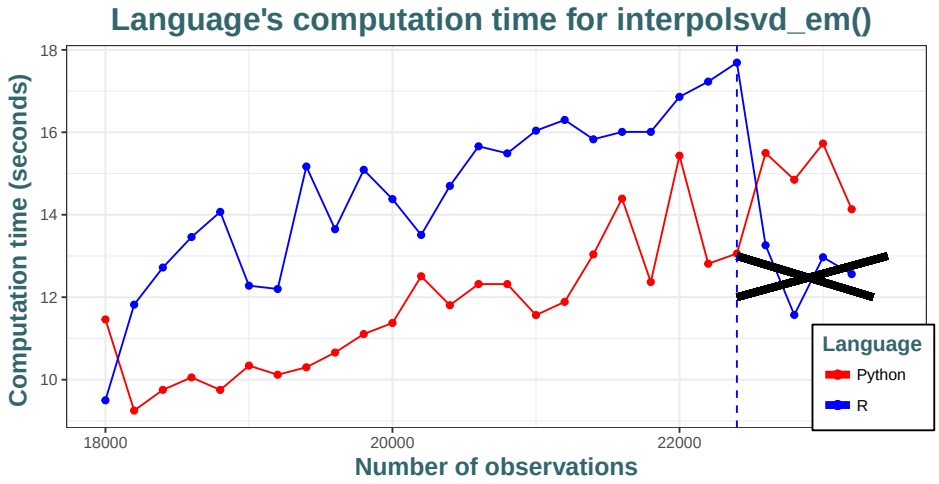


Figure 1: *shows the computational complexity analysis between Python and R for the implemented method* `interpolsvd_em()`.

We have to be prudent concerning this comparison as the computation time is very variable and it depends on lots of conditions. Hence, we must be prudent when interpreting it. We illustrate this here, because we did the 'error' to do another computation in the same time as doing the R simulation (and not in parallel) and hence, the big decrease around 22500 (that is, right to the vertical dotted line, i.e. the region that is crossed in black) is not reliable.

Regarding the R computations, we had to take a slightly refined function (which is a bit faster) to perform the tests as the R IDE were crashing all the times during the simulation. We expect our machine not to be powerful enough and this shows us how huge is the computation of this method. The function we take is the one we will use for cross-validation (`interpol_CrossVal()`, see section **4**).

We remark that before the first iteration, R is faster and this is perhaps thanks to the more efficient implementation of the function, which is notably smoother before the main loop.

From our point of view, as R is statistical-oriented language, it is more 'efficient' (and intuitive) in the sense that more refined methods exist to compute the large amount of statistical this method requires. An example of a R method is `sweep()` which permits to easily compute the normalization steps.

2.1 Improvements : Low-level Language

We have seen that the methods perform slowly. Hence, it is very important to take advantage of compiled languages such as C++ which should easily do the job. R facilitates implementation of C++ since the apparition of `Rcpp` thanks to Eddelbuettel et al. (2011). Moreover, the `RcppEigen` package (see Bates et al. (2013)) provides a huge library for numerical linear algebra. Together with other similar packages such as `RcppArmadillo`, and assuming a global knowledge of the C++ language [6], the work would be very precious to consider.

From that, drastic gains in computation time are expected. When looking at the computation times we had with he three languages, and when thinking of further deep cross-validation analysis (see section 4), this could lead to considerable improvements on the (hyper)parameters tuning. It is also easily possible to export some part of the functions such as those we have found quite slow (`smooth_gauss()`, see just after) and then reuse it in the main function.

2.2 Profiling

Indeed, we did some profiling in R, relying on both the package `profvis` and the profiling tool proposed by the Rstudio[7] IDE. It showed us, in particular, that the sole `smooth_gauss()` takes nearly 50% of the total time computation of `interpolsvd_em()`. An other 40% is taken for the SVD co mputation by `rank_reduce()` and the rest comes from all the (re)normalization steps made on the matrix.

Hence, assuming the SVD computation will be hard to improve, it is advised to either improve the R implementation of `smooth_gauss()` which is not computationally efficient, or to pass this function to a low-level language implementation, or even to find another smoothing method that could perform equally and which is less time consuming. This will be in fact the subject of the section **3.2**.

3 Improvements : Other Methods and Comparisons

So far, we have followed the provided method and we did not make significant departures from this method. Now, we will present three other methods that comes from different sources and that can make good benchmarking of the implemented method of section **1**. These methods include :

- A completely different method coming from C.Ritter. (section **3.1**).
- A slightly refined method from Dudok de Wit (2011) by replacing the interpolation method (section **3.2**).

[6] We sure expect this will not be a problem for physicians...

[7] see `https://support.rstudio.com/hc/en-us/articles/218221837-Profiling-with-RStudio`

- There also exist a global method that is builded by experts the **SILSO** (section **??**). This is included in the Shiny application for relevant comparisons purpose we we will not develop it here. More information can be found in `http://sidc.oma.be/silso/newdataset` .

3.1 Method implemented by C.Ritter

The method implemented by C.Ritter, namely the "Chris method", relies on the Anscombe Transformation to stabilize variance and make ≈Gaussian with constant variance the initially Poisson distributed SSN . Then, linear regression on these transformed data and the gaps are filled with the fitted values. The 'global' SSN is computed by taking the median accross stations and a new linear regression is made on these new values. Finally, imputation is made and the 'global' SSN is computed by a trimmed mean.

More information can come from C.Ritter. We still need to update this method to be able to 'functionize' for sake of generalization.

3.2 Splines as Smoothing method

This method simply replace the interpolation method from a gaussian smoothing to a smoothing with splines. Besides, this 'simple' refinement could have drastic impacts. This method comes from the fact that we have found the gaussian interpolation method too slow in the preceding section. We did not really play with the parameters (not necessary here) and we 'only' the `na.splines()` from `zoo` package.

Note that there exist a bunch of for doing interpolation and this could be easily study. Again, a cross-validation procedure (similar to that later presented in section **5**) would be recommended to choose between the interpolating method.

3.3 Comparisons

The following small table will summarize the computation time for the 3 method :

Methods →	Gaussian interpol.	Splines interpol.	Chris' method
Computation time (seconds)	305.24	203.64	≈20

where the time for Chris' method is more approximative as the methodology is quite different and we did not have time to "functionize" this method. We see that the other methods are well faster.

Visual comparisons between these methods are made through the Shiny application (see section **0.2**)

3.4 Improvements : based on SoftImpute

This is **not** (yet?) **included** in this project but there is amount of literature on this subject and there are high similarities with the method we have presented.

The idea relies on the nuclear norm regularization. This can be formulated by the following convex problem :

$$\min \ ||\hat{y}||_*, \qquad \text{s.t.} \sum_{i,j \in \omega} \left(y_{ij} - \hat{y}_{ij}\right)^2 \leq \delta, \tag{3}$$

where $\omega \subset \{1, \ldots, n\} \times \{1, \ldots, m\}$ is the set of observed entries and where the nuclear norm $||\cdot||_*$ is another approach to relax the *rank* of the matrix. Eq.(3) aims to minimize the complexity subject to a maximum error.

This method has been first presented by Mazumder et al. (2010) and a lot of research have emerged from the same idea, the so-called *online matrix completion* problem as presented in Dhanjal et al. (2014) for example, among others. The R package `SoftImpute` has also emerged from Hastie et al. (2014).

We think that this can represent a non-negligible source of information from which we can create novel ideas for improving the existing method.

4 Parameter Tuning : Cross-Validation

In this section we aim at looking more deeply at the (hyper)parameters of the model and then tune them accurately. Following Dudok de Wit (2011), the three tuneable parameters are :

- The number of significant components retained (`ncomp`).
- The number of scales into which the data are decomposed (`nsmo`).
- the embedding dimension (`nembed`).

4.1 Mathematical Notations

For smoother comprehension in this section, we re-adapt the notation of Dudok de Wit (2011). Hence, let $y_{i,j} = y(t_i, \lambda_j)$ denote the matrix of observed entries for the SSN with i rows and j columns. Thus, t denotes the time index in days and λ is the set of stations. More specifically here, it means

$$i = \Big[1/1/1924;\ 5/4/2015\Big], \qquad j = \Big\{\text{wnA3}, \ldots, \text{wnUC3}\Big\}. \tag{4}$$

for the complete series. But here for computational (and error) concerns, we let i start at ≈ 1950.

4.2 Computational Problems

A difficulty arises for memory and computational load as the complete final matrix involves a number of $i = 33333$ rows and $j = 52$ stations.

Whereas parallel computing will not improve drastically computational performance (around 50% of saved time), it is a good idea to focus on locating (parts of) the analysis on a external cluster such as proposed by Apache Spark. This could considerably decrease the computational complexity.

4.3 Methodology

Now, we present the *cross-validation* procedure which consist in checking how the residual error of the model varies with those parameters (the 3 above). Dudok de Wit (2011) focused on the normalized error :

$$\epsilon_K(\lambda_j) = \frac{1}{\sigma_{\lambda_j}} \sqrt{\frac{1}{N_{\text{gaps}}} \sum_{i=\{\text{gaps}\}} \Big(y(t_i, \lambda_j) - \hat{y}_K(t_i, \lambda_j)\Big)^2}, \tag{5}$$

where $\hat{y}_K$ denotes the reconstructed estimate matrix with `ncomp`$= K$ significant modes for the SVD. It is a kind of a normalized RMSE. We will rather use the RMSE by omitting the normalization by $\sigma_{\lambda_j}^{-1}$. This could be improved. We build the `cvFromInterpolsvd()` function to compute this, where methodology follows this path :

1. Remove randomly a fraction of the observations which are not missing for each stations (say 5%), namely the *synthetic gaps*. This set is denoted by $\{\text{gaps}\}$. This fraction must be high enough for the cross-validation criterion to be accurate, but small enough to leave stations with enough kept observed values to train the model. This is the famous **bias-variance** dilemma. This fraction is controlled by `min_keep_frac` parameter ($\in [0, 1]$) which takes the fraction of the number of observed values for the station that has the smallest amount of observed value. This will prevent errors due to the low number of observed values for a particular station.
2. Compute the `interpolsvd_em()` method on the new matrix with replaced observations as NA and save the new filled values of the synthetic gaps.
3. Compute the error criterion by the traditional *Root Mean Squared Error* (RMSE) as explained above, comparing the synthetic gaps of the filled values and the true values.

Note that we have suppressed some unnecessary steps in the initial `interpolsvd_em` function (such as the case where `ncomp`< 1, the display parameter,. . .) to slightly fasten the cross-validation. This new function is called `interpol_CrossVal()`.

Example

The figure 5 depicts the output of this method made in parallel and with repetitions (see section 4.5), for the default value of the parameters and splines smoothing. However, it gave unexpected results as we obtained decreasing (CV) RMSE with respect to the number of components, and this for all the experiments we have made on this function. This is probably an example of the possible bias (underestimation of the true error.) when this is computed for only a fraction of the observed values.

4.4 K-fold cross-validation

As the first method was a bit "limited", we have considered the "*k-fold*" cross-validation.

Indeed, with this variant, the algorithm takes the whole set of non-missing values and place them in groups (=*folds*), and then it applies the methodology on each of them iteratively, for each folds.

Note that the number of folds now also determines the number of synthetic gaps. For example, if we make it 10-fold, we will have 16000 synthetic gaps, assuming there are 160000 observed values in the considered matrix.

Bias-Variance Dilemna : the number K

As we said, in the k-fold scheme the number of folds K will now determine the number of synthetic gaps we will create. Hence,

- If K $\nearrow \implies$ number of synthetic gaps will $\searrow$ in each groups and the number of groups will then $\nearrow$.

Hence, this will induce more bias but make the results less variable, and the reverse if K$\searrow$.

Examples

We have run several simulations with different 'parametrizations'. The outputs are shown in figure 2, together with their parametrizations and the computation time. For all simulations, we took the 5 stations that has the largest number of observed values.

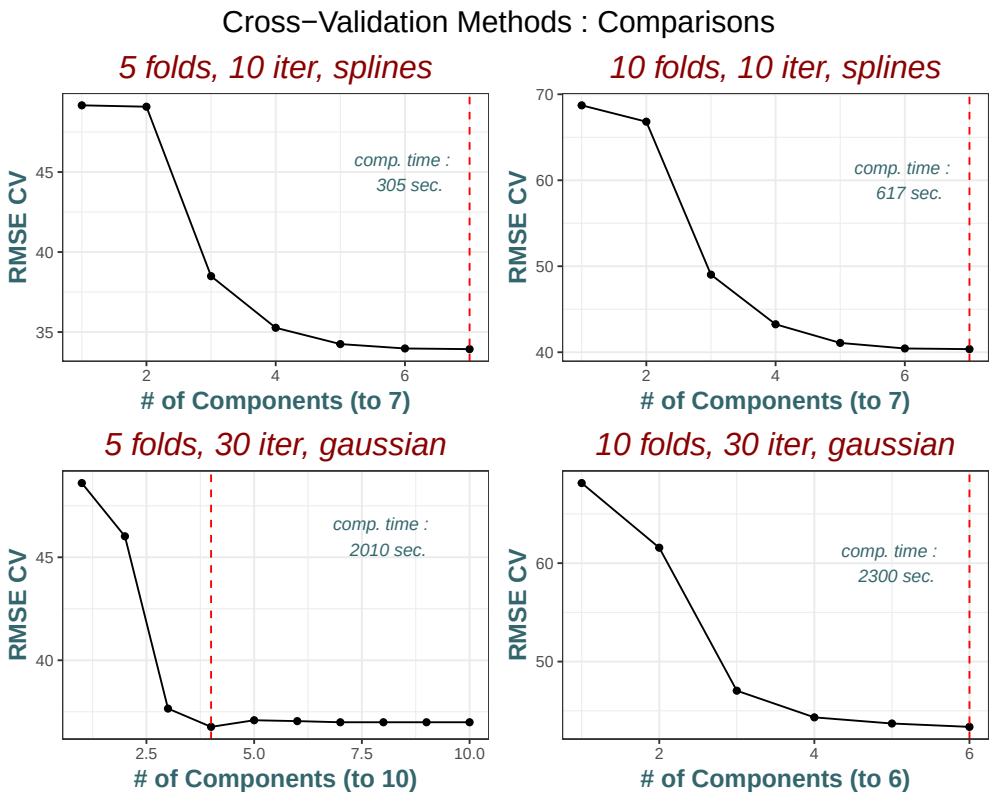


Figure 2: representing the outputs of 4 k-fold cross-validation procedures with `cvFromInterpolsvd_kfolds()` .

The results are 'weird' and we remark that the influence of the (controlled) parameters are in fact very important. In fact, we retrieve the initial finding of "4 components" only in one configuration... This must be analysed further and we put doubt on this finding. The next subsection will try to reduce the variability.

An interesting finding may be that we forgot to change the seed between these 4 computations. Hence, the "random" sampling of the NA indices are the same for all the 4 methods...

4.5 Repeated Cross-Validation

One could consider the usual cross-validation giving too variable results. Hence to obtain more accurate estimate of the error, we consider a repeated cross-validation scheme (see e.g. Burman (1989) for detailed results). The drawback of this scheme is the computational load which is roughly multiplied by the number of repetitions.

Hence, we did it in **parallel** in R, thanks to the `foreach` and the `doParrallel` packages. It actually saved us $\approx 50\%$ of computation time by using 7 cores. We did 10 repetitions but note that it would have been more efficient to do a number of repetitions which is a multiple of 7. .

The results are shown in the same figure 5 as before but there were no 'improvements" here, compared with the method with no repetitions (not shown here), thus nothing more to say than in section **4.3**. Relying on the simple `cvFromInterpolsvd()` is too short.

4.6 Improvements

Repeated K-fold and other

As we found so variable results in figure 2 it would be interesting to do the repeated cross-validation scheme on the K-fold CV in order to get more reliable results. But we will need cores....

Individual tuning vs Grid tuning

Following Dudok de Wit (2011), the two other parameters are not really important. Hence, he proposed the individual (sequential) tuning method we have followed, fixing the other parameters.

This idea now would be to take a grid for the 3 parameters of interest. And hence, for each "triplet" of values, compute the algorithm and see how the RMSE varies. But this will drastically make the algorithm more complexity as it will roughly follow $\mathcal{O}(n^3)$ where n is the (assumed equal) grid length for each parameter.

5 Conclusion

During this project, lots of results have been found and we are now questioning the results found in Dudok de Wit (2011) concerning the optimal parameters.

We note that a key element of the method that we presented in section 2 is the assumption of a linear relationship between records of different stations. We have not strictly checked this assumption and this

should interesting to verify it, to perhaps better identify an other smoothing method as we found the one used is very slow.

Considering the cross-validation is an important issue and we have to rethink the implementation of the methods which are too slow to obtain reliable results. Moreover, the cross-validation criterion can be thought as a minimal check but it will not save us if there are serious non-ignorable missingness (e.g. large values more likely than small values to be misreported), but it can be thought of as a minimal check.

References

Douglas Bates, Dirk Eddelbuettel, and others. Fast and elegant numerical linear algebra using the RcppEigen package. *Journal of Statistical Software*, 52(5):1–24, 2013. URL http://cran.irsn.fr/web/packages/RcppEigen/vignettes/RcppEigen-Introduction.pdf.

Prabir Burman. A comparative study of ordinary cross-validation, v-fold cross-validation and the repeated learning-testing methods. *Biometrika*, pages 503–514, 1989. URL http://www.jstor.org/stable/2336116.

F. Clette, L. Wauters, D. Berghmans, A. Koeckelenbergh, R. a. M. Van der Linden, and P. Vanlommel. From the Wolf number to the International Sunspot index: 25 years of SIDC. *Advances in Space Research*, 40(7): 919–928, 2007. doi: 10.1016/j.asr.2006.12.045,10.1016/j.asr.2006.12.045. URL https://publi2-as.oma.be/record/233.

Thierry Dudok de Wit, Laure Lefvre, and Frdric Clette. Uncertainties in the Sunspot Numbers: Estimation and Implications. *Solar Physics*, 291(9-10):2709–2731, November 2016. ISSN 0038-0938, 1573-093X. doi: 10.1007/s11207-016-0970-6. URL http://arxiv.org/abs/1608.05261. arXiv: 1608.05261.

Arthur P. Dempster, Nan M. Laird, and Donald B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977. URL http://www.jstor.org/stable/2984875.

Charanpal Dhanjal, Romaric Gaudel, and Stphan Clmenon. Online Matrix Completion Through Nuclear Norm Regularisation. *arXiv:1401.2451 [stat]*, January 2014. URL http://arxiv.org/abs/1401.2451. arXiv: 1401.2451.

T. Dudok de Wit. A method for filling gaps in solar irradiance and solar proxy data. *Astronomy & Astrophysics*, 533:A29, September 2011. ISSN 0004-6361, 1432-0746. doi: 10.1051/0004-6361/201117024. URL http://www.aanda.org/10.1051/0004-6361/201117024.

Dirk Eddelbuettel, Romain Franois, J. Allaire, John Chambers, Douglas Bates, and Kevin Ushey. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL http://r.adu.org.za/web/packages/Rcpp/vignettes/Rcpp-introduction.pdf.

Trevor Hastie, Rahul Mazumder, Jason Lee, and Reza Zadeh. Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares. *arXiv:1410.2596 [stat]*, October 2014. URL http://arxiv.org/abs/1410.2596. arXiv: 1410.2596.

Rahul Mazumder, Trevor Hastie, and Robert Tibshirani. Spectral Regularization Algorithms for Learning Large Incomplete Matrices. *J. Mach. Learn. Res.*, 11:2287–2322, August 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1859931.

A Appendix

A.1 Figure : Shiny application Example

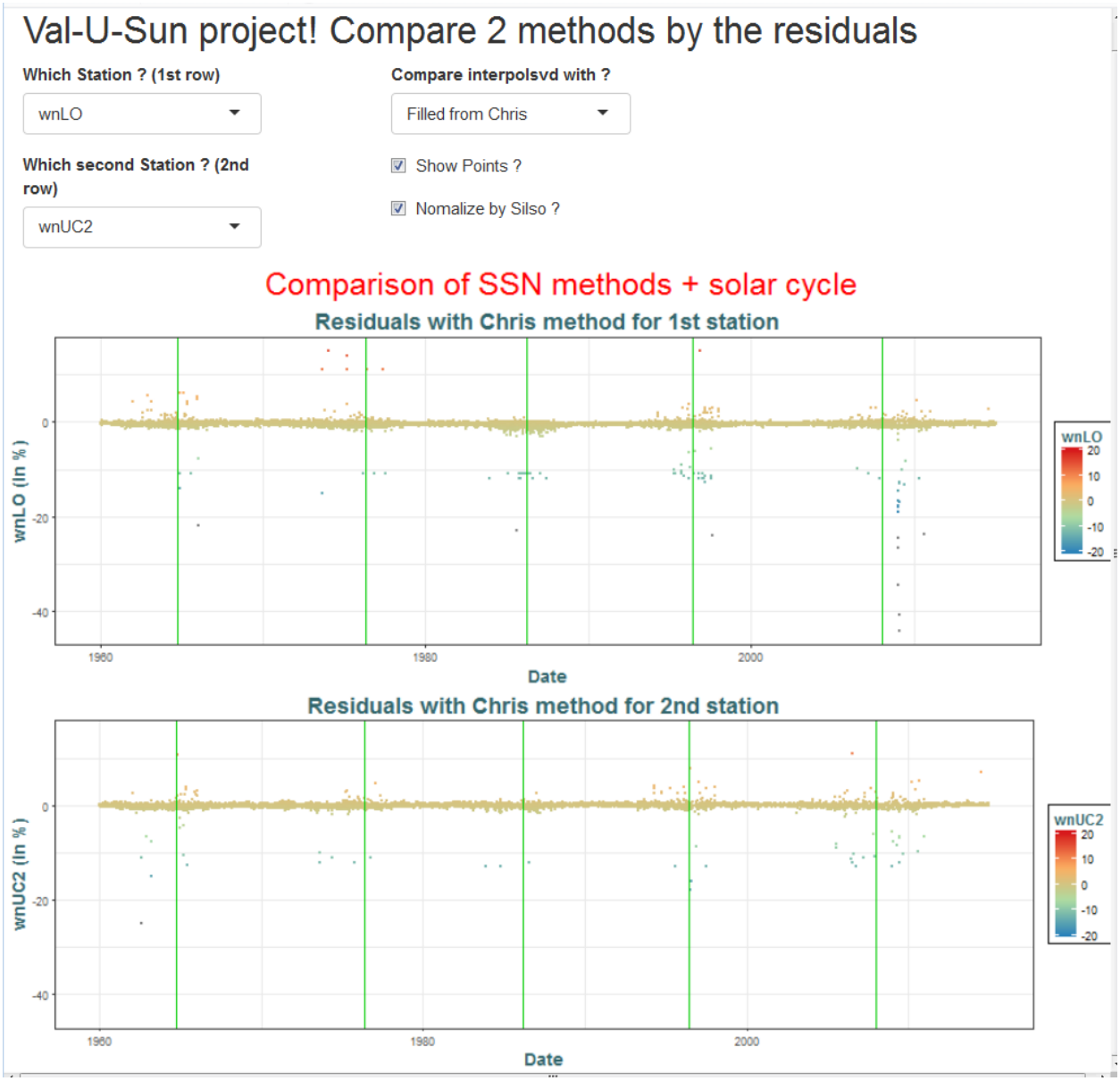


Figure 3: Static *printscreen* of the created application comparing our (re-)implemented imputation method with the imputation method of C.Ritter. This compares the station of Locarno with Uccle2 with points (instead of lines), put on the SILSO scale. Here, a major remark goes for a series of observation in January 2009 in Locarno. This means in this period we have days that have quite higer SSN with the method of Chris and this must be inspected (...)

A.2 Figure : Example of the filling method for Uccle2

A.3 Figure : Cross-Validation example

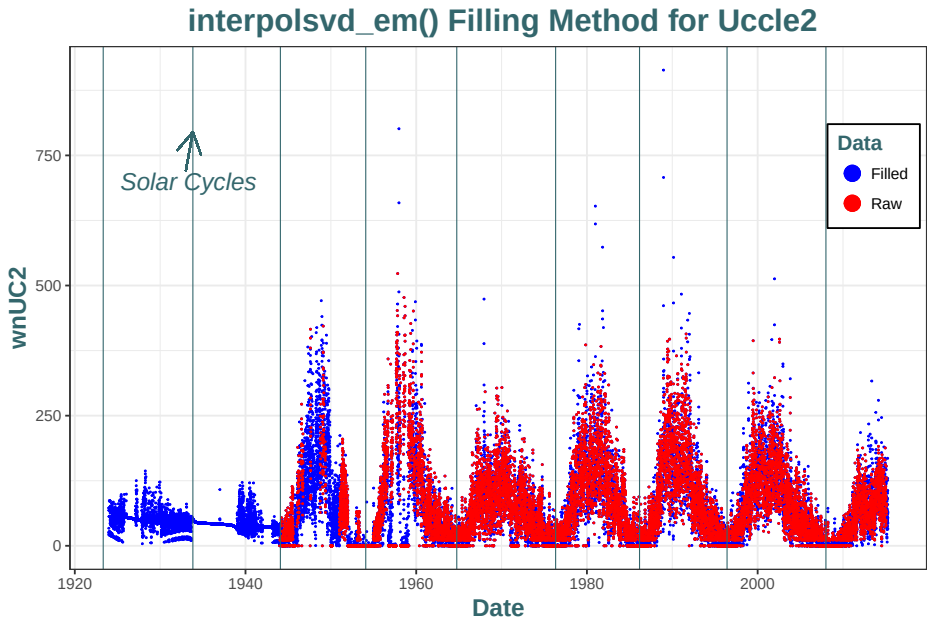


Figure 4: Representing the comparison between the filled values from the `interpolsvd_em()` method (in blue) and the observed values (in red).

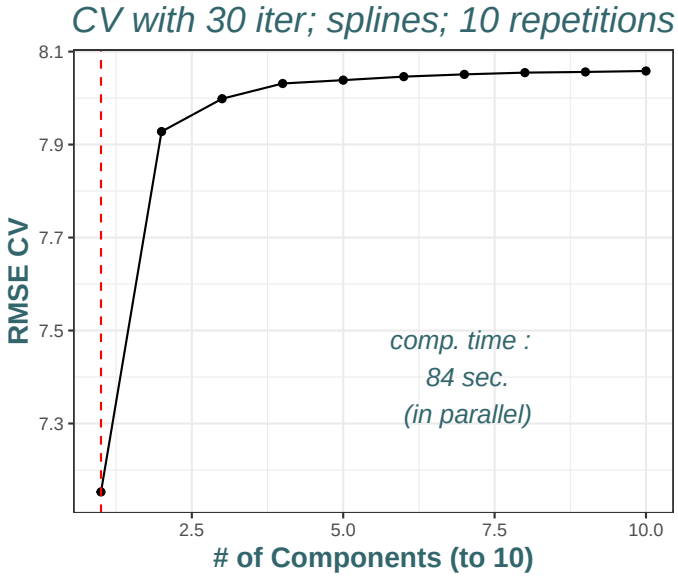


Figure 5: Repeated cross validation with the splines smoothing method (30 iterations), and 10 repetitions

A.4 R Package : /data

We saved the important data that have been computed during the project with the methods we have used. This allows to use them inside the R package.
Each file contains one R object which has the same name of the file without the extension `.RData`.
So far, the following files are stored in **/data/** :

- `data.mat.RData` : The complete SSN data.frame in matrix form with all stations in columns. The first columns represent the time : "year", "month", "day", "decdate", "Date".
- `data.mat2.fin.RData` : The complete SSN data.frame in matrix form with all stations **that have enough observation,** that is more than 5%, thus 52 columns and no columns for the time index. The observations are ordered by time. Used to compute the function on it.
- `SSN_filled_all.RData` : The value given by the algorithm `interpolsvd_em()` (see section **1.1**) for the 52 kept stations with filled gaps, for all the time period we had at hand (since 1924).
- `z_final60.RData` : Same as above but values are taken since 1960. (for comparisons with Chris's method).
- `silsoSSN.RData` : data.frame with the value of the SILSO index, from 1924 and a column for decdate.
- `ssn_chris.RData` : data.frame with 120 columns giving the values of the SSN with filled gaps for 59 stations with the method of Chris' (see section **3.1**), starting at year 1960. The other columns are boolean showing if the value has been filled or not.
- `ssn_splines.RData` : Matrix giving the 52 stations with filled values by the method of splines (see section **3.2**) for the full period since 1924.
- `ssn_splines81.RData` : Matrix giving the 52 stations with filled values by the method of splines (see section **3.2**) since 1981.

A.5 R Package : R scripts

Additional informations on the use of the functions, interpretations, etc... are available directly in the scripts (comments).

Located in the /R folder : i.e. the functions that are made available by the package for the user. Since they are used through the `roxygen2` package, the structure is adapted. These are all in `.RData` format :

- `Cross-ValidationFun.R` : contains the functions which allow to compute the cross-validation.
- `cross-val_kfolds.R` : contains the function allowing to compute the K-FOLD Cross Validation.
- `UsedFunc.R` : contains all the functions that are used in the final `interpolsvd_em()` method, that have been translated from Dudok de Wit (2011).
- `interpolsvd_em.R` : contains the main method allowing to directly compute the method of Dudok de Wit (2011) with a gaussian smoothing method for interpolation.
- `interpol_splines.R` : contains the same method as `interpolsvd_em()` but with a spline smoothing method for interpolation, which is quite faster.

- `interpolsvd_em2.R` : Slightly different version of `interpolsvd_em()`, but not used from now.
- `runExample.R` : Function which enable to run the shiny application.

Located in the /Scripts folder : i.e. some various scripts that have been realized during the project :

- `Comparison_caro.Rmd` : Markdown script to compare the results of CARO for the initial function and the result from R. This allowed to see the error of "not using the squared root transformation".
- `Cross-Validation.R` : Scripts which computes the cross-validation schemes used in the project.
- `Test_interpolem_piss2.R` : Script using and testing the initial methods, drawing some plots, etc...

Sunspot Number
Exploration of the uncertainties

Lorraine Dawirs

April 2017

Executive summary

Sunspot number has been studied for years as a record of sunspot activity. Sunspot number is based on the observation of multiple observers across the world. Unfortunately, there are uncertainties about this number coming from two majors sources: the observers (dispersion errors) and the source itself (time-domain errors). This work attempts to decompose and studies these errors. It is based on an article from Thierry Dudok de Wit [3].

As expected, we shown that both errors depend on the sunspot number and are higher at high sunspot number. It also indicates that dispersion errors are more important than time-domain errors and that the difference is more important at high sunspot number. Also, as expected, replacing missing values by an estimation leads to higher dispersion errors. Unexpectedly, we shown that variance stabilization transformation (square root, anscombe and Haar-Fisz) do not break the relation between dispersion errors and sunspot number even if global variability looks more random.

We also shown that, on the opposite of the methodology used in the article, an ARIMA model is more adapted than an AR(8) model in term of model validation, AIC and RMSE. But we also found that the difference is not big.

Exact relation between errors and between time-domain error and sunspot number are station-dependent but have the same shape (square root).

1 Introduction

The sunspot number is an astronomical indicator related to the solar activity. It is the longest available record for the solar activity. Solar activity and so sunspot number have various influence on Earth. For example, solar activity is related to climate change and atmospheric drag.

Different observers among the world count, day by day, the number of sunspot since years. All these observations are used to determine and compute the International Sunspot Number [1]. As this number stand for the real sunspot number and as it is a combination of different observations, it is important to evaluate the uncertainties of these observations.

The sunspot number is calculated as the Wolf number, N_w, and has been introduced in 1848 by Rudolf Wolf. It is computed as the combination of the number of sunspot groups, N_g and the number of isolated sunspot, N_s.

$$N_w = k(10N_g + N_s)$$

where k is a scaling factor associated to each observer.

This work focuses on sunspot uncertainties and is based on an article from Thierry Dudok de Wit [3]. A part of the work consists in the reproduction of the article and another part focuses on alternative method or complementary analysis.

2 Data

As the Sunspot Index and Long-Term Solar Observations (SILSO) database is hosted, since 1981, by the Royal Observatory of Belgium, data used in this work are all available data since the 1st January 1981.

Two databases are used. The Total Sunspot Number (TSN) from the SILSO database between 1st January 1971 to 9 February 2015 is considered as the true sunspot number. The other dataset is the sunspot number observed for 52 stations during the same period.

The TSN database contains an estimation for the sunspot number for each day but the observations database contains only the sunspot number observed by the observers of the station thus this file contains missing value. Missing values can have several origins: sun not visible on the days due to whether condition, observers did not give their observations, ...

3 Methodology

This work compares different methodology for variance stabilization and studies to types of errors: observers and time domain errors.

The uncertainties about the real sunspot numbers can come from different origins. Firstly, the sun himself is a natural cause of variation. Sunspot Number, N_w, is a combination of sunspot groups and sunspots. Small spots, N_s, can emerge and disappears in a time-scale between hours and weeks while groups of spot, N_G, have a time-scale that can go to month. The sunspot number is not fixed at one days, it varies all along the time and therefore, one important part of the uncertainties of the sunspot numbers is due to the solar variation itself. It is estimated by the "**time domain errors**".

Secondly, the spots are counting by human eyes and it is another important source of variation. Two observers, even if they observed the same sun, will not count the number of groups and spots in the same way. The differences between observers is a combination of different factors that can lead to different results: telescope resolutions, seeing conditions, ... These uncertainties are estimated by the "**dispersion errors**".

3.1 Missing values

The implementation of time-series domain requires data without missing values. As the observations data from the different observers contains a lot of missing values, we cannot only consider period and station for which we do not have missing values.

Therefore, it is necessary to fill the missing values by an estimation. The missing values estimation method used is described in an article from Thierry Dudok de Wit [2].

3.2 Variance stabilization

The sunspot number follows the solar cycle and consists in time-series data. Therefore, major of analysis require a homoscedastic variance with time. Sunspot number are typically heteroscedastic because variance increases with sunspot numbers. One of the goal of this analysis is to compare three different variance stabilization method and their effect on errors.

The three different method for stabilization are the following:

- Square Root transform: $\sqrt{SSN}$
- Simple Anscombe transform: $2\sqrt{SSN + 3/8}$
- Haar-Fisz transform: based on wavelet (more information in appendices 10.1.1)

3.3 Dispersion errors

This error, as explain previously, attempts to explain the part of uncertainties due to the observer's conditions. The methodology used in this work is the same methodology than in the article about uncertainties in the Sunspot Number from Thierry Dudok de Wit [3].

To estimate the dispersion, the first step consists in scaling the sunspot number observed by each station at each time to the International Sunspot Number from SILSO, ISN.

$$N^*_{W,i}(t) = \gamma_i N_{W,i}(t) \text{ where } \gamma_i = ISN/N_{W,i}$$

Where γ_i is estimated by total least squares.

Then, for each station i, the residual errors are computed as:

$$\epsilon_i(t) = N^*_{W,i}(t) - ISN(t)$$

Finally, for each time, t, the time-dependent standard deviation is obtained:

$$\sigma(t) = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}\epsilon_i^2(t)}$$

We only consider, $\sigma(t)$, for which we have at least 2/3 of observations.

3.4 Time domain errors

This error is focus on the uncertainties due to the sun itself. It is computed as the prediction errors based on a time-series model.

Two time-series models have been taken into account (more information about these models are available in appendices 10.1.2):

- auto-regressive model with 8 parameters: AR(8)
- auto-regressive - integrated - moving average model: ARIMA

For the **auto-regressive model AR(8)**, the 8 parameters $(a_1, ...a_8)$ of the model are estimated first and then they are used in order to predict "new observations":

$$\hat{x}(t_i) = a_1 x(t_i - 1) + a_2 x(t_i - 2) + ... + a_8 x(ti - 8)$$

An **ARIMA model** is characterized by three meta-parameters : p, d and q that define an ARIMA(p,d,q) model. p is the number of auto-regressive parameters, d is the degree of differencing and q is the number of moving average parameters. In the following work, d will be fixed to one. Therefore, the prediction model is defined as:

$$\hat{x}_t = \mu + \varphi_1 x(t-1) + ... + \varphi_p x(t-p) - \theta_1 e(t-1) - ... - \theta_q e(t-q)$$

For each new observation estimated, the **time-domain error** is computed as the difference between the real sunspot number observed and the sunspot number predicted :

$$\eta(t_i) = x(t_i) - \hat{x}(t_i)$$

4 Variance stabilisation

Wolf number follows solar cycle and is collected each day by different stations. The sunspot number of one particular day is depending on the solar cycle and on previous observation. One solar cylce takes almost 11 years. As each sunspot number depends on the previous sunspot numbers, time-series analysis is required. Time-series analysis can deal with high correlation between successive observations and are made for temporal data. It requires equal variance over time. Figure 1 represents the variation of the data over time. Each boxplot is made on 1000 following observations. If variance

were constant over time, each box would have the same size even if they aren't at the same level. Here, it appears that when the sunspot number is lower, the box is smaller and inversely. It means that the variance depends on the time. Therefore, it is necessary, in order to apply time-series analysis, to stabilize the variance over time.

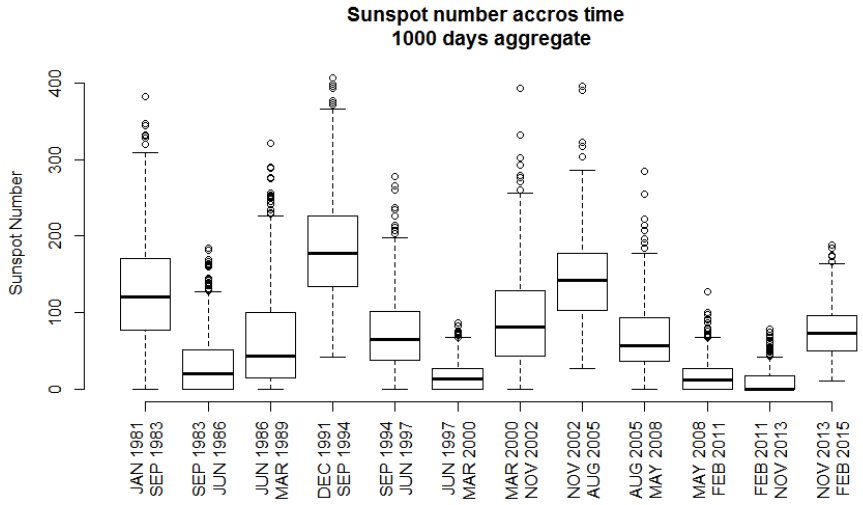


Figure 1 – Boxplot of sunspot numbers for UC2 stations. Each boxplot is an aggregation of 1000 consecutive days.

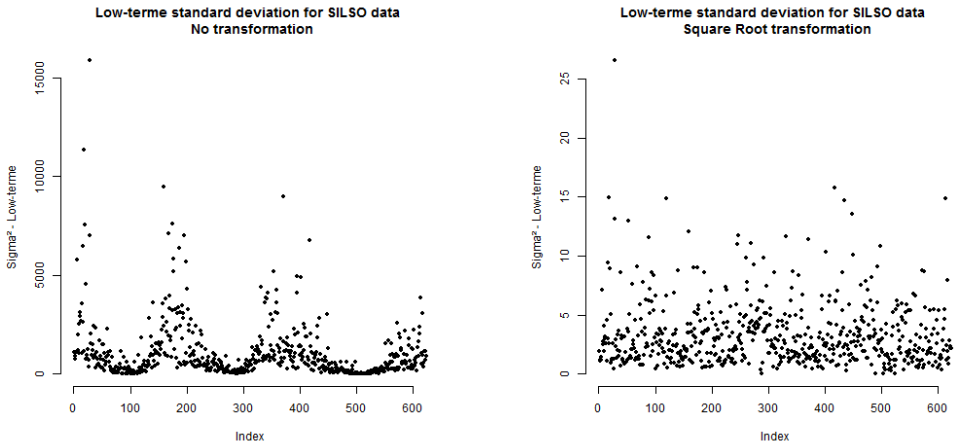
As variance is time-dependent, three variance stabilization have been compared: square root, simple anscombe and Harr-Fisz as presented in the methodology section.

4.1 Short-term variability

Figures 2 and 3 represent the short-term variance of International Sunspot Number before (a) and after transformation (b-c-d). Short-term standard deviation is computed as the classical standard deviation but on a subsample of data. Here, it is estimated on subgroups of 100 observations.

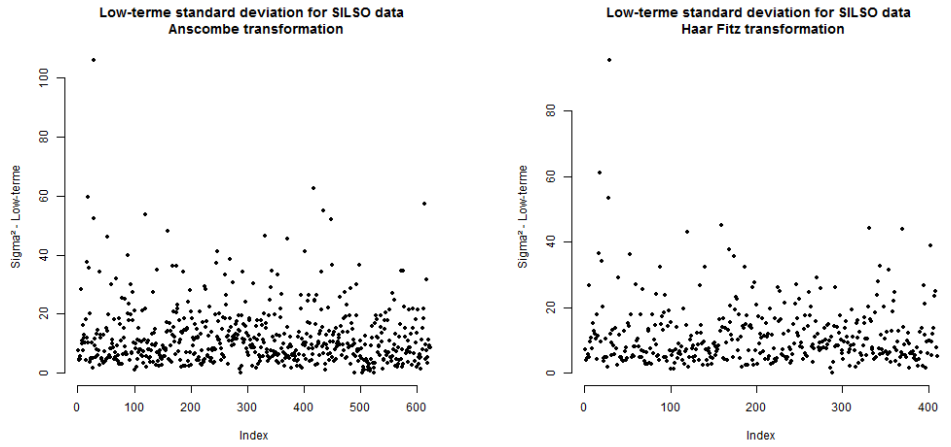
On the top-right [figure 2(a)], when no transformation is applied to the data, it is clear that the short term standard deviation is not constant over time. Indeed, we can retrace the solar cycle. Lower variations correspond to lower sunspot number and higher variations correspond to high sunspot number.

On the top-left [figure 2(b)], when the standard deviation is computed after square root transform of the data, we cannot distinguish the solar cycle anymore. Even if the short-term variability is not fully constant over time, it seems to be more random that when no transform was applied. Same conclusions are available for data after anscombe transform [figure 3(c)] and after Haar-Fisz transform [figure 3(d)].



(a) (b)

Figure 2 – Comparison of short term variability before (a) and after (b) square root transformation



(a) (b)

Figure 3 – Comparison of short term variability after (c) anscombe transform and after (b) Haar-Fisz transformation

5 Dispersion errors

Different data transformations were applied. The goal of these transformations was to find the more appropriate way to stabilize the variance. The homoscedasticity of the data is not required for the dispersion errors analysis but is required for time-series models. In order to compare directly the two sources of errors, same transformation will be used for both analysis.

The dispersion errors are studied on three different axis. First, we discuss about the dispersion errors based on data without any transformation. Secondly, we discuss the effect of transformation on dispersion errors and finally we discuss the effect of replace missing values by an estimation on dispersion errors.

5.1 Dispersion errors without transformation

Methodology used to compute the dispersion errors is detailed in the methodology section. On the article [3], dispersion error is computed based on 13 stations that "exhibit high and rather uniform

time-coverage between 1967 and 2014". On this work, we did not follow the same strategy and we choose to use all the 52 stations.

Figure 4 represents the evolution on time of the dispersion errors. As expected, it clearly shows that this error is time dependent. We also observe that the dispersion errors (black points) follow the sunspot number fluctuation (blue line). High sunspot numbers are related to high standard deviation and inversely. It indicates that, in average, observers make more mistakes when then have to count a lot of sunspot number or group and make less mistakes when the sunspot number is lower.

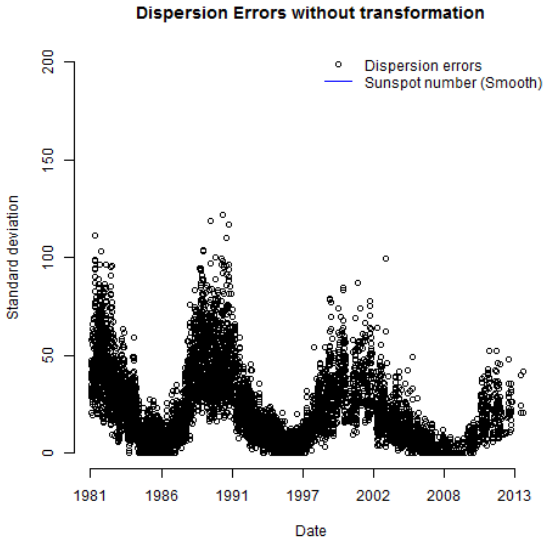


Figure 4 – Dispersion Errors without transformation of sunspot number.

5.2 Effect of transformation on dispersion errors

In this part, we look at the effect of the variance stabilization on dispersion errors. Haar-Fisz requires absence of missing values and so only two transformation will be compared: square root and anscombe.

On figure 5, dispersion errors of square root data (a) and anscombe data (b) are represented. It shows that, even if the variance is globally stabilized and is not dependent on the sunspot number, dispersion errors is still depending on the sunspot number.

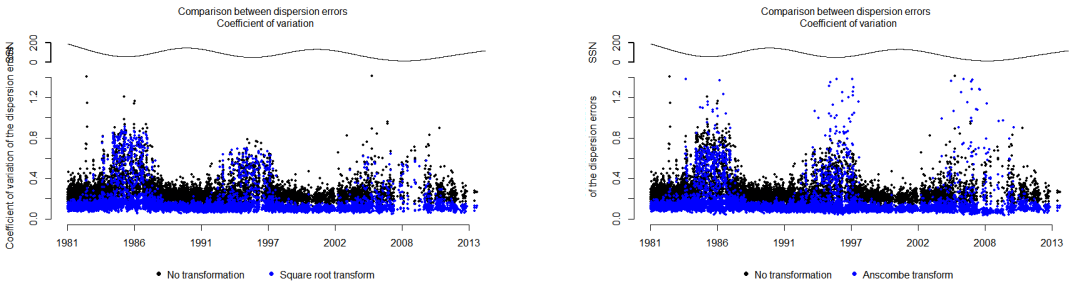
(a) (b)

Figure 5 – Comparison of dispersion errors after square root transform (a) and anscombe transform (b)

Dispersion errors for in the different case cannot be directly compare because of difference in the scaling. To face this issue, dispersion errors can be expressed as a coefficient of variation:

$$\text{Coefficient of variation } = \frac{\sigma(t)}{SSN(t)}$$

On figure 6, coefficient of variation of dispersion errors are represented before (black points) and after square root transformation (a - blue points) or after anscombe transform (b - blue points).



(a) (b)

Figure 6 – Comparison of dispersion errors express as a coefficient of variation before (black points) and after square root transform (a) and anscombe transform (b)

These figures clearly show that the dispersion errors are time-dependent. Ratio of dispersion errors on sunspot number is not constant over time and appears more important when the sunspot number is low. Figure 6 (a) indicates that ratio is less important when data are square root transform. Similarly, figure 6 (b) indicates that ratio is less important when data are anscombe transform. Form both transformed data, it also appears that the coefficient of variation is less important when the sunspot number is higher. It is link to the fact that for both transformation, we take the square root of the sunspot number. Therefore, 2 high sunspots numbers, even with a "big" difference between them will be "shrink" to small and closer values. Difference between the observed sunspot number and the "true" sunspot number appears lower.

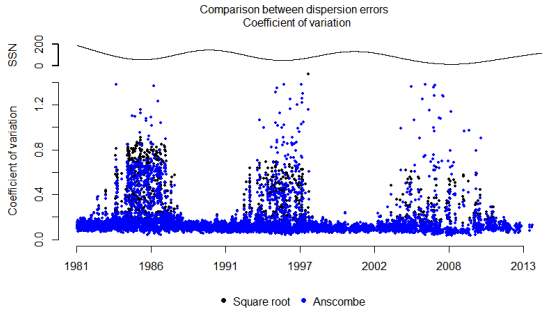


Figure 7 – Comparison of dispersion errors express as a coefficient of variation after square root transform (blue points) and anscombe transform (black points)

Figure 7 represents the coefficient of variation of the dispersion errors on the sunspot number. For high sunspot number, both transform leads to similar values. For low sunspot number, anscombe transform gives bigger coefficient of variation.

5.3 Effect of replace missing value on dispersion errors

As explain the precedent part, Haar-Fisz transform requires absence of missing values. Therefore, the effect of the estimation of missing values will only be observed for data without transformation

and with square root and anscombe transform.

Figure 8 compares the dispersion errors with and without missing values replaced for initial data.

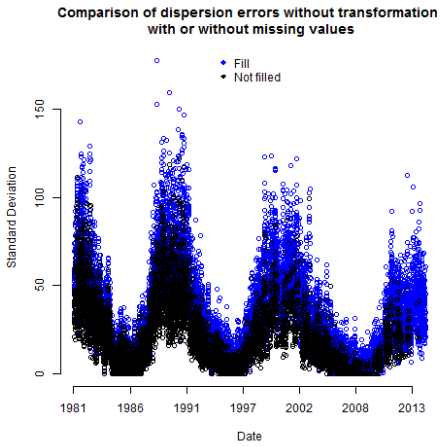
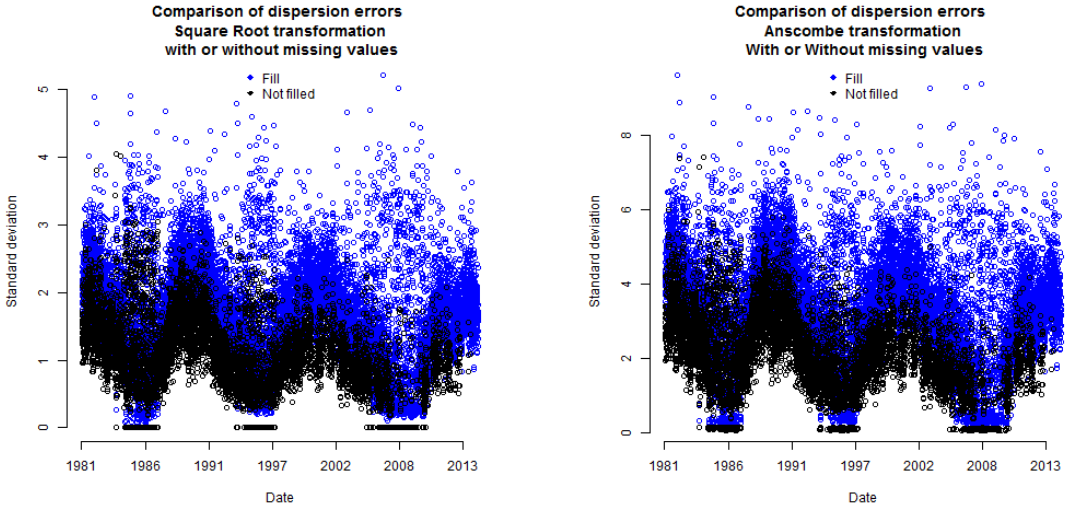


Figure 8 – Comparison of dispersion errors before (black points) and after have estimated missing values (blue points) - Based on initial data - No transformation

On the look of figure 8, it appears that considering no data-transformation, replace missing values by an estimation increases the dispersion errors. This effect is observed for low and for high sunspot number.



(a) (b)

Figure 9 – Comparison of dispersion errors before (black points) and after (blue points) the replacement of missing values. (a) data after square root transform and (b) data after anscombe transform

Figures 9 also represent the dispersion errors before and after the replacement of missing values but are based on square root data (a) and anscombe data (b). Same observations is available: as expected, replace the missing values increases the dispersion errors. This effect seems more important for lower sunspot number.

6 Time-domain errors

Time-domain errors are prediction errors of the best ARIMA model according to AIC criteria. AIC and parameters are available in appendices 10.2.1 By this error, we attempt to represent the uncertainties in sunspot number due to the sun itself.

In the article of Dudok de Wit et al. [3], time-domain errors correspond to the prediction errors based on AR(8) model.

In this work, two types of model were compared: AR(8) and ARIMA. The parameters signification and theoretical model are available in the methodology section. Models were compared based on AIC (Akaike Information Criterion) criterion. The best model for each station is estimated in order to minimize this criterion.

If we compare the different models obtained in term of AIC and in term of RMSEP (Root Mean Square Error of Prediction), it appears that, in all case, ARIMA models give better results (minimization of AIC criteria). If we compare the different transformations applied in term of RMSEP, square root transformation gives better results (minimization of RMSEP criteria). It is important to notice that the effect of transformation is not very important and that the difference between stations is more important. Full tables with results is available in appendices 10.2.1.

Relation between errors and sunspot number and comparison between errors are studied on non-transformed data and ARIMA model.

7 Relation between errors and sunspot numbers

7.1 Dispersion errors

Plotting the sunspot number on the dispersion errors reveals a shape of square root relation [figure 10]. Relation between dispersion errors and sunspot number is based on non-transformed data. Best modelization is obtained with the red model: $SSN = \alpha + \beta * \sigma_{Disp}(t)^{\gamma}$

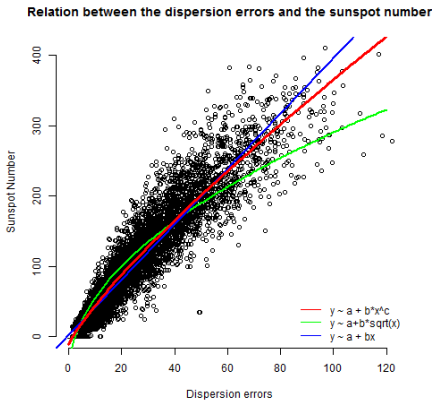


Figure 10 – Relation between dispersion errors and sunspot numbers - Data without transformation

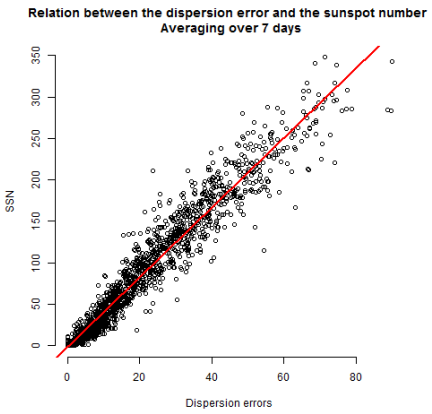


Figure 11 – Relation between mean of dispersion errors and sunspot numbers - mean of 7 days

The estimated model is:

$$SSN = -11.7 + 8.3 * \sigma_D(t)^{0.8} + \epsilon$$

As expected this relation shows that dispersion error grows up with sunspot number.

In the previous figure [figure 10], plot is done on all data date by date. In the figure 11, each point is an average of 7 consecutives values (both for SSN and dispersion errors).

It appears that taking an average over 7 days leads to more linear model. Red model corresponds to the following linear model:

$$\overline{\sigma_D(t)} = 1.5 + 0.2 * \overline{SSN}$$

7.2 Time-domain errors

Same type of relation can be modelized for time-domain errors. Time-domain errors used here correspond to prediction error of the best ARIMA model for station based on centered data (no transformation applied).

Dispersion errors is studied on global errors and a sort of average is taken for all stations. Here, time-domain errors are station dependent and no global errors are estimated. Studied the relation between time-domain errors will be made station by station.

Figure 12 represents the time-domain errors on the sunspot number. Each point is an average of 27 consecutive points.

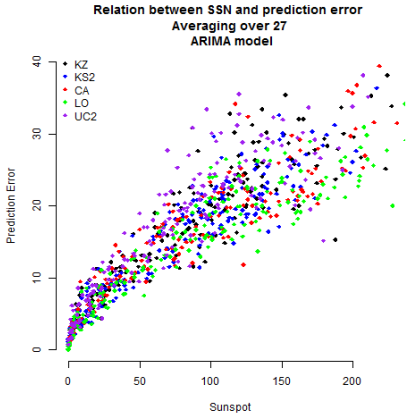


Figure 12 – Relation between time-domain errors and sunspot numbers.

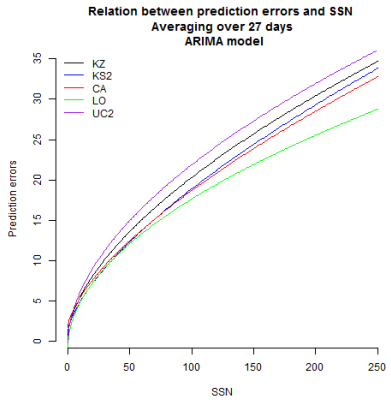


Figure 13 – Relation between time-domains errors and sunspot numbers for five stations

Globally, for the five different stations studied, relation between SSN and prediction errors has a square root shape. If we look at each station separately [figures in appendices 10.2.2], it is clear that the modelization is station-dependent but has always a square root shape. Data are modeled by the following model:

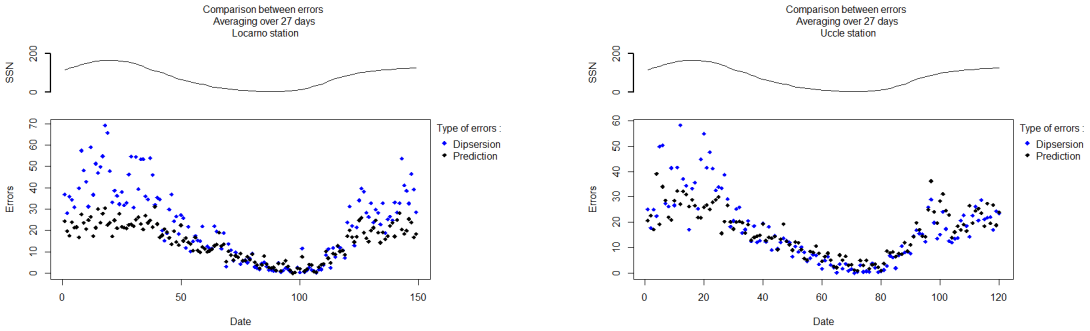
$$\overline{\sigma_T(t)} = \alpha + \beta * \overline{SSN}^{\gamma} + errors$$

Coefficient of the different models are also available in appendices 10.2.2. It appears that the five γ coefficients are around 0.5.

8 Comparison between errors

This section is based on non-transformed data for both errors and ARIMA model for prediction errors. Dispersion errors are not the global dispersion errors but correspond to the intermediate dispersion errors, station-dependent.

Figures 14 compare the dispersion errors and the time-domain errors for two stations: Locarno (a) and Uccle (b). Figures are made on an averaging over 27 days.



(a) Locarno station

(b) Uccle station

Figure 14 – Comparison between dispersion errors and time-domains errors

These figures show that not only both errors depend on sunspot numbers but the relation between errors depends on the station. On low sunspot number, both errors seem equivalent in both cases but for high sunspot number, relation is not the same. For Locarno station, dispersion error is more important at both rise of the sunspot numbers but for Uccle station, dispersion error only diverges from prediction error at the first rise of the sunspot number. It highlights the fact that relation is station-dependent.

If we represents the relation between the two errors for the five stations [figure 15], it appears that the relation also have a square root shape but that the exact relation is station dependent.

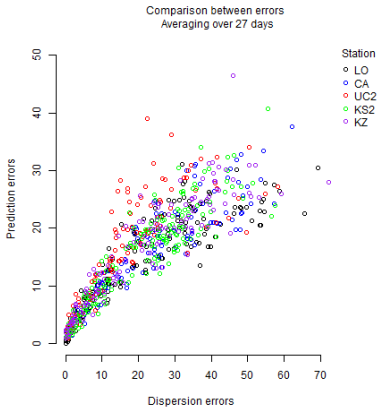


Figure 15 – Relation between dispersion and time-domain errors for five stations

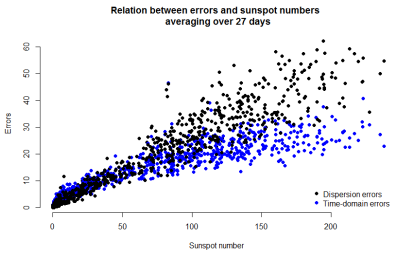


Figure 16 – Relation between errors and sunspot numbers. Data for five stations: LO - CA - UC2 - KZ - KS2

Relation between sunspot numbers and errors [figure 16] shows, as expected, that dispersion errors is more important than time-domain errors, moreover at high sunspot number.

9 Conclusion

This work focused on different aspect of uncertainties of sunspot number. Firstly, we looked at different data transformation in order to stabilize the variability. Secondly, errors have been separated in time-domain errors (representing the errors due to the sun variability) and in dispersion errors (representing the errors due to the observers). For each error, global comportment and relation with the sunspot number are studied. Impact of replacing missing values is studied for dispersion errors and impact of variance stabilization is studied for both errors. Finally, comparison between errors is performed.

In term of variance stabilization, we shown that applied one of the three transformation (square root, anscombe and Haar-Fisz) on data allows us to have data with more random variation. Without transformation, variation directly follows solar cycle but transform the data "breaks" this direct relation.

In term of dispersion errors, different points have been studied.

Firstly, as expected, we illustrated the fact that dispersion errors are directly related with sunspot number and that dispersion error is more important at high sunspot number. As expected, we also shown that replacing missing values by an estimation leads to higher dispersion errors.

Unexpectedly we also found that, even if global variation seems random, data transformation did not "break" the relation between dispersion error and sunspot number. It means that even when we worked with transformed data, dispersion error is still higher at high sunspot number and inversely.

Study of time-domains error shown that an ARIMA model is more adapted than an AR(8) model in term of model validation, AIC and RMSEP. But it also indicates that differences are not big.

As expected, we found that time-domain error depends on the sunspot number and, as for the dispersion error, time-domain error is higher when the sunspot number is high.

Relation between errors and sunspot number indicates that the relation of sunspot number on dispersion errors has a shape of square root relation and if we average the observation, this relation becomes more linear.

Relation between sunspot number and time-domain errors have shown that the relation also has a square root shape but that the relation is station-dependent.

Finally, comparison of errors shown, as expected, that both errors rise with sunspot number and that dispersion errors is higher than time-domain errors. The difference is more important at high sunspot number.

References

[1] F. Clette, D. Berghmans, P. Vanlommel, R. A. M. Van der Linden, A. Koeckelenbergh, and L. Wauters. From the Wolf number to the International Sunspot Index: 25 years of SIDC. *Advances in Space Research*, 40(7):919–928, 2007.

[2] T. D. d. Wit. A method for filling gaps in solar irradiance and solar proxy data. *Astronomy & Astrophysics*, 533:A29, Sept. 2011.

[3] T. D. d. Wit, L. Lefèvre, and F. Clette. Uncertainties in the Sunspot Numbers: Estimation and Implications. *Solar Physics*, 291(9-10):2709–2731, Nov. 2016.

10 Appendices

10.1 Methodology : details

10.1.1 Haar-Fisz transform

Haar-Fisz transform is a method used to stabilize the variance in a non-parametric way. This method consists in four major step:

1. applied an Haar wavelet transform to the data
2. estimated the mean-variance relation between the smoothing and the detailed wavelet coefficient
3. divide the wavelet detail coefficient by smooth of mean-variance relation
4. perform the inverse Haar wavelet transform

10.1.2 ARIMA and AR(8) model

An ARIMA model tries to modelize temporal data and is characterized by three meta-parameters: ARIMA(p,d,q). ARIMA stands for auto-regressive - integrated - moving average model. The three meta-parameters are the following:

- p: the number of auto-regressive parameters
- d: the degree of differencing
- q: the number of moving average parameters

The prediction model of an ARIMA is:

$$\hat{x}_t = \mu + \varphi_1 x(t-1) + ... + \varphi_p x(t-p) - \theta_1 e(t-1) - ... - \theta_q e(t-q)$$

Auto-regressive p : auto-regressive model are model that estimate a new observation on the basis of the p previous values (linear relation).

Integrated d : one of the hypothesis of ARIMA model is the stationarity of variance over time. It means that not only mean but also variance is constant in time. The best method to obtain this type of series is to differentiate the data. Here, for all stations, d is fixed to one. It means that two successive values of sunspot number are constant:

$$y(t) - y(t-1) = \mu + \epsilon(t)$$

Moving average q : moving average model indicates that data vary around a mean value. The new observation estimation is based on an average of the q latest observations.

An AR(8) model is an ARIMA model were :

- p, the number of auto-regressive parameters, is fixed to 8
- data are not differentiated: d=0
- q, the number of moving average parameters is null: q=0

10.2 Results : details

10.2.1 Time-domains errors : ARIMA models

The following table contains the best ARIMA model obtained for five stations (CA, LO, UC2, KS2 and KZ) and for the four transformation (centered, square root, simple anscombe and Haar-Fisz).

Station	Centered	Square root	Anscombe	Haar-Fisz
CA	(4,1,5)	(4,1,3)	(4,1,3)	(1,1,5)
LO	(5,1,3)	(5,1,5)	(3,1,5)	(3,1,2)
UC2	(5,1,2)	(3,1,1)	(3,1,1)	(1,1,1)
KS2	(3,1,5)	(4,1,2)	(4,1,2)	(3,1,4)
KZ	(5,1,2)	(4,1,5)	(4,1,5)	(1,1,5)

Table 1 – Best ARIMA model obtained - ARIMA (p,d,q) - p : number of auto-regressive parameters - d : degree of differentiation - q : number of moving average parameters

The following table contains the AIC criteria for the different model of the different station and transformation.

Station	Centered		Square Root		Anscombe		Haar-Fisz	
CA	6513	5874	3699	3053	3849	3207	4081	3670
LO	1742	1055	944	215	1224	565	2368	1929
UC2	13130	12533	9659	9088	9718	9148	8151	7745
KS2	8525	7852	5726	5075	5871	5225	4013	3568
KZ	8145	7468	5427	4770	5593	5012	5541	5158

Table 2 – AIC values for the best ARIMA model obtained and for AR(8) model

The following table contains the RMSE criteria for the different model of the different station and transformation.

Station	Centered		Square Root		Anscombe		Haar-Fisz	
CA	0.314	0.306	0.283	0.275	0.284	0.275	0.316	0.304
LO	0.239	0.232	0.240	0.233	0.243	0.236	0.280	0.80
UC2	0.362	0.353	0.328	0.319	0.329	0.320	0.384	0.371
KS2	0.310	0.302	0.287	0.278	0.288	0.280	0.280	0.300
KZ	0.308	0.301	0.299	0.292	0.301	0.294	0.347	0.338

Table 3 – AIC values for the best ARIMA model obtained and for AR(8) model

10.2.2 Time-domain errors : Relation between errors and SSN

Relation between sunspot numbers and time-domain errors for the five stations. Each point is an averaging over 27 days

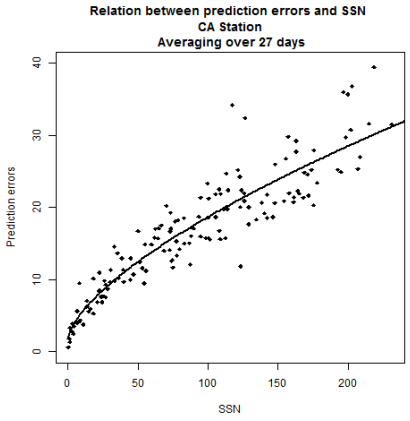


Figure 17 – Relation between Sunspot Numbers and Prediction Errors - CA station

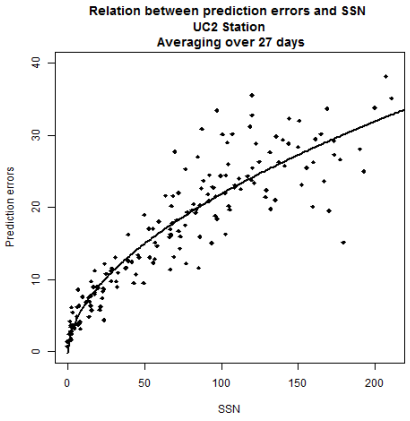


Figure 18 – Relation between Sunspot Numbers and Prediction Errors - UC2 station

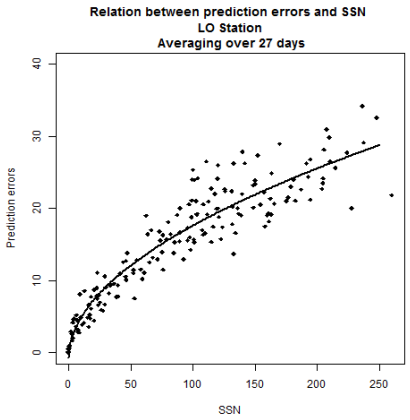


Figure 19 – Relation between Sunspot Numbers and Prediction Errors - LO station

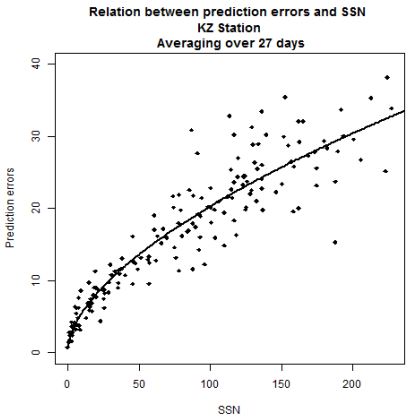


Figure 20 – Relation between Sunspot Numbers and Prediction Errors - KZ station

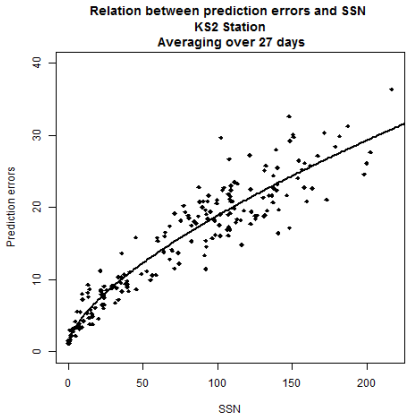


Figure 21 – Relation between Sunspot Numbers and Prediction Errors - KS2 station

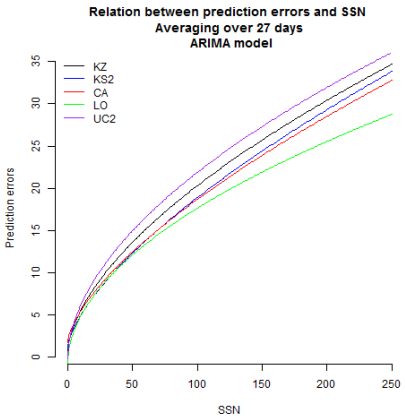


Figure 22

Theoretical model used:

$$\overline{\sigma_T(t)} = \alpha + \beta * \overline{SSN}^{\gamma}$$

Estimation of the coefficients for the five stations:

Station	α	β	γ
LO	-0.71	1.69	0.52
CA	1.75	0.81	0.66
UC2	0.18	1.82	0.54
KZ	0.54	1.27	0.60
KS2	1.09	0.84	0.66

Table 4

Analysis of the variability of the International Sunspot Number

VALUSUN project for the Royal Observatory of Belgium

LSTAT2390 Statistical Consulting - UCL

Executive summary

The Sunspot Number (SSN) is an astronomical indicator measuring the number of sunspots present on the surface of the sun on a daily basis. The purpose of this report is to analyse the uncertainty structure of the SSN data collected between 1981 and 2015 for over 40 observation stations, using a 2016 article on uncertainties in the sunspot numbers as a starting point.[1] The article divides the error in the SSN data in two parts: time-domain errors, linked to natural variations in solar activity, and observation or dispersion errors, linked to differences between observers.

A preliminary objective was to identify a method to stabilise the variance of the time series. The square root, simple Anscombe and Haar-Fisz transforms were assessed. All three methods are suitable to stabilise the variance, but the square root transform is sufficient to obtain a good model fit.

The 2016 article on uncertainties estimates the time-domain error of the SSN by fitting an autoregressive model with 8 parameters (AR(8)) on centred SSN data. This approach was implemented for a subset of 5 stations and 4 data transforms and compared with a standard time series approach based on autoregressive integrated moving average (ARIMA) modelling. In all 20 cases, the ARIMA model was a better fit for the data and more closely met the validation conditions.

However, the root mean square error in prediction (RMSEP) for the ARIMA models was only marginally smaller than for the AR(8) models. The results obtained with the standard time series approach therefore confirm the magnitude of the time-domain error estimated in the 2016 article.

Time-domain errors have a square root relationship to the SSN, while observation errors follow the solar cycle and are proportional to the SSN. Both errors are of similar magnitude at low points in the solar cycle, while observation errors tend to be more important when the solar cycle is in a high point. This highlights the importance of reducing the dispersion of observations between stations through better guidelines or the selection of a smaller selection of stations.

Two other approaches to the time-domain errors could be further explored: seasonal time series modelling and the use of one overall ARIMA model for all stations. Computing time-domain errors with one overall ARIMA model could be a more robust alternative to the AR(8) approach in future analyses.

Context and objectives

Sunspots are manifestations of solar magnetism appearing as dark spots on the solar surface. The Sunspot Number (SSN) is an astronomical indicator measuring the number of individual sunspots and groups of sunspots present on the surface of the sun on a daily basis. The sunspot data stretches back to the 17th century, with the introduction of the Wolf Number W in the 19th century. The formula W = 10 Ng + Ns is still used today to compute the SSN, with Ng the number of sunspot groups and Ns the number of single sunspots observed on a given day. Since 1981, the Royal Observatory of Belgium hosts the Sunspot Index and Long-term Solar Observations (SILSO) database. The Observatory compiles the observations made in over 30 countries to compute the SSN per station and the overall International Sunspot Number (ISN).

The overall objective of this project is to better understand the SSN data set to improve its processing. More specific research objectives identified during the project are:

1. Categorisation of the stations

[1] Dudok de Wit, T.; Lefèvre, L.; Clette, F. (2016) Uncertainties in the Sunspot Numbers: Estimation and Implications, Solar Physics, Volume 291, Issue 9-10, pp. 2709-2731 http://adsabs.harvard.edu/abs/2016SoPh..291.2709D

2. Analysis of a missing values imputation MATLAB code[2] and transposition into R and Python
3. Analysis of the sources of variability

This report addresses the third objective, using an article from 2016[3] on the uncertainties in the sunspot numbers as a starting point. The purpose is to better understand the uncertainty structure in the data by reproducing some of the analyses presented in the article, comparing them with other approaches and evaluating the results.

Methodology

Data

The analyses were conducted on a set of 52 series of sunspot observations collected between 1981 and 2015 and coming from over 40 stations that are part of the SILSO network. The measure of interest is the SSN (and not the group or individual counts) as computed daily for a single station.

The original data set includes missing values, which may be due to the weather, changes in participation in the network or other factors. These were imputed using a method based on singular value decomposition[4] for the time-domain errors analysis.

Due to the computation time needed to fit time series models, a subset of five stations was selected for the time-domain errors analysis: Locarno (LO), Uccle (UC2), Catania (CA), Kislovodsk (KS2) and Kanzelhöhe (KZ). These stations have an adequate coverage for 1981-2015 with a limited number of missing values. Uccle data (UC2) was used to illustrate the general behaviour of the SSN data.

Approach

In a preliminary phase, different variance stabilisation methods were compared as subsequent steps require a stabilised data set. The analysis then followed the structure of the 2016 article on uncertainties, which distinguishes between time-domain errors (errors over time) and observation or dispersion errors (errors between stations).

The present report will focus on the time-domain errors. In the article, time-domain errors were estimated using an auto-regressive model with 8 parameters (AR(8)) based on centred data. This approach was compared with a more standard ARIMA approach. Observation errors were computed as per the article. The errors were then compared with each other and with the SSN.

All analysis was implemented in R using additional packages, existing functions written by UCL researchers (FonctionsSeriesChrono file[5]) and custom-made code.

Variance stabilisation

Figure 1 shows the SSN for Uccle for the selected time frame. The series clearly follows the solar cycle, with high points in the early 1990s, 2000s and 2010s. The data variability is time-dependent: the variance increases with the SSN and is the largest during the high points in the cycle.

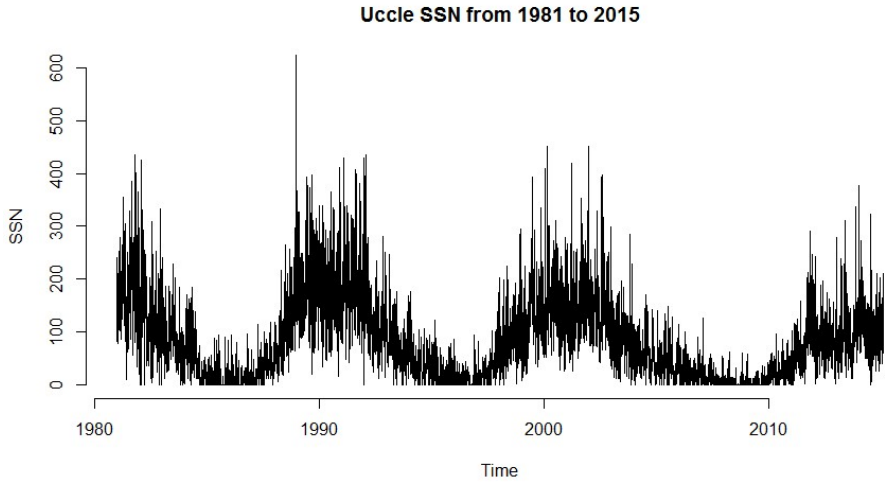
[2] Dudok de Wit,T. (2011), A method for filling gaps in solar irradiance and solar proxy data, Astronomy & Astrophysics, 533 http://adsabs.harvard.edu/abs/2011A%26A...533A..29D

[3] Dudok de Wit, T.; Lefèvre, L.; Clette, F. (2016) Uncertainties in the Sunspot Numbers: Estimation and Implications, Solar Physics, Volume 291, Issue 9-10, pp. 2709-2731 http://adsabs.harvard.edu/abs/2016SoPh..291.2709D

[4] Dudok de Wit,T. (2011)

[5] Please see file for details of the authors and copyrights.

Figure 1 – The Sunspot Number series for Uccle from 1981 to 2015



Standard time series modelling typically requires a data set with a stable variance. An adequate variance stabilisation method needs to be identified before performing the main analyses. Three approaches were compared:

- Square root transform: $\sqrt{SSN}$
- Simple Anscombe transform: $2\sqrt{(SSN+3/8)}$
- Haar-Fisz Transform (HFT)[6]: data-driven transform based on Haar wavelets and Fisz coefficient scaling

The HFT was implemented using the DDHFm package in R.[7] [8] It consists of four steps: 1. Haar wavelet transform 2. Isotonic regression to estimate the mean-variance relationship 3. Scaling of the wavelet coefficients by the results of the regression 4. Inverse Haar wavelet transform.

Figure 2 and Figure 3 (as well as Figure 17 in annex) present the results of the variance stabilisation methods. The three transforms lead to a stabilisation of the variance, with the square root and simple Anscombe transforms leading to very similar results. The HFT can only be applied to series of 2^J time points and had to be applied to a subset of the data. A major difference with the other two transforms is the shorter length of the transformed time series.

Two blips appear in the Uccle time series. In December 1988, the SSN for Uccle went above 500 for two days, which is reflected in the square root and Anscombe transforms. In August 2008, the series was very close to 0, which leads to a blip in the Haar-Fisz transformed data series.

[6] P. Fryzlewicz, V. Delouille and G. Nason (2007), GOES-8 X-ray sensor variance stabilization using the multiscale data-driven Haar-Fisz transform. Journal of the Royal Statistical Society Series C, 56, 99-116.

[7] Delouille, V., Fryzlewicz, P. and Nason, G.P. (2005), A data-driven Haar-Fisz transformation for multiscale variance stabilization. Technical Report, 05:06, Statistics Group, Department of Mathematics, University of Bristol

[8] Package 'DDHFm' documentation available on https://cran.r-project.org/web/packages/DDHFm/DDHFm.pdf

Figure 2 - Impact of the variance stabilisation - time series

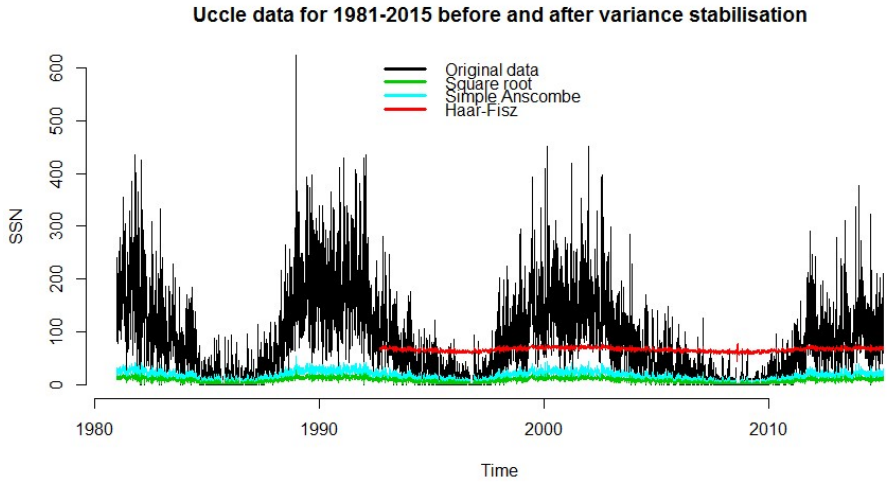
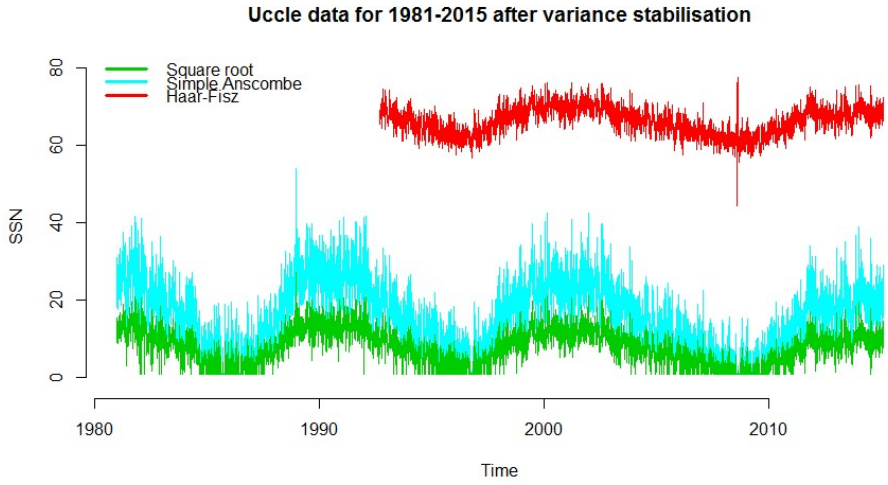


Figure 3 - Impact of the variance stabilisation - transformed time series



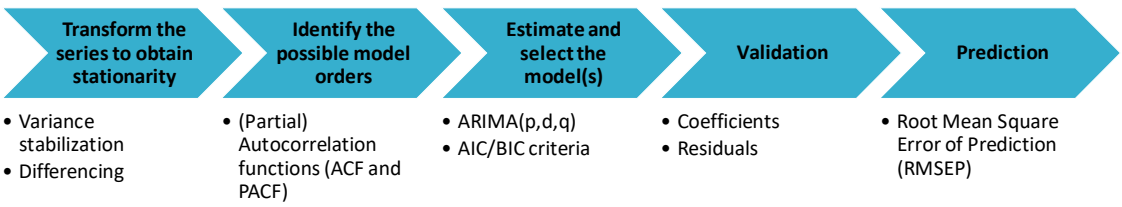
Since all three transforms lead to a stabilisation of the variance, the different approaches will be compared further in the modelling sections.

Rationale for the ARIMA modelling

Standard approach to time series

Figure 4 presents the steps of the standard Box-Jenkins approach to ARIMA modelling. These steps were applied to the SSN time series to provide an alternative approach to the AR(8) model previously applied to the data.

Figure 4 - Steps of the ARIMA modelling approach[9]

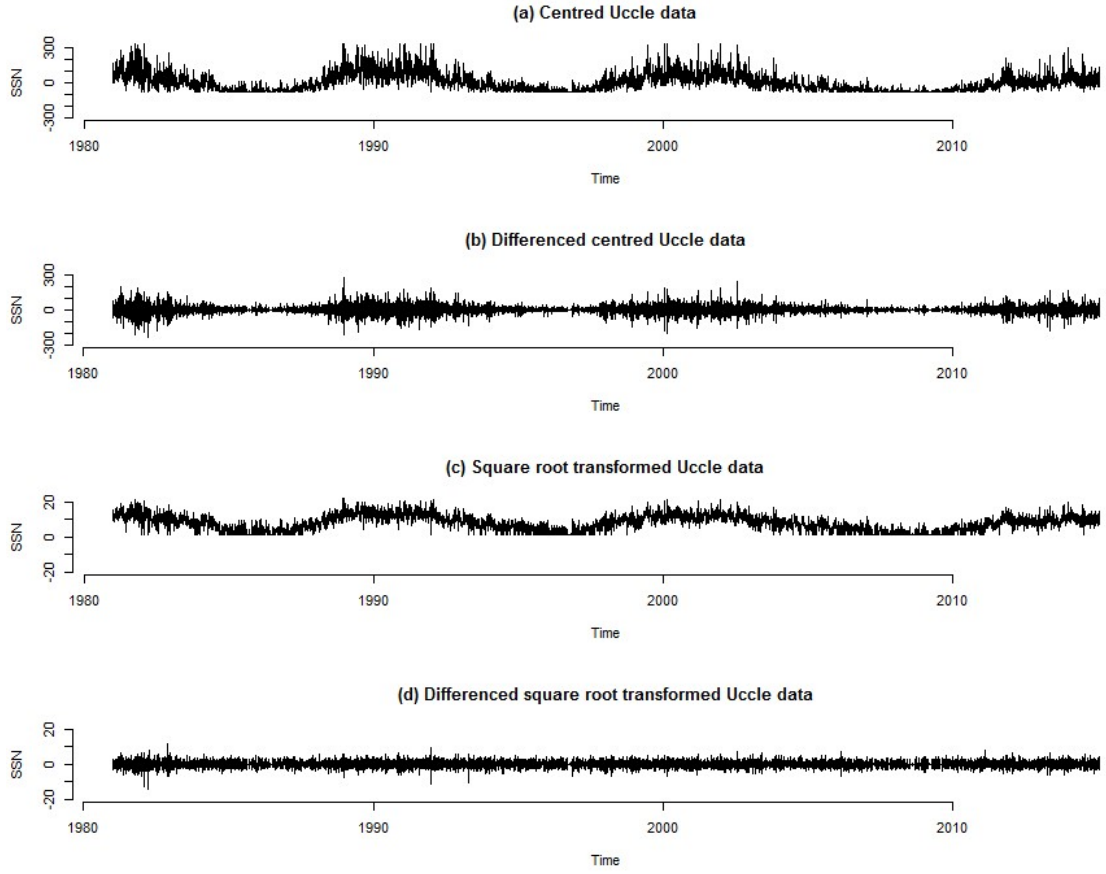


[9] Based on *Brockwell, P. J., & Davis, R. A. (2002). Introduction to time series and forecasting. New York: Springer*

Stationarity

The first step of the process is to obtain a (weakly) stationary time series[10]. Figure 5 (a) shows that the centred data, used for the AR(8) models in the 2016 article, is not stationary. Both the mean and variance of the series are time-dependent. Differencing the data with a lag of 1 stabilises the mean (b) while transforming the data leads to a stabilisation of the variance (c). Both are needed to obtain a weakly stationary time series (d).

Figure 5 - Impact of transforming and differencing the data

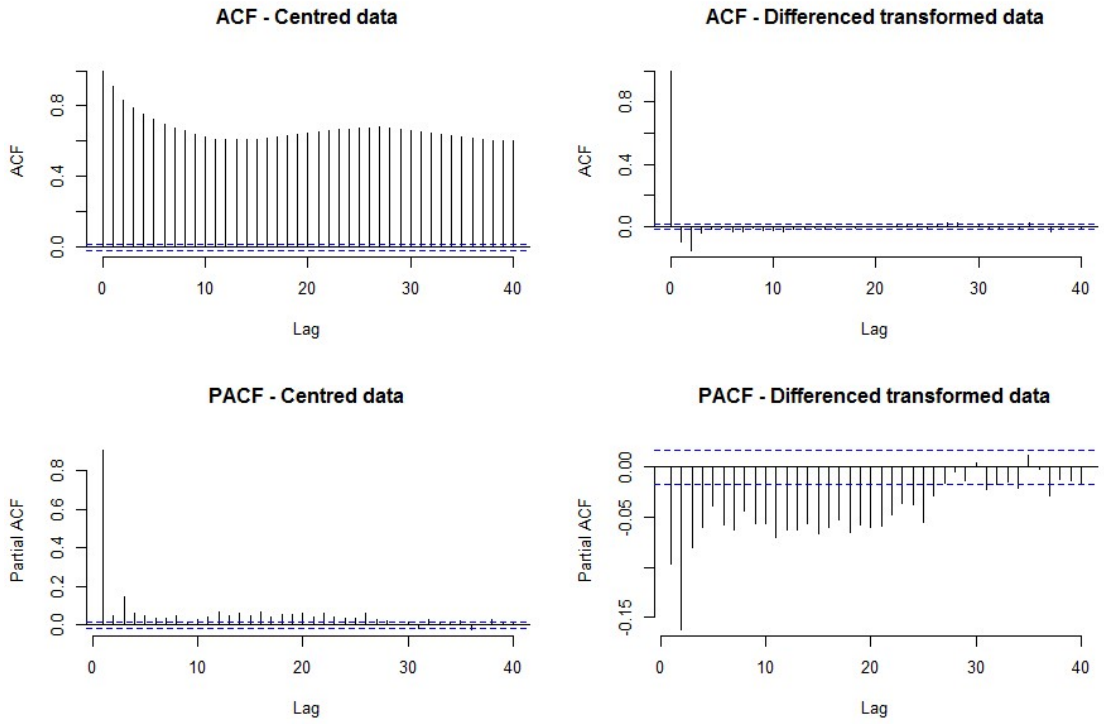


(Partial) Autocorrelations Functions

After transforming and differencing the data to reach weak stationarity, the (partial) autocorrelation functions (ACF and PACF) can be used to identify possible model types (Figure 6). The ACF for the differenced transformed data is 0 after a few lags, while the PACF tends to 0. This points towards a moving average (MA) model and indicates that an ARIMA approach may be more suitable than a pure autoregressive model.

[10] A weakly stationary process has a finite variance and a mean and autocovariance that do not vary with respect to time, i.e. a constant mean and a covariance structure which only depends on the lag between observations.

Figure 6 - (Partial) Autocorrelation Functions for Uccle based on the centred data and on the differenced and square root transformed data



ARIMA modelling of the time-domain errors

Approach

For each of the five stations included in the analysis, four data sets were compared: the centred data, the square root transformed data, the simple Anscombe transformed data and the Haar-Fisz transformed data.

To easily compare the RMSEP between different models, all data sets were centred and reduced after the transformations (Figure 18 in annex).

Given the large number of models to compare and the required computing time, the best ARIMA model for each series was selected automatically using the auto.arima function of the forecast package, which selects models based on the corrected Akaike's Information Criterion (AICc).

The root mean square error in prediction (RMSEP) of the model was computed with the "one step ahead" approach on the final t observations of the series. The full data set was used for modelling but only non-missing values in the original data set were included in the calculation of the RMSEP.

Following preliminary analyses, t was set at 6000. This is close to half of the available observations and provides a good compromise between model robustness, the number of predictions and the computing time (+/- 10min per model). Besides, the RMSEP on the original data follows the solar cycle, with a smaller RMSEP when the solar cycle is at a low point. A large t therefore ensures that the RMSEP is computed over more than a full solar cycle.

Model selection and comparison

ARIMA models have three parameters (p,d,q): p is the number of AR parameters, d the degree of differencing and q the number of MA parameters. An AR(8) model is therefore an ARIMA (8,0,0) model.

Figure 7 presents the ARIMA models selected for each station and data set. In all cases, models with a degree of differencing of 1 and at least one moving average parameter were selected. This indicates that differencing the data and including moving average parameters could give a better fit than pure autoregressive models.

Figure 7 – ARIMA models selected by auto.arima for 5 stations and 4 transforms

Station	Centred	Square root	Simple Anscombe	Haar-Fisz
LO	(5,1,3)	(5,1,5)	(3,1,5)	(3,1,2)
UC2	(5,1,2)	(3,1,1)	(3,1,1)	(1,1,1)
CA	(4,1,5)	(4,1,3)	(4,1,3)	(1,1,5)
KS2	(3,1,5)	(4,1,2)	(4,1,2)	(3,1,4)
KZ	(5,1,2)	(4,1,5)	(4,1,5)	(1,1,5)

Figure 8 presents the Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC) for all models. In general, the lower the criterion (for a given data set), the better the model fit. Based on the AIC and BIC, the ARIMA models are a better fit for the data in comparison with the AR(8) models in all cases.

Figure 8 - Akaike's Information Criterion (AIC) and Bayesian Information Criterion (BIC) for all models

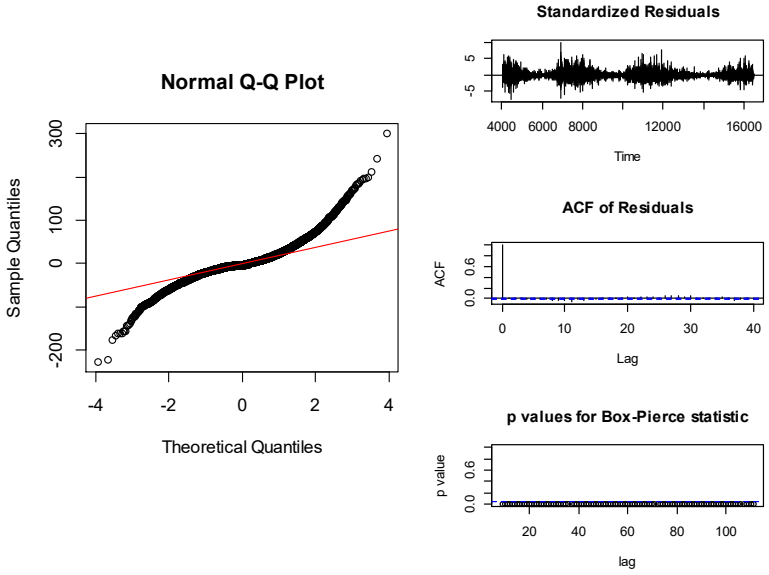
		Centred		**Square root**		**Simple Anscombe**		**Haar-Fisz**	
Station	Criteria\Model	AR(8)	ARIMA	AR(8)	ARIMA	AR(8)	ARIMA	AR(8)	ARIMA
LO	AIC	1742	1055	944	215	1224	565	2368	1929
	BIC	1816	1122	1018	304	1298	632	2438	1971
UC2	AIC	13130	12533	9659	9088	9718	9148	8151	7745
	BIC	13204	12592	9734	9125	9793	9185	8221	7766
CA	AIC	6513	5874	3699	3053	3849	3207	4081	3670
	BIC	6587	5948	3773	3113	3924	3267	4151	3720
KS2	AIC	8525	7852	5726	5075	5871	5225	4013	3568
	BIC	8599	7918	5780	5127	5946	5277	4084	3623
KZ	AIC	8145	7468	5427	4770	5593	5012	5541	5158
	BIC	8219	7528	5502	4845	5667	5012	5612	5207

Model validation

As per the standard time series approach presented above, the model validation includes an analysis of the model coefficients and residuals. Coefficients were generally significant at the 95% level for all models tested. The section below presents the general results obtained in terms of model validation, with illustrations based on the Uccle data. Results for other stations were similar unless otherwise stated.

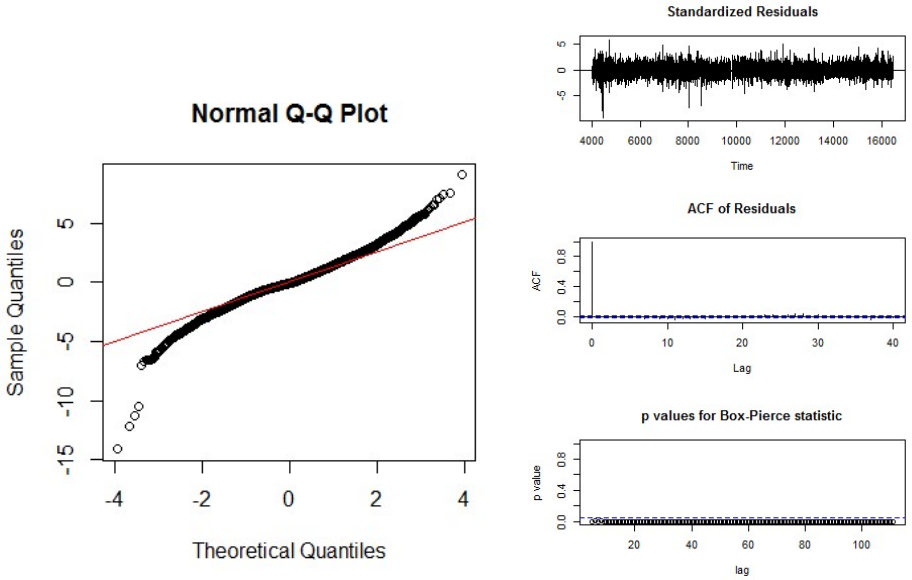
The residuals of a valid ARIMA model should be Gaussian (QQ-plot close to a diagonal line), uncorrelated (ACF within the boundaries), white noise (non-significant values for the Box-Pierce statistic) and have a stable variance (stable variance of the standardised residuals). When the data is only centred, the residuals generally do not meet any of these criteria, whether for the AR(8) or ARIMA models (Figure 9).

Figure 9 - Model validation for an AR(8) model on the centred Uccle data (QQ-plot, plot and ACF of residuals, p-values of the Box-Pierce statistic[11])



Transforming the data leads to residuals with a stable variance, but residuals for the AR(8) models are not necessarily Gaussian and fail the white noise test (Figure 19 in annex).
The ARIMA residuals are generally closer to being Gaussian (Figure 10). The residuals still cannot be considered white noise according to the autocorrelation function and the Box-Pierce statistic, but this varies across stations, with for instance residuals closer to white noise for Kislovodsk (Figure 20 in annex).

Figure 10 - Model validation for an ARIMA model (3,1,1) on the square root transformed Uccle data



Overall, the ARIMA models applied to transformed data meet most validation criteria, while AR(8) models and models on centred data cannot be considered valid.

Root Mean Squared Error in Prediction (RMSEP)

Figure 11 present the RMSEP per station and data set. The first two columns (Centred – Original data) are based on the original data and give an idea of the magnitude of the RMSEP before transforming the data. These results show that the RMSEP is between 20 and 30 and that this

[11] The null hypothesis of the Box-Pierce test is that the data is white noise. P-values below the threshold indicated on the plot correspond to a rejection of the null hypothesis : the residuals cannot be considered white noise.

estimate of the time-domain error is comparable to the results obtained in the 2016 article.

The « Reduced » columns allow for a comparison of the different models. Overall, the RMSEP is almost always smaller for the ARIMA models than for the AR(8) models, but the difference is small. ARIMA models fit the data better than AR(8) models, but lead to an equivalent estimate of the time-domain error.

Furthermore, the Haar-Fisz transform leads to slightly higher RMSEP than the other transforms. The square root transform gives similar results to the Anscombe transform. Given all three method stabilise the variance, the square root transform seems to be the best choice, with a simple formula and a small RMSEP.

Figure 11 - RMSEP for each data set and station

Station	Centred - Original data		Centred Reduced		Square root Reduced		Anscombe Reduced		Haar-Fisz Reduced	
	AR(8)	ARIMA	AR(8)	ARIMA	AR(8)	ARIMA	AR(8)	ARIMA	AR(8)	ARIMA
LO	21.6	22.2	0.239	0.232	0.240	0.233	0.243	0.236	0.280	0.280
UC2	27.1	26.4	0.362	0.353	0.328	0.319	0.329	0.320	0.384	0.371
CA	24.4	23.8	0.314	0.306	0.283	0.275	0.284	0.275	0.316	0.304
KS2	22.0	21.5	0.310	0.302	0.287	0.278	0.288	0.280	0.280	0.300
KZ	25.1	24.6	0.308	0.301	0.299	0.292	0.301	0.294	0.347	0.338

Observation errors

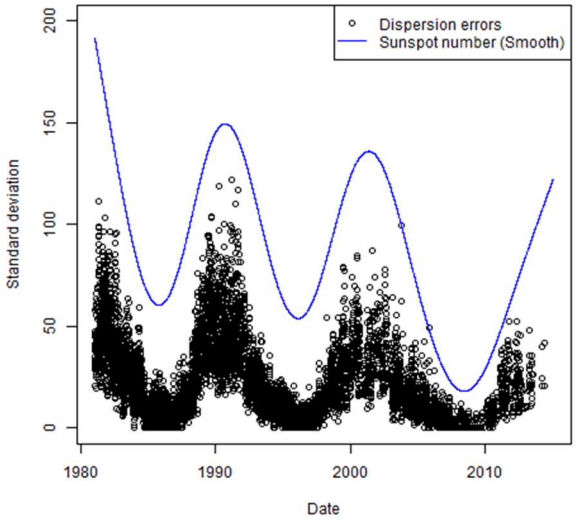
The observation errors correspond to the dispersion between observers. They were computed for each station and each time point in three steps:

1. Scaling the SSN to the International Sunspot Number: $SSN^*_{Station}(t) = SSN_{Station}(t) * \gamma_{Station}$
2. Computing residual errors: $\epsilon_{Station}(t) = SSN^*_{Station}(t) - SSN_{SILSO}(t)$
3. Computing the standard deviation of residual errors for each time point with at most one third of missing values:

$$\sigma(t) = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \epsilon^2_{St}(t)}$$

The standard deviation of residuals is used as an estimate for observation errors. Figure 12 show the observation errors over time. They clearly follow the solar cycle and are more important during the high points in the cycle, when more sunspots need to be counted and results are more likely to differ between observers, and less important during the low points.

Figure 12 - Observation errors over time compared with the smoothed SSN



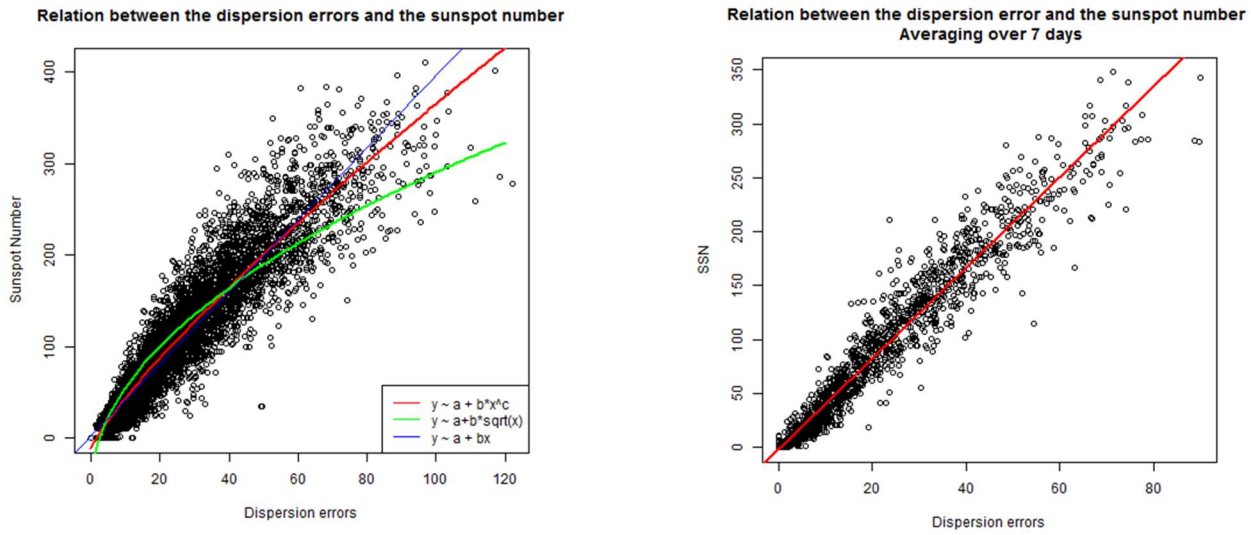
Stabilising the variance does not change the behaviour of the series, which remains time-dependent (Figure 21 in annex). Imputing missing values tend to increase dispersion errors (Figure 22 in annex).

Relationship between errors and with the Sunspot Number

Relationship between errors and the Sunspot Number

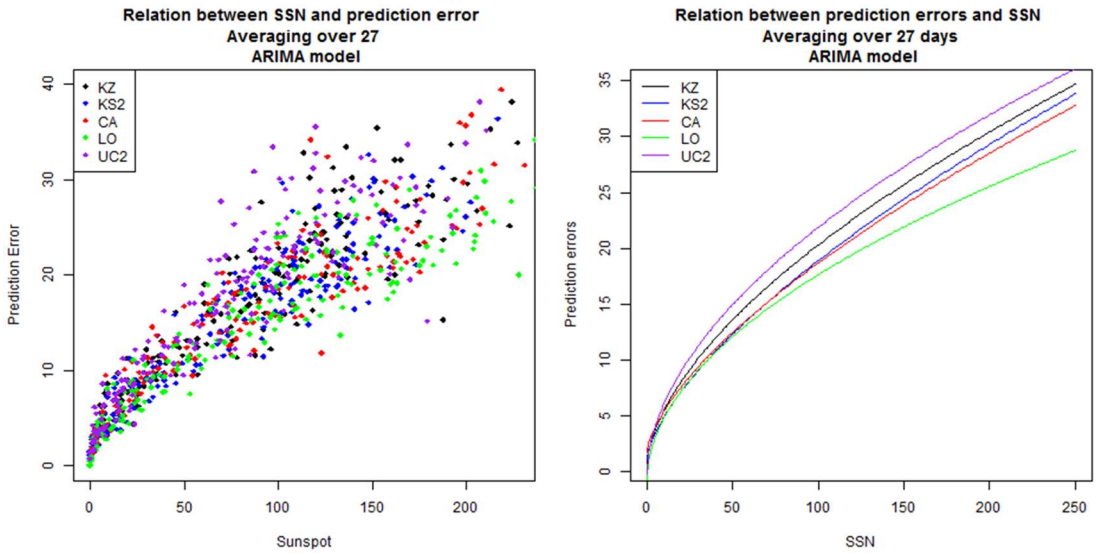
The relationship between the SSN and observation errors is proportional and can be modelled with a polynomial model (Figure 13, left). However, averaging over 7 days leads to a linear relationship between the errors and the SSN (Figure 13, right), which is unexpected and should be further analysed.

Figure 13 - Relationship between the SSN and observation errors



The relationship between the time-domain errors and the SSN varies between stations but is close to a square root relationship (Figure 14).

Figure 14 – Relationship between the SSN and time-domain errors

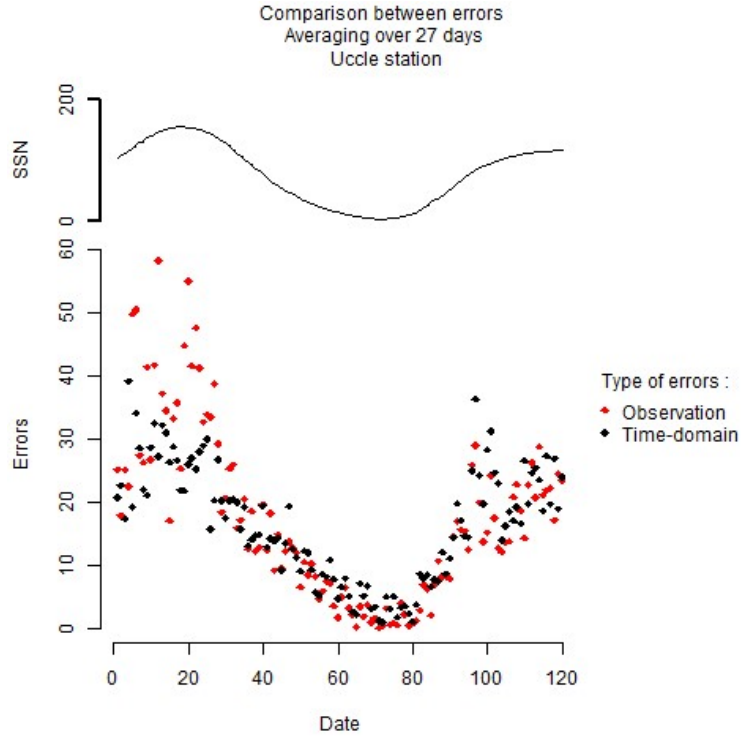


Comparison between errors

Figure 15 presents the smoothed SSN together with the observation errors and the time-domain errors for Uccle, both averaged over 27 days. The errors tend to be of similar magnitude when the solar cycle is in a low point. When the solar cycle is in a high point, the observation errors tend to be become more important than the time-domain errors, but this is not the case for all cycles, nor for all

stations.

Figure 15 - Comparison between the two types of errors and the SSN



Ideas for further analyses of the time-domain errors

SARIMA

Given the periodicity of the sun's activity, Seasonal ARIMA (SARIMA) models could also be used to model the data. However, fitting a SARIMA model based on the solar cycle periodicity failed. This could be linked to the fact that the solar cycle is not of stable length and, while it is close to 11 years, cycles can vary between 9 and 14 years in practice.

SARIMA models with a seasonality of 27 days (one solar rotation) were also fitted to the data but gave worse results in terms of AIC/BIC and validation diagnostics than the ARIMA models presented in this report.

Further analyses could assess whether other methods are available to model time series data with varying periodicity or with a cyclic variance.

Overall ARIMA

The results obtained for ARIMA models were based on the best model selected individually for each data series. It could be interesting to compare the AR(8) model with a "blanket" approach of one overall ARIMA model for all stations.

A comparison of ARIMA models with p=1 to 5 and q=1 to 5 shows that most ARIMA models have a lower AIC than the AR(8). Based on preliminary analyses, a (3,1,3) ARIMA model is a good candidate for an overall model. Figure 16 presents the results of a comparison between the AR(8) and ARIMA(3,1,3) models on square root transformed data for the subset of 5 stations. These results show that using the same model for all stations also tend to give better results than the AR(8) models, in terms of AIC, BIC and RMSEP. The models also seem to meet the validation criteria to the same extent as the individual ARIMA models.

Selecting an overall ARIMA model therefore seems a valid approach. Further analyses to assess the performance of an overall ARIMA model on all stations in the data set could be useful, as these models have a better fit and rely on a more robust approach than the AR(8) model while giving comparable results.

Figure 16 - Comparison of the AIC and BIC for the AR(8) and ARIMA(3,1,3) models on square root transformed data

		Square root	
Station	Criteria\Model	AR(8)	ARIMA(3,1,3)
LO	AIC	40217	39549
	BIC	40291	39601
	RMSEP	1.16	1.14
UC2	AIC	46569	45995
	BIC	46644	46032
	RMSEP	1.44	1.41
CA	AIC	41009	40361
	BIC	41084	40420
	RMSEP	1.27	1.23
KS2	AIC	41987	41333
	BIC	42061	41385
	RMSEP	1.23	1.20
KZ	AIC	43362	42702
	BIC	43436	42776
	RMSEP	1.37	1.34

Conclusions

Stabilising the variance of the data can be implemented through square root, simple Anscombe or Haar-Fisz transforms. The square root transform is sufficient to obtain a good model fit and has one of the smallest RMSEP. This transform could therefore be used as the main variance stabilisation method for the SSN data.

Fitting individual ARIMA models to the data gives a better fit than the AR(8) models. However, the RMSEP of the more robust ARIMA models is only marginally smaller than the one for the AR(8). The results obtained with the standard time series approach confirm the magnitude of the time-domain error estimated in the 2016 article.

Observation errors are time-dependent and proportional to the SSN, while time-domain errors have a square root relationship to the SSN. Both errors are of similar magnitude at the low point in the solar cycle, observations errors tend to be more important when the solar cycle is in a high point.

This highlights the importance of reducing the dispersion of observations between stations. This could be obtained through standard guidelines to the stations or the selection of a smaller selection of stations producing more robust results.

Two approaches to the time-domain errors could be further explored: seasonal time series modelling and the use of one overall ARIMA model for all stations. First tests with seasonal ARIMA models gave poor results, but further research in this area could lead to more interesting results. Applying one overall ARIMA model to all stations, for instance an ARIMA(3,1,3), gave promising results and could be a more robust alternative to the AR(8) approach to compute time-domain errors.

In summary, the analysis of the SSN data presented in this report confirms the magnitude of the time-domain errors and the relationships between the errors and the SSN outlined in the 2016 article. It further suggests the use of the square root transform to stabilise the data variance and of an overall ARIMA model to compute time-domain errors in a more robust fashion.

ANNEXES

Figure 17 - Impact of the variance stabilisation - histograms[12]

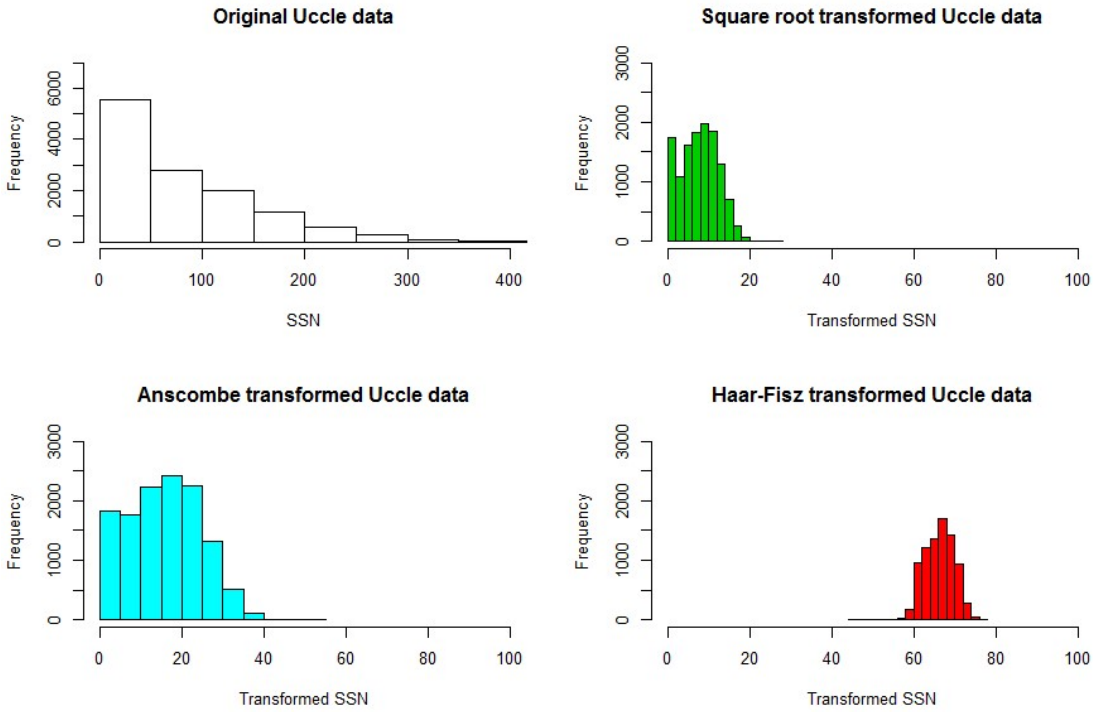
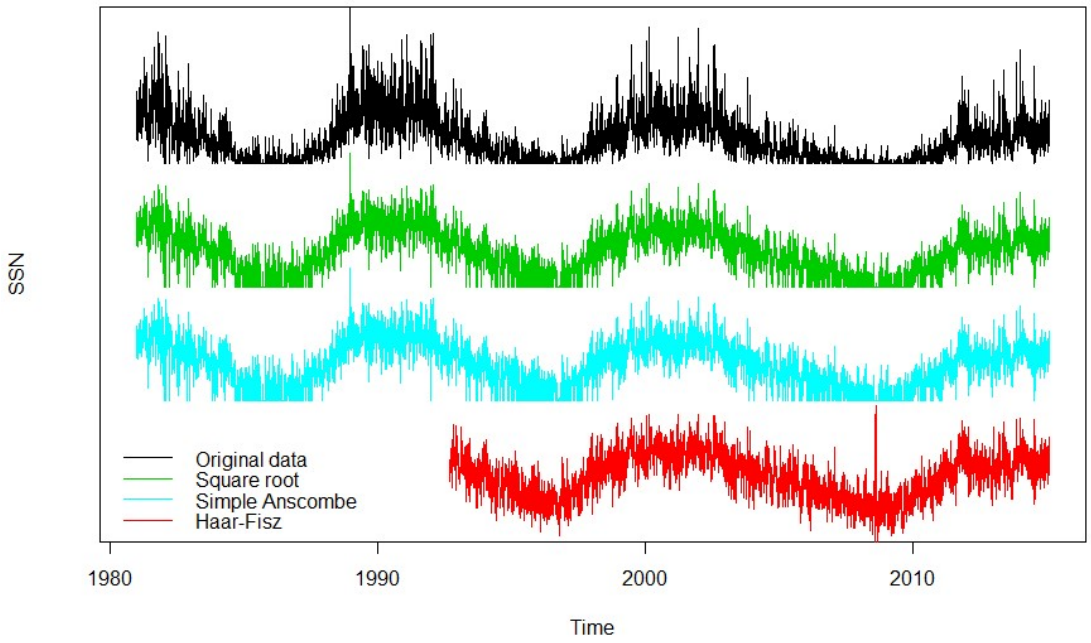


Figure 18 - Transformed, centred and reduced Uccle data – Data series on shifted scales



[12] Please note the original data is on a different scale than the transformed data.

Figure 19 - Model validation for an AR(8) model on the square root transformed Uccle data

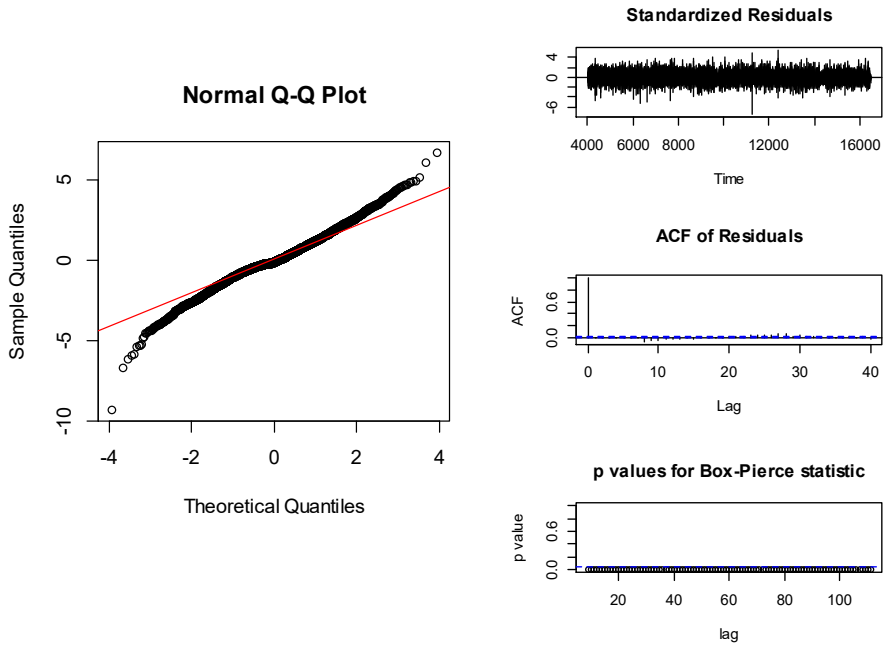


Figure 20 - Model validation for an ARIMA(4,1,2) on square root transformed data for Kislovodsk

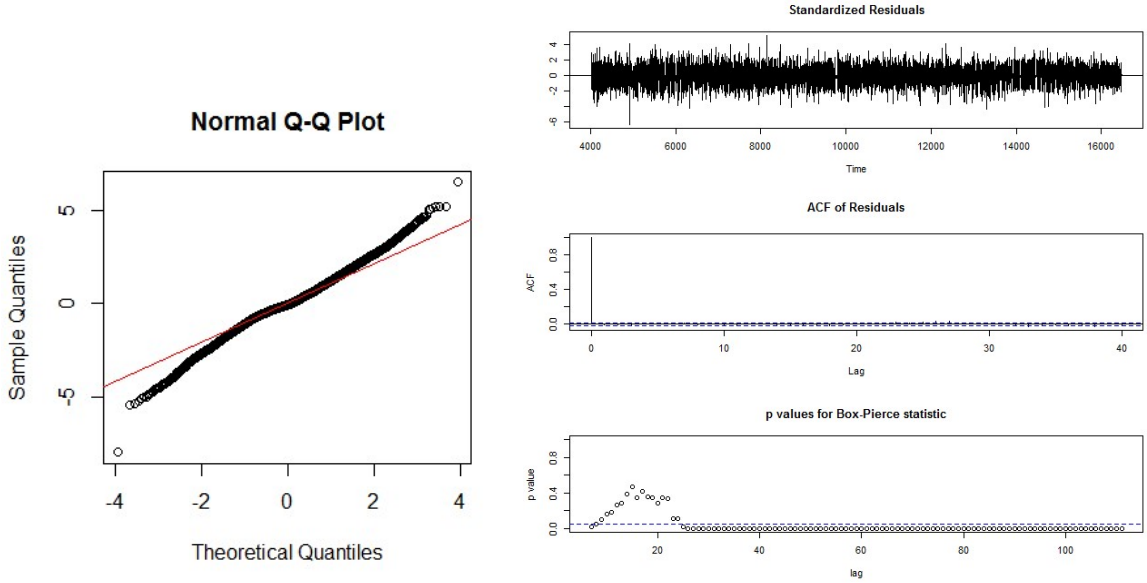


Figure 21 – Observation errors - Effect of variance stabilisation

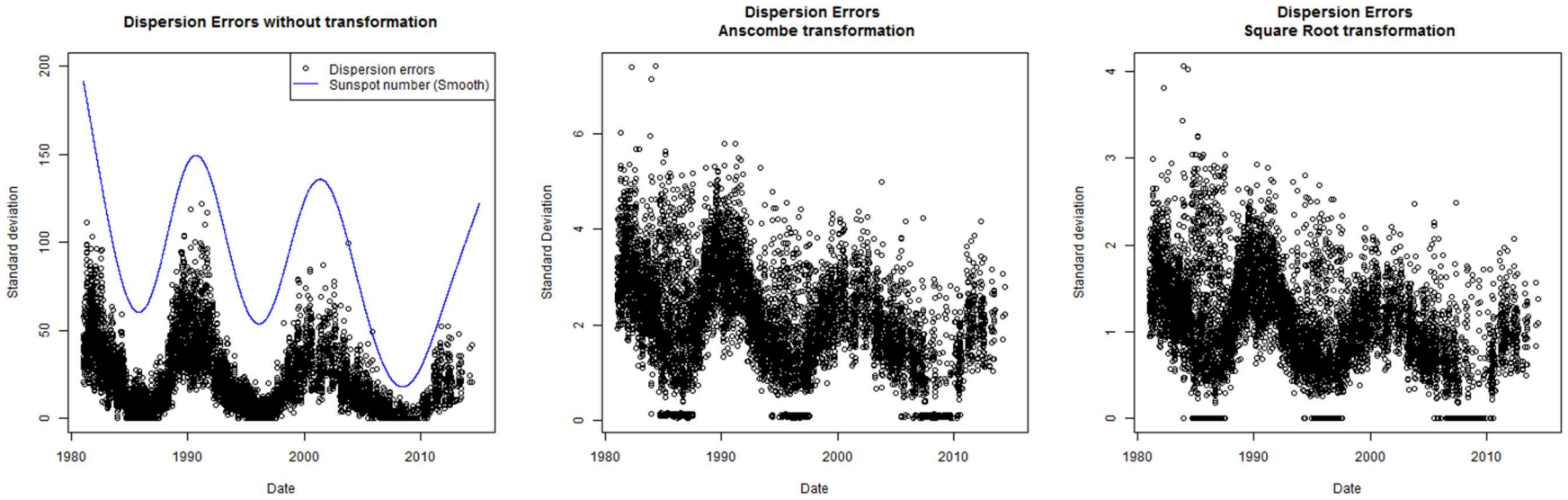


Figure 22 – Observation errors - Effect of missing values imputation