Vrije Universiteit Amsterdam

Universiteit van Amsterdam

Master Thesis

Using Spark to Extract, Compare and Validate Remote Sensing-based Phenological Metrics on a Large Scale

Author: Viktor Bakayov 11397918

1st supervisor: Rob van Nieuwpoort
daily supervisor: Romulo Goncalves
2nd reader: Adam Belloum

A thesis submitted in fulfillment of the requirements for the joint UvA-VU Master of Science degree in Computer Science

August 30, 2018

Abstract

Time series of remote sensing (RS) images can be used to characterize land surface phenology at continental to global scale. For this, the images are typically transformed into various vegetation indices (VI) such as the normalized difference vegetation index (NDVI) or the enhanced vegetation index (EVI). These indices can then be used to extract various vegetation metrics (e.g., start and end of vegetation season). However, there is no universally accepted method for extracting such vegetation metrics, and furthermore, results can vary per region, vegetation or VI used. In addition, extracting vegetation metrics is a computationally intensive process which in high spatio-temporal resolution can be a time-consuming procedure if handled by a single machine.

To compare and validate the EVI and the NDVI based vegetation metrics using various extraction methods, but also efficiently extract such metrics on a high spatio-temporal resolution, a distributed Spark-based solution was implemented. The solution integrates the existing software TimeSat for phenology computation.

The NDVI and the EVI time series were used to detect the start-of-season (SOS) at 1-km spatial resolution using 10-days composites for the period from 1999 to 2017 on North America and more specifically the USA. Further, the two VIs were intercompared with three fitting functions for seasonality extraction both spatially and temporally. Also, to evaluate the accuracy of our experiments the results were compared on a species level to volunteered phenological observations curated by the USA national phenological network.

The results revealed that SOS vary within and among VIs and fitting function used. The biggest differences are observed between VIs. In central and east USA there are minimal differences (0-5 days), however, in Central-West regions, as well as in Florida, differences can go up to 50-65 days. The fitting function has less effect on the results. The comparisons with the ground observations show that RS phenology is a viable method for determining SOS. The EVI shows consistent better results with MAE between 20 and 30 days per species studied.

The study demonstrates how phenology metrics can be derived at continental or even global scales at high spatio-temporal resolution, using cluster distributed computation. Furthermore, we can now better understand the relationships between different VIs and fitting functions.

Acknowledgements

I would like to thank my thesis supervisor dr. Romulo Goncalves from the Netherlands eScience Center (*NLeSC*) for proposing such an exciting project, for the time he devoted to guide me and for his valuable expertise.

I would also like to thank prof.dr. Raul Zurita-Milla and dr. Emma Izquierdo-Verdiguier from the *Faculty ITC of the University of Twente* who supported my work, provided valuable knowledge in the phenology domain and helped me get results of higher quality.

In addition, I would like to acknowledge prof.dr. Rob van Nieuwpoort from the University of Amsterdam as the first reader of this thesis, and I am gratefully indebted to his very valuable comments.

My love and gratitude also goes to the people dearest to me, who have always been extremely supportive and believed in me.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

VI	Vegetation Indice
NDVI	Normalized Difference Vegetation Index
EVI	Enhanced Vegetation Index
RS	Remote Sensing
SOS	Start of Season
EOS	End of Season
LAI	Leaf Area Index
LC	Land Cover
AVHRR	Advanced Very High Resolution Radiometer
VM	Virtual Machine
DAG	Directed Acyclic Graph
RDD	Resilient Distributed Dataset
HDFS	Hadoop Distributed File System
GeoTIFF	Georeferenced Tagged Image File Format
NaN	Not a Number
SD	Standard Deviation
AG	Asymmetric Gaussian
SG	Savitzky-Golay
DL	Double Logistic
USA-NPN	USA National Phenology Network
MAE	Mean Absolute Error
RMSE	Root Mean Square Error

Chapter 1

Introduction

This Chapter presents the relevant theory and context for the project, what are the reasons behind it and gives an overview of the dissertation.

1.1 Context

"Phenology is the study of periodic plant and animal life cycle events and how these are influenced by seasonal and interannual variations in climate" (Mack et al., 2014). Because of this, the life cycle events vary from year to year and from place to place (Zurita-Milla et al., 2017). Understanding this variability is critical to quantify the impact of climate change on our planet and can have a strong impact on a plethora of public sectors and real word applications (Zurita-Milla et al., 2017).

There are several sources of phenological data, such as ground observations, phenocameras or satellite sensors. Latter have allowed deriving remote sensing (RS) vegetation indices (VIs) to characterize land surface phenology at continental to global scale. Such indices (typically normalized difference vegetation index (NDVI) and enhanced vegetation index (EVI)) are used to extract various vegetation metrics (e.g. start-of-season (SOS) and end-of-season (EOS)) in order to determine the plant's phenophase state.

1.2 Motivation

Several challenges are at hand. Currently, there is no universally accepted method to extract phenological metrics from RS images. Applying different methods to the same RS data might result in a difference of the average day-of-year estimates of up to 60 days (White et al., 2009). Moreover, multiple VIs can be used to study phenology, producing different results (LI Hongjun, 2007). Also, the spatial resolution of each time series varies from sensor to sensor, making the integration of images a challenging task. With the varying remote sensors, different data sets, and temporal and spatial variations it is difficult to establish a uniform approach to study land surface phenology at large scale.

In an effort to validate the RS-derived phenology metrics the results are typically compared to ground phenology data. Another challenge is that the RS measures usually compare poorly with field-observed phenology (Schwartz and Hanes, 2010) (White et al., 2009). In particular, ground phenological studies focus on a limited number of individual plants (Liang et al., 2011). Moreover, most of the ground observations are isolated observations, which lack the spatial coverage that can match satellite pixels. However, the primary source of uncertainty comes from the inability to account for within-pixel phenological variability with the coarse-scale RS data (point-vs.-pixel comparison errors) (Liang et al., 2011).

Earlier studies have shown that satellite data is a valuable source for vegetation mapping (Justice et al., 1985). Liang et al. (2011) concludes that landscape phenology derived from satellites could be problematic, but using appropriate techniques for extracting seasonal markers, such as logistic curve, is promising. However, another source of errors and uncertainties is the landscape heterogeneity; the satellite measures are more accurate with internally consistent and homogeneous vegetation covers (such as deciduous plants) (Doktor et al., 2009) (Soudani et al., 2008).

The satellite-derived phenology is also influenced by the temporal (Zhang et al., 2009), spatial (Klosterman et al., 2014), and spectral resolutions (Atzberger et al., 2013) of the remote sensing data. To minimize the within-pixel phenological variability, one can use images at higher spatial resolution. The RS data is available at different scales; from 8 km. to 10 meters (VITO, 2018b). Moreover, in order not to penalize on the temporal accuracy, one might want to work with frequent samples; from 15-days composites to daily images. Therefore, to have best accuracies when producing phenology products (or RS validation), high spatio-temporal resolutions are required, resulting in a Big Data challenge.

The higher the spatio-temporal resolution, the greater the computational and storage requirements. That already can rise some challenges to use a single thread file-based solution. As we will show, a distributed multi-core solution outperforms a single thread solution, making it feasible to complete computation in a couple of hours, instead of days. In the future, we foresee even a bigger challenge when working with even higher resolutions. Currently, 10-meter resolution data products are being created for Sentinel-2 with larger scales, such as continental or even hemisphere scales (esa, 2018b).

1.3 Objectives

The main objective of this project is to study the validity and coherence of NDVI and EVI based Start-of-Season(SOS) phenology metrics at continental scale. To do such analysis, at large scale and high-resolution, we decided to use distributed computing and more specifically Spark, not only because it will allow us to handle the increased problem size, but also because we can easily scale when the problem size increases/decreases. Therefore, the project goal is separated into two specific objectives:

- Objective 1: Build a Spark-based Big Data solution for phenology extraction. This objective composes of four parts:
  - The first challenge is to study the existing phenology extraction programs and the possibility of integrating them into Spark.

  - Once a phenology software is determined suitable for integration, build a platform for distributed phenology computation.
  - Compute the NDVI and the EVI using the satellite's spectral bands, extract SOS, and produce SOS products with georeference information encoded.
  - Finally, evaluate the platform's *efficiency* and *scalability*.

- Objective 2: Study the validity and coherence of NDVI and EVI vegetation indices and intercompare them with different phenology extraction functions. This objective composes of three parts:
  - To study the impact of using one or another VI and fitting function on the estimated day of SOS, we compare the SOS products on a temporal and spatial scales by calculating the average and standard deviation SOS values for all years. Furthermore, we compare the difference between the SOS experiments.
  - To estimate the accuracy of the VIs and the seasonality extraction methods, we study the validity of the results on a species level by comparing them to ground truth; a manually build validation dataset.
  - Finally, we determine the best performing approach (best VI and best fitting function).

All procedures were built in a customizable way for anyone interested to be able to reproduce our results. We would like to have a project which is sufficiently and well documented to allow reproducibility and good data provenance. What is more, we provide other researchers a Big Data solution they can work with and be able to efficiently extract various phenometrics.

1.4 Structure

The remainder of this thesis is structured into the following Chapters:

Chapter 2 - Phenology introduces the reader to phenology in general and RS phenology in particular and describes the dataset used for calculating the NDVI and the EVI.

Chapter 3 - Phenology Software discusses the preliminary research on the available phenology software and which solution was chosen to be integrated with Spark.

Chapter 4 - The Computational Platform describes the Big Data processing platform used for this project.

Chapter 5 - TimeSat Spark Big Data Processing explains the integration of the phenology software with the Spark platform and how the SOS products were derived and calculated.

Chapter 6 - Platform Evaluation evaluates the efficiency and scalability of the implemented Big Data solution.

Chapter 7 - Phenology Analysis presents and evaluates the results of the generated SOS products.

Chapter 8 - Conclusion summarizes the dissertation and gives suggestions for future work. Also, discusses the limitations of the project and the learning outcomes.

Chapter 2

Phenology

This Chapter will introduce the reader to phenology (Section 2.1) and will review the main sources in phenological data (Section 2.2). Furthermore, RS sensors and RS vegetation indices are discussed in Section 2.3 and the RS products, as well as the dataset used for this project, are discussed in Section 2.4.

2.1 Phenology Introduction

Earth is the only object in the known universe known to harbor life with terrestrial vegetation widely distributed all over the globe. The study observing the changes in the vegetation timing and animal biological phases, their causes and their interrelations is called Phenology (Demarée, 2011). Phenology is an important component in the Earth's life since it influences the distribution, timing and abundance of organisms, food webs, and global cycles of water and carbon (usanpn, 2018). For example, many birds will start nesting when there are insects available for their offspring (Sedinger and Raveling, 1986). For people in the public health sector, through phenology studies, it can be determined when hay fever or plan based allergies are about to begin (WHO, 2003). Food production can be better managed with determining the timing of agricultural activities. Farmers and gardeners will exceed when they know the plant's and insect's development cycles so they can determine when is the best time to apply pesticides or fertilizers. They also will benefit from knowing when to plant to avoid frosts and when to harvest to maximize the agricultural production (Peña-Barragán et al., 2011).

Phenology, however, may be altered by changes in precipitation and temperature (usanpn, 2018). The seasonal timing of plants and animals varies from year to year and from place to place because it is strongly influenced by weather and climatic variability. The recent 30 years of warmer temperatures have affected the development and seasonal events in many groups of organisms, including birds (Crick et al., 1997) and plants (Fitter and Fitter, 2002), but also the range and distribution of species and the composition and dynamics of communities (Walther et al., 2002). The ecological complexity interactions make it difficult to extrapolate from studies of individuals, through the community or ecosystem level to global-scale studies of atmospheric and oceanic processes (Walther et al., 2002). Studies using satellite-derived phenology data or many newly established initiatives try to find answers to such connections.

2.2 Phenology Data Sources

There are four main sources of phenological data: ground observations, phenocameras, phenology models and satellite remote sensing. Our validation strategy is to compare the satellite data with another source with established authenticity and validity. Ground observation collections which are gathered by volunteers who annotate the timing of phenological events for a specific plant or animal and locations are our main ground truth. The approachability of the field makes it suitable to public participations (Primack and Miller-Rushing, 2012) and as a result there are established networks and organizations around the world (e.g., in the United Kingdom, France, Germany, Turkey, the Netherlands) which engage both amateur naturalists and professional scientists who collect, store and share phenological information (Schwartz, 2003).

In the USA, the USA National Phenology Network (USA-NPN, usanpn.org) was built to provide the connection between ground observers and researchers (or anyone interested), starting phenological monitoring program and gathering both historical phenology datasets and producing new observations (Schwartz et al., 2012). Such continued data accumulation provides a great opportunity to better understand remotely sensed continental-scale derived data and facilitates its integration and validation (Liang et al., 2011). We are going to use this dataset as a ground truth in order to validate the results produced from our methods derived from other phenological data sources.

Another source, a near-surface remote sensing phenocameras have emerged as a useful tool for capturing phenology metrics. They offer phenological data at an in-between scale (to that provided by ground and satellite data). Phenocams are a web of digital cameras purposely mounted to monitor vegetation activity. A wide PhenoCam network was established in 2008 in the Northeastern United States to provide automated vegetation monitoring at high-frequency resolution (typically 30 minutes resolution) (Richardson et al., 2018). The distribution of the phenocams is shown in Figure 2.1. The imaginaries can be processed with image processing software in order to derive quantitative data on, for example the color of vegetation, which is a proxy to its phenological state. The phenocameras hold a valuable mean for evaluation satellite-based phenology, but due to the similar technique both use; recording vegetation reflectance change; phenocameras are more akin to RS if compared to conventional observations.

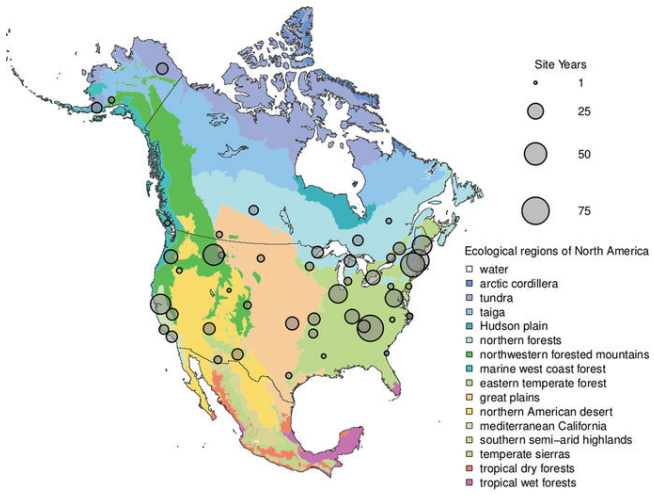


Figure 2.1: Spatial Distribution of PhenoCam Data Across Ecological Regions of North America (Richardson u. a., 2018)

Another source of spatio-temporal phenological data are models that combine such ground observations with environmental data (e.g. weather) to understand which factors drive phenology and to be able to map and predict phenological events. Understand the effects of the climate change is important in order to predict the consequence of future changes. However, understanding the processes is not straightforward. Phenology changes and trends can be concealed by short-term inter-annual climate variations and to correctly recognize this, long datasets are needed (Badeck et al., 2004). Therefore, validating such models, or in general, choosing a study, has been dictated by the availability of the field records (Robbirt et al., 2011).

The main source comes from time series from satellite data. RS images derived from satellite data posses significant potential for observing vegetation dynamics at regional and global scale because of its regular temporal sampling (Azzali and Menenti, 2000). The images can be used to characterize land surface phenology (LSP) at large scale. This is the study of spatio-temporal development of the vegetation in relation to climate as revealed by satellite sensors (De Beurs and Henebry, 2005). Applications of remote sensing are urban and hydrological modeling, drought predictions, energy and water flux estimation, but also it is indirectly related to plant phenology via absorption and reflection of radiation. However, it is influenced by cloud and snow cover, bidirectional reflectance effects and non-climatic factors influencing the land surface (e.g. anthropogenic disturbances) (White et al., 2009).

2.3 Remote Sensing Phenology

We will focus on the analysis of time series of RS data because it provides both long-term time series and allows studies on continental to global scale that can improve our current understanding of phenology and phenological changes in space and time (Reed and Sayler, 1998). Using spectral data from different space missions, such as the Systeme Pour l'Observation de Ia Terre (SPOT), different data products are derived. The most known data products are normalized difference vegetation index (NDVI), enhanced vegetation index (EVI), leaf area index (LAI) or land cover maps (LC) (Melesse et al., 2007). The NDVI and the EVI are the ones we will use in our projects since SOS products are derived from them.

2.3.1 Remote Sensing Sensors

At first, satellites were mainly used for purposes of reconnaissance, mapping, surveying and military surveillance. During the Cold War, data was mostly gathered for military purposes. However, the technology developed for mapping during this times slowly transitioned into environmental and natural resources applications (Melesse et al., 2007).

The beginning of the meteorological satellite sensors consisted of the polar orbiting National Oceanic and Atmospheric Administration (NOAA) satellites carrying the Advanced Very High-Resolution Radiometer (AVHRR) sensor (Kramer, 2002). At this period data started to be freely available and analysis of environmental applications at global scale became possible.

Next, the Landsat era followed with the launch of Landsat-1 (1972) and multiple other satellites, the last being Landsat-8 launched in 2013 (USGS, 2016). The Landsat era has good sun-synchronous land satellites such as the Systeme Pour l'Observation de Ia Terre (SPOT)

of France and, Indian Remote Sensing Satellite (IRS) of India (Jensen, 2009). Those satellites have high resolution (nominal 2.5 - 80 meters). This period gave the true wide environment application of the remote sensing data. With the launch of Terra (1999) and Aqua satellites, the applications have multiplied. They were carrying sensors such as the Moderate Resolution Imaging Spectroradiometer (MODIS) that have daily re-visit (Melesse et al., 2007). Sentinel-2 is a mission of two satellites (Sentinel-2A: launched 23 June 2015 and Sentinel-2B: launched 7 March 2017) that will provide images of high resolution and aim to continue the Landsat and SPOT observations (Delegido et al., 2011).

Figure 2.2 shows the availability of RS time-series based on sensor type (eoPortal, 2016). To choose our sensor source we have to study what options we have. The AVHRR sensor provides the longest time series, spanning from 1978 to present date. For the rest, if we are to study longer period, we need to combine different types of remote sensing sensors, which theoretically can affect precision. The MODIS data and phenometrics are sparse (because of cloud coverage) or suffer from other sensor and/or processing related issues (Butt et al., 2011).

The combination of VGT1 on SPOT-4, VGT2 on SPOT-5 and the PROBA-V sensor provides relatively long time-series of 19 years with minimal noise caused by the sensor type variability. The mean difference of reflectance between VGT1 and VGT2 is 2%, ultimately having no significant difference from a users point of view (Vito, 2016). Furthermore, the sensor on the PROBA-V satellite is designed to have spectral and radiometric performance identical to the VGT sensors (eoPortal, 2016). The most recent data comes from the Sentinel-3 satellites which disrupt the cycle by bringing in 10-meter high-resolution data which raise for sure new challenges.

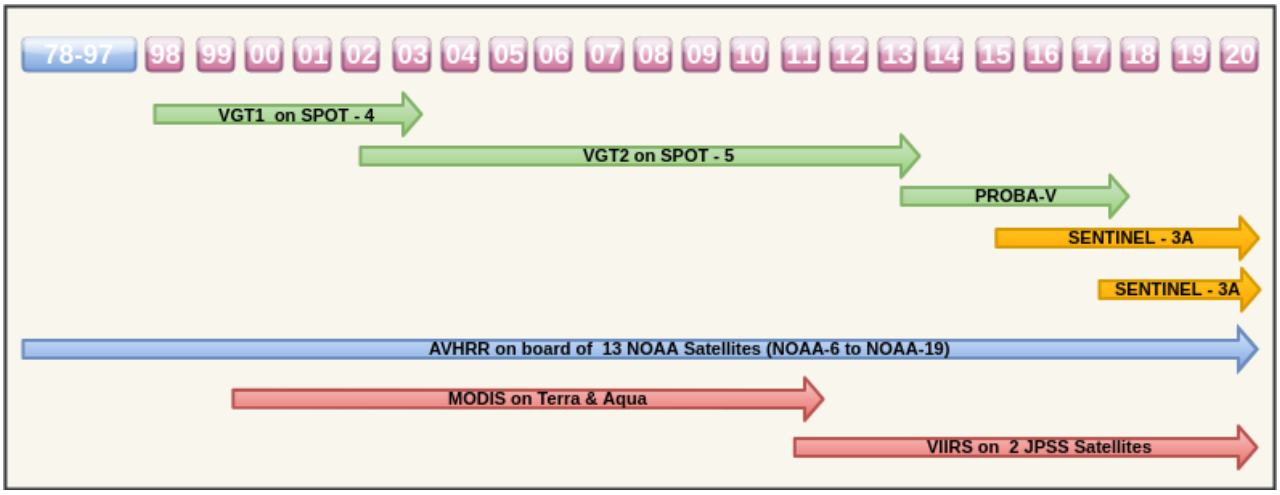


Figure 2.2: Availability of Remote Sensed Time Series Based on Sensor Type/Source. (image adapted from (eoPortal, 2016)). The top scale represents years.

2.3.2 Remote Sensing Vegetation Indices

The sensors carried aboard those satellites measure the wavelengths of light absorbed and reflected by green plants. Some pigments in the plant's leaves strongly absorb visible (red) lights and correspondingly strongly reflect near-infrared light (usgs, 2000). As plant canopy changes from spring to fall, these reflectance properties also change (usgs, 2000). With mathematical algorithms, this raw data can be transformed to RS phenology products and vegetation indices.

"A vegetation index is an indicator that describes the greenness — the relative density and health of vegetation — for each picture element, or pixel, in a satellite image" (usgs, 2000). The two main indices are the NDVI and the EVI. They are useful for assessing the biophysical properties of the land surface and are used to characterize vegetation phenology.

The most widely used vegetation index is the Normalized Difference Vegetation Index (NDVI). The NDVI formula (see Formula 1) uses the near-infrared (NIR) and red (RED) channels of a satellite's sensor (GISGeography, 2018). Figure 2.3 shows two scenarios with different reflective properties and how the NDVI is derived.

$$NDVI = \frac{NIR - RED}{NIR + RED} \tag{1}$$

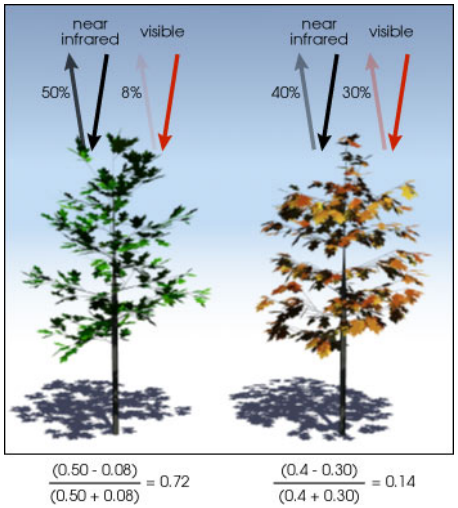


Figure 2.3: NDVI Method: Reflections (and Absorption) of Red and Near-Infrared Light. The figure shows how the reflectance property of plants is used to derive NDVI and ultimately judge a plant's phenology state (GISGeography, 2018)

The valid NDVI range is between 0 and 1. High values (0.6 to 0.9) correspond to dense vegetation, for example in the tropical forests or crops at their peak growth stage, whereas low NDVI values (0.1 or less) represent barren rock, sand or snow (usgs, 2000). Grasslands may result in moderate NDVI values. The NDVI can compensate for changing viewing angle or illumination conditions but tends to saturate over dense saturation, and it is sensitive to underlying soil color (usgs, 2000).

To improve on those shortcomings, the MODIS Science Team proposed a new formula called Enhanced Vegetation Index (EVI). The EVI better copes with ground cover below the vegetation, does not become saturated as easily as the NDVI when viewing rainforests and corrects for distortions caused by reflection of light in air particles (Observatory). The two indices complement each other in global vegetation studies. The EVI formula is shown in Formula 2 where near-infrared/blue/red are the atmospherically corrected surface reflectances, L is a canopy background adjustment, C1 and C2 are coefficients adjustments for the aerosol resistance and G is a gain factor (Liu and Huete, 1995). The values adopted for the coefficients in the equation are C1=6, C2= 7.5, L= 1, and G= 2.5 (Verhegghen et al., 2014). The EVI ranges from -1 to 1 and similarly has a practical range of 0 - 1 (GISExchange, 2016).

$$EVI = G \times \frac{NIR - RED}{NIR + C1 \times RED - C2 \times BLUE + L} \tag{2}$$

2.4 Dataset and Study Area

There are many available remotely sensed phenological products (see Appendix C.1). Several options and criteria must be decided when choosing a dataset to work on. We have many choices of datasets, at different spatial and temporal resolutions. That is, multiple RS datasets can be used to study vegetation phenology and the spatial resolution of each time series varies from sensor to sensor. For example, the AVHRR global data is available at about 8 km. whereas Sentinel-2 data is available at 10 meters.

In addition, some space and government agencies provide already computed vegetation phenometrics, but they can be of moderate quality. For instance, the USGS AVHRR phenometrics are negatively influenced by the orbital drift of the sensor and was not considered when the product was made (Ji and Brown, 2017). Furthermore, these phenometrics often cannot be immediately used for comparison because each agency has selected a different deriving method.

Given the scope of this project, we were to focus on already pre-processed data. That is, our initial plan was to work with already derived NDVI and EVI data, arranged at the spatial-temporal resolution of interest. However, only the NDVI was available. In our desire to compare both indices we decided to step back in the processing and compute the EVI from the radiometry sensor data of the satellites. VITO Earth Observations provides 10-days 1-km radiometric composites of the same SPOT-VEGETATION sensors (VITO, 2018a). By applying the EVI formula described in Section 2.3.2 we can compare the two indices for the 1998-2014 range.

The EVI calculation had three issue: shorter temporal range, grid misalignment, and cell size difference. The EVI radiometric dataset was 3 years shorted as it was not including the PROBA-V satellite data. Also, the 1 km grid was slightly misaligned by 200-300 meters, having slightly different cell size, thus introducing an additional source of noise. To best compare the NDVI and EVI, both indices must be in the same spatial and temporal resolution. Furthermore, to minimize noise, it is desirable that they have been generated from the same satellite sensor/s and data provider.

To circumvent these, we used a 3rd dataset provided by Copernicus Global Land Service (Service, 2018) which spanned on the whole temporal range of 19 years (1999-2017), combining the SPOT-VEGETATION (1998-2014) and PROBA-V (2014-present) satellite data. With it we can calculate the EVI and the NDVI coming from the same radiometric sensor data product, minimizing any source of additional noise. The product is offered in 1 km spatial resolution, 10-days composites on a global extent (covering the whole world).

We selected North America (more specifically the USA) as the analysis region, because in this area the seasonality vegetation is evident, and the data is least contaminated by solar zenith angle effects at mid- to high latitudes (Slayback et al., 2003) (Piao et al., 2006). Moreover, the USA-NPN ground-based dataset as described in Section 2.2 is present for this region.

Chapter 3

Phenology Software

In order to decide which software program can be best integrated with Spark, preliminary research on the available software was performed. An alternative would be to re-write the software's logic with Spark components and offer phenology metrics on our own, but this can easily turn in a complicated, time-consuming task and a thesis on its own. Although this decision to rely on "legacy" code (if we are to look through the Big Data prisma), would provide us with phenology metrics, we still have to find a possible and efficient solution to integrate it with our Big Data platform. Three software packages were identified capable of calculating phenological metrics from regular time series of RS data. The tools are TimeSat [1], Spirits [2], and Sen2-Agry [3].

We found that all the software is modular and the phenology based modules can be separated from the main program and independently run. However, they would need a certain degree of integration and customization in order to be deployed on a cluster environment. Also, the software requires certain file types in specific order and format. Moreover, different techniques of parallelization are required to distribute the load of processing.

After close consideration, we have decided to proceed with TimeSat. This software is modular, and we can separate the module for SOS generation from the rest of the programme. In addition, it has a way to define the processing area to be a subset of the whole area of interests, allowing us to distribute the load in Spark. Also, it offers several fitting functions and options which we can evaluate during our phenology analysis. On the contrary, the major drawbacks of Spirits, is that it is Windows-based and required additional layer of integration to run it on our Ubunto cluster, as well as it does not provide an option for spatial work distribution. Similarly, we decided to not proceed with Sen2-Agry which required a set of CentOS dependencies.

Brief introduction of TimeSat and how the seasonality metrics are derived can be found in the next Section 3.1, whereas more detailed description of the functionality and the possibility to integrate Spirits and Sen2Agry can be found in Section 3.2.

[1] http://web.nateko.lu.se/timesat/timesat.asp

[2] http://spirits.jrc.ec.europa.eu/

[3] http://www.esa-sen2agri.org/

3.1 TimeSat

TimeSat is a software package for analyzing time-series of vegetation index derived from satellite sensors. However, other types of data can also be processed: fire data, meteorological index, and eddy co-variance carbon flux data (Verbesselt et al., 2006) (Le Page et al., 2010). The program creates a fitted model using several fitting functions from which a number of seasonality parameters can be derived. The program's numerical and graphical routines are coded in Matlab and Fortran. The Fortran routines are highly vectorized and efficient for use with large data sets (Eklundha and Jönsson, 2017). TimeSat can be run both from Matlab and as a standalone executable.

To successfully integrate TimeSat in our Big Data platform, we studied the whole process of deriving seasonality metrics. The TimeSat processing logic consists of 5 steps. At each step, we paid close attention to the required structure, output and input parameters and the options provided (or not provided). A full description of the process and how the individual TimeSat modules were used can be found in Appendix A.1.

The TimeSat module for extracting seasonality parameters is called "TSF_process". It is an executable pre-compiled Fortran program. Since it is a standalone executable, it is possible to be run in parallel on different machines. It is this module we used for our Spark integration. As input, the programme requires the path to a text file, which is referred as "the settings file". Through the settings file, the input parameters and the settings for the seasonality extraction are defined as well as the files for processing. The "TSM_GUI" module was used for early exploration of the data and easily tunning the input parameters as it provides a mean of visualization between the time series and the fitting methods.

The required input format is headerless binary files (two-dimensional spatial arrays). TimeSat is image oriented and does not consider geographical coordinate systems other than column and row. The seasonality output is delivered in binary (.tpa) files accompanied by index files (.ndx). More information on the structure of the files and the seasonality parameters produced can be found in Section 5.4.2 (Figures 5.6a and 5.6b).

3.1.1 TimeSat Seasonality Extraction

In this Section we will briefly introduce the core concepts of seasonality parameter extraction. For more information refer to the TimeSat manual (Eklundha and Jönsson, 2017). Each image is given an index value. By extracting the values at a position (row,col) from each image, a time-series is obtained for each pixel pair (Figure 3.1).

Next, a smoothing function will be fitted to the time-series and used to extract the seasonality parameters (Figure 3.2a). Thirteen parameters can be determined based on the fit, but we are only interested in SOS. (Figure 3.2a). TimeSat implements 3 fitting functions for seasonality extraction: Savitzky-Golay, Asymm. Gaussian and Double Logistic. The function's parameters can be tuned for optimum results. TimeSat provides an additional preprocessing step to remove spikes and outliers with two methods: median filtering and weights from STL. Based on the fit, the seasonality parameters are extracted (Figure 3.2a). In the general case a time-series spanning n years will give seasonality parameters for the **n − 1 center-most seasons** and therefore TimeSat is programmed to extract n-1 seasons. Figure 3.2b shows such a scenario where for 5 years, 4 seasons are extracted.

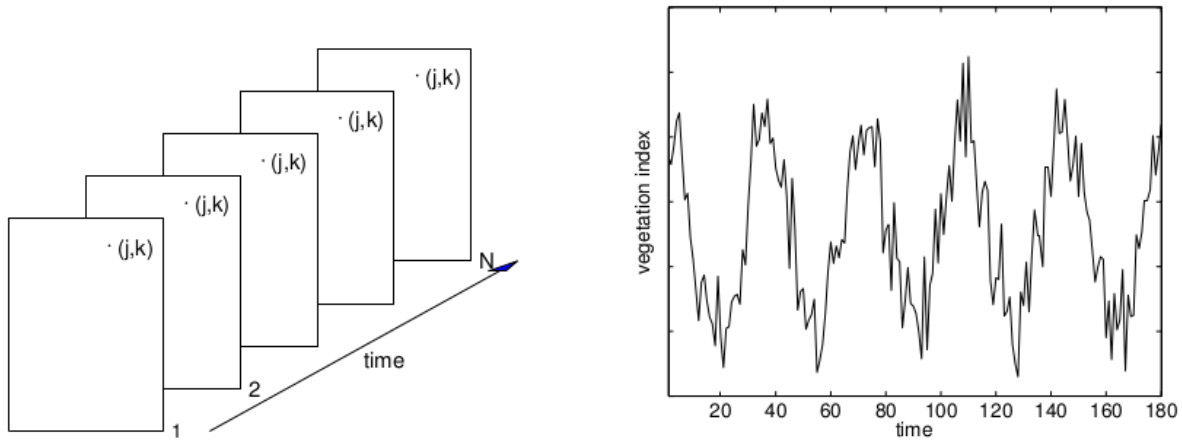
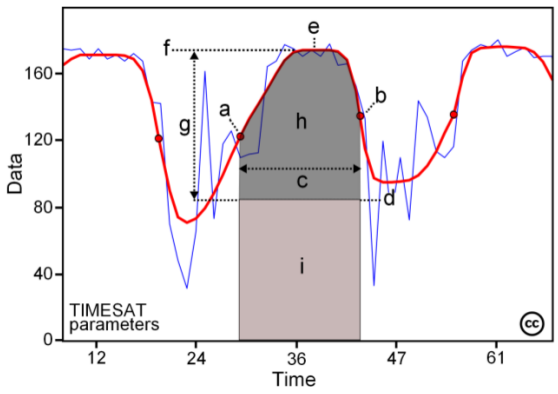
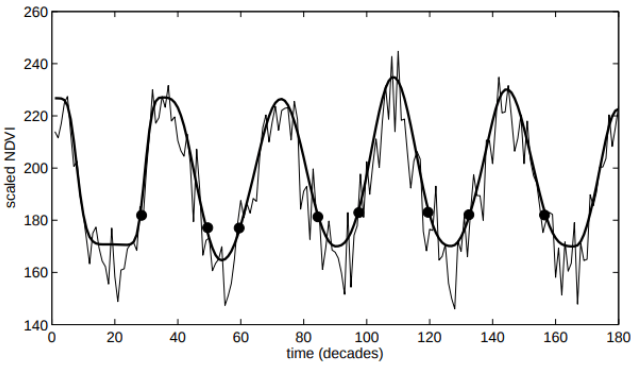


Figure 3.1: Transformation of Image Data to Time Series (Eklundha und Jönsson, 2017)



(a) Seasonal parameters: (a) start of season, (b) end of season, (c) length of season, (d) base value, (e) time of middle of season, (f) maximum value,(g) amplitude, (h) small integrated value



(b) Time-series covering a period of 5 years. Only seasonality parameters from the 4 full seasons can be determined from the fitted functions. Circles show start and end of season

Figure 3.2: TimeSat Seasonality Extraction (Eklundha und Jönsson, 2017)

3.1.2 Standalone Execution

To better understand the mechanisms, we followed the TimeSat tutorial and created a Scala script to locally test the TimeSat capabilities and explore the end-to-end pipeline from reading NDVI files to producing seasonality images. We also wanted to understand how we can run TimeSat in a distributed environment. The module of interest, as mentioned, is "TSF_process". The script was not run in the cluster or using Spark API but rather started in a single laptop machine.

TSF_process is a single threaded, however, TimeSat provides a script to run multiple instances in a single machine. In such a mode, TimeSat creates a bash script which will run the executable several times. The process scheduler will execute the processes in parallel on each available logical core. To test that, we created a spatial work division (effectively load-balancing the work among the processes), and when the script was run, it started the executable several times for each of the spatial regions defined. For details on how the test was run and the set-up refer to Appendix A.2. Figure 3.3 shows running TimeSat (i.e TSF_process) in parallel on 7 out of the 8 available cores.

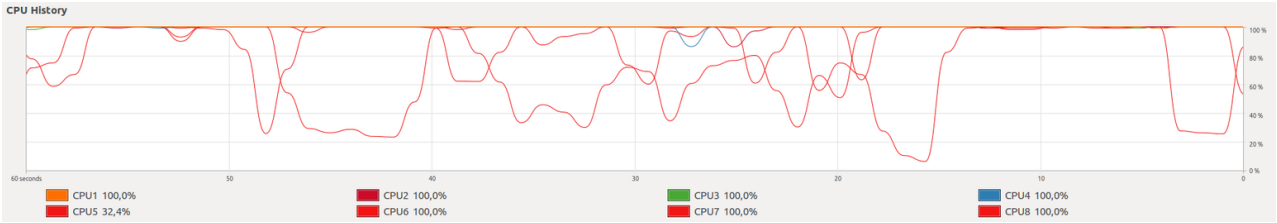


Figure 3.3: CPU load on 7 out of 8 cores

The executables fully utilize the CPU power per core, whereas the process scheduler switches between the available 8 cores. The graph also shows the computation of phenology metrics is CPU bound. With this experiment, we demonstrated the possibility of integrating and parallelizing the TSF_process executable within a distributed environment. Through the ”settings file” we can distribute and load-balance the work in our cluster by specifying a different area (i.e. region of interest (ROI)) on which the program will produce results.

To assess the performance, a seasonality extraction test was performed on a subset of 3 years. The parallel script was run on the full spatial range on 2, 4 and 7 Intel Core i7-4710HG 2.50 GH cores using the Savitzky–Golay fitting function. The computation finished for 8h 31m., 4h 17 m., and 3 h, respectively.

It is important to note that the provided parallel script, unfortunately, does not support inter-process communication neither allow for the processes to share the data. It simply runs the single-process variant multiple times. This same technique will be used for our Spark integration, as we will explain in Section 5.3, where the Spark task scheduler will do this automatically.

3.2 Other Software

This Section is to describe the software packages Spirits and Sen2-Agri, which were evaluated but deemed unsuitable to be integrated with Spark.

3.2.1 Spirits

Spirits is another software that provides time series processing and seasonality analysis. The installation is straightforward. The main requirement is Java 1.6 (or above) to be installed on the target computer. The Spirits tutorial was followed to understand the software capabilities (Vito, 2018b).

The Spirits accepts as input the so-called ”modified ENVI” file format. Similarly to TimeSat, the raster images must be simple flat-binary files (*.img). The difference is that now a small associated ASCII-TXT annotation file (*.hdr) is required. This separate text header file provides information about the dimensions of the image, bands, data format etc. The ENVI header file is based on the ENVI principles with some additional fields that are a requirement for the time series analysis. For example, ”flags” can be defined to specify fixed values for ”snow”,”water”,”clouds” etc.

Spirits can process series of images with a temporal resolution of one day, 10-days, one month or a year. The names of the files must agree with the general template [prefix][DATE][suffix].[ext]. As a prefix, the user can define the sensor used to produce the image, and the periodicity. As suffix one can set the image variable (for example NDVI). The data must be specified according to twelve data formats (Vito, 2018b).

The software is based on several main concepts. A user might create a map template which will be used to automatically create a set of maps over the time series. This allows for a large number of maps to be generated with only a few clicks. A "scenario" can be defined in order to save the parameters and be re-used on a later stage. The functionalities are divided into tools. In the context of seasonality computing 3 tools are available: "Phenology", "Detect seasons", "Progress of Season" (Vito, 2018b).

The Phenology tool can extract twelve different phenological parameters from a series of periodic IMGs. For the available options and input settings, please refer to Figure 3.4. The executable name is called "PHENO.exe". The "Detect Seasons" detects (per pixel) the number of seasons (0,1,2) for a given "target year". Please refer to Figure 3.5 for detailed input parameters. For each detected season the dates of the start (SOS), the maximum (MOS) and the end (EOS) of the season will be derived. The output is 6 images with dekad of SOS/MOS/EOS for 2 seasons. The "Progress of Season", as the name suggests, can calculate the progress of season, per dekad, over the complete season. The tool operates on the phenological IMGs (SOS/EOS) created by the "detect seasons" tool.

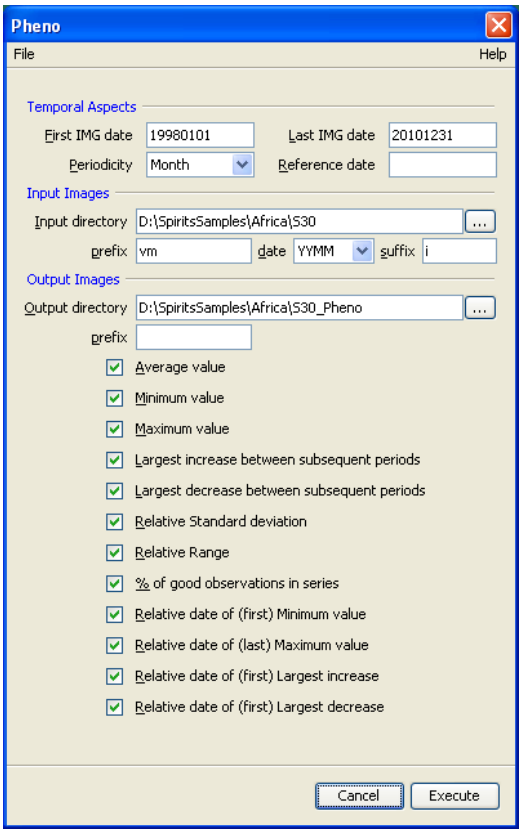
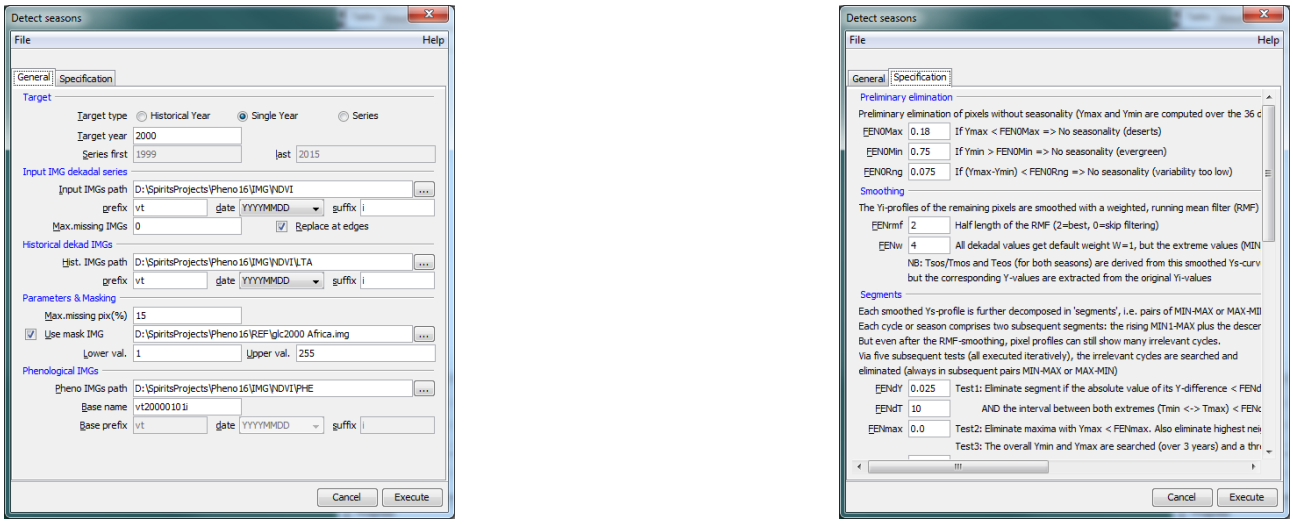


Figure 3.4: Tool to Extract Phenological Parameters from a Series of Periodic IMGs. Twelve Different Phenological Parameters can be Extracted.



(a) General Parameters

(b) Specifications

Figure 3.5: Tool for Deriving Phenological Parameter IMGs

The Spirits project invokes executables from two main libraries - Geospatial Data Abstraction Library (GDAL) and Global Image Processing Software (GLIMPSE). The phenology based tools mentioned above are all GLIMPSE executables. The GDAL executables are Windows-based executables.

With ”wine”, a programme providing a compatibility layer capable of running Windows applications on several POSIX-compliant operating systems, such as Linux, macOS etc. (Wine, 2018) the GDAL executables were successfully invoked. The glimpse executables, however, seem to have been adapted. Although valid from within the Spirits GUI, they were unable to run from the console. When invoked from the command prompt the modules would ’seg’ fault, giving little insight on what is the issue. Eventually, a ”LaunchGLIMPSE.bat” script was found in the projects’ source and with wine we successfully ran the modules in the terminal. However, this approach requires having another layer of integration, such as the ”wine” software.

We found that the tools can be run from within the nodes. However, they not provide an option to specify the region of interest, but rather work on the full image dimensions. Also, there is no raw output, but a set of images for a ”target year”. A distribution can be achieved by creating a temporal split. However, as a requirement by the fitting function minimum of 108 images or 3 full years must be supplied for a given target year in the middle. Therefore, we have to provide this two-sided buffer each time we want to compute for a target year. This would increase the computation load significantly when compared to creating a spatial load distribution.

3.2.2 Sen2-Agri

Sen2-Agri can generate a set of products from Sentinel-2 and Landsat-8 time series. The software is open-source, allowing for anyone interested to tailor it to his needs (Consortium, 2018). The software package is rather big and sophisticated. The system is composed of a set of independent processing modules orchestrated by a data-driven approach (Sen2Agri, 2018). These modules are composed of a set of tools which can be re-used into other systems.

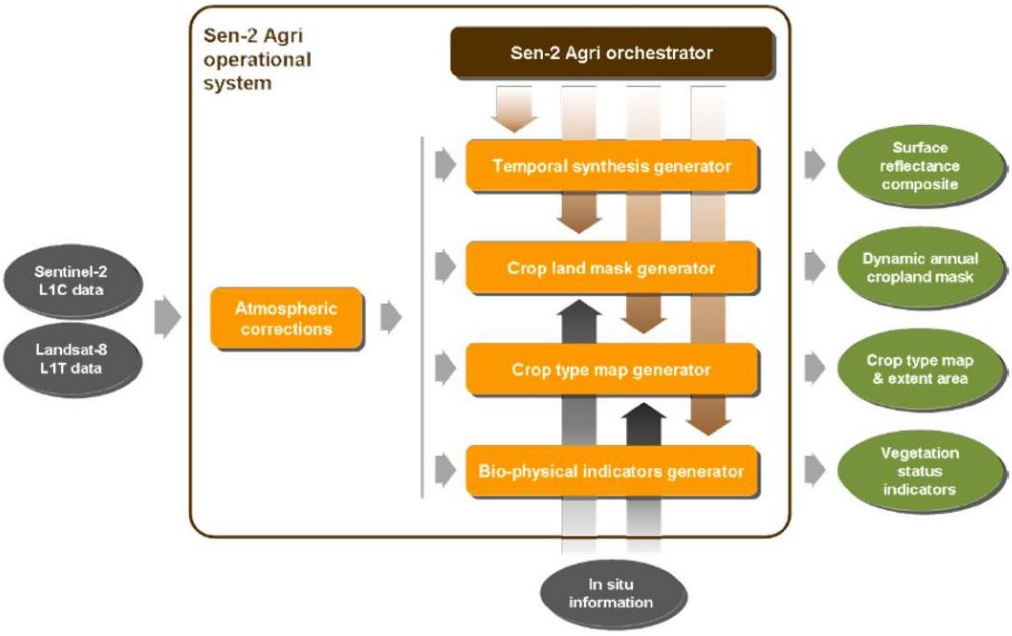


Figure 3.6: Logical Data Flow of the Sen2-Agri System (Sen2Agri, 2018)

The procedure and the data flow is shown in Figure 3.6. A user should download either Sentinel-2 (L1C) or Landsat 8 (L1T) data and run the mandatory atmospheric correction (L2A). The output is used as an input to several so-called processors that will generate a set of different products for agricultural monitoring. Through the biophysical indicators generator, and more specifically the L3B processor, one can generate vegetation status indicators.

The biophysical vegetation (L3B) processor provides three types of products about the evolution of the green season: NDVI, LAI and Phenology indices (Sen2Agri, 2018). The Phenology indices, also referred to as NDVI metrics, inform about specific key parameters of the growing season (starting date, season length, and date of maximum growth rate). The specifications of all products generated by the biophysical vegetation processor are shown in Appendix B.1. The NDVI metrics are extracted from the time-series through a double logistic function (Sen2Agri, 2018).

The software requires a set of dependencies in order to be run successfully. First and foremost it is a CentOS distribution, and therefore we are unable to install it in our Ubuntu cluster. The required atmospheric Correction (L2A) module also requires a CentOS based library called MACCS (MACCS, 2018). Although they support RedHat distributions, the rest of the Sen2-Agri installation would fail due to dependencies only available in CentOS.

The Sent2-Agri software was successfully installed and ran from within a virtual box running CentOS. As the software is opened- sourced, it would be possible to extract the required code and run it on our cluster. Also, there is a possibility to install the core components of the software in manual mode. When running the ”sen2agriPlatformInstallAndConfigCore.sh” the script will use the ”yum” packet manager and install the Sen2-Agri processors provided as python scripts. The L3B processor is exposed to manual command line execution through the ”pheno_processing.py”. With it, we can extract NDVI phenometrics both SOS and EOS. However, due to the mandatory CentOS requirement, we decided the opt-out from this software and to proceed with TimeSat.

Chapter 4

The Computational Platform

Deriving phenology metrics at a continental scale is a challenging task when done by a single machine. Working with datasets with high spatial and temporal resolutions imposes certain computational requirements. The procedure involves CPU intensive curve fitting algorithms. They work on a pixel level, extracting values at a pixel (row,col) for consecutive times to obtain a time-series for this pixel. Due to this fine-grained granularity, and by the results from the local performance test as described in Section 3.1.2, the task in hand is best solved by a distributed computing solution.

One of the decisions for our analysis was to find the appropriate technology and tools in order to distribute the process of our large dataset efficiently and reliably. Two commercial frameworks exist for large-scale data processing - Hadoop[1] and Spark[2]. We have chosen the latter over Hadoop MapReduce, not only due to its novelty, but also the emphasis placed on speed, flexibility and hiding much of the complexity. Its flexibility allowed us to easily experiment with different methods and configurations. The architecture of the platform using Spark was designed and implemented by The Netherlands eScience Center and University of Twente [3].

The platform allows the user to solve large-scale data science challenges and to do a simple exploratory analysis. The main design goals were to be scalable and easy to use (Zurita-Milla et al., 2017). To better understand the workflow and the structure we are to first look at the platform's main components. The platform's architecture (see Figure 4.1) is organized in three layers: storage layer (Section 4.1), processing layer (Section 4.2) and *JupyterHub* services for user-interaction (Section 4.3). The platform's deployment is discussed in Section 4.4.

[1]https://hadoop.apache.org/
[2]https://spark.apache.org/
[3]https://github.com/phenology/infrastructure

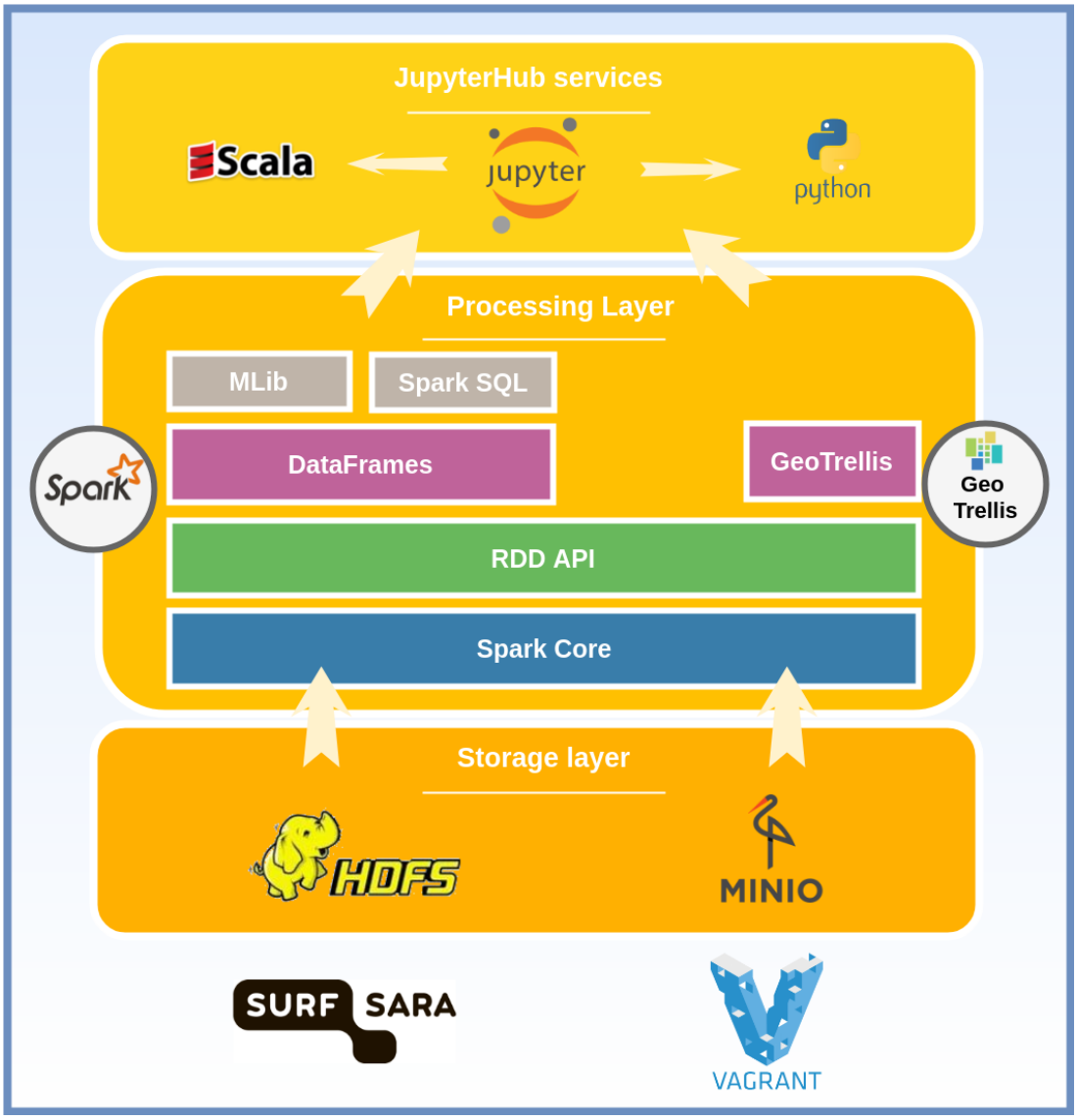


Figure 4.1: Architecture of the Computational Platform

4.1 Storage Layer

The storage layer is where we save our input data, intermediate data and results. We used both the file-based Hadoop Distributed File System (HDFS)[4] and the object-based Minio storage with Amazon S3 compatible API[5]. Both storage systems are used to store results and input data.

HDFS is used by Spark to exploit data locality and store intermediate files when processing. It offers redundant, fault-tolerant data storage. The data is distributed through the cluster. This allows storing big amount of data, which would otherwise not fit in one machine. Due to its replication factor, the system is reliable in case of node failures. What is more, computation takes place near the data. The HDFS's rack awareness assists Spark to place the computation near the data by contacting the nearest replica. Such optimization is visible on big data sets, it reduces the network traffic and increases the throughput (tutorialspoint, 2018).

[4]https://hadoop.apache.org/docs/r1.2.1/

[5]https://www.minio.io/

By exploiting the Minio's object storage, we have an easy way to access our data (through HTTP) as well as other advantages such as agility. With the Minio's S3 API, which is becoming the de-facto API standard for object storage systems provided by commercial cloud platforms such as Amazon S3[6], Microsoft Azure[7] and Google Cloud[8], one can easily migrate to other commercial systems.

In addition, object storage is usually used as a data bank to store the phenology or other remote sensing data products. That is because large data banks of data, in this case, remote sensing data, should be hosted in separate storage. Often in commercial or national infrastructures there is such an external data storage used to store large amount of remote sensing data (such as Landsat's data at Amazon) (AWS, 2018).

To provide for TimeSat to read and write into a POSIX file system, we mount the Minio buckets as local file system directories. This was a requirement imposed by TimeSat as it can only load local files. More detailed information can be found in Section 5.3. At first 'goofys'[9] (a high-performance, POSIX-ish S3 file system) was installed. The software proved to be unreliable as files were partially downloaded or corrupted. As a result, we resorted to the official Amazon S3 fuse - 's3fs'[10]. Generally, S3 cannot provide the same performance as a local file system. Appending to files require rewriting the entire file. Also, listing files in a directory have poor performance due to network latencies. However, a local disk cache on the Bucket will cache the data locally. The subsequent runs use the local copy, effectively removing the network overhead completely.

4.2 Processing Layer

The processing layer is the place where all the data crunching is performed. The computations will use distributed data structures and Spark internals to efficiently distribute processing. The main used components are the Spark's Resilient Distributed Datasets (RDDs), its machine learning library *SparkMLlib*[11] and *GeoTrellis*[12] - a project for high-performance geographic data processing using Spark. With *GeoTrellis* the datasets are read into RDDs; thus we can exploit the Spark's internals for distributed data processing. *DataFrames* are used to take advantage of the SQL Spark interface, combining the benefits of the RDDs.

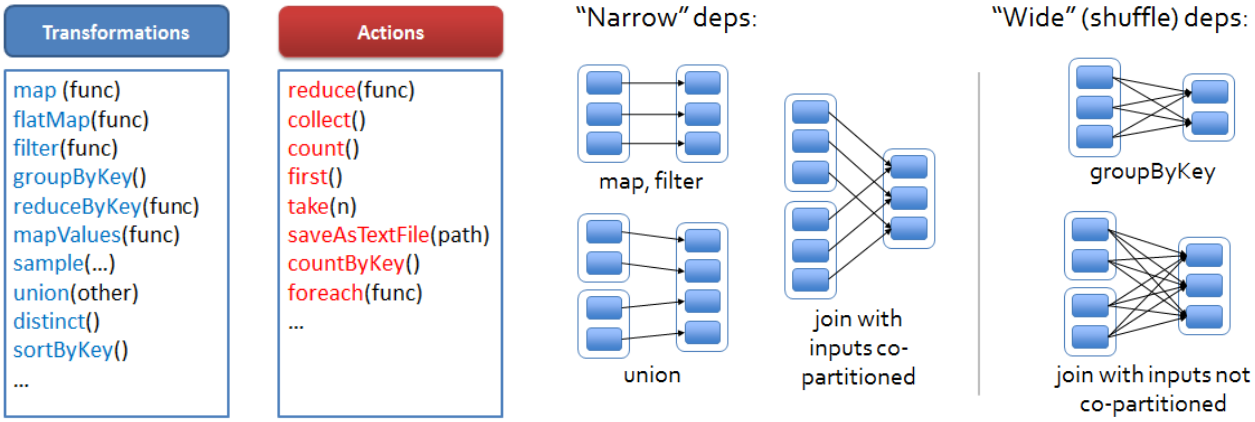
Spark Core Components Introduction

Before going further into the next Sections, it is important to introduce some Spark components and concepts. The RDD is the primarily data abstraction used as a mean of parallelization. Furthermore, the RDD's **partitions** are logical chunks (a.k.a split) that help parallelize and distribute the data across the cluster. In the following paragraphs, the Spark's operations are introduced and later the execution process is outlined.

[6]https://aws.amazon.com/s3/
[7]https://azure.microsoft.com/services/storage/blobs/
[8]https://cloud.google.com/storage/
[9]https://github.com/kahing/goofys
[10]https://github.com/s3fs-fuse/s3fs-fuse
[11]https://spark.apache.org/mllib/
[12]https://geotrellis.io/

RDD Operations

Each RDD can have 2 sets of parallel operations: *transformations* and *actions*. The most popular operations are shown in Figure 4.2a whereas the operation's dependency types are shown in Figure 4.2b.



(a) Types of RDD Operations (Nguyen, 2014)

(b) RDD Dependency Types (Bharadwaj, 2016)

Figure 4.2: RDD Operations and Dependencies Between Operations

- A **Transformation** is a function that applied to an RDD produces one or more new RDD(s) (jaceklaskowsk, 2018). This is because RDDs are immutable and therefore transformations always require creating new RDDs. RDD lineage graph is built when multiple transformations are created, connecting the parent RDDs with the final RDD(s). The graph contains a chain of lazily build RDD dependencies. It is also necessarily a **direct acyclic graph (DAG)** since a loop is impossible to be present in it.

  A transformation can be either "narrow" or "wide":

  - In **narrow transformation**, each partition of the parent RDD is used by at most one partitions of the child RDD (see Figure 4.2b). This implies that work can be executed locally on each of the nodes and we don't have to do expensive data movement (i.e. shuffle). A map, filter, flatMap or union operations are narrow operations. If two RDD are similarly hashed with the same number of partitions, a join operation can be narrow operation as well. This is because the same key will always be in the same partition in the two parent RDD. However, a join operation is usually a "wide" transformation.
  - In **wide transformation** multiple child partitions may depend on one partition of the parent RDD (Nguyen, 2014). This required an expensive shuffling of data (unless the parent RDD(s) are hash-partitioned) since all shuffle data must be written to disk and then transferred over the network. Such wide operations are *sortByKey*, *groupByKey*, *join*, *cartesian product*.

- An **Action** applies all the transformations in the RDD lineage graph and then it performs the action to obtain results. When we are lazily building the DAG with different transformations we are keeping track of a bunch of meta-data. It is not until we call an action when the Spark driver JVM decides how to carve up the DAG into stage boundaries.

A Spark Job

There are several concepts of work in Spark with decreasing granularity: application, job, stage, and task.

1. An **Application** is the main functionality of a Spark programme.
2. A **Job** is created when an action is invoked on an RDD. A job is a work submitted to Spark. For each 'action' invoked, a job will be created, followed by building the DAG based on the RDD operations (Figure 4.3).
3. A **Stage** or multiple stages is/are created for each job based on the shuffle boundaries. The process of dividing the job into stages is done by the **DAG Scheduler** (Figure 4.3) (Farooqui, 2015). Because of the lazy evaluation build up, the DAG Scheduler will be able to optimize the stages before submitting the job. It will pipeline the narrow operations within the stage, try to minimize shuffle based on the join algorithm used, reuse previously cached data etc. (Nguyen, 2014).
4. **Tasks:** Each **stage** will be submitted to the **task scheduler** which will further divide the stages into tasks based on the number of partitions in the RDD. Now the driver will schedule individual tasks to be run on individual executors.(Figure 4.3)

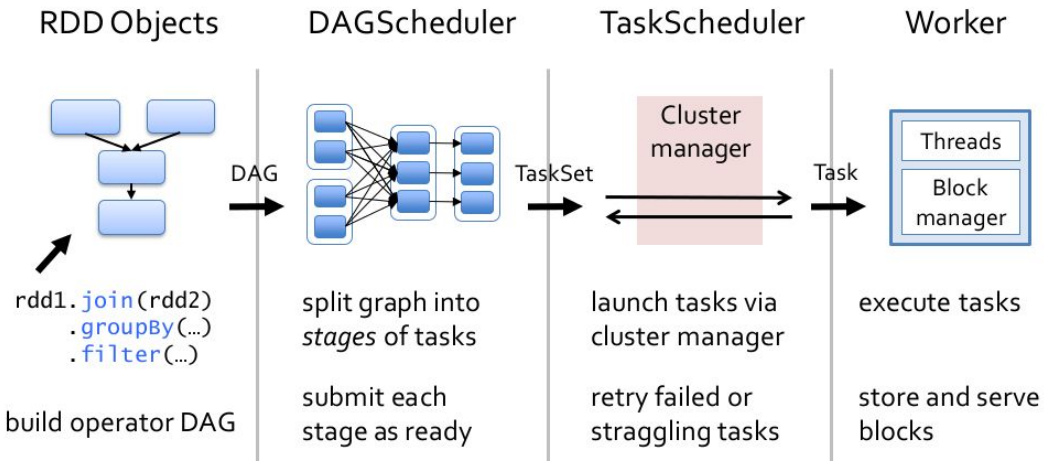


Figure 4.3: The Spark Job Submission Process (quangnhathoang, 2014)

4.3 User Interface

Spark provides bindings to popular languages like Python and R, as well as the more enterprise-friendly Java and Scala. To express the computation in these languages Jupyter notebooks [13] are used. With *JupyterHub* we have multiple notebooks in different languages interacting with Spark. The interface provides (among others) Python (pySpark) and Scala kernels, allowing us to code in the most suitable language. The language of choice for this project is Scala.

Among the many advantages of *JupyterHub*, it allows us to easily collaborate and share live code and narrative text between students or colleges, but also we can work with multiple "kernels" and push execution to the Spark cluster directly through the Apache Toree [14] interface.

[13] http://jupyter.org/

[14] https://github.com/apache/incubator-toree

4.4 Platform Deployment

The platform can be simulated on a local machine, i.e., provide a local development environment or run on a remote cluster. For deployment, a project named Emma [15] is used. Emma helps the users to deploy the same platform in different locations. The configuration of the cluster and versions of the different libraries can be saved for tools provenance.

The platform runs on an infrastructure of machines (i.e. nodes) that are prepared/constructed by either prepared cloud virtual nodes or constructed Vagrant boxes in the case of a local set-up. Once the nodes are prepared they are provisioned using Ansible[16], an automation tool for IT infrastructures. With Ansible we can deploy a platform with the same features at different locations, such as local cluster, national infrastructure or even a commercial cloud provider. Ansible playbooks are used to install and configure the platform's components shown in Figure 4.1 and with a host inventory file, a user can define which nodes Ansible should connect to (see Figure 4.4). The configuration is run from a host machine (or node) and via SSH, Ansible will provision the target nodes.

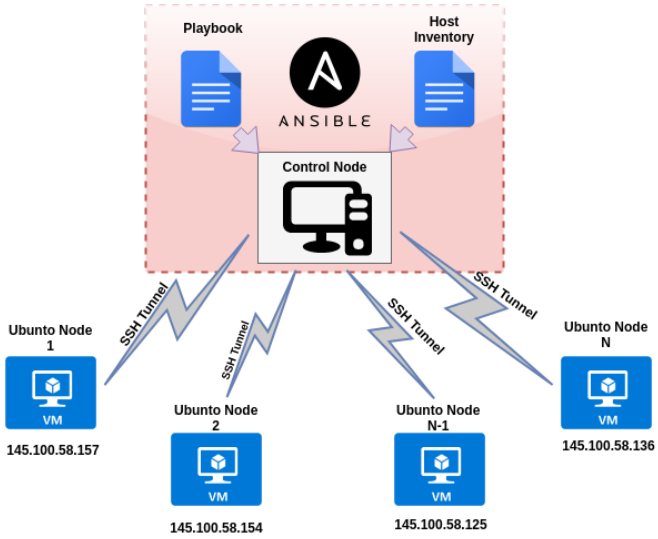


Figure 4.4: Ansible Provisioning and Node Management

At first, a local deployment with Vagrant boxes was used to test the technology. Vagrant allows to deploy on local infrastructure at the universities, but also to create a local test/development environment without having to spend credits from a cloud provider. This allowed for early exploration of the data and hands-on experience with the platform. The laptop on which was run was upgraded to 16 GB of RAM which allowed for running 2 Vagrant boxes with the virtualization software - *VirtualBox*[17].

The Vagrant integration made managing the Virtual Boxes easy and in a single flow. Initially, an experimental approach was taken, and we only worked with small amount of data. When the full data processing had to be started, we migrated to the SURFsara HPC Cloud facility [18]. The migration was done with ease since only the hosts node addresses and the private keys had to be changed. The SURFsara interface allows us to configure and manage our own cluster in the cloud. The project was done with the SURFsara cloud, but one can easily migrate to commercial solutions, such as,for example, Amazon EC2 [19].

[15] https://github.com/nlesc-sherlock/emma
[16] https://www.ansible.com/
[17] https://www.virtualbox.org/
[18] https://userinfo.surfsara.nl/systems/hpc-cloud
[19] https://aws.amazon.com/ec2/

Chapter 5

TimeSat Spark Big Data Processing

This Chapter describes the process of integrating TimeSat with the computational platform as well as its distributed configuration. Figure 5.1 displays the steps required to reach our final goal of comparing the SOS products with the ground measurements. From the raw satellite data we are going to compute the EVI and the NDVI, which we will use as input to TimeSat to extract the SOS. Using the three extraction methods and both indices we will intercompare the results with the ground measurements. This Chapter describes the technicalities of the process, whereas the platform's evaluation is shown in Chapter 6 and the phenology results are shown in Chapter 7.

The following Sections show more in-dept description of our methodology. The pre-processing steps and a description of the radiometric satellite dataset are shown in Section 5.1. Section 5.2 describes the EVI and the NDVI calculations. Sections 5.3 describe the distributed configuration whereas Section 5.4 describes the generation of the SOS products. Section 5.5 briefly discusses the technical aspects used to compare and evaluate the results.

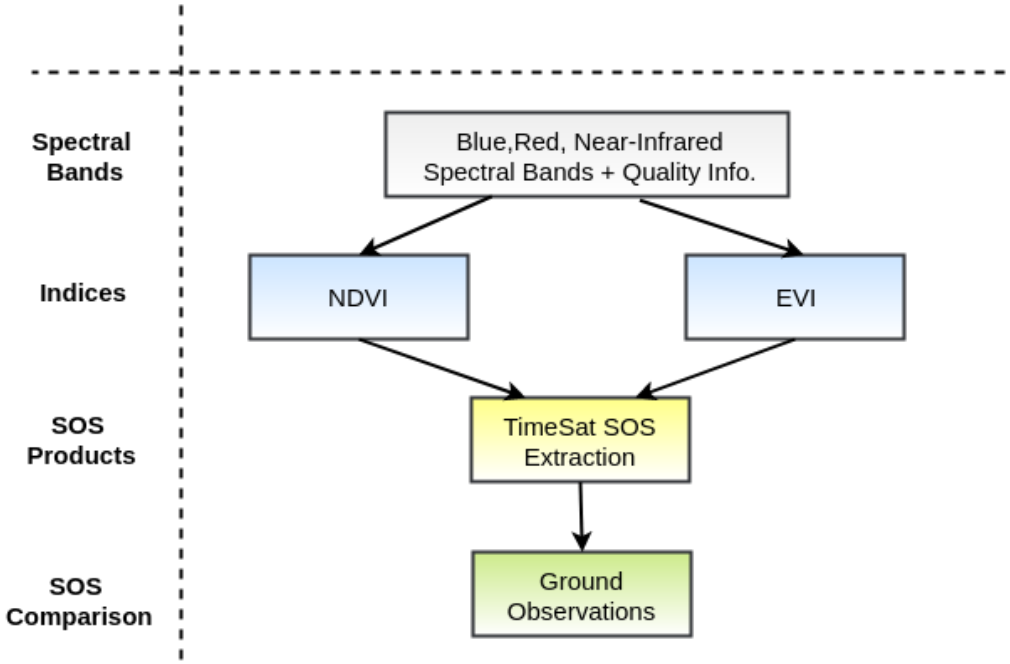


Figure 5.1: Workflow of the Processing Steps

5.1 Data Preparation - Radiometric Satellite Data

As described in Section 2.4 the dataset is provided by the Copernicous Global Land Service and combines the SPOT-VEGETATION and PROBA-V satellite data and it was downloaded with FileZilla, resulting in 767 GB of size. The 684 files are in NetCDF(.nc)[1] format contain several layers inside.

Before loading the data in HDFS, the files were locally pre-processed. Since, the files contain several additional layers, with a python script only the layers of interest were extracted. Such layers are: the blue, red and near-infrared bands, which map to the layer names as *REF_NOR_BLUE*, *REF_NOR_RED*, *REF_NOR_NIR* and a status map indicating the quality of the data referred as *TOCR_QFLAG*. The status map was later used to filter only the good quality pixels.

Additionally, as the layers are global composites, they were cropped out to cover only the USA region. This contributed to reduced data load. The resulting dimension of the images is 6500 by 3000 pixels. Furthermore, the data values are efficiently stored in Digital Numbers (DN). That is, they are stored in 8 bits, ranging from 0-255, instead of representing the true vegetation indice floating-point range with 16 bits. Finally, the bands plus the status map were arranged in folders and put on HDFS for further processing.

5.2 Calculating NDVI and EVI

The formulas used for the NDVI and EVI generation were described in Section 2.3.2 and accordingly applied. Additionally, the status map was used to filter out sea/water, snow, mixed aerosols, out of range or invalid input values and saturated bands pixels. Its structure is displayed in Table 5.1. Once calculated, the results were saved to headerless binary images (two-dimensional spatial arrays, *.img), since it is the only supported format by TimeSat, ready to be used as input to extract the SOS.

BIT	BIT = 0	BIT = 1
Bit 1: Land/Sea	Land	Sea
Bit 2: Snow status	Clear	Snow
Bit 3: Cloud/shadow status	Clear	Suspect
Bit 4: Aerosol status	Pure	Mixed
Bit 5: Aerosol source	Modis	Latitudinal gradient
Bit 6: Input status	OK	Out of range or invalid
Bit 7: Blue band (B0) saturation status	OK	Saturated
Bit 8: Red band (B2) saturation status	OK	Saturated
Bit 9: Near infrared band (B3) saturation status	OK	Saturated

Table 5.1: 9-Bit Status Map Structure (Service, 2018)

[1] https://www.unidata.ucar.edu/software/netcdf/docs/netcdf_introduction.html

The DAG of the whole execution plan is shown in Figure 5.2. Each stage consists of narrow operations pipped together (map, filter, etc.), whereas between stages connections are caused by wide operations (joins, sortBy, etc.). The boxes refer to the Spark operations that the user calls in his/her code. The dots in these boxes represent RDDs created in the corresponding operations (Or, 2015). We are going to annotate each stage with numbers (1 to 5) and inside the 5th stage number the operations (5.1 to 5.6).

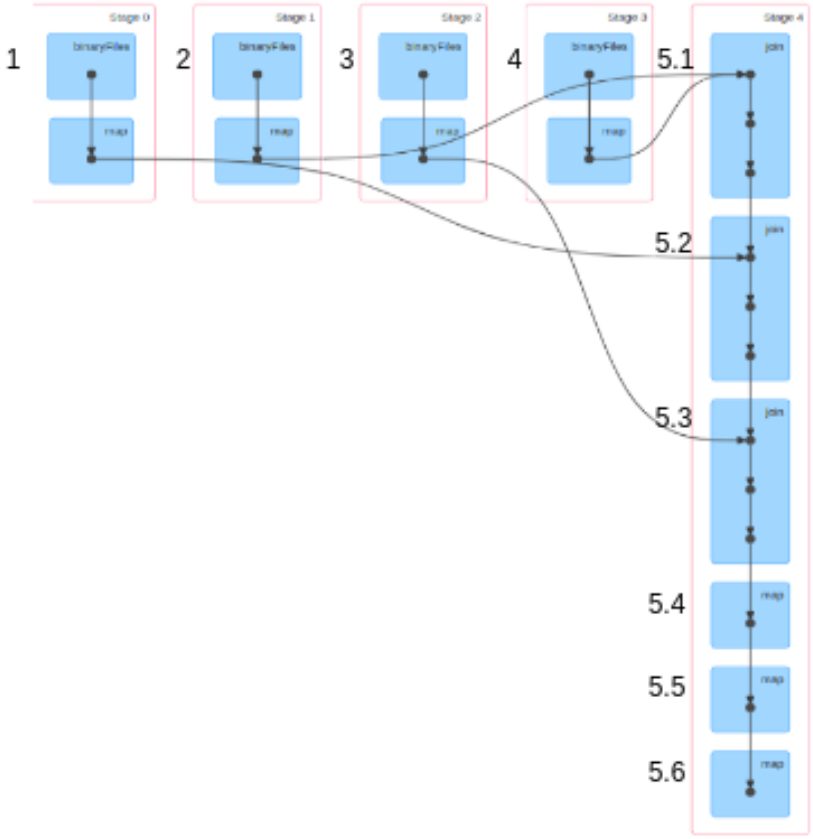


Figure 5.2: Spark Stages and Operations for Generating the EVI and the NDVI

The operations in stages 1 to 4 load the data from HDFS into Spark RDDs using the Spark's *binaryFiles* [2] method. Also, a subsequent map operation is called to crop out the file's name from the rest of the path. To most efficiently calculate the indices, all the required information must be in the same RDD record in order to avoid shuffle or unnecessary filters. To achieve this, 3 joins are performed (5.1, 5.2, 5.3) on the file's name. Operation 5.4 is to map the output of the joins to one tuple of 5 elements in the form *RDD[(fileName, blue band, red band, nir band, quality info)]*.

In operation 5.5 the VI is calculated, one run calculating the EVI and one run calculating the NDVI. To convert from digital numbers (DN) to physical values, a scale factor of 0.0005 is applied to each band. The scale factor converts the 8-bit values to floating point reflectance values in the range [0 - 1]. The EVI or NDVI values are calculated per pixel only if the criteria for quality data is met; for each pixel from any of the bands the radiometric quality must be good, the pixel must represent land, there must be no ice or aerosols, and the input value must be valid (bits 1,2,4,6,7,8,9 = 0) (Table 5.1). If those conditions are not met, a value outside the valid range is returned (i.e. -5). In 5.6 the results are saved to Minio, ready to be ingested by TimeSat.

[2]https://spark.apache.org/docs/1.5.0/api/java/org/apache/spark/SparkContext.html

5.3 Data Loading and TimeSat Parallel Execution

The standalone version, presented in Section 3.1.2, has shown that data parallelism is easily exploited to scale up. To scale out we intend to use Spark, however, three major questions need to be solved. In the following paragraphs, we are going to describe our approach. The challenges are:

- How to make the indice data available on each node?
- How to define and separate the load for processing?
- How to run the TimeSat module in parallel with Spark?

Providing a Local Storage - Mount the Minio Buckets

First step is to make the data available on each of the nodes. The TimeSat executable has two main restrictions - First, it only can work with local files. That is we can't pipe an RDD of the files as it is not a Spark executable neither read the files from HDFS. However, this gives us the opportunity to explore the object storage component of our platform. We provided a local shared space for all the nodes. With the s3fs fuse (introduced in Section 4) a remote Minio bucket can be locally mounted on each of the nodes, effectively providing a common local path (see Figure 5.3). When TimeSat is run, the s3fs fuse will download the files locally. The bucket storing the data needed for the TimeSat processing is cached, and the other bucket storing all other files is not. This enables local file caching which minimized downloads.

The second restriction is that by default the output files are saved in the local path where the execution is run, and TimeSat does not provide control over that. What is more, the local path will differ per tile as it based on a combination of the job_id and the task number as set by Spark. What we do is to move the output from TimeSat from within the Spark's task over the shared Minio space.

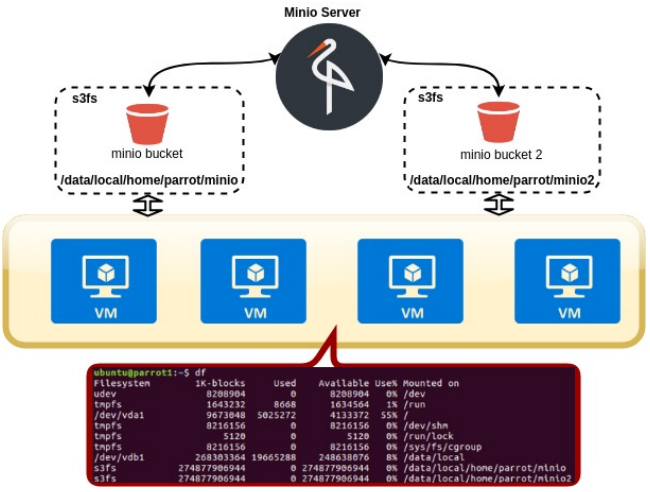


Figure 5.3: Figure shows how the remote Minio buckets were locally mounted on each of the nodes with the s3fs fuse

Data Partitioning

The load division can be either spatial or temporal. That is, we can either work on the whole USA map and separate on years range or divide the USA map into smaller regions and work on the full temporal series. A temporal partitioning has two major drawbacks. First, the time-series must be of a minimum length of 3 years, a restricting imposed by TimeSat. That would drastically impact on our granularity. Moreover, TimeSat is set to find n- 1 seasons, where n is the number of years. To account for this, the temporal separation must be overlapped, further reducing our performance. Whereas, in spatial partitioning, neither of the drawbacks exists as there will be no unnecessary computation or hampering the reduction of the extent of our granularity. Therefore, we have chosen to perform spatial partitioning. Through a "settings file" we can easily define the so-called "region of interest" and distribute the load in the cluster.

In the first iteration, we divided the space row-wise. However, due to the USA territory shape, much more terrestrial regions will end up in the rows in the North. This unevenly divides the workload. To improve on this, we made a column-wise separation as well. This way each region of interest will have a quadrilateral polygon shape with 90 degrees of internal angles. We are going to refer to a single region of interest as a "tile". The space will now appear as a grid of tiles, where all the tiles cover the whole extent. Figure 5.4 shows a 35 x 85 grid (2975 tiles). A separate "settings file" is created for each tile defining its spatial region. Please note that the width and height on each depicted tile are equal, resulting in a square, but in reality they can defer, resulting in any rectangular shaped tile.

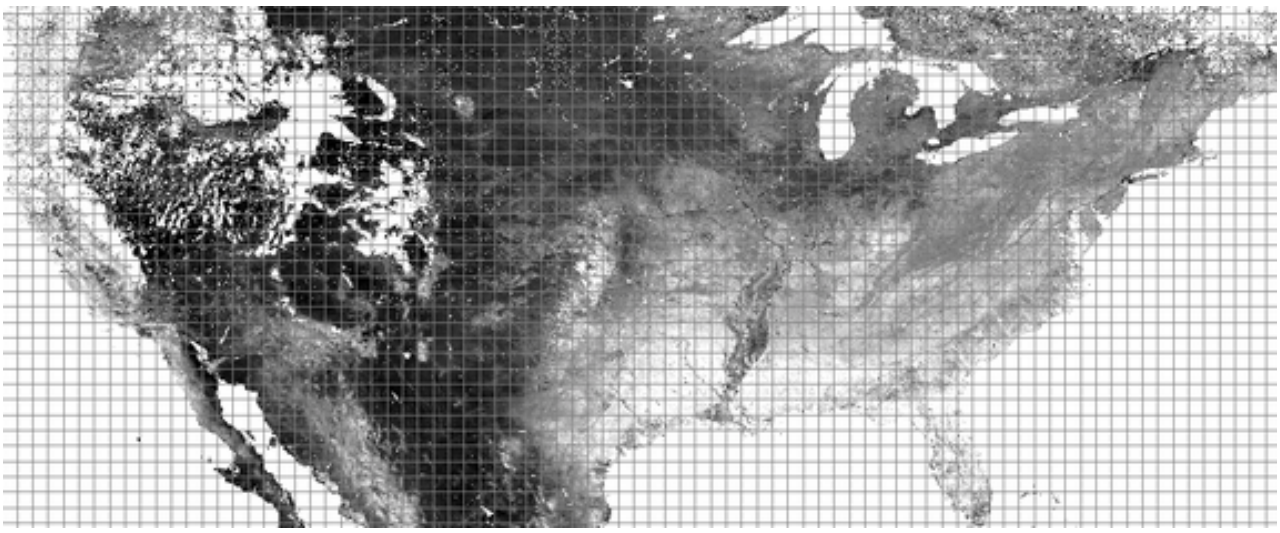


Figure 5.4: Spatial Load Separation on NDVI USA Image; The Image is Overlaid with a Grid, Representing Tiling

A user has the control over the number of tiles per row and the number of tiles per column. That is a (10 x 20) grid will result in 200 tiles. An alternative would have been to build the grid based on tile's width and height (measured in pixels). This will provide fine explicit control over the tile's dimensions but less control over their total. Whereas, the current approach provides explicit control over their total and implicit control over their size.

Data and Task Parallelism

The final step is to find a way to run the TimeSat module in parallel. The RDD Spark abstraction is exactly for this; to parallelize our execution. Therefore, an RDD is created holding the input parameter for the TimeSat executable, that is the path to a "settings file" which defines the area on which to produce results. To invoke the execution of the external TimeSat process, similarly to as it was running from the command line, the Scala.sys.process is used (Scala, 2018). The trick is to call the TimeSat executable inside the map function of the RDD. For each tile (or record in the RDD) the map function will execute a separate TimeSat instance.

To improve on the degree of parallelism, we had to tune our settings first since, by default, the RDD will have only 2 partitions. Having 2 partitions is way too low since only 2 tasks will be created and launched in parallel on just two nodes (Ryza, 2018). Moreover, the tile separation is now mostly invalidated, because each tile is grouped on either of the two partitions. Although the RDD records will still run in parallel on the available cores, the whole computation will complete when both tasks finish, resulting in an unbalanced cluster. To solve this, we increase the number of tasks by repartitioning the RDD. We set the number of partitions to be equal to the number of records (i.e. tiles). Now each tile will run in a separate task. This partition strategy will allow us to fully utilize all resources in the cluster. When a task is completed, the Spark Scheduler will launch the next not yet scheduled task (i.e. tile) on any available core in the cluster.

Spark is intelligent enough to fully utilize the CPUs since, by default, Spark allocates a single core per task (i.e TimeSat process). Furthermore, the task scheduler will launch execution on any available core in the cluster. That is, for example, Spark will run 8 instances of TimeSat on 2 nodes with 4 cores each. With that, we achieve both data and task parallelism.

An instance of the TimeSat executable will be run for each tile on its defined area for processing. Each instance will request and open the files in order to build the time-series, fit the fitting function and extract the seasonality for the temporal range of the time-series. Each instance will output the results in a seasonality (*.tpa) and an index (*.ndx) files which will be later read-in in Spark for further processing. That is, for each tile we have a separate set of results, which we later collect and load in an RDD.

5.4 Processing TimeSat Output and Results Construction

First, we need to find a way to load and extract the TimeSat produced SOS results into an RDD, perform necessary SOS adjustments and calculations, and then generate geo-aware SOS products (*GeoTIFF*), one for each year. The procedure consists of several steps, described in the following Subsection:

1. Read the TimeSat output into an RDD
2. Extract the seasonality information
3. Align Seasons
4. Calculate SOS per year
5. SOS GeoTIFF Products Construction

The DAG of the whole process is shown in Figure 5.5. If a stage is cached it will be grayed out and the cached RDD with a green dot. We are going to annotated each stage with numbers (1-6) and inside each stage number the operations (example: 1.1,1.2). The numbering will be used in the descriptions and discussions in the following paragraphs.

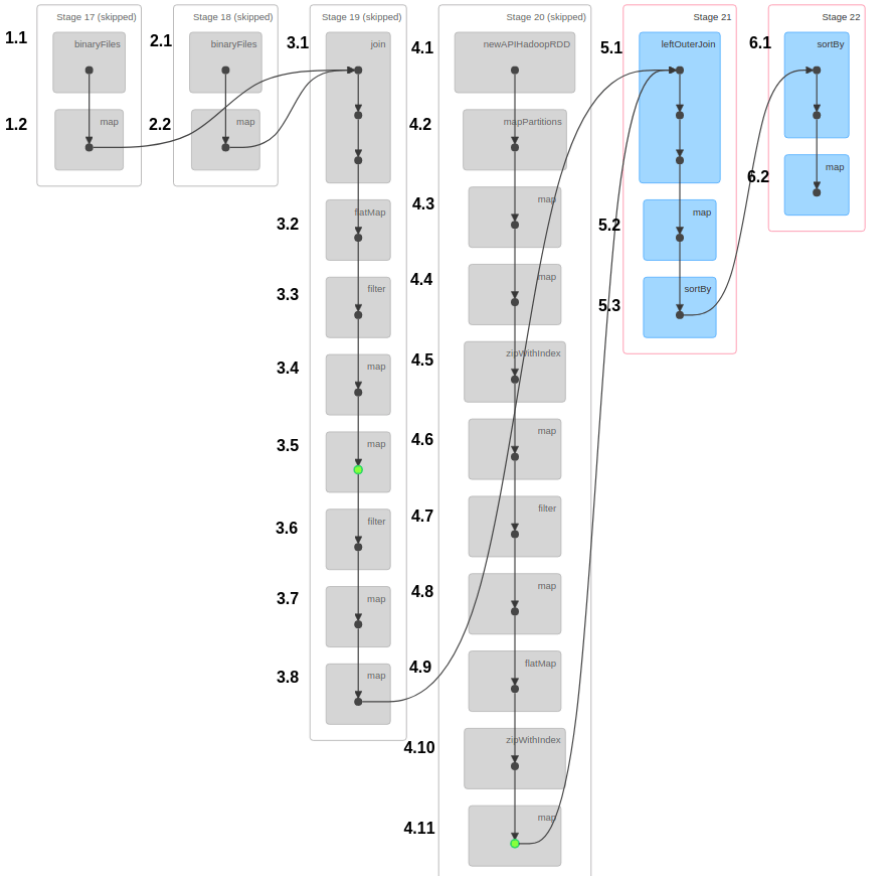


Figure 5.5: Spark DAG for the Results Construction: Spark Stages and Operations for Generating GeoTIFF Files

5.4.1 Read the TimeSat Binary Output into an RDD

First, we load the seasonality files (*.tpa) (1.1) and the index files (*.ndx) (2.1) with the Spark *binaryFiles* method. The method will serialize and stream the binary data that is on HDFS to the compute nodes through the Spark's *PortableDataStream* class.

Several different alternatives were tried before successfully and efficiently transferring the data to each worker. Initially, the reading was implemented with the Java *RandomAccessFile* method, which is performing random access I/O on the files. However, it accesses the underlying runtime system, and as such, tends to be performance expensive. Also we cannot read the whole file outside the map function, because the method has no serialization implemented and Spark doesn't know how to pass the data to each node.

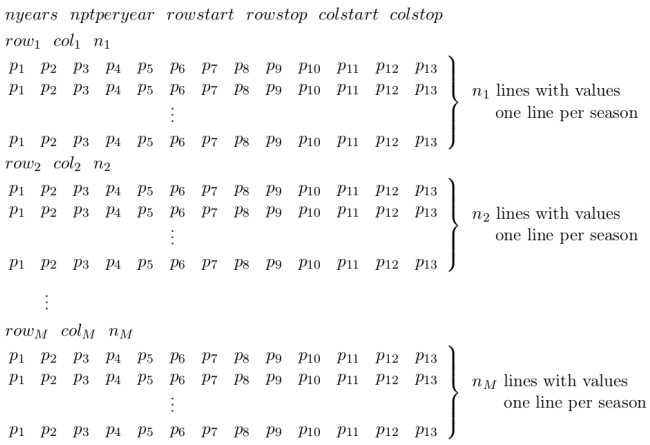
An alternative was to use *Files.readAllBytes* which will read all the content of the file in a byte array. Arrays and bytes in Scala are serializable, therefore if initialized outside the map function, the Spark will auto serialize the structure and ship to the workers. Since the method only can access files on a regular file system, the files could not be read from HDFS using this method. After several attempts, using the Spark's *binaryFiles* method solved all mentioned challenges and allowed for efficient file reading.

Next, we have to decode the binary data and transform it into a suitable format for further processing. Operations 1.2 and 2.2 (as shown in Figure 5.5) just map the file paths as provided from the *binaryFiles* method to file names on which we can perform a join operation and bring the seasonality and the index data into one RDD. That is, the seasonality data and the index data will now be aligned, having the same RDD record.

5.4.2 Extract Seasonality Information per Pixel

To parse and load the seasonality parameters produced by TimeSat into a Spark RDD, we first have to familiarize with the file's structure. The output is arranged in (.tpa) binary files with accompanying index files (.ndx). The structure of the seasonality file is shown in Figure 5.6a, whereas the byte order and the data types can be found in the TimeSat manual (Eklundha and Jönsson, 2017). For each pixel (row, col) the seasonality parameters (p1, p2, . . . , p13), as shown in Figure 5.6b, are given for each season found.

The index file (.ndx) is in the format (row, col, number seasons, starting location in the seasonality file). Keeping the number of seasons as entry per (row, col) pair implies that the number of seasons can vary per (row, col) pair, and therefore no structure and no constant record length can be assumed. Visually inspecting a small part of the output with "TSM_printseasons", however, shows that the season's count is kept constant and when the fitting function could not find a value for a corresponding season, 0s were inserted as placeholders. Nevertheless, to be safe, we assume that the seasons can be of varying length.



(a) Structure of the (*.tpa) file

```
Start-of-season time _____________1
End-of-season time _____________2
Length of season _____________3
Base value _____________4
Time of middle of season _____________5
Maximum value value of fitted data ____6
Amplitude _____________7
Left derivative _____________8
Right derivative _____________9
Large integral _____________10
Small integral _____________11
Start-of-season value _____________12
End-of-season value _____________13
```

(b) Seasonality Parameters Produced

Figure 5.6: Structure of the (.tpa) file as produced from TimeSat, storing the seasonality parameters in a binary form as well as the parameters produced (Eklundha und Jönsson, 2017)

Since we assumed a pixel pair can have a varying number of seasons, we used the index file to find out how many seasons there are and from what position they start in the binary file. The seasonality information is extracted from the (.tpa) files, as produced by each tile, and loaded into an RDD containing all the seasonality parameters (5.6b).

We used a *flatMap* operation (3.2, Figure 5.5) to map the single RDD record (i.e. .tpa file) to the numerous *[(row, col, season, p1, ... , p13)]* tuples. The filter in 3.3 handles the case that some files might have no results as they were over water. In such a case, TimeSat produces empty *.tpa files. In task 3.4 we map only the desired seasonality parameter. In our case, Start-of-Season (SOS) but a user can choose any of the rest. The resulting RDD is of the form *RDD[(row, col, season, SOS)]* where row $\in$ {1 : n rows}, col $\in$ {1 : n columns}, season $\in$ {1 : number seasons found}, SOS $\in$ {1 : file index range}.

5.4.3 Align Seasons

To correct an inconsistency in the seasonality output produced by TimeSat, we need to align the seasons. The issue is that it can be the case that the 1st season found is for the second year in the time series (Figure 5.7). This is because TimeSat is programmed to find the **n- 1 center most** seasons (Eklundha and Jönsson, 2017). That results in either getting the first year and skipping the last or skipping the first year and getting the last (Figure 5.7). The solution is to check if the first seasons starts from the second year and shift all seasons for the corresponding pixel pairs by one. To improve on the accuracy also check if the last season is either 0 or falls in the last year.

In operation 3.5 (Figure 5.5) we alight the seasons to be correct for each year. As our RDD contains a season per record, we can't check both for the first seasons and last season in one map function. The technique used is to run two separate map functions and then intersect on the results. The result was broadcasted to each of the nodes in order to distinguish which records' seasons must be incremented from the parent RDD. Broadcasting improved performance by delivering the data once to all the nodes instead of having to serialize it with each task. There are 8 million pairs for which the 1st season is for the 1st year and 1 million pixel pairs for which the 1st season is for the 2nd year. Aligning the seasons mean that the first year is partially incomplete and the last year is partially complete. As mentioned, this is because TimeSat finds only the n-1 center most seasons for the full range of n years. We cache this RDD since will be used multiple times when generating the GeoTIFF for each season.

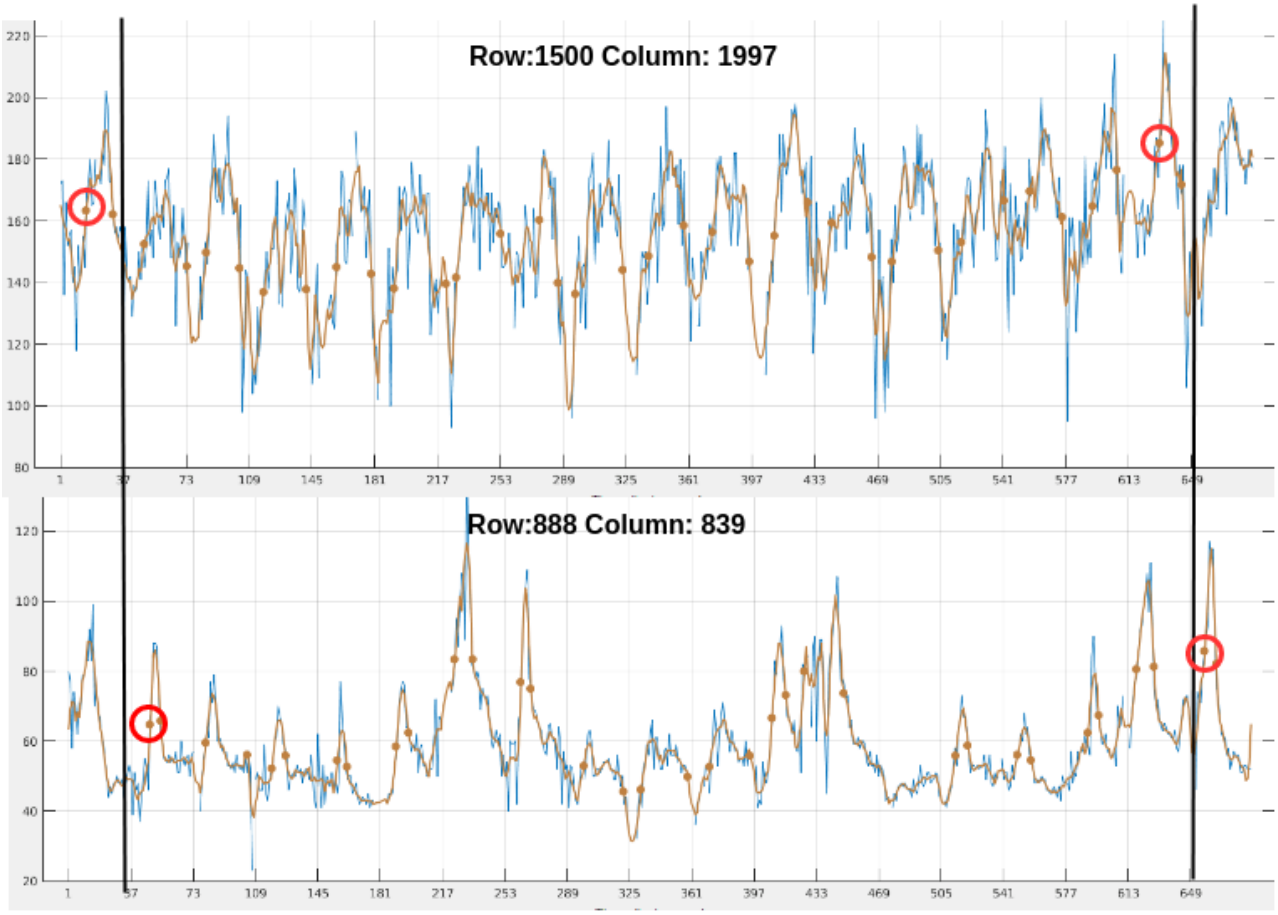


Figure 5.7: Seasons Nonalignment: TimeSat gets the n-1 center most seasons; the lines separate the two outer most years of either side from the rest of the series. The red circles mark the SOS detected. The picture show that in some cases the first season found is for the 1st year (top), whereas in other cases the first season found is for the 2nd year (bottom).

5.4.4 Calculate SOS per Year

The SOS values TimeSat produces are the file number index in the time-series sequence. When producing *GeoTIFF* per year, we need to transform that value to represent the day of the displayed year. The corresponding date for the SOS value is found, and the passed days since the beginning of the corresponding year are calculated.

The process of mapping the seasons to years have one major limitation; distinguish between early and late springs. As a result, the seasons may not match exactly to corresponding years. Figure 5.8 shows such a situation where the SOS detected for year 2 is too early and falls in the range of year 1. To solve this, we assume there is no possible SOS value later than the 270th day of any year. If values greater than 270 are detected, 365 is subtracted from them, indicating early springs (SOS that started in the previous year). In the example in Figure 5.8 the SOS for year 2 (seasons 2) is detected to be the 338th day of the previous year. By subtracting 365 the result is -27 days, indicating the SOS started 27 days before the begging of the year. Therefore, similarly to other SOS products produces for the studied region (Zurita-Milla et al., 2017), we define an early spring to be any value between 270 and 365 of the previous year, which is accordingly transformed to a negative number in the range [-95,0]

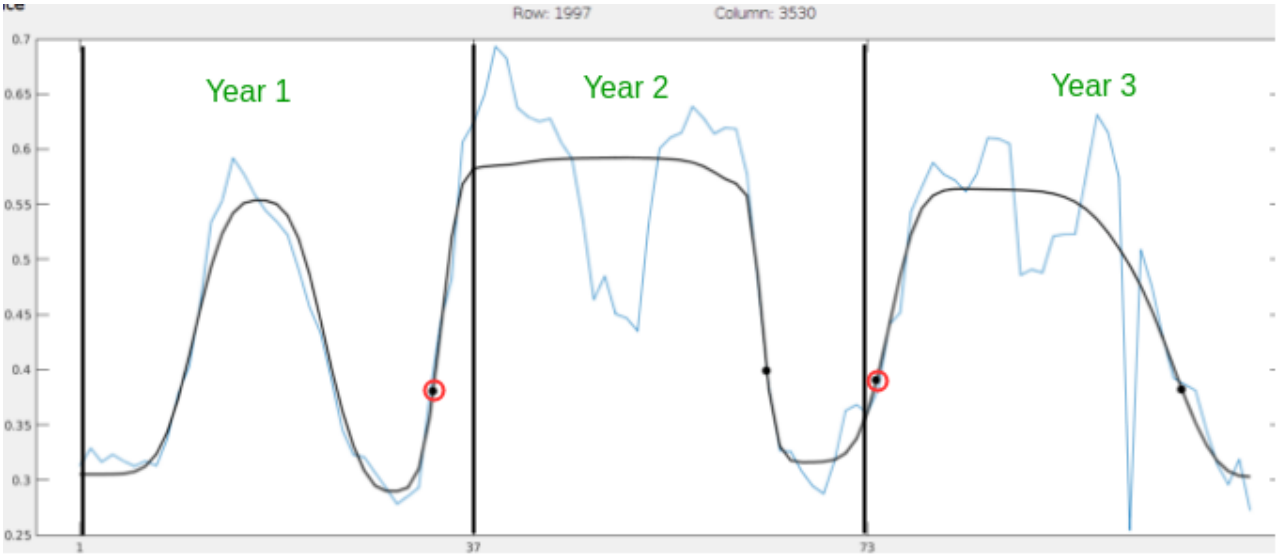


Figure 5.8: Example of SOS detected for the previous year (first red circle); vertical lines represent year separation, the red circles are the detected SOS values, the blue time-series is the NDVI, the black time-series is the fitted Asymm. Gaussian function

Our calculations are based on the assumption that in the current geographical regions there will be no SOS values greater than 270 days (1st of October) for any of the years, indicating late spring. This implies there will be no early springs before the 270th day of any year. Our results show the values are well within the range, however the problem distinguishing late and early springs is still challenging since theoretically it is possible to misclassify a late spring as early spring.

5.4.5 SOS GeoTIFF Product Construction

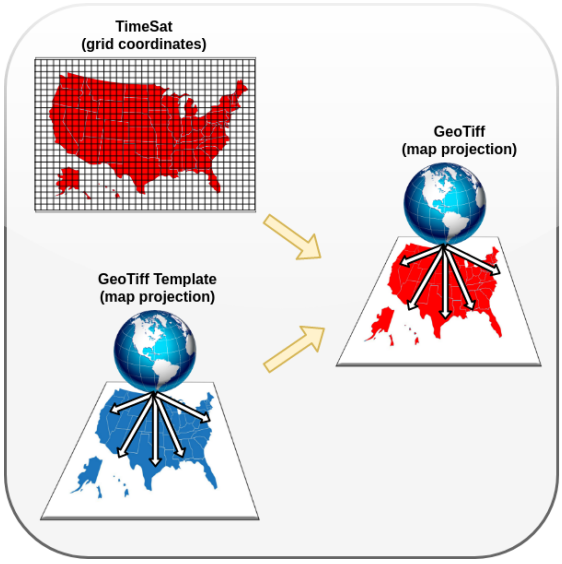
To produce SOS images with georeference information encoded we have to know the map projection of the images. That is the information how the (row,col) coordinates in our grid/plain, map to latitude/longitude coordinates on the globe. However, we don't have any geo-information (location on the map, pixel size, projection, etc.) provided by TimeSat, since the software is image oriented and does not consider geographical coordinate systems other than columns and rows. What we can do is to transform one of the input spectral bands files to a *GeoTIFF* and load it in *GeoTrellis*. Since the file has exactly the same spatial resolution and projection, we can use it as a template to get all the required geo-information. We then combine the seasonality values as produces from TimeSat with the geo-information from the template to produce our SOS products with geo-referencing information embedded (see Figure 5.9a).

Operation 4.1 (Figure 5.5) loads the template's tile into *GeoTrellis*, 4.3 extracts the values as *RDD[Tile]*, 4.4 transforms them as *RDD[ArrayDouble]*. From operations 4.5 to 4.11, we index the template's values, results in an RDD of type *RDD[Array[Index, Value]]*. The index is required because the TimeSat seasonality output is sparse and only produces seasonality information for a subset of the pixels. That is because a lot of NDVI/EVI data could be missing for (row,col) pixel pairs due to out of range values, clouds, water pixels, etc. However, to construct our image, we need the full range of values. Our approach is to use the index and fill in the TimeSat values (if present).

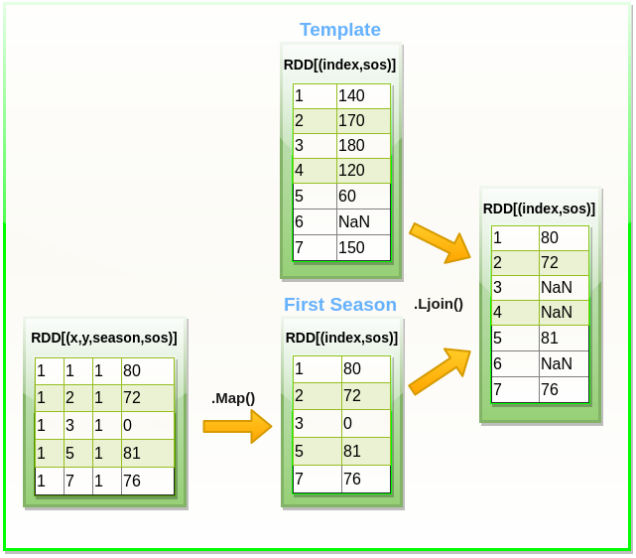
Still, our data is not in a matching format. From the TimeSat output we have a grid of (row, col) values and from the template, the data is indexed in an array. To translate grid coordinates to array index the formula *number cols.* * *row* + *col* is used (3.8).

Now the data from the template and the TimeSat output is both indexed, we can perform a left outer join on the template (5.1) which will keep the full index range. If SOS value is found and it is not 0, then put it as a value, otherwise put "NaN" for the corresponding index (5.2). A simple scenario for just one season is displayed in Figure 5.9b.

Finally, since the join in 5.1 (Figure 5.5) shuffled our data, we need to re-order based on the index to correctly reconstruct the *GeoTIFF* with the new SOS values (6.1). Then, only the values are collected into a *DoubleArrayTile* (6.2). With the extent and projections from the template, we write the new SOS image to the Minio Storage. For each season, a GeoTIFF imaginary is produced.



(a) Figure shows how we apply lat/long map projection to our results in order to generate geo-aware SOS products



(b) Figure shows how we re-use the index and NaN values from the template (if SOS output is not produced for a pixel) to construct the SOS products

Figure 5.9: GeoTIFF Product Construction

5.5 Building the Evaluation Models

In this Section, we will briefly describe the technical aspects of our SOS analysis in Chapter 7. For justification of the experiments and the results, please refer to Chapter 7. The main evaluation techniques used in the analysis are: calculating the mean and standard deviation for all years, calculating the difference per experiment, and comparing the models with the ground observations per species.

Mean and Standard Deviation

For calculating the mean and standard deviation (sd) for each pixel across all the years, the results were put into HDFS for processing and further loaded into an RDD. Each pixel is labeled with an index, indicating its position in the image. All pixels from all years are flattened in an RDD with type *RDD[(index, SOS day)]* and further grouped on their indexes. It is important to filter out NaN values first (water, missing SOS, etc.). Now that SOS values from all years are grouped on their position in the image (row, col) coordinates, we can easily calculate their statistical metrics. The Spark class *StatCounter* was used to calculate the mean and the standard deviation of the values. A *GeoTIFF* image is constructed containing the results. For improved visualization, the *GeoTIFFs* were translated to color gradient PNG images with a python script using *GDAL*[3] to open and load the files into *numpy* array, used by *pyplot*[4] to plot and visualize the results.

Ground Measurement Comparison

For comparing with the ground observations, the *.csv* file containing the ground measurements was loaded into a Spark *DataFrame* for easy filtering and data management. The columns and the data itself was filtered in order to reduce the input size, but also gather only the entries of interest. For further processing, the backing RDD of the *DataFrame* was used. The position of the observations were encoded in 6 digits of precision latitude, longitude coordinates, which on a qualitative scale that can be identified as individual humans (Wikipedia, 2018). To translate from map coordinates system (latitude, longitude) to grid coordinates (row, column) we used the geoTrellis' function *mapToGrid*.

To translate to index positions the formula *cols * row + col* is applied. The studied species was filtered and the resulting RDD is of type *RDD[(year, index), SOS observation]*. The TimeSat SOS products were loaded and transformed to an RDD of the same type *RDD[((year, index), SOS prediction)]*. Now that the two RDDs have the same key we can join them together by bringing the SOS observations and predictions together. Since there can be multiple observations per plant and multiple plants per tile the earliest SOS observation is taken per (year, index) pair and the resulting RDD is of type *RDD[((year, index), SOS observation, SOS prediction)]*.

The SOS observations and predictions are fitted into a *MLlib* regression model and the R^2 metric derived to determine how much of the variance is explained by the model. The Mean Absolute Error (MAE) was calculated with map and reduce RDD operations and the formula accordingly applied.

Calculating the Difference of the SOS products

The difference between one of the generated SOS products with the rest was calculated with the QGIS software since it already provided the required functionality using the "raster calculator" tool (QGIS, 2018a). Averaging the SOS on ecological regions was done with polygon layer containing the regions using the QGIS "zonal statistics" plug-in (QGIS, 2018b).

[3]http://www.gdal.org/

[4]https://matplotlib.org/api/pyplot_api.html

Chapter 6

Platform Evaluation

The platform evaluation was performed on two main criteria: efficiency and scalability. We did a performance profile to understand better how the systems works and identify its pitfalls. We wanted to understand what the optimal configuration is, how to partition our data, how many cores and nodes to use. The results were used to further configure and tune our jobs to run faster, more stable, and more cost-efficient.

To find out if the platform is efficient and scalable we studied how to exploit the available resources to get optimal performance metrics for the three stages of our phenological analysis: TimeSat seasonality extraction, NDVI and EVI calculation and GeoTIFF SOS product generation. We analyzed the best Spark and Big Data practices (Section 6.1) in order to write efficient jobs. Furthermore, several experiments were performed to identify the optimal partitioning scheme when TimeSat processing, and also to assess the scalability with different cluster configurations and data resolutions.

Section 6.1 describes identified techniques and heuristics for optimizing Spark jobs. Section 6.2 discusses the VI generation process and its performance. In Section 6.3 the platform's partitioning and load-balance is studied. Platform and scaling solutions are discussed in Section 6.4.

6.1 General Efficiency and Performance Optimizations

We used several techniques to optimize our Spark jobs (Azure, 2018; jaceklaskowski, 2016b; Spark, 2018; Dahms, 2018). Please refer to the list below:

- **Optimize Data Handling**: An important concept in big-data processing is to reduce the amount of data as much as possible as early as possible. A primary goal was to always work with the minimal amount of data. Since the provided input data is on global composites, covering the entire world, a preprocessing step was to crop out only the area of interest (USA) in order to reduce the amount of data or minimize unnecessary computations.

  Another heuristic technique used was to apply RDD filters first to reduce on only the necessary data. In practice, that means to filter out the SOS data from the rest of seasonality output produces by TimeSat. In addition, when generating the GeoTIFF images per year, the data required for that year is filtered out as early as possible.

- **Shuffle Reduction**: As described in Section 4.2 shuffles are fairly expensive operations since they involve data movements. In the cluster, by default, the data nodes of HDFS co-exist with the worker nodes of Spark, which is a good configuration if data locality exists.

  The VIs generation procedure, as described in Section 6.2 demonstrates, how we took advantage of HDFS and data locality. However, if shuffle is required, the primary goal when choosing an arrangement of operators is to reduce the number of shuffles and the amount of data shuffled. Therefore, we tried to refrain from operations which require shuffle, unless necessary. Also, if the data is small enough to fit in the nodes' memory, we took advantage of broadcast variables and performed a map side operation.

- **Level of Parallelism and Partitioning**: The resources of the platform will not be fully utilized unless the right level of parallelism is chosen. That is, inadequate partitioning of data may result in an unbalanced cluster, and inefficient jobs.

  In our Spark jobs, the default number of partitions returned by transformations like map, join, reduceByKey was set to 1000. Further adjusting the size and number of partitions was not required since the value is greater than the number of cores and distribution was good. The task durations for the tasks usually range between milliseconds and seconds and no cause for underutilization was found.

  For the TimeSat processing (i.e. SOS extraction), no heuristics are present and therefore experimental approach was taken to find the optimal data partitioning and load distribution. As described in Section 5.3 we divided our TimeSat processing into a number of tiles to distribute the work in the cluster and furthermore tune the degree of data parallelism by setting the number of RDD partitions to be equal to the number of tiles (as seen in Section 5.3). To evaluate on our task and data partitioning several tiling experiments were performed in Section 6.3.

- **Cache Reusable RDDs**: RDDs can be cached in memory or on disk in order to be re-used in subsequent operations. Caching is used when generating the SOS GeoTIFF images. The RDD right before the filter selecting the data for a particular year is cached as well as the template's data. Now, we don't have to recompute the whole chain of RDDs from the beginning but use/ re-use the intermediary results.

  Experiments were run on a 4 nodes with 4 cores each, 1 worker and 1 executer per node with and without caching. The results show caching speeds up the image generation by 14 minutes. One should remember that it is up to the user to un-persist the cached RDDs in order to free the memory for other operations (if required).

- **Optimize Transformations** Another rule-of-thumb was to use "rich" transformations, i.e. do as much as possible in the context of a single transformation. That often means to combine several map functions into one and let the Spark's DAG Scheduler decide on the execution plan. Also, it is important to pipe narrow transformations together (if possible) and then apply the wide transformation, instead of interleaving between the two. This would reduce the shuffled data since wide transformations may or may not depend on a shuffle. This technique was used throughout all our jobs.

6.2 NDVI and EVI Calculation Evaluation

To evaluate our VI index generation, three runs were performed on the EVI generation. The experiment was run on 4 nodes with 4 cores, one worker and executer per node. The best time of three runs is 6.3 minutes. There are 684 records arranged into 1000 partitions. As shown in Section 5.2 each record holds the blue, red, near-infrared and the quality data. The experiments show the default partitioner will assign 4, 8, 12, 16 records per task, whereas few tasks will be empty.

All tasks have a local data locality, which is the most optimized data locally option. That is, the data is local to the required computation, and therefore data movement will be none or minimum. Spark will schedule tasks on the same node where the required HDFS blocks are stored. With HDFS the Spark driver contacts the *NameNode* about the *DataNodes* containing the required blocks, and then schedules the work to the Spark Worker hosted on the same node (if possible) (i.e data locality) (jaceklaskowski, 2016a) (Docs, 2018).

By taking advantage of the Spark's locality-aware scheduling and since our data is co-partitioned, we can minimize data movement over the network and minimize network bandwidth bottlenecks. This greatly improves on the shuffle based (i.e. wide) transformations as described in Section 4.2. Therefore when reading the files from HDFS, the shuffled write is only 16.1 MB and when performing the VI calculations, the shuffled read is only 64.6 MB.

6.3 Tiling- Partitioning Strategies

To assess what is the optimum number of tiles and tune the load distribution in the cluster, we ran a set of experiments where the number of tiles were roughly doubled on each experiment. We choose to keep the dimensionality ratio between height and width of a tile for the cost of not precisely doubling the number of tiles for each subsequent experiment. The test experiments were set on 30 (8x8), 121 (11x11), 256 (16x16), 529 (23x23), 1024 (32x32), 2025 (45x45) tiles.

The experiments were run on 19 years of NDVI data with the TimeSat Asymmetric Gaussian function. The files were cached on each of the nodes. The test setup was 4 nodes with 4 cores each having 1 worker and 1 executor per node. Each node has 4 cores resulting in a total of 16 cores in the cluster. The node's memory was set in 3:1 ration; 12 GB for Spark jobs and 4 GB left for TimeSat processing. That is, there are 684 files with a dimensionality of 6500 x 3000 pixels. Each experiment was run three times to minimize OS or network interference. (Coffin and Saltzman, 2000).

First, we wanted to understand how well the VM resources are utilized during TimeSat processing. To check that, the monitoring tool "nmon" [1] was installed on each of the nodes. With it, the CPU utilization, the network I/O and the disk and memory usage was observed. The results are presented in Section 6.3.1. Once we understood the process, we proceeded evaluating our partitioning strategy. A main evaluation strategy was to observe the total computation time and observe the cluster's load-balance but also to study the individual tile's duration and the tiles' distribution.

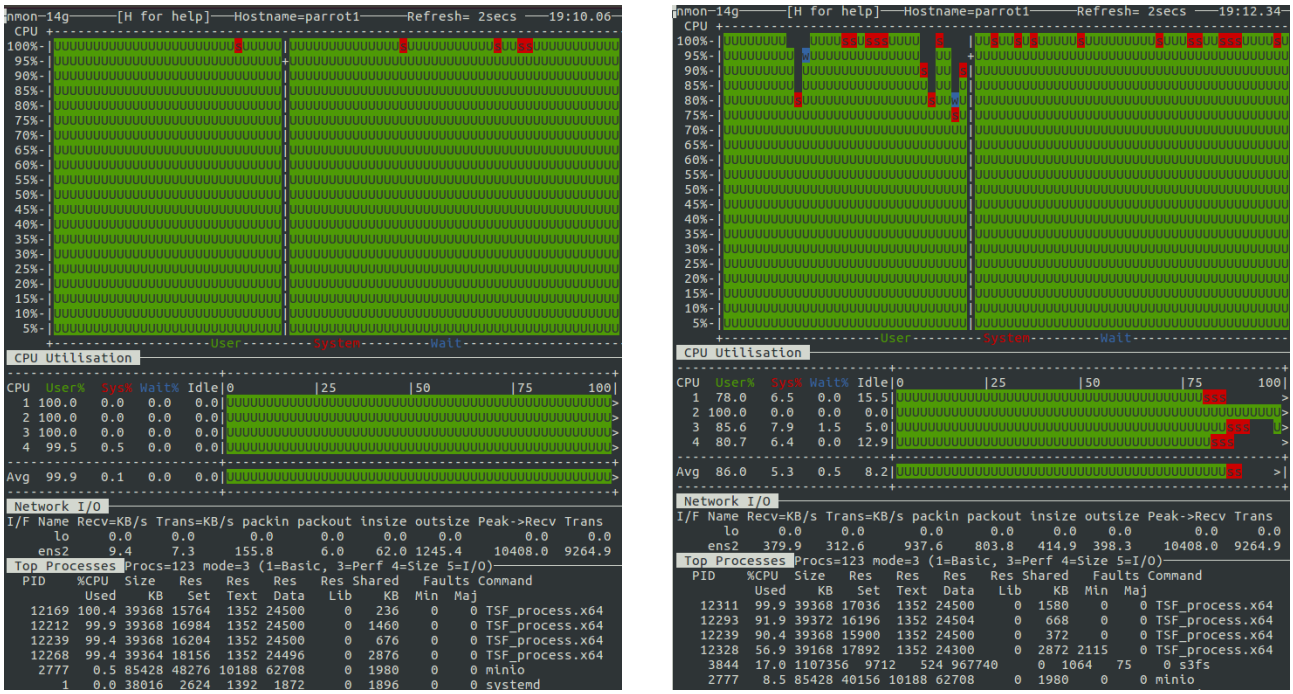
[1] http://nmon.sourceforge.net/pmwiki.php

6.3.1 VM Monitoring

During computation, the node's 4 cores are fully utilized by the TimeSat executables almost the whole time (Figure 6.1a). On TimeSat completion, however, few CPU cycles will be taken from the *s3fs* and Minio to copy the files over, but also to read the files for the new tile. The network I/O will briefly spike. During such transition times the CPUs will be slightly underutilized because it will wait for Minio and s3fs to complete its I/O. Also Minio and *s3fs* will 'steal' few CPU cycles. Figure 6.1 display such a situation. The graph is refreshed every 2 seconds; the green color stands for the TimeSat executable and the red for the s3fs/minio system processes. Each TimeSat instance uses 40 MB virtual and 16 MB of physical memory, which is 0.1 % of the available memory.

If the files are not physically present on each of the nodes (e.g. run for the first time), they will be downloaded from Minio through the *s3fs* fuse. In case of such input/output task, the CPUs will enter a 'waiting' state and will remain in it until the download is complete. Caching the files improves the performance, as they are not requested from Minio on subsequent runs.

To measure the network performance in the cluster, the software package *iperf* [2] was installed. On one of the nodes, the software was run in a server mode (receiving packages), and each of the other nodes was acting as clients, sending packages. The min/max performance averaged out of 5 pings was 8.34 Gbits/sec and 9.20 Gbits/sec. This shows the cluster infrastructure is on a fast network.



(a) VM Processing Load

(b) VM Load on Tile Completion

Figure 6.1: Performance monitoring for a single node. The figures display the CPU utilization (both ephemeral and long-term), the Network I/O, and the top processes running on a node with 4 cores

[2] https://www.gbmb.org/gbps-to-mbs

6.3.2 30 Tiles

The first test was performed on a 6x5 grid with 30 tiles. All tasks had finished for 14 hours. However, this tiling had a terrible load distribution. Two of the nodes had finished after 4 hours, one node after 8 and one long-running task was running for 14 hours.

Two issues were identified:

1. **Varying tile duration**: Due to load in-balance a long task was holding on the job's completion. Tiles have varying duration because pixels values can be missing due to clouds, snow, oversaturated bands or water regions. **We defined the worst case performance scenario to be the case when the longest tile is run right before the completion of the first CPU in the cluster**. This is the case when the whole cluster will be underutilized for the longest time. The identified solution was to reduce the load per tile by reducing its dimensions (i.e. increasing the tiles number, would reduce the time spend per tile). This is to be confirmed in the subsequent experiments.

2. **More than one tile assigned per task**: Repartitioning on the number of tiles as described in Section 5.3 does not assure we will get a uniform distribution across partition_ids. The default partitioner will try to assign the RDD records uniformly to partition_ids, but we observed that in some tiling scenarios there are empty tasks and others have 2 RDD records assigned. The fine-tuning the work per task to each executor is possible through adjusting the partition scheme.

   A second experiment with the *HashPartitioner* was performed, however, even worst grouping was observed since it assigns partition_ids based on the key's hashCode modulo the number of partitions [3]. Therefore, it is very possible 2+ keys (RDD records) are assigned to the same partition_id (i.e. task). Since tasks are the most fine-grained parallelization entity, grouping tiles on tasks is counter-productive in our effort to achieve better results.

   Finally, a custom partitioner was implemented that assures data is in uniform distribution: 1 block_id per partition. We repartitioned based on the number of elements and for each partition one RDD record was assigned. Now each task will contain a single RDD record (i.e. tile). This partitioning scheme produced best results since a task is assigned at most one tile and improved on the uniform work scheduling and load distribution.

6.3.3 121 Tiles

The 121 tiles' experiment had a best time of 6h and 43 minutes. The tasks durations are displayed on an event timeline shown in Figure 6.2. Each horizontal line separates neighboring workers. Each green rectangle represents a separate task (i.e tile) and its length indicates the duration, whereas a horizontal alignment of tasks represents they were run in the same worker thread. The graph has been overlaid with three vertical lines: the blue line displays when the first CPU had finished (and there are no more tasks to schedule), the orange line displays when the first executor had finished, and the red line indicates the completion of all tasks. Since the executor have multiple parallel threads per node, and there are greater than the number of cores, the white gaps are not representative for underutilization. However, if

[3] https://github.com/apache/spark/blob/branch-2.2/core/src/main/scala/org/apache/spark/Partitioner.scala

there are less then 4 tiles per node vertically (the space between the blue and the red line), the cluster was underutilized. Optimal performance will be achieved when the three lines are as close as possible to each other. The difference between the blue and red line is 1h and 17 minutes. During this time the cluster was not fully utilized. To reduce on that time, further reduction on the tile's size was performed.

Figure 6.2: 121 Tiles; Event Timeline for Tasks

6.3.4 256 Tiles, 529 Tiles, 1024 Tiles

During the 256 tiles, 529 tiles, 1024 tiles experiments further positive reduction was observed. The best completion times are 6h 50 min., 6h 22 min, and 6h 20 min respectively.

Figures 6.3 and Figure 6.4 show the ordered maximum task duration with the corresponding number of tiles for the 256 and 1024 experiments respectively. In both tests, we see tiles cluster among the longest or shortest tiles with linear growth between the shortest and longest tasks. Both show there are no outliers of much longer tasks. The duration of the tasks has also substantially decreased.

Figure 6.5 shows that the time between the first and the last CPU finishing have reduced (8 minutes). The maximum tile's duration is 14 minutes. Thus, as described in Section 6.3.2 the worst case performance penalty is 14 minutes. In this particular experiment, the cluster in-balance time was 8 minutes.

.

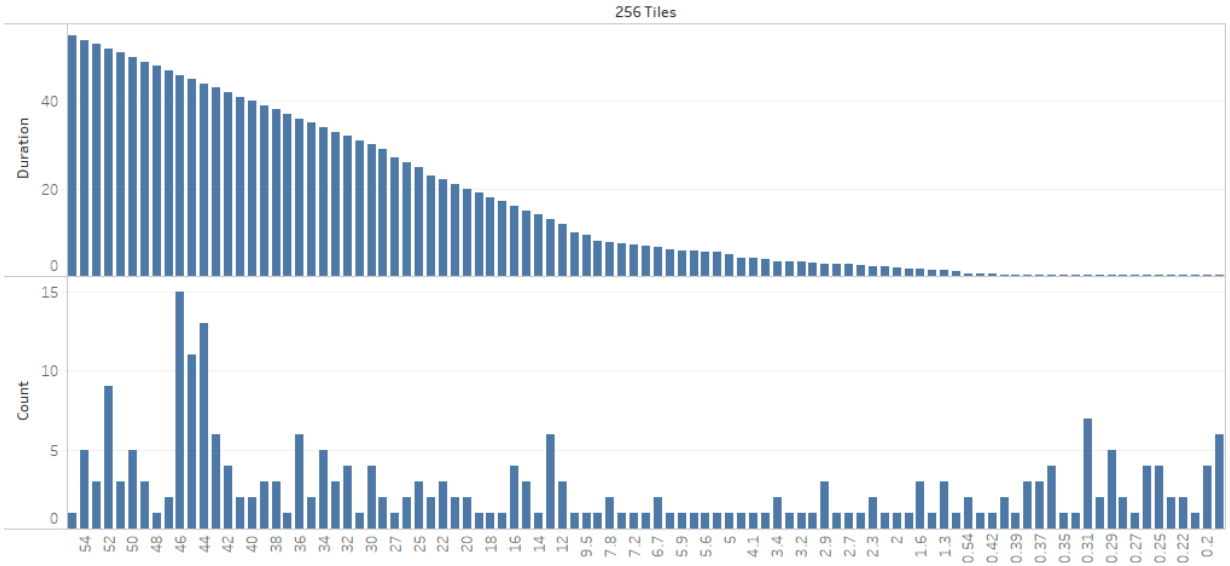


Figure 6.3: 256 Tiles; Ordered Maximum Task Durations (Top) and Number of Tiles for the Corresponding Duration (bottom)

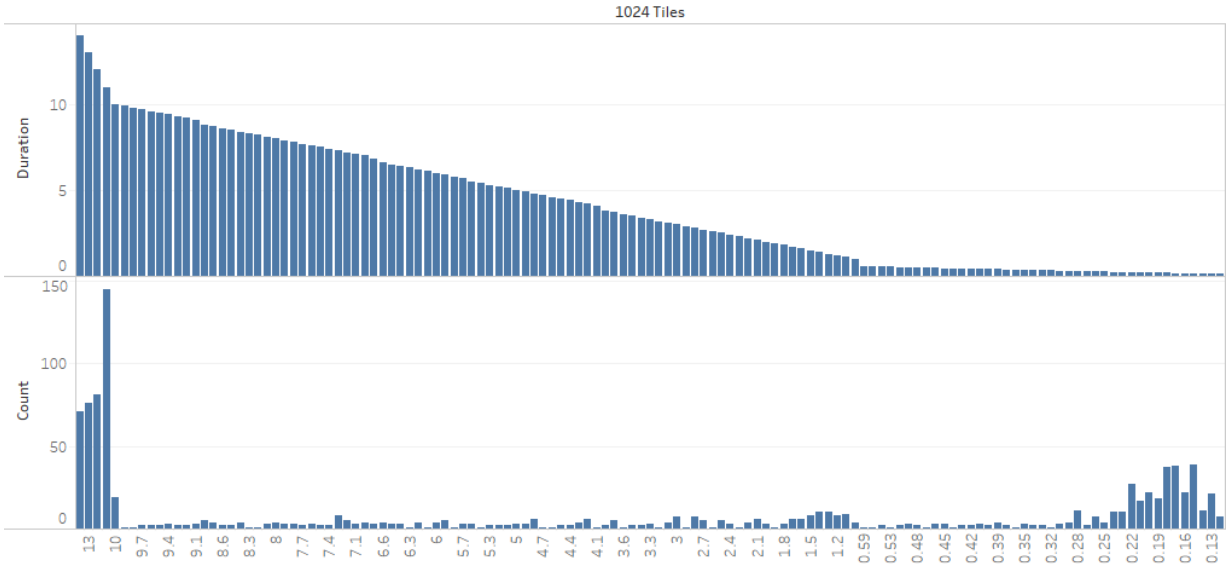


Figure 6.4: 1024 Tiles; Ordered Maximum Task Durations (Top) and Number of Tiles for the Corresponding Duration (bottom)

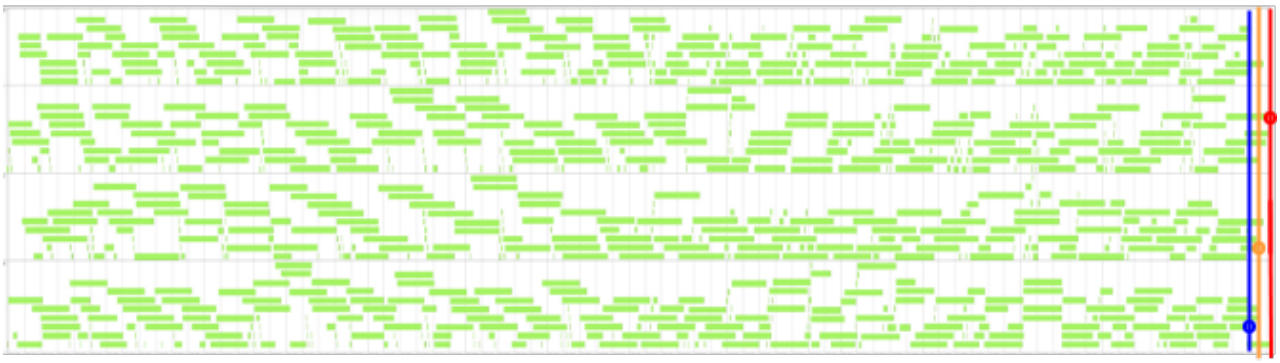


Figure 6.5: 1024 Tiles: Event Task Timeline

6.3.5 2025 Tiles

To bridge the gap between the finishing times of the first and last tasks, we increased the tile count drastically (45x45= 2025 tiles), reducing the tile size, effectively shortening the execution times for each task. The best time for the 2025 tiles' experiments is 6h 46 m. Interestingly, this run is slower than both runs with 1024 tiles. This shows an increase of 26 minutes when compared to the previous best time. The time shows there is a 1,5 seconds penalty for each tile.

We noticed that a tile over a water region which contains only NaN values can take minimum 7 seconds. Since, no (little) computation is involved, that is the time that TimeSat takes to read-in all the input files for a tile, traverse the pixels and output the results. Since the executable does not have memory sharing capabilities, this process is repeated for each tile.

We can infer that the increased tile size has some cost on time, in terms of opening/reading-in the input files for each tile. Investigation shows that the local disk is capable of serving the files as it spikes to around 10 % in performance if the files are requested simultaneously. The reason for this underperformance is the combined effect of TimeSat opening the files each time and the s3fs resolving and serving the requests from the local disk cache. The cumulative effect from all tiles creates a trade-off between smaller tiles vs. more tiles.

6.3.6 Tiling Summary

Table 6.1 shows the finishing times of the trials and in Figure 6.6 the best times are displayed. The best performance is observed in the 1024 trial. Increasing the tile count does improve the load balance, but at at the cost of increasing the time for initial file read (as each tile reads the files separably). This is a trade-off relationship at which at some point this cost will overtake the benefit and will become counter-efficient. For this particular data the turning point is somewhere between 1024 and 2025 tiles. Since the maximum observed tile duration is 14 minutes this is the worst case cluster imbalance time.

Table 6.1: Durations for Different Tiling Experiment

Tiling Experiments	Tiling	Run 1	Run 2	Run 3	standard deviation (minutes)
64	8x8	7h 30 m	7h 29 m	7h 31 m	1
121	11x11	6h 52 m	7h 02 m	6h 43 m	10
256	16x16	6h 50 m	6h 50 m	6h 51 m	0.5
529	23x23	6h 27 m	6h 22 m	6h 23 m	3
1024	32x32	6h 22 m	6h 20 m	6h 22 m	1
2025	45x45	6h 46 m	6h 52 m	6h 47 m	3.2

To better understand and compare the tile's durations across experiments, in Figure 6.7 each tile's duration is plotted for each experiment. The x-axis shows the frequency of occurrence, and the y-axis represents the duration in minutes. For the 121 trial and the 529 trial, the box plot is relatively short suggesting a high level of agreement in the durations of times, whereas a higher spread can be observed in the 256, 1025, 2025 trials. Besides, the times are similar for smaller ranges, whereas for higher ranges are quite varied judging by the length of the whiskers (lower and upper extremes). Understandably, the tile's durations are decreasing with each subsequent test.

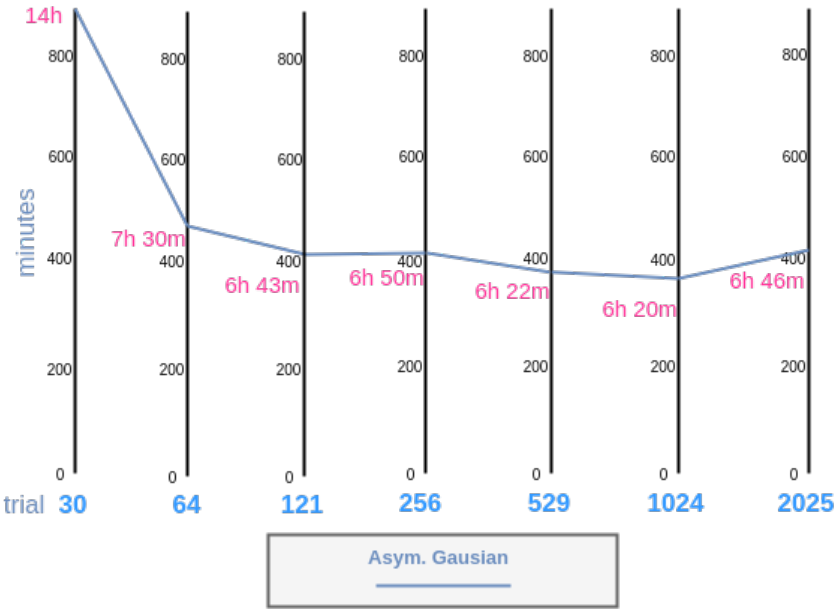


Figure 6.6: Parallel Coordinates Graph of the Trial's Times (y-axis times displayed in minutes)

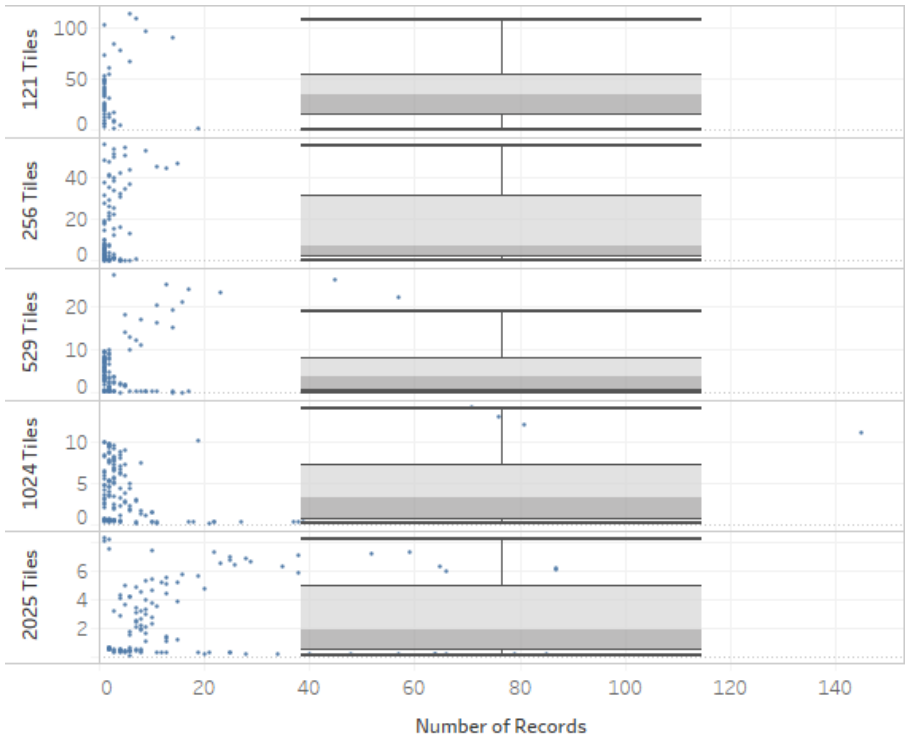


Figure 6.7: Parallel Coordinates Graph of the Trial's Times. Each Entry(Dot) Represents a Tile

6.4 Scaling

Our Spark platform is designed in such a way for easy scaling. With Ansible, as described in Section 4.4, a user can flexibly provision and manage new sets of VMs once deployed with the SurfSara interface. There are four common bottlenecks in Big Data processing platforms: the performance can be CPU, memory, disk or network bounded.

Our primary interest is to access the TimeSat SOS computation component of our platform. As described in Section 3.1.2 the TimeSat computation is CPU bound. As a general rule, if there are performance problems due to long-running jobs caused by longer time-series or increased resolution, one should aim for more processing power. Therefore, to asses the performance, but also to further identify potential bottlenecks and effects of increased cluster power or data load, we perform several experiments. In Section 6.4.1 we discuss different scaling configurations. In Section 6.4.2 we study data scaling and perform tests on three datasets with different spatial resolutions.

6.4.1 Platform Scaling

Once the need for scaling has been identified, the specific scaling approach should be chosen. There are two commonly used types of scaling: scale out and scale up. To scale horizontally (or scale out) means to add more nodes to the cluster. To scale vertically (or scale up) means to add resources to a single node. To investigate what are the effects of scaling out/up the CPU power several tests were performed.

For the scale-up experiments we used 4 nodes with 4, 8, and 12 cores. For the scale-out experiments we used 4, 8 and 12 nodes with 4 cores each. Therefore the total CPU power would add to 16, 32 and 48 cores. Results are shown in 6.2. The SD for the 3 runs was calculated as well an improvement factor related to the best time in the 4 nodes 4 cores experiment. The results show no significant difference between a scaling-up or scaling-out approach, although the scaling-up approach is showing slightly better results. The differences in our tests can be a result of a cluster underutilization, or even the VMs physical location can have a role. The improvement in time in relation to the computational power is almost linear, however we observer a performance cost. We identified Minio and the *s3fs* fuse to be a cause for this.

Table 6.2: Platform Scaling Experiments

Scaling Experiment	Technique	Times Increase	Run 1	Run 2	Run 3	SD (minutes)	Factor
4 nodes; 4 cores	default	1x	7h 11m	7h 08m	7h 10m	1.53	1x
4 nodes; 8 cores	scale-up	2x	3h 40m	3h 40m	3h 42m	1.15	1.95x
8 nodes; 4 cores	scale-out	2x	3h 47m	3h 43m	3h 44m	2.1	1,92x
4 nodes; 12 cores	scale-up	3x	2h 37m	2h 36m	2h 33m	2.1	2.80x
12 nodes; 4 cores	scale-out	3x	2h 36m	2h 39m	2h 39m	1.72	2.74x

The way the *s3fs* fuse maintains the cache is by calculating and comparing MD5 checksums (ETag HTTP header) against the checksums of the files stored in Minio (gitHub, 2018b). The calculated MD5 checksums are stored in a 'stat' cache, which is big enough not to cause any trouble. The underperformance comes from the HTTP requests send to Minio to give the new Etag header to asses the 'freshness' of the cache. Also, in the *s3fs* codebase there is a "mutex" lock, which prevents for race conditions, but contributes for the underperformance (gitHub, 2018a). To confirm this, the scale-up experiments were re-run factoring out Minio and the *s3fs*. Instead, the files were copied over with a bash script on each of the nodes and accessed directly from the disk. The running times for the 16, 32 and 48 cores experiments are 6h 29m, 3h 25m (1.9x). and 2h 17m.(2.85x). The running times show not only faster times, but also slightly better scalability. Therefore our biggest performance loss comes from accessing the files through the *s3fs* on Minio.

During scaling-up more cores per node will put an increased load on the disk. Although, this is a minor detail since reading the files is per tile and can rarely happen in the same time, if one is to stack many CPUs on one disk, the disk can become a bottleneck. Our experiments show that with 8 cores per node the disk load can briefly spikes up to around 30 % when reading from the disk (and several disk I/O requests overlap). As a general rule, one can scale-up the CPU power to around 20-25 cores per node and then perform scaling-out.

6.4.2 Data Scaling

The quantity of the data can be increased in 3 ways: increasing the length of the time series, increasing the scale, or increasing the resolution. All will increase the number of pixels to process. To study the effects of that we downloaded and preprocessed three sets from the MODIS sensor with three different resolutions of NDVI data: 1 km., 500 m., 250 m.[4]. To minimize noise and differences, the resolutions all come from the same sensor and cover the same area of USA of length 1,100 x 1,100 kilometers of central USA.

The experiment was run with 23x23 tiling for 3 years with 32 cores. Results in Table 6.3 show that with the increase of resolution logically the execution time increases. An improvement factor was calculated in relation to the previous best time.

Results show the times of the 500 m. and 250 m. show (roughly) a linear increase with the problem size. That is, the 250 m. time of 31 minutes is roughly four times faster than the 500 m. time of 8.6 m. The effects of opening the files for each tile are most evident by the 1 km. time duration. The time is spent more for opening the files rather than doing computation for such a small region.

Table 6.3: Data Scaling Experiments

Scaling Experiment	Dimensionality	Times Increase	Run 1	Run 2	Run 3	SD (minutes)	Improvement Factor
1 km	1200 x 1200 px	1x	5.1 m	5.1 m	5.0 m	0.05	X
500 m	2400 x 2400 px	4x	8.8 m	8.6 m	8.8 m	0.11	2,3
250 m	4800 x 4800 px	16x	31 m	31 m	32 m	0.57	1,1

The next generation of remotely sensed data is provided by the Sentinel-2 satellite, having 10-meter resolution (esa, 2018a). Such scale would require far more computing power. With 10 m. resolution, the same extent would be covered by 120,000 x 120,000 pixels resulting in 1,44 x 10^8 data point in a single image or 625 times increase when compared to the 250 m. resolution. Computing with 32 cores on the same area would take almost 14 days. To finish within a reasonable time, one should scale to 256 cores in order to finish for approximately 1 day and 17 hours in order to compute for the same area and the period of 3 years.

Based on the best time of 2h and 33 minutes for the 48 cores experiments, to perform the analysis on 19-years of data with 10 days composites on 10 meters resolutions, one should increase the CPU power by 128 times (6144 cores) to finish for 8 days and 10 hours. The times assume a linear increase/decrease, the same number of missing pixels due to snow/clouds, same TimeSat configuration, no additional network, platform, or disk bottlenecks.

[4]https://modis.gsfc.nasa.gov/data/dataprod/mod13.php

6.5 Platform Evaluation Summary

When working with Spark, we followed the heuristics described in Section 6.1. As a general rule, the amount of data should be reduced as early as possible as much as possible to avoid unnecessary data loads or computations. A preprocessing step was taken to crop out the USA region from the global composites. When generating the GeoTiFFs, filtering operations were applied to further reduce the required information from the rest.

In addition, we optimized our transformations and took advantage of RDD caching to minimize re-computation of data re-used several times. In particular, we cache the RDD holding the SOS data for all seasons. This way when the following filtering per season/year is applied the whole chain of RDDs is not re-computed from the beginning, but the cached RDD is used. We observed a significant speeds-up of 14 minutes when generating the SOS GeoTiFFs.

Also, one should be aware of the memory required when Spark processing (or caching). Spark is flexible and will flush its content to disk when there is not enough RAM on the node, but for optimal results, the result set should be kept small enough to fit into the main memory. If this is not possible, more RAM should be added for optimal results.

By taking advantage of HDFS, data locality and co-partitioning, we can minimize the shuffle in our cluster when performing joins (or wide transformations in general). During EVI computation we observed minimal data movements between the nodes.

Monitoring the nodes show the CPUs are fully utilized during TimeSat processing with minimal Network I/O. The dedicated memory for the TimeSat executables is enough to cover their requirements. Disk usage can spike to 10 % with 4 CPUs and to 20 and with 8 CPUs per node. The network bandwidth is fast enough and sufficient for our requirements.

When performing phenology analysis with Spark, the right degree of data partitioning should be set to achieve good level of load-balance. We divided the processing area into tiles, however, due to water, snow, or clouds a lot of data is missing and therefore tiles can have varying processing time, resulting in big cluster-in-balance. By having more but smaller tiles, we can reduce the individual tile duration and improve the load balance. In our experiments, 32x32 (1024 in total) tiling showed best results with a time of 6h and 20 m.

We found a performance penalty of having more tiles. For each tile, the files are served by the s3fs fuse from the local cache and opened by TimeSat. Depending, on the degree of parallelization this effect can increase the total job duration. As a general rule, one should reduce the individual tile duration by increasing the tile's number, however, should be careful not to overdo it as performance penalty of opening/loading the files for each tile will occur. As an alternative and speed up, one can directly read the files from the disk, excluding Minio and *stfs* bottlenecks, however, the ease of managing the files as well as the capability to access data remotely stored is lost.

In addition, within Spark, the level of parallelism was adjusted. We repartitioned the number of partitions to be equal to the number of tiles. Furthermore, during the 30 tiles' experiment, a custom partitioner was implemented to map at most one partition for a task, i.e. assure a uniform data distribution. Hence, an unwanted grouping of tiles was avoided. The Task Scheduler will distribute and schedule a single task (tile) at a time on any available CPU in the cluster.

If scaling is required, more CPU power (cores) should be added since TimeSat is CPU bounded. In our experiments scaling-up showed slightly better results. Having more cores per node the disk load will increase when cores start requesting files. However, since more of the time is spend in CPU computation, disk I/O bottlenecks will show up at a very later stage.

We consider our Spark jobs to be efficient; however, a constraint is that due to the TimeSat limitation to have as input only local files, we can't take advantage of HDFS. In the next Chapter, the platform is shown in action, performing phenology analysis.

Chapter 7

Phenology Analysis

This Section presents the phenology analysis performed with the implemented Spark-based platform for extracting vegetation seasonality. Section 7.1 describes the experimental set-up and Section 7.2 shows an analysis on the function fit and and VIs data. To study the impact of using one or another vegetation index and fitting function on the estimated day of SOS we perform a spatial and temporal comparison based on mean, standard deviation (SD) and the difference in Section 7.3. In Section 7.4 we study the validity and accuracy of the experiments by comparing the SOS results with the ground observations. Section 7.4.4 summarizes the results.

7.1 Experiments Set-up

To compare the SOS derived from the NDVI and EVI extracted from the data provided by the SPOT-VGT and PROBA-V sensors we ran 6 experiments with the 3 fitting functions provided by TimeSat for seasonality extraction: Asymmetric Gaussian (AG), Savitzky-Golay (SG) and Double Logistic (DL). For each indice we applied the 3 functions, resulting in a total of 6 experiments. The additional parameters provided by TimeSat are fixed for all experiments. The parameters are:

- **SOS retrieval method**: The SOS calculation is based on **0.3 %** of the amplitude of the series. Another option is based on a fixed threshold, however the indices' range is different with varying amplitude from place to place, therefore such an option will not universally perform well (White et al., 2009). For example, in areas with rich vegetation the indice can be in the range [0,5- 0,85], whereas in mountainous regions [0,25- 0,45] and in wastelands [0,1-0,2]. Also, some areas are rapidly responding to precipitation, resulting in fast onset of greenness following spring rainfall, however other areas like the desert shrublands have minimal photosynthetically active vegetation cover and thus change in VIs within a season is low (Bradley et al., 2007). Therefore, a curve fitting and SOS extraction procedure must be flexible enough to accommodate such ranges of phenological amplitudes (Bradley et al., 2007) at a continental scale.

  The amplitude percentage is of particular interest. In time-series with a big slope (i.e. gradient vector), the amplitude percentage value has a little effect on the result. Such areas are observed in central-south regions of USA. However, in time-series with smaller

slope, differences of up to a week are observed. Furthermore, a low value can wrongly mark spikes occurring before the true vegetation seasons as SOS, whereas a high value may be not representative (too late) for SOS. (White et al., 2009) performed experiments with a fixed threshold of 0.2 and 0.3, and concluded that the 0.2 experiments were constantly early. After several tests and visually inspecting the time-series and the fitted data, we decided that an amplitude percentage of 0.3 % will best derive the SOS.

- **Spike removal**: To cope with positive and negative outliers that can seriously impact the function fit, we defined a preprocessing step based on median filtering to remove spikes and outliers. We used the default value of 2, denoting that spikes larger than 2 standard deviations from the running median will be removed.

- **Seasonal parameter**: We set this parameters to *1* indicating there is at most 1 vegetation season per year (NASS, 1997).

- **Number of envelope iterations**: Since the most noise from remotely sensed data is negative, the fit is adapted to the upper envelope of the data (Eklundha and Jönsson, 2017). By setting the number of envelope iterations to 2, the fitting function is adjusted during the second iteration with weights (derived from the 1st iteration) of the low data values decreased by some factor.

- **Amplitude cutoff value**: This setting can be useful for areas with minimal seasonal variation e.g. deserts. That is, time-series with smaller average amplitude then the cutoff value defined will not be processed. Although, we have a lot of arid areas in the studied region we choose to take results on all the available data. Therefore a value of 0 was used, processing all the data no matter the seasonal amplitude.

- **Savitsky-Golay window size**: The SG function provides a filtering option to smooth the data and suppress disturbances by replacing each data value with a linear combination of nearby values in a window, similar to a moving average (Eklundha and Jönsson, 2017). The window size determines the degree of smoothing, but it also affects the ability to follow a rapid change. We wanted to disqualify sharp differences, but also not penalize too much on the ability to capture sudden change since in semi-arid areas the vegetation almost instantaneously responds to precipitation (Jönsson and Eklundh, 2004). Therefore, the width of the moving windows was set to 4.

- **Valid data range**: To define the valid VIs range is described in Section 2.3.2, we set the parameter to [0,1].

7.2 Assessing Function Fit and Data Behavior

In an effort to study the curve fitting and the data itself, a subset of NDVI derived SOS data was observed, with temporal range of 3 years (2011-2013) . We noticed there is a lot of missing data during winter periods. From the quality assessment report of the reflectance product we are using to calculate the indices, we found that the spatio-temporal continuity is poor over latitudes higher than 45º North, with a percentage of missing values up to 100 % (Figure 7.1) (Vito, 2018a). This is explained by the lack of clear-sky observations. Lack of observations impede the fitting functions to fit a curve and therefore results in a substantial among of missing values in high latitude regions. Therefore, this reduces our SOS results in high-altitude regions.

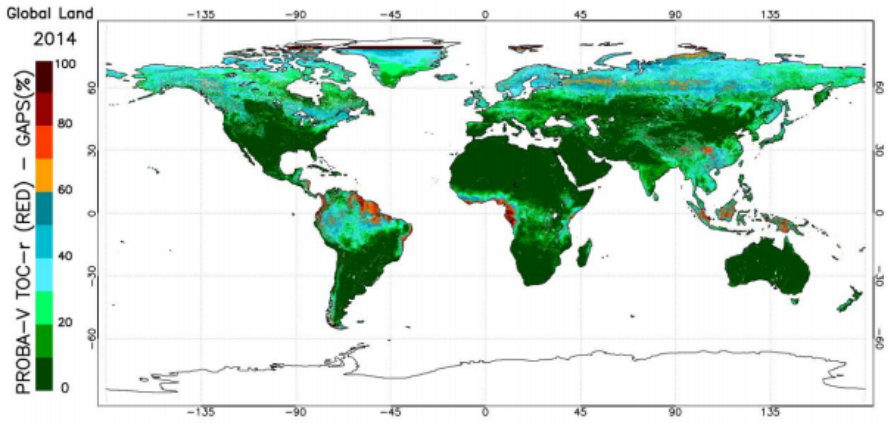


Figure 7.1: Percentage of missing values for PROBA-V TOC-r (red band) product considering all land pixels. (Vito, 2018)

Generally, both VIs indicated a good/logical amplitude throughout North America. The example shown in Figure 7.2 displays an excellent time-series with high amplitude during the beginning of the calendar year. Such patterns are observed in the South where the vegetation is rich (Louisiana, Mississippi, Arkansas). The SOS results, indicating the days passed since the beginning of the year, for the first year for AG, SG, DL are (74,83,65) and for the second (81,71,72), respectively. The variability is relatively small but still substantial.

Figure 7.3a shows less than perfect time-series with low amplitude value and a small slope for the second year. The results from the three functions vary more (-67,74,18). The negative value as described in Section 5.4.4 indicate the SOS started in the previous year. What we also see is that the SG function is over-fitting the time-series. The AG function is least susceptible to long-term variation, whereas the DL function is more affected by local trends. This is also observed in Figure 7.3b where an increase in the time-series' amplitude during the second year, causes an evident difference in the function fit. The predictions of the AG, SG and DL functions are 200, 76 and 127 respectively, demonstrating significant differences.

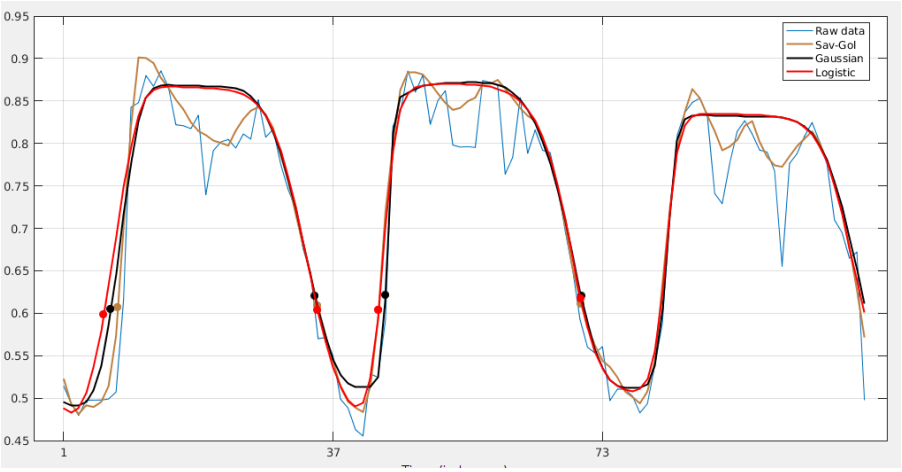
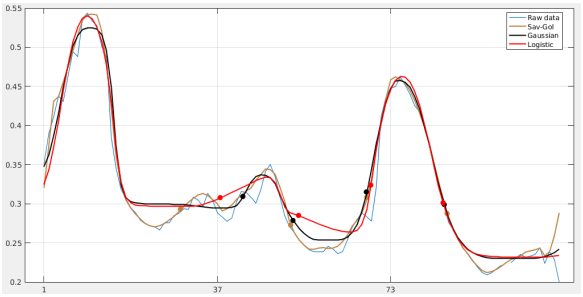
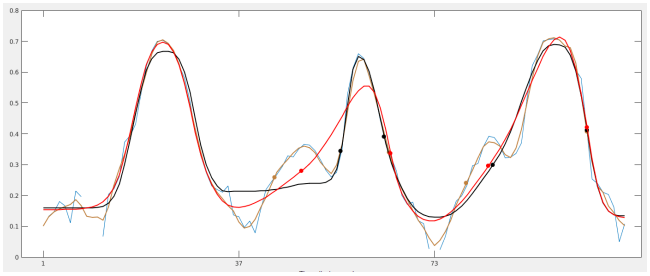


Figure 7.2: Raw data + fitting of the AG, SG and DL functions: Example of Smooth, Regular Time-Series (Dots Indicate SOS and End-of-Season (EOS), x-axis is the time interval, y-axis VI values.



(a) Functional Fit Comparison: Low Amplitude and Small Slope



(b) Functional Fit Comparison: Spike Before the True Seasonal Amplitude Increase

Figure 7.3: Functional Fit Comparison: Irregular Time-Series (Dots Indicate SOS and EOS)

7.3 Compare SOS Products and Functions

In this Section we present our comparison results. Section 7.3.1 shows the mean and Section 7.3.2 shows the standard deviation (SD) across the temporal range of the time-series (1998-2017). Section 7.3.3 displays the spatial and temporal difference between the EVI-AG experiment with the rest. Across our experiments, we plot the results on maps to study the results spatially.

When geographical regions are discussed, we use both the usual geographical cardinal directions (north, east, south, and west). We further separate on USA regions to more accurately describe and pinpoint geographical locality. A map of the USA regions can be found in Appendix D.3.

7.3.1 Mean

The average value per pixel was calculated for the whole range of the time-series from 1998 to 2017. Table 7.1 shows the minimum and maximum predictions for each of the experiments. To remove the outliers, the highest and lowest 2 % of values were discarded. Results show, the NDVI has a higher overall spread. This results in lower minimum values which can go negative, indicating SOS in the previous year, and higher maximum values. The spread between the min. and max. observations is about 10 days higher when compared to the EVI. The results for the AG and DL showed generally similar values when compared per index, however the SG function shows earlier SOS by several days, more noticeable in the EVI-SG experiment.

Table 7.1: Average Min/Max and Mean Values for Each of The Experiments

Experiment	Min	Max	Mean
NDVI- AG	-10	204	113.8
NDVI - SG	-1.6	201.77	107.9
NDVI - DL	-8.92	205.64	113.5
EVI- AG	13	192	107.2
EVI - SG	8	186	100.6
EVI - DL	15	191	107.3

Figure 7.4 shows a histogram of the value distribution for each of the experiments. The results are fairly consistent, with most of the values clustering around the 100th day. The histograms are skewed to the left indicating a higher percentage of values greater than 100. In the 150-200 days range, the biggest differences are observed between VIs, where the NDVI experiments show a higher occurrence of values.

Figure 7.4: Histograms for the mean SOS per experiment.

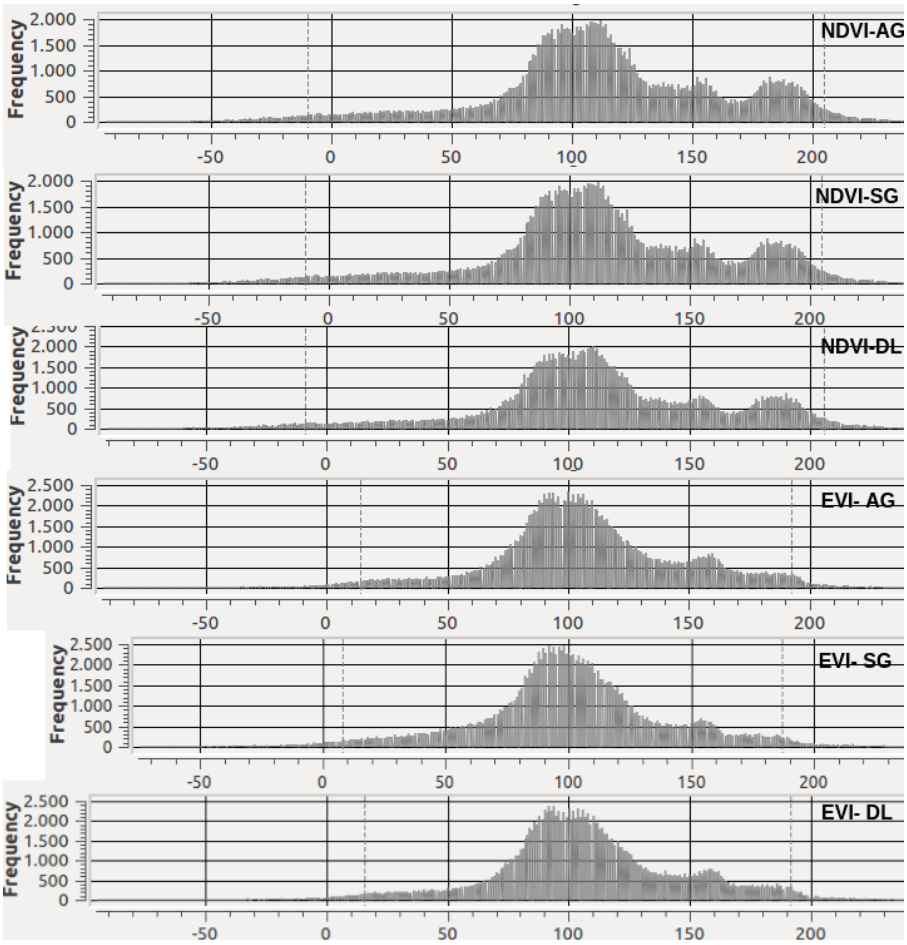
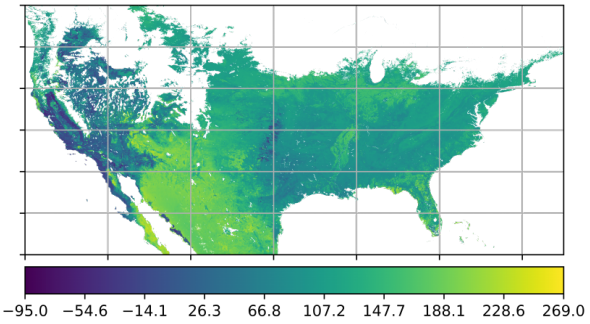


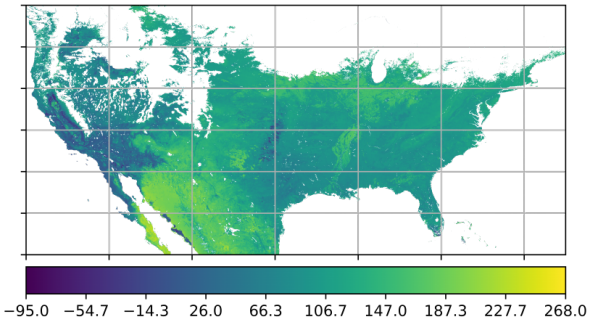
Figure 7.5 shows a SOS map for each of the experiments. We see, the phenological pattern depends strongly on latitude. The SOS values advance northward from 30 to 100 days in South USA to values ranging from 100 to 180 days in North USA. The growing season starts, in most parts of the central USA between the 80th and 120th day, but on all models a small patch of values is observed with detected early season. In West USA all models show the earliest season in the range -50 to 80 days. In the mountainous and desert region of South-West North America (see Appendix D.2 for elevation map), the SOS is the latest, having a range of 140-190 days.

Figure 7.5: Average SOS over the 1998-2017 period

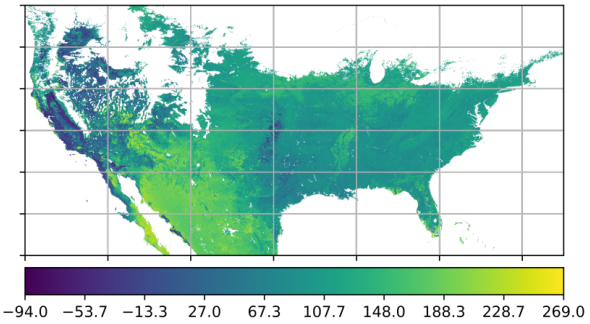
(a) NDVI -AG



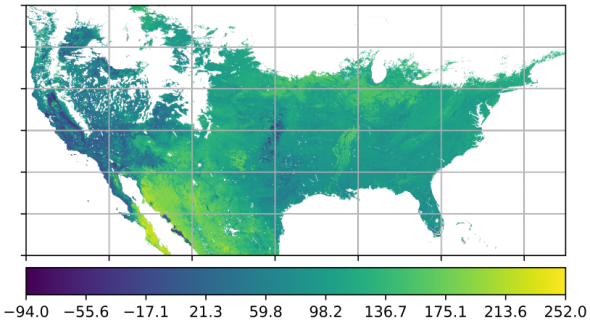
(b) EVI - AG



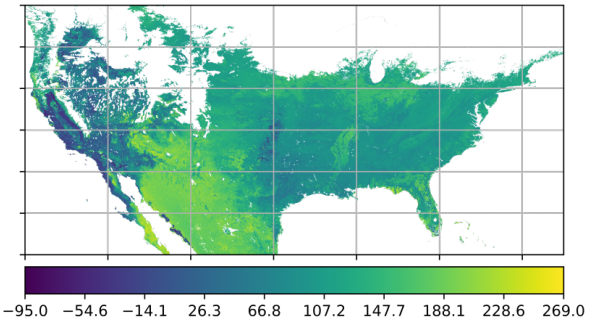
(c) NDVI - SG



(d) EVI - SG



(e) NDVI - DL



(f) EVI - DL

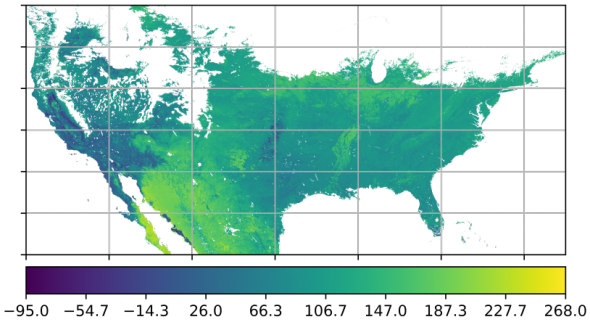


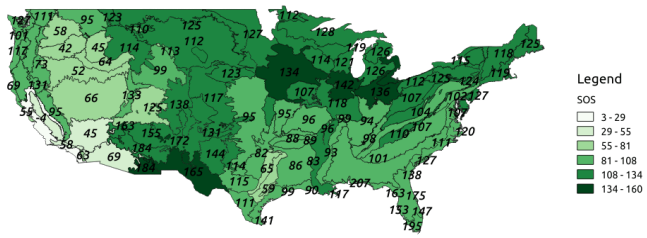
Figure 7.6 further separates the average values on ecological regions. An ecoregion is a "recurring pattern of ecosystems associated with characteristic combinations of soil and landform that characterise that region" (Burgess et al., 2004). For example, desserts, forests or corn plains are ecological regions. In Appendix D.4 the names of the ecological regions are displayed. Although we have much smaller number of values in high longitudes, the average was calculated on all the available valid pixels. We see in the Eastern Temperate Forest values between 100 and 150 days are observed. The Western, Central and Eastern Corn Belt Plaints show relatively high SOS as well in the range 130-150, whereas in the West deserts the range is lower between the 30th and 80th day.

By observing Figure 7.6, we can better judge the spatial difference between the experiments, since we have a more distinct visualization. In the central and Midwest regions, minimal differences are observer across VIs and functions (0-10). Across other regions greater variations start to show with biggest differences observed between VIs. For example in Florida, an ecoregion classified as Eastern Temperate Forests, biggest differences are observer, where the NDVI predicts later values (by 30 to 70 days), as well as in the West Coast of the USA, where the difference is between 0 and 20 days. However, the NDVI shows earlier predictions in the North America Deserts (0-30 days), but later predictions in the Chihuahuan desert.

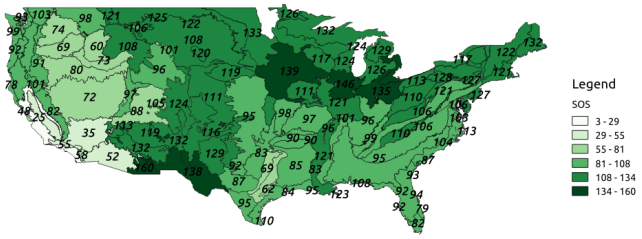
There are minimal differences (0-3 days) between the AG and DL function for both of the VIs, whereas the SG function generally shows earlier predictions across the whole geographical region (0-10 days), with an exception being the "Sonoran Basin and Range", where the SOS prediction is 20 to 30 days later, when compared to the other functions.

Figure 7.6: Average SOS over the 1998-2017 Period on Ecological Regions

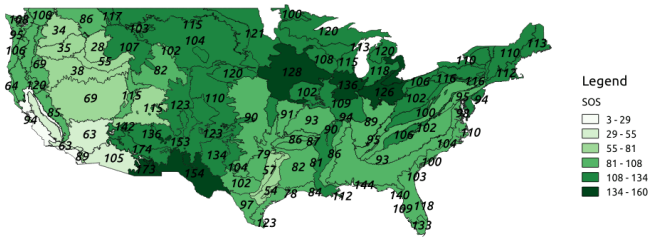
(a) NDVI -AG



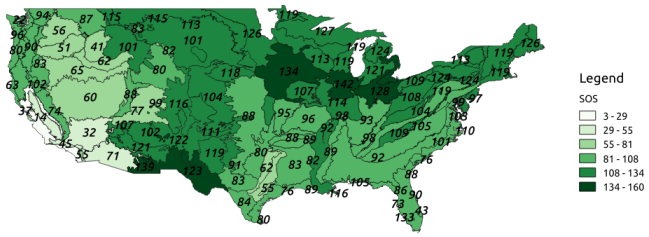
(b) EVI - AG



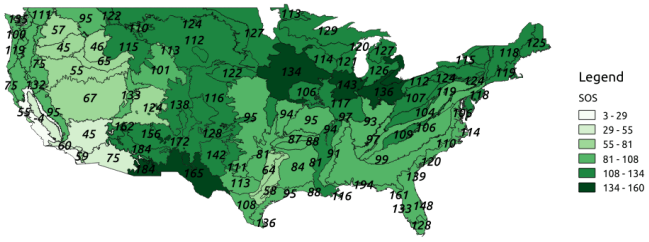
(c) NDVI - SG



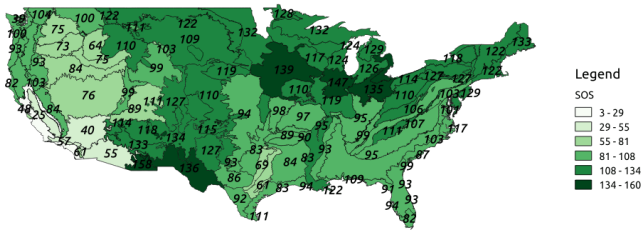
(d) EVI - SG



(e) NDVI - DL



(f) EVI - DL



7.3.2 Standard Deviation (SD)

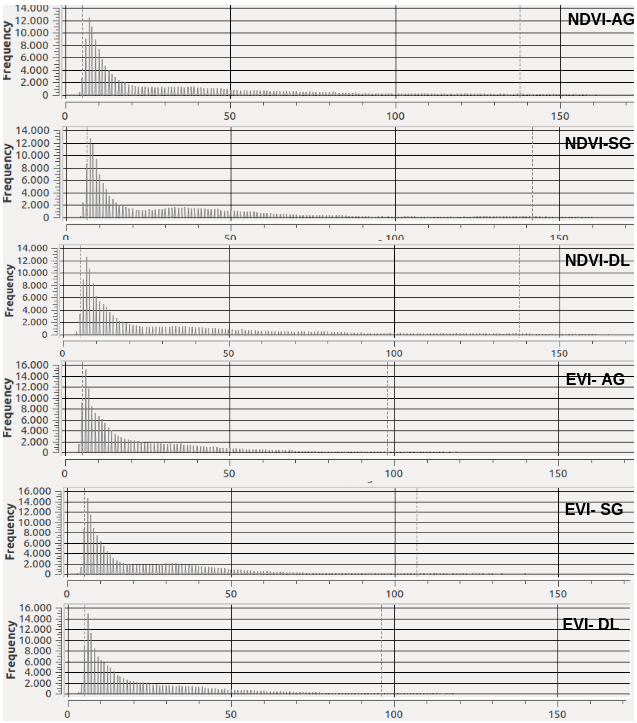
To assess the seasonal spatial change across all the years the standard deviation (SD) was calculated for the 1998- 2017 period. Results shown in Table 7.2 display the min/max SD observed after the highest and lowest 2 % of values were removed in order to separate the outliers. The minimum boundary is fairly consistent across experiments ranging from 4.92 to 5.04 days, however we have higher deviations in the max boundary with 95.03 for EVI- AG to 147 for NDVI-SG. We observe much higher SD values for the NDVI (30-40 days) if compared to the EVI, similar AG and DL results across VIs, and higher SD values for SG if compared to the rest of the functions.

Table 7.2: SD Min/Max Values for Each of The Experiments

Experiment	Min	Max
NDVI- AG	4.92	137.80
NDVI - SG	6.01	146.77
NDVI - DL	5.04	137.88
EVI- AG	5.07	97.91
EVI - SG	4.98	106.81
EVI - DL	4.95	95.93

To better assess the SD SOS distribution, the histograms for all the experiments are shown in Figure 7.7. Most of the values are clustered between 4 and 25 days with a significant percentage of values going up to 50 days. Less than 1,000 pixels per value unit have SD more than 50. A relatively low frequency of pixels have SD greater than 100 days. All histograms show identical behavior, however small differences are observed. For example, the SG function for both EVI and NDVI (2nd and 5th panel) shows a slightly greater SD spread. This is most likely as a result of the function overfitting the time-series and being more susceptible to variations and spikes.

Figure 7.7: Histograms for the SD SOS per experiment



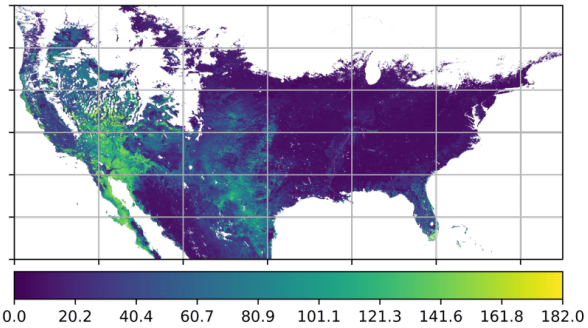
To better evaluate the spatial differences the SD results are plotted on maps in Figure 7.8 and further Figure 7.9 shows the same result, however, to zoom-in in our evaluation, only SD values less than 50 days are shown. In South-East, Mid-West, Central and eastern part of South Central USA, all models show similar results, with slight differences in the South-West and West regions. Central and Mid-West regions show minimal SD between 5 to 10 days, occasionally going up to 15 days. Usually, in such regions the time-series is smooth with good seasonality, observations stay high, then drop, and the pattern repeats from one year to another. In the West, South-Central and in northern Mexico the SD is significantly greater ranging from 20 to 100 days. The highest variability is in the western USA in the dead valley and the approximating regions. The NDVI index shows a greater dispersion of high SD values, observed in the Nevada region as well.

There can be several reasons for the higher SD values:

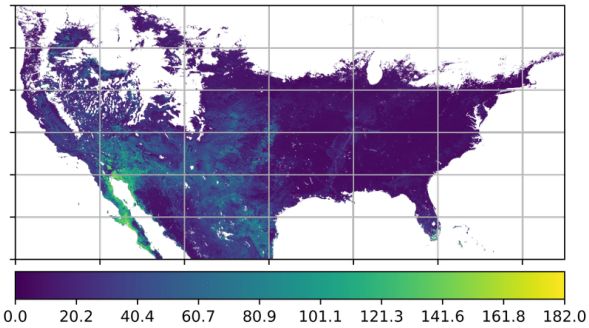
- The SOS vary per year. In the South West region, the SD can go up to the 20-50 range 7.9, A typical time-series for the region is with low VI values and low amplitude variation. Appendix A.3 shows such a scenario.
- Furthermore, some years show high amplitude values (in the VIs) (0.4-0.5), whereas other years show minimal amplitude variations, effectively years with little to no seasonality. This implies such regions do not have regular seasonality, resulting in varying SOS. Appendix A.2 visualize such a scenario.
- Another case is when the time-series is wobbly, noisy or incomplete, resulting in varying function fit as well. In such cases the SOS value can't be adequately derived.
- Finally, the highest SD values observed in the arid desert region of Arizona (Figure 7.8) is due to the extremely low amplitude variation (<0.03) due to the little or non-existent vegetation, which again results in various function fit (Appendix A.4). Furthermore, if minimal vegetation exists the VIs amplitude can be triggered by rainfall from the Pacific Ocean from November through March and can be incorrectly increased by an accumulation of snow during the winter periods (wrcc, 2014) (Appendix A.5). When SOS vary through the whole year's range, big differences can be observed, additionally highlighted by early springs and negative numbers.

Figure 7.8: Standard Deviation for SOS over the 1999-2017 period (full SOS range)

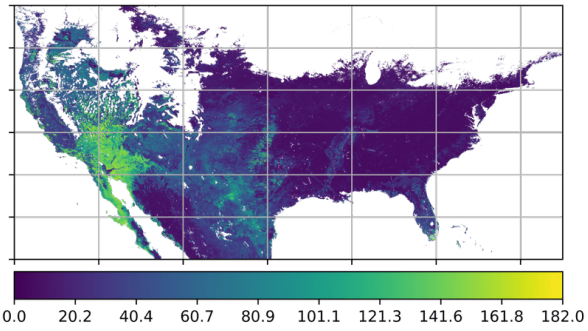
(a) NDVI -AG



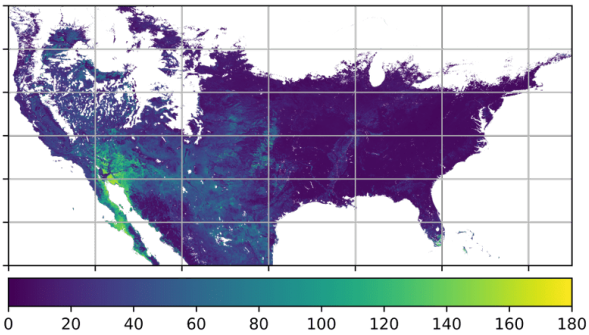
(b) EVI - AG



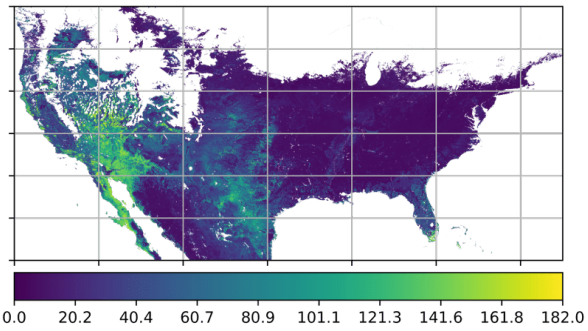
(c) NDVI - SG



(d) EVI - SG



(e) NDVI - DL



(f) EVI - DL

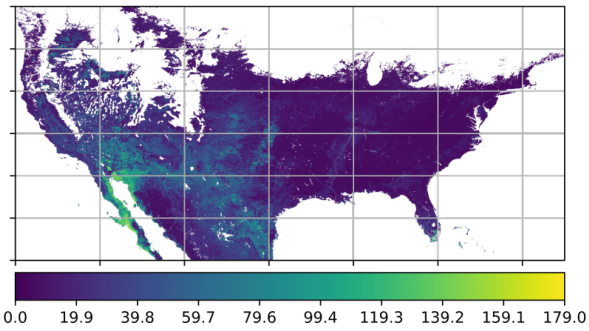
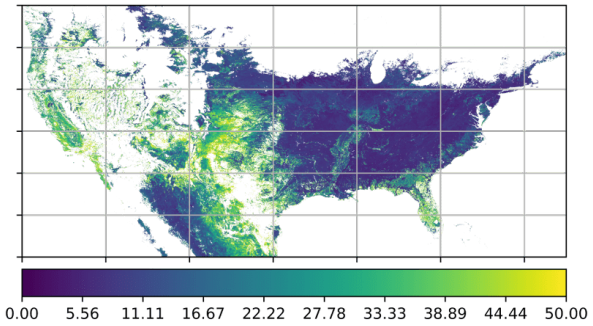
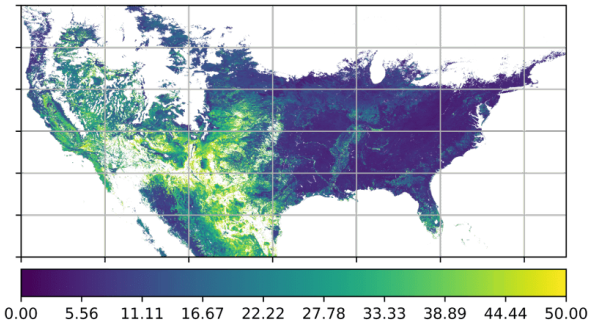


Figure 7.9: Standard Deviation for SOS over the 1999-2017 period (0-50 range)

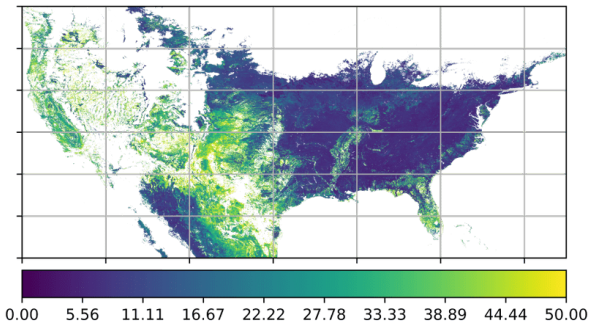
(a) NDVI -AG



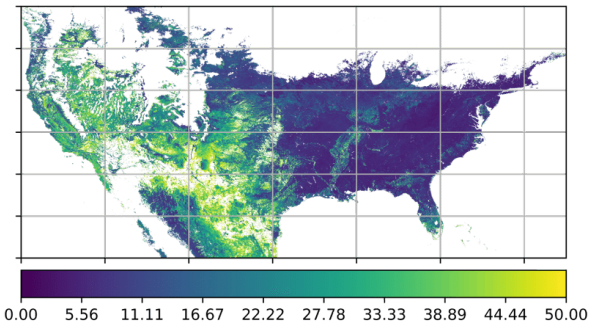
(b) EVI - AG



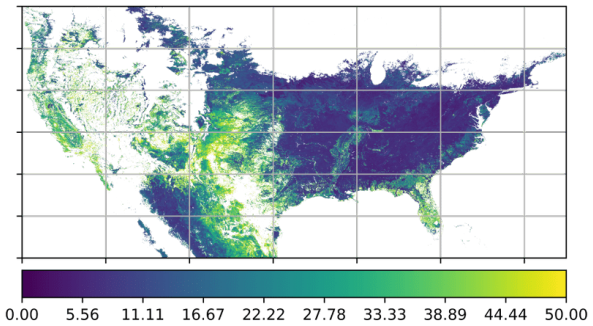
(c) NDVI - SG



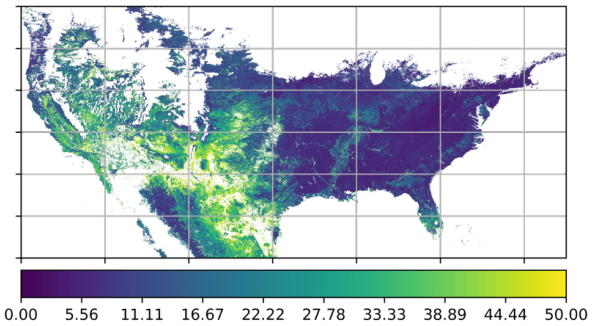
(d) EVI - SG



(e) NDVI - DL



(f) EVI - DL



7.3.3 Compare Difference of SOS Experiments

To assess how the different functions and indices compare to each other, their average difference across the years (2000-2016) was calculated, whereas in Appendix D.1 the difference for the year 2016 only is displayed. The process involves calculating the difference per year (and per pixel) and taking the average difference across all the years. For this experiment, we fix the EVI-AG results and subtract the rest of the experiments from it.

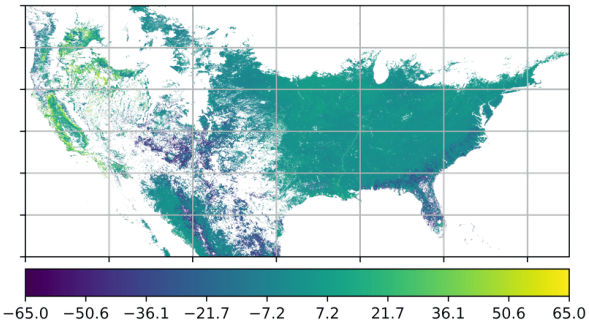
To better visualize the results shown in Figure 7.10 all values below -65 and above 65 were removed. Again, we observe similar behavior. In central to East regions there are minimal differences (0 up to 5 days). For the same region, there are some observations which can have a greater positive difference, such as Figure 7.10b where the difference can go up in the range [20,30], indicating the NDVI-SG reports earlier predictions.

In general, we see there is a minimal,but still some, differences observed per function choice; the EVI-SG (7.10c) and EVI-DL (7.10e) show minimal differences with the EVI-AG. The same pattern can be inferred by the similar differences observed in the NDVI -AG (7.10a), NDVI-SG (7.10b) and NDVI-DL (7.10d) experiments. However, the latter three also show there are significant differences between VIs (when the EVI experiments are compared with the NDVI experiments).

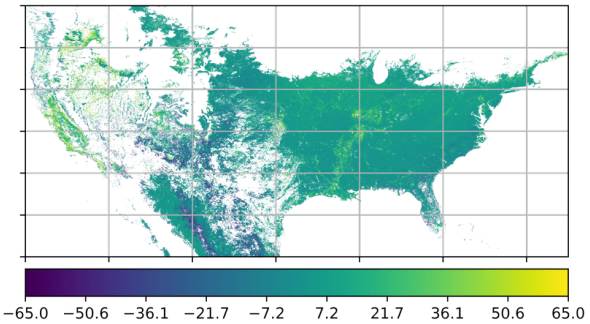
The biggest difference is observed in the South-West region, as well as in Florida, where EVI- SG (7.10c) reports earlier SOS (by 20 to 50 days), whereas NDVI-AG (7.10a), NDVI-SG (7.10b), NDVI-DL (7.10d) show later SOS (by 10 to 50 days). However, the latter 3 models show earlier SOS to the West. If we are to correlate our finding from 7.1, we can now say from which regions the greater NDVI range comes from.

Figure 7.10: Difference between SOS experiments for the range 2000-2016

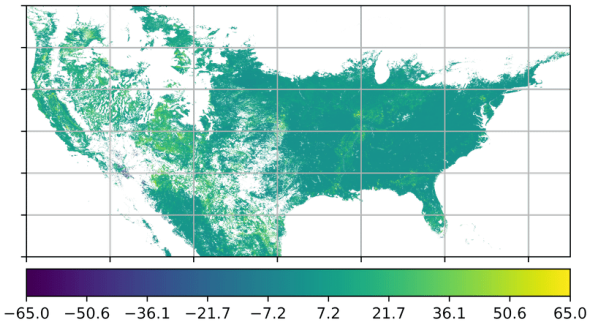
(a) EVI-AG minus NDVI- AG



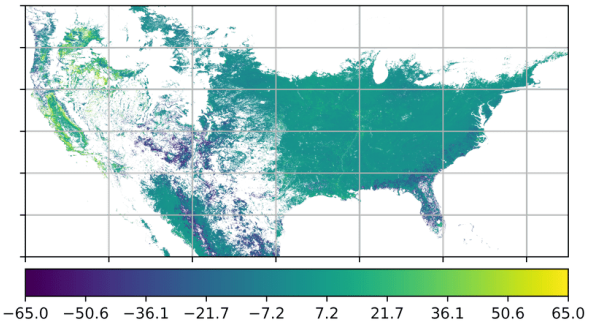
(b) EVI-AG minus NDVI-SG



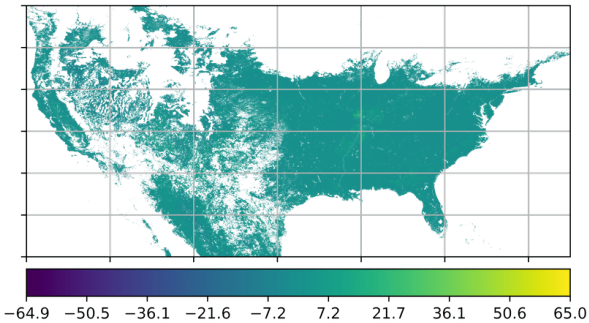
(c) EVI-AG minus EVI- SG



(d) EVI-AG minus NDVI - DL



(e) EVI-AG minus EVI - DL



7.4 Compare SOS vs Ground Observations

In this Section we compare the results from our experiments with the ground observations provided by volunteers, and curated by the USA-National Phenology Network (USA-NPN) initiative as described in Section 2.2. We decided to perform the comparison on a species level, rather than the whole *Plantae* Kingdom, since different species can have different vegetation cycles and patterns (LaLiberte, 2016). The results from each experiment are going to be compared with the top 5 species with the highest frequency of occurrence. In Section 7.4.1 an exploration of the ground measurements data is performed, whereas the set-up and the results are presented in Sections 7.4.2 and 7.4.3, respectively.

7.4.1 Validation Dataset Exploration

To understand the characteristics of the ground observation dataset, data exploration was performed. This is an important step because we will have a better understanding of the ground observations, but also we can detect missing data, outliers or irregularities. The dataset spans the period 1956 - 2017 and combines a historical dataset (1956-2006), and the data collected for the USA-NPN. Our SOS results range from 1999 to 2016, and that is the period we can perform our comparison.

Figure 7.11 displays the distinct sites which provide SOS indicators. We see that during the early years, between 1955 and 1968 the observations were in an uptrend, however after that period, they were at a decline. Between 1995 and 2005, the phenology ground observations field was dormant with as little as 50 sites per year, however, with the USA-NPN effort in 2005, the number of sites sharply increased, having more than 1400 sites in 2017. The graph roughly shows with how many data points we can compare the RS results.

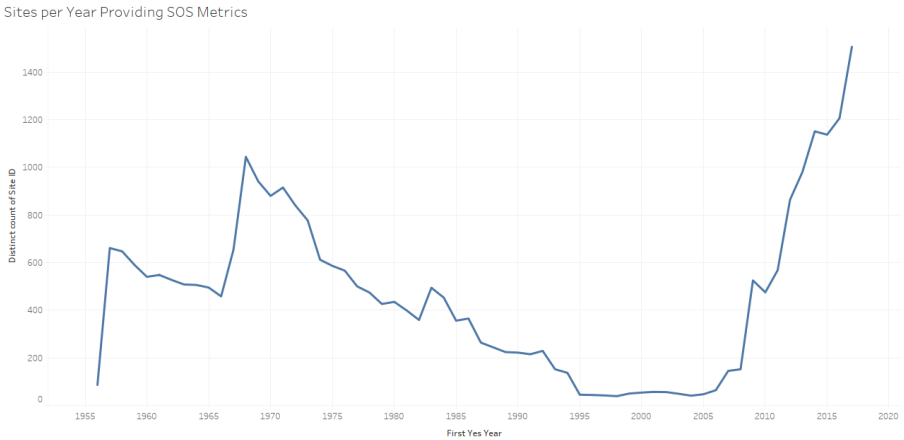


Figure 7.11: Sites per year having SOS phenology states, Figure shows the decline of phenology observations between years 1970 and 2005 and the proliferation after year 2005

To study the spatial properties of the dataset, we plot the observation locations over the USA with a yearly color gradient (see Figure 7.12). Most of the observations are clustered in the West and the East regions. The color gradient shows the temporal distribution of the locations, and we can see the recent observations are mostly in the East-Central USA. It is the observations from 1999 to 2016 which will overlap with out RS results and be able to compare with.

The SOS values are studied by plotting the average SOS "yes" month per site (see Figure 7.13), that is the first month when SOS was observed. The earliest SOS is along the West and East coastlines as well as in the West-South-Central region, consistent with our finding in 7.3.1. However, several observations can be seen as having high SOS value.

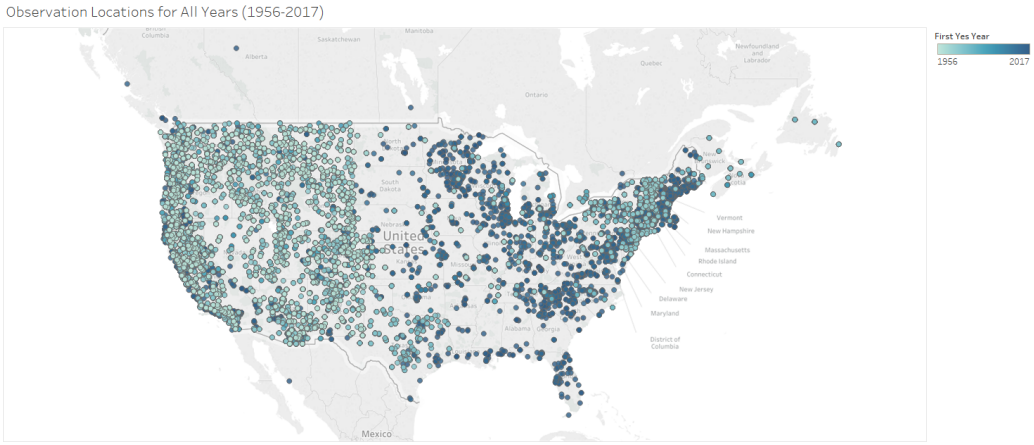


Figure 7.12: Observation Locations for All Years (1956-2017). The Figure shows the initiations of sites across the USA from the 1956 until 2017

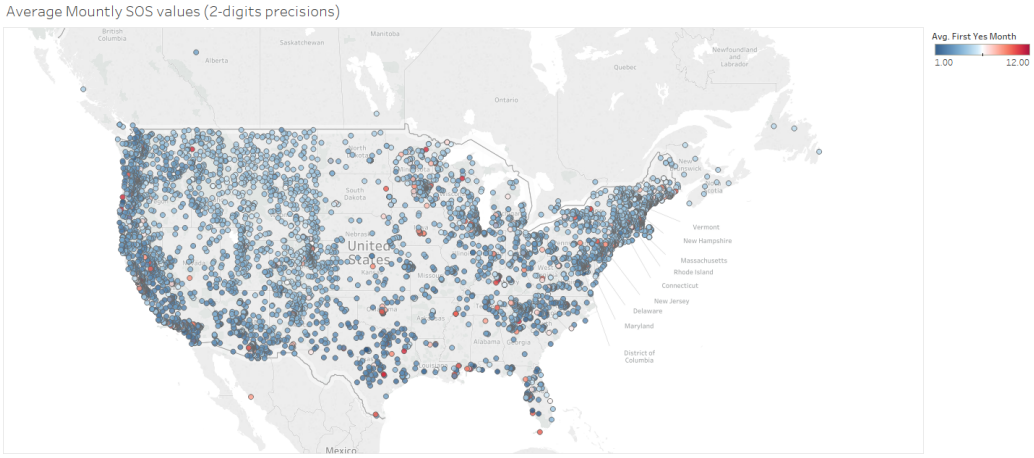


Figure 7.13: Average Monthly SOS Value Clustered per Site for All Years (1956-2017). The Graph Shows the Average SOS Month for All Observations in the *Platae* Kingdom

7.4.2 Experiments Set-up

The dataset contains various phenological phases indicating the plant's phenophase state. The phenophases indicating the beginning of season (SOS) were identified to be *First Bloom*, *"Breaking Leaf Buds"*, *"Flower or Flower Buds"* and *"First leave (historic lilian/honeysuckle)"*.

The top 5 species, on which the study was conducted, with the highest number of matches between TimeSat predictions and ground observations are: red rothomagentic liliac (*syringa chinensis*), red maple (*acer rubrum*), flowering dogwood (*comus florida*), tulip tree (*liriodendron tulipifera*) and black elderberry (*sambucus nigra*).

When we performed the comparisons, only the earliest observation per plant for a year was taken into account since we want to see the onset of the SOS phenophase. This is because phenological status is reported by the volunteers by ”yes” or ”no” answers to a series of questions, for example, “Do you see leaves? (Rosemartin et al., 2018). This approach allows for the quantification of uncertainty around the actual date of the beginning or end of a phenophase (Rosemartin et al., 2018). For deriving the SOS date, we used the first ”yes” observation. However, the actual event might have begun before the first ”yes” observation, but still no prior to the last ”no” observation.

Since our RS data comes in 1 km resolution, it is possible that several ground observations (i.e. plants) are contained over the 1 km. by 1km. RS region (pixel). Three options are present: comparing the single TimeSat SOS value with several ground observations, taking the earliest observation per region (i.e pixel) or averaging all the observations contained in the same region (if more than 1 observation exists). Since most of the observations are 1 per pixel, we decided to be consistent and take the minimum value of the plants in cases when more than 1 observation is present per pixel. The same approach was used by White et al. (2009). Additionally, if the average is taken, the value might be incorrect scaled-up by observations with large uncertainty on the exact occurrence of the SOS (if the first ”yes” observation is far from the last ”no”). Figure 7.14 displays such a situation. In this theoretical case the comparison will be as 60 - min(59,65) = 60 - 59 = 1.

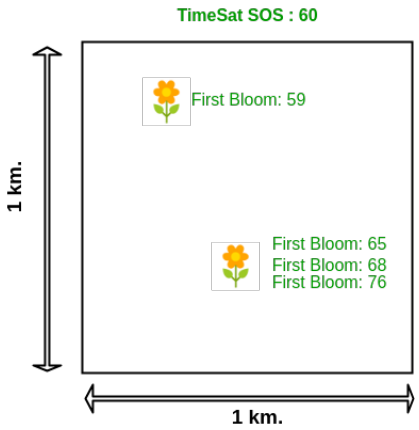


Figure 7.14: Representation of a 1 km. Tile Having More than 1 Ground Observation

For the experiments only values between 0 and 270 were considered both from TimeSat and from the ground measurements. That is we don’t assess early spring values (between 1st of October and 31st of December) since the ground measurements do not distinguish between early spring or late spring.

As a main metric of evaluation, we used the mean absolute error (MAE) to estimate the difference between the satellite-derived SOS and the ground-observed SOS. The MAE is the average of the absolute errors calculated by the formula:

$$MAE = \frac{1}{n}\sum_{i}^{n} |(P(rs)_i - P(ground\ observation)_i)| \tag{3}$$

where $P(rs)_i$ and $P(\text{ground observation})_i$ are the satellite-derived SOS and the ground-observed SOS at pixel i, respectively, n is the number of samples across all years. This metric was chosen over the root-mean-square error (RMSE) since RMSE will give unnecessary weight to higher deviations (Medium, 2016).

A regression model was fitted with the predicted vs. observed SOS, from which we calculated its R^2. The R^2 ranges from 0 to 1 and with it we can asses the ratio of explained to total variance between SOS predictions and SOS observations.

7.4.3 Results

The following paragraphs summarize the results. The evaluation metrics are displayed into tables, whereas scatter plots show the predicted vs. observed values. For easier assessment, the identity line is shown (predicted = observed). The x-axis displays the TimeSat predictions, whereas the y-axis shows the ground measurements observations.

In all experiments the EVI shows better MAE error and R^2 value, whereas the best function results vary per species. For example, the best statistics for the tulip tree experiments are observed with the EVI-SG, whereas the best statistics for the black elderberry are observed with the EVI-DL. The MAE variability is minimal across function choice (2-3 days), with a bit greater variability across VIs (3-4 days, where EVI is performing better).

Generally, all experiments show very good predictions close to the identity line, however several outliers are observed, which contribute to the relatively low R^2 metric. It is difficult to determine if those observations are outliers in the observations or TimeSat mispredictions. For example, in Figure 7.16 several ground observations are noted as having late SOS, which is significantly different from the SOS of the rest of the sample size, likely being outliers in the ground measurement's dataset. However, in Figure 7.15, several observations are classified as having SOS in the range (100,250) by the NDVI experiments, whereas the EVI predicts SOS in the range (0,100). In this particular case, the NDVI models mispredict the SOS.

Table 7.3 shows the experiments for the **red liliac** species. Overall, the MAE varies between 24.1 and 29.5 days, with the EVI experiments having noticeable better results. The EVI- AG experiments show the least variability with R^2 close to 0.1. When we observe Figure 7.15 we see there are some very good predictions close to the identity line, however most of the observations are about 15 to 20 days away from the line. The EVI-SG experiment shows the best MAE calculated and the best centrality of the bulk of observations along the identity line. In the NDVI experiments we can notice some misclassification of early SOS, but predicated TimeSat late SOS (bottom- right corner), which contributes for the higher MAE. In all experiments for this species there are few outliers of more than 100-150 days.

Table 7.4 shows the experiments for the ***red maple*** species. The MAE calculated ranges from 27.4 to 30.2. The EVI experiments show slightly better results, with best metrics observed in the EVI-AG experiment. The best centrality along the identity line is observed in the NDVI - SG experiments (Figure 7.16). However, most of the values are below the line, inferring TimeSat predicts later SOS in general.

Table 7.5 shows the experiments for the ***flowering dogwood*** species, with the EVI-AG and EVI-SG having best statistics. There is a good centrality along the line with a substantial number of points at close proximity (5-10 days) (Figure. 7.17. Similar pattern with the red liliac experiment is observed as the NDVI experiments mispredict few early observations to be considerably higher. All models show twenty to thirty outliers with higher errors.

Table 7.6 shows the experiments for the ***tulip tree*** species. The experiments show the best MAE statistics (21.3 MAE). Due to less number of outliers the R^2 is better. All models show a good distribution along the line, again with slightly later TimeSat predictions if compared to the ground measurements (Figure 7.18). The EVI - SG experiment shows slightly earlier predictions when compared to the other models, with slightly fewer outliers, resulting in having the better statistics (Figure 7.18).

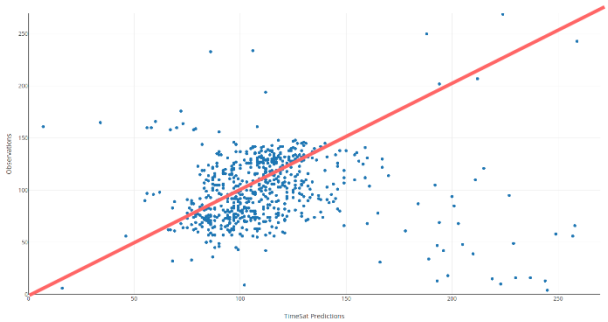
Table 7.7 shows the experiments for the ***black elderberry*** species. The experiments show the best R^2 metric ranging from 0.31 - 0.33 for EVI and 0.09 - 0.24 for the NDVI experiments. The EVI experiments show best results with MAE ranging from 28 to 32 days with the best performing method being EVI-DL. Figure 7.19 shows a good spread of the values along the identity line. The early SOS observations are correctly predicted by TimeSat (bottom left corner).

Experiment	MAE	R^2	Values Compared
NDVI - AG	28.6	0.007	656
NDVI - SG	28.8	0.007	660
NDVI - DL	29.5	0.004	662
EVI - AG	24.5	0.098	667
EVI - SG	24.1	0.062	665
EVI - DL	24.8	0.095	670

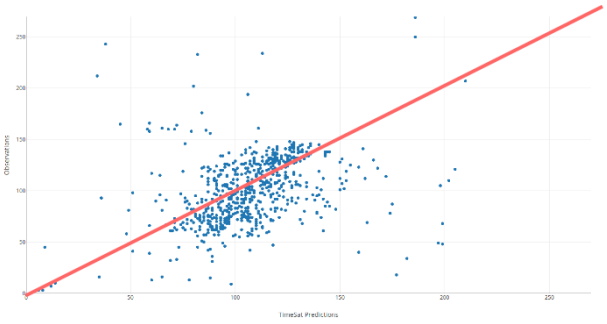
Table 7.3: Red Liliac Experiments Results

Figure 7.15: Comparison on the **red liliac** species (x-axis shows TimeSat predictions, y-axis ground observations, the red line is the identity line (x=y))

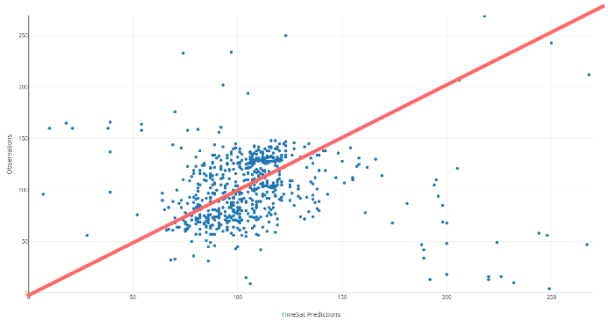
(a) NDVI- AG



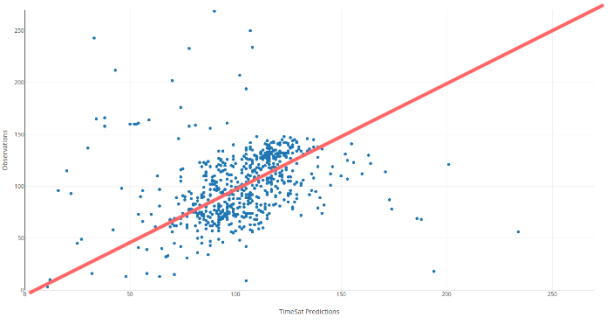
(b) EVI-AG



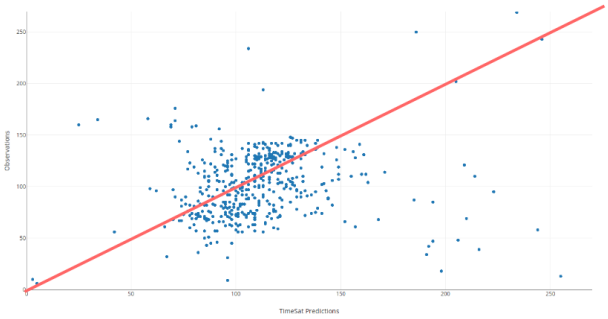
(c) NDVI-SG



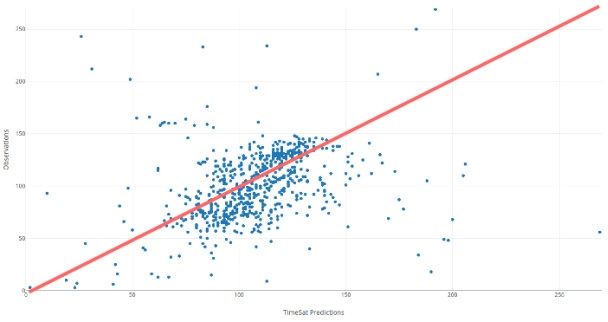
(d) EVI-SG



(e) NDVI-DL



(f) EVI-DL

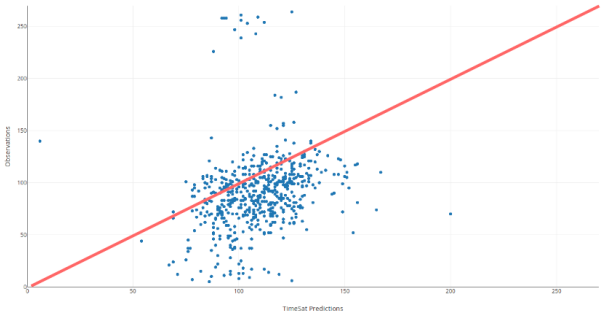


Experiment	MAE	R^2	Values Compared
NDVI - AG	27.4	0.06	578
NDVI - SG	30.1	0.05	579
NDVI - DL	30.2	0.05	576
EVI - AG	29.14	0.07	576
EVI - SG	28.3	0.05	576
EVI - DL	29.7	0.07	576

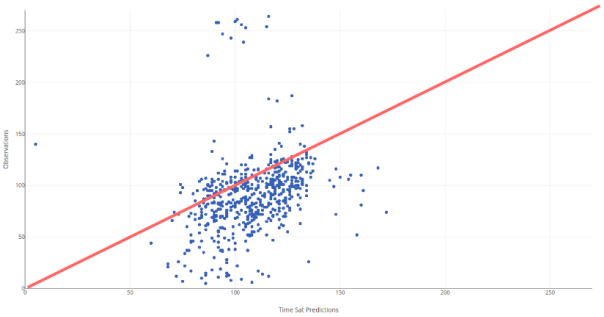
Table 7.4: Red Maple Experiments Results

Figure 7.16: Comparison on the **red maple** species (x-axis shows TimeSat predictions, y-axis ground observations, the red line is the identity line (x=y))

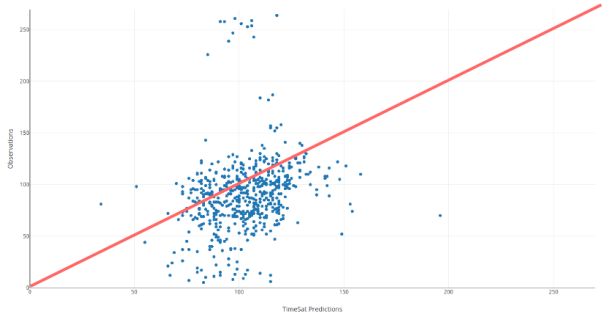
(a) NDVI- AG



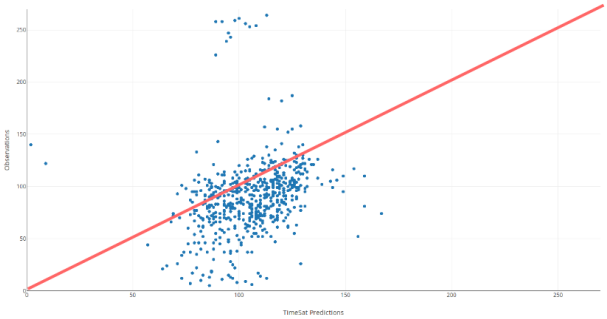
(b) EVI-AG



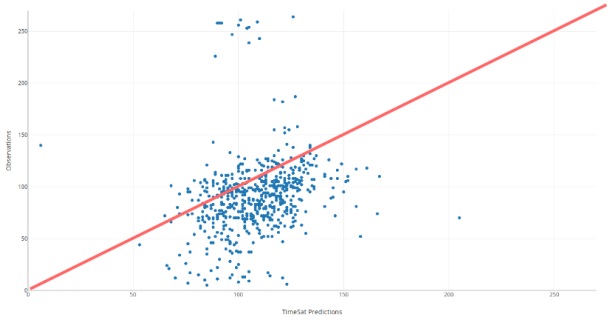
(c) NDVI-SG



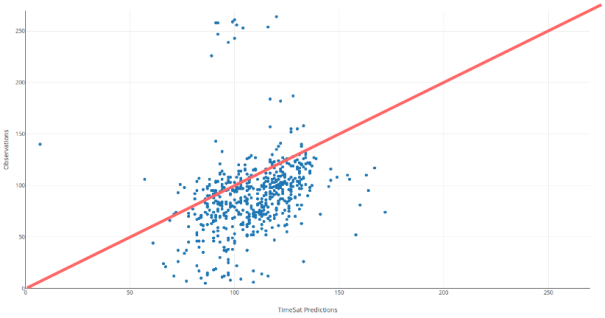
(d) EVI-SG



(e) NDVI-DL



(f) EVI-DL

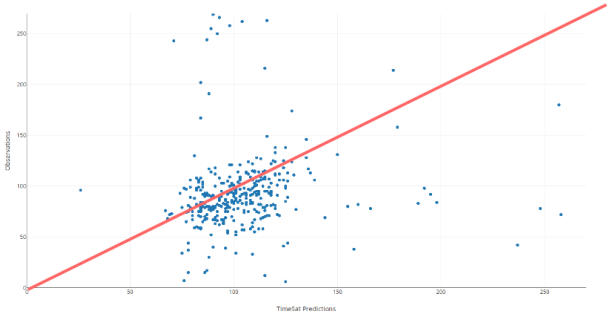


Experiment	MAE	R^2	Values Compared
NDVI - AG	26.4	0.008	364
NDVI - SG	28.5	0.013	361
NDVI - DL	27.2	0.012	359
EVI - AG	24.1	0.03	369
EVI - SG	23.7	0.03	368
EVI - DL	24.7	0.03	370

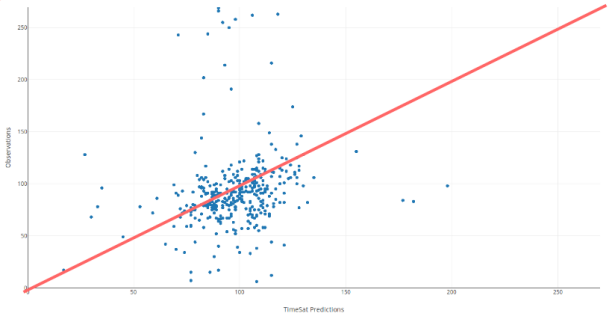
Table 7.5: Flowering Dogwood Experiments Results

Figure 7.17: Comparison on the **flowering dogwood** species (x-axis shows TimeSat predictions, y-axis ground observations, the red line is the identity line (x=y))

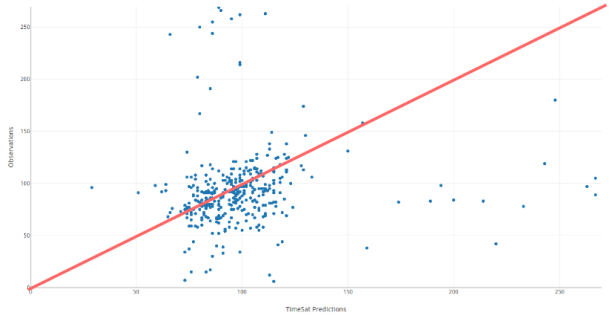
(a) NDVI- AG



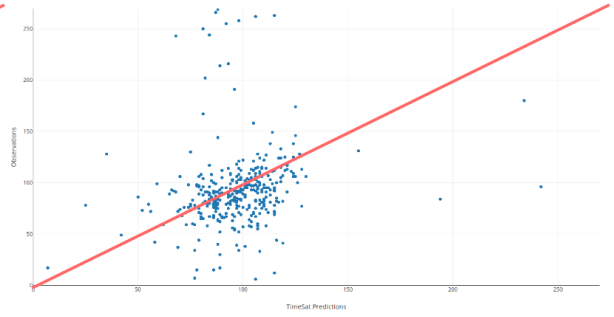
(b) EVI-AG



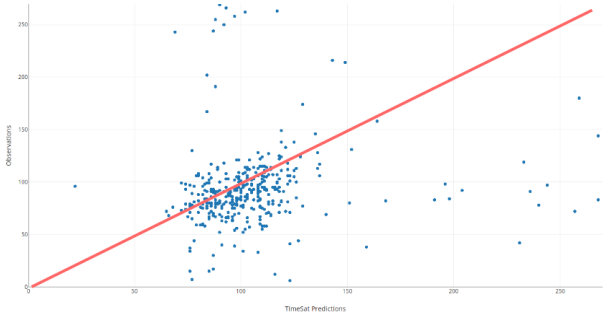
(c) NDVI-SG



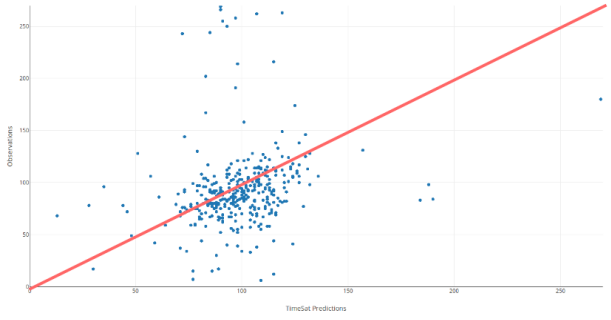
(d) EVI-SG



(e) NDVI-DL



(f) EVI-DL

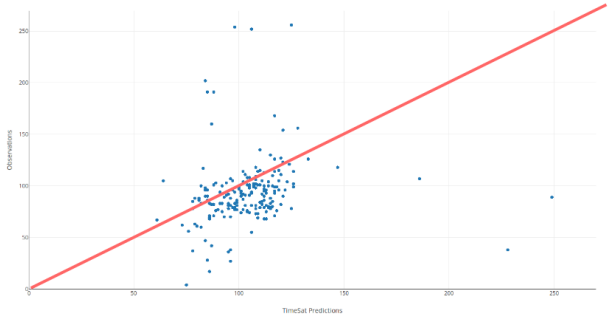


Experiment	MAE	R^2	Values Compared
NDVI - AG	23.0	0.01	195
NDVI - SG	24.6	0.02	194
NDVI - DL	24.5	0.02	194
EVI - AG	22.7	0.04	196
EVI - SG	21.3	0.1	195
EVI - DL	23.4	0.03	196

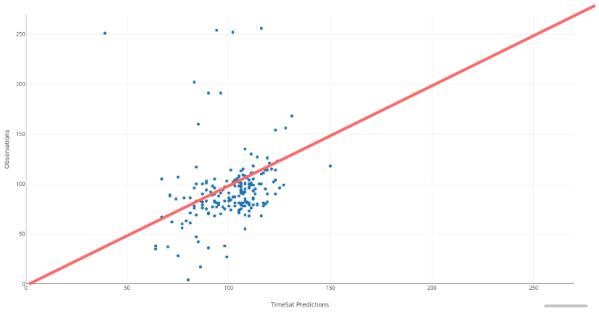
Table 7.6: Tulip Tree Experiments Results

Figure 7.18: Comparison on the **tulip tree** species (x-axis shows TimeSat predictions, y-axis ground observations, the red line is the identity line (x=y))

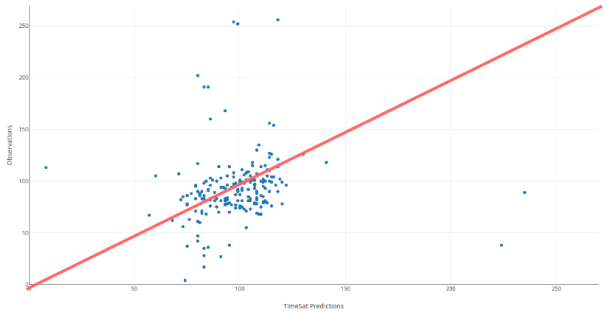
(a) NDVI- AG



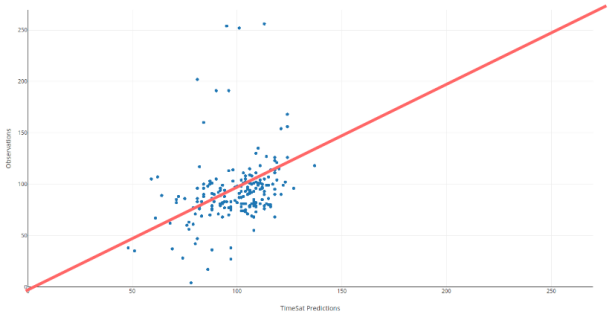
(b) EVI-AG



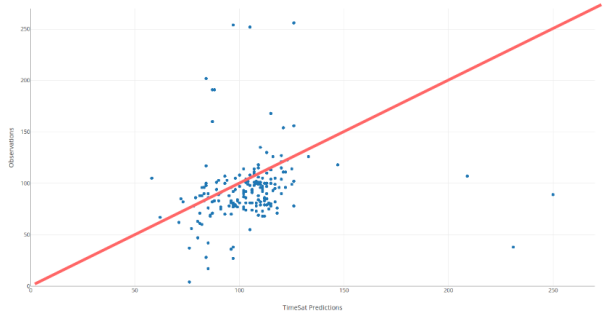
(c) NDVI-SG



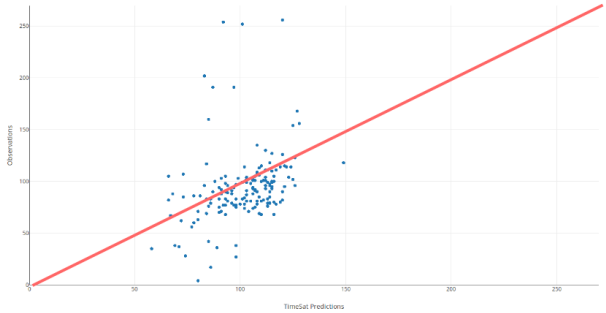
(d) EVI-SG



(e) NDVI-DL



(f) EVI-DL

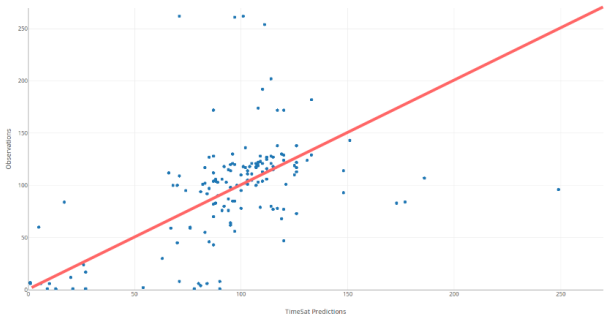


Experiment	MAE	R^2	Values Compared
NDVI - AG	32.4	0.09	148
NDVI - SG	31.0	0.19	151
NDVI - DL	29.7	0.24	149
EVI - AG	28.9	0.32	177
EVI - SG	29.4	0.31	168
EVI - DL	28.4	0.33	177

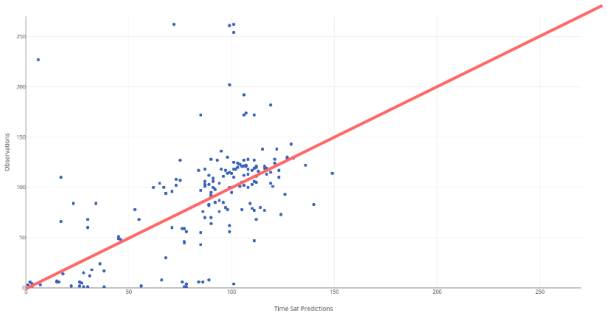
Table 7.7: Black Elderberry Experiments Results

Figure 7.19: Comparison on the **black elderberry** species (x-axis shows TimeSat predictions, y-axis ground observations, the red line is the identity line (x=y))

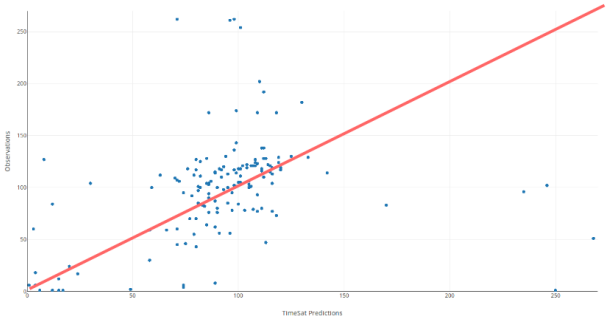
(a) NDVI- AG



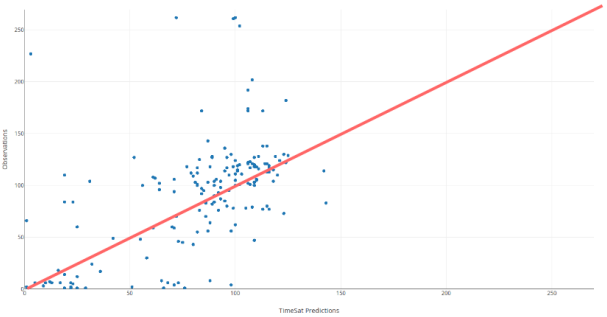
(b) EVI-AG



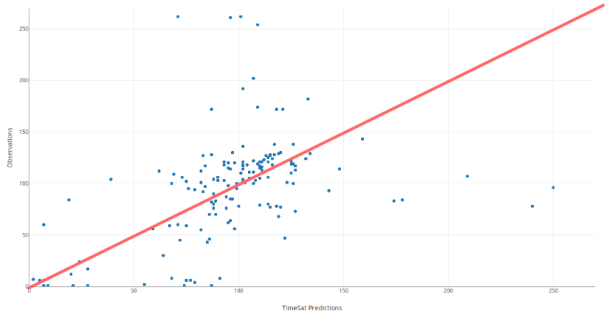
(c) NDVI-SG



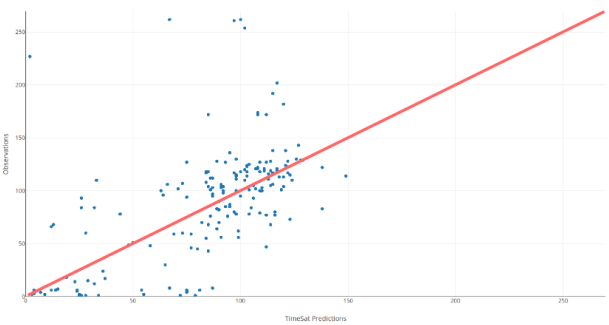
(d) EVI-SG



(e) NDVI-DL



(f) EVI-DL



7.4.4 Phenology Evaluation Summary

In this evaluation, we studied how the SOS varies geographically and temporally, but also determined where differences in the predicted SOS between VI or function choice can occur. Throughout most parts of USA, in central, Midwest and parts of the South, minimal differences are observed (0 to 10 days). However, there are substantial variations in other regions. We found biggest differences are observed between VIs. The NDVI has a higher overall spread between minimum and maximum predictions. We found in the West Coast, the NDVI shows later SOS values (0-20 days), but earlier predictions in the North America Deserts (0-30 days), with an exception being the Chihuahuan desert. Substantial differences are observed in part of Florida (30-80 days) and South-West North America(10-50 days) where NDVI models show later SOS. When compared per function the AG and DL functions show similar results, whereas the SG function shows earlier observations, more noticeable in the EVI experiment.

In parallel, we characterize the spatial and geographical patterns and found that they strongly depend on latitude coordinates. The SOS values advance northward, from values between the 30th and the 100th day in South USA to value between 100th to 180th day in North USA, whereas in most parts of USA (the South and the Mid-West) the SOS ranges between the 80th and 120th day. We further characterized the landscape on ecological regions. In the mountainous and desert region of South-West North America, the SOS is the latest, having a range of 100th to 230th day. The earlies SOS is observed in the North American Deserts with SOS values between 30 and 80. The Western, Central and Eastern Corn Belt Plants show relatively high SOS in the range 130 to 150. We note that the exact range varies between VI or function choice, but all models are consistent on ecological, and geographical patterns.

Similar spatial patterns are observed when studying the SD. Throughout the most of the study area, in regions with good seasonality and homogeneous plant cover (Central and Mid-West areas), the SD is reasonably consistent with minimal SD variations (5-10 days). Higher SD values are observed to the Central-West. Furthermore, in the North America Deserts and South-West North America, regions with high altitude, landscape heterogeneity, and sparse vegetation SD ranges from 15 to 80 days are observed. In such areas, the NDVI shows higher SD values if compared to the EVI. The AG and DL functions show similar and close results for both VIs, whereas the SG experiments show higher SD values, more noticeable in the NDVI experiment.

The high SD can be a result of a varying seasonal start, but also other factors can influence the results, such as the RS methodology being inaccurate in areas of low vegetation. Based on our results, and by manually inspecting the time-series we believe in the South-West indeed the seasonal start varies, however when extreme values are observed (>120 days), other factors can influence the accuracy of the results. Such factors can be wobbly, noisy or incomplete time series, minimal amplitude due to little or non-existent vegetation (i.e., desert and rock regions) or SOS derived on the two sides of our definition for 'early spring' (i.e., the 270th day).

The ground measurement comparisons show that remotely sensed phenology is a viable method for determining SOS. The MAE for all species studied is roughly between 20 and 30 days. The majority of predictions are less than 20 days of error, however, our generalized metrics are influenced by outliers, with more than 50 days of error.

The EVI experiments conclusively show better predictions, but the best function choice varies depending on the species studied. Generally, the AG and DL functions show similar results, whereas the SG slightly differs. This is an effect of the SG function over-fitting the time-series. All three functions were best performing for at least one of the species (SG -> tulip tree, DL -> black elderberry, AG -> red maple). When viewing our experiments on a species level, best results were observed with the tulip tree, (EVI-SG, MAE = 21.3). The best R^2 metric was observed with the black elderberry species (EVI, DL, $R^2 = 0.33$).

The black elderberry, red liliac, and flowering dogwood experiments show a good and roughly equal distribution of points between the two sides of the identity line (x=y). The red maple experiments, however, show more points on the right of the line, meaning in general TimeSat predicts later SOS than actually observed. This infers those species have earlier SOS than the rest of the vegetation in the area.

To better approximate the red maple observations it is possible to reduce the SOS predictions by setting the amplitude percentage to a lower value. This will not guarantee significantly better results, but in time-series with smaller slopes, differences of few days can be observed. However, throughout the rest of experiments amplitude percentage of 0.3 % showed consistent results. As another point of improvement, we note that setting a minimal cutoff amplitude value can remove otherwise meaningless results. The experiments were conducted on all the available data with no amplitude cutoff value. However, there can be bare rock areas; for example, the Grand Canyon, where there is no vegetation and therefore no seasonality. Experiments show a value of 0.1 may be too much as it invalidates areas with noticeable amplitudes, however, a value of 0.05 can be more appropriate.

There are several sources of uncertainties in our comparison with the ground measurements. As described in Section 1.2, the primary source of uncertainty comes from the inability to account for the within-pixel phenological variability of the 1km resolution. That is, the species monitored may or may not represent the general landscape phenology, a single point observation may or may not represent the overall pixel characteristics. Furthermore, the uncertainty of the exact occurrence of the SOS observation event (between the first "yes" and last "no" observations) can introduce additional inaccuracies. Also, the temporal resolution of our data is 10 days composites, introducing some noise. Finally, as described in Section 2.3.1 the PROBA-V and SPOT-VEGETATION sensors have non-significant radiometric variability, but it is uncertain how accurate they are for assessing phenology. With this study, we also evaluated their accuracy indirectly.

Chapter 8

Conclusion

8.1 Contributions

To attain our first objective, as described in Section 1.3, we successfully designed and implemented a distributed Big Data platform for phenology extraction by integrating the existing software programme TimeSat with Spark. The platform allowed us to extract the SOS phenology metric at a high spatio-temporal resolution at continental scale and furthermore was optimized for best performance. Our results show it will be able to scale with even higher resolutions and also be able to adjust accordingly with the increase/decrease of the problem size. The code for this project is freely shared and documented on the development platform GitHub, allowing for anyone interested to re-use it or be able to reproduce our results (GitHub, 2018).

To attain our second objective, we used the platform to conduct phenology research on North America, and more specifically the USA, for the period 1999 to 2017 at a 1-km resolution with 10-days composites. In addition, we calculated the VIs from the raw satellite radiometric data, extracted the SOS on a continental scale and produced SOS products with embedded geo-reference information for easy comparison with the ground measurements.

During the phenology evaluations, we showed that RS imagery is a reliable source for phenology computation, with MSE between 20 and 30 days. Best results were observed with the tulip tree (MAE = 21.3), whereas the best R^2 metric was observed with the black elderberry ($R^2 = 0.33$). However, all fitting functions showed best results for at least one spices. The EVI was the better performing VIs, since it showed consistently, better results (improved MAE of 2 to 4 days).

In addition, we demonstrate that SOS estimates vary within and among VIs and fitting functions. We showed the biggest differences are observed between VIs, where, in central and East regions there are minimal differences (0 up to 5 days), but in South-West regions as well as in Florida, the NDVI predicts up to 50 or even 65 days later SOS, whereas in the West deserts the SOS difference can be up to 30 days earlier. We note there are minimal differences between the AG and DL functions, whereas the SG function predicts slightly earlies SOS (up to 5 to 7 days). In parallel, we identified the geographical and ecological patterns and found the SOS strongly depends both on latitude coordinates and ecoregion.

8.2 Future Work

There are many directions for future work. The results of the RS derived phenology metrics do vary per ecological region (White et al., 2009), as also shown in this study, and land cover (Schwartz and Reed, 1999). To further evaluate the accuracy of RS phenology per such regions, the calculated SOS products can be compared with the ground measurements per land cover and per ecological region, and similarly, calculate the corresponding error metrics.

TimeSat does not provide goodness-of-fit statistics between the time-series and the fitted function, and there is no straightforward way to evaluate the curve fitting. However, it is possible to output the fitted time-series, compare it with the VI time-series and calculate, for example, the MSE. Such an evaluation can be performed per pixel and will give more formal and concrete insight on which regions are problematic for deriving phenology metrics.

Our analysis was performed on the start-of-the-growing-season (SOS) phenology metric. Rather than studying only one phenological marker, an analysis can be performed on the complete time-series, since TimeSat provides 12 more seasonal parameters (e.g. EOS). Therefore more insight can be derived for the whole vegetation cycle.

(Moulin et al., 1997) and (Justice et al., 1985) analyze vegetation phenology on global scale, but on coarse resolutions (1 lat & 1 long and 15 to 25 km, respectively). The platform will allow a phenology analysis to be performed at a similar continental or global scale, but with higher spatio-temporal resolutions. The global imaginary of the Sentinel-2 data at 10-meter resolution will allow us to launch phenology studies at unprecedented scales and resolutions. Therefore we can zoom-in in our analysis for the USA or perform it on a different continent (e.g. Europe).

8.3 Limitations and Recommendations

There were few limitations and undesirable effects involved in the project. We defined the boundary between early spring and late spring to be the 270th day of any year. In the studied region this caused no issues, however, it is possible to misclassify a later spring as early spring. An additional heuristic or technique is required to improve the accuracy of border case predictions, in regions where such observations can be frequent.

We were unable to take advantage of HDFS when deriving the SOS since TimeSat only can read files stored on the local POSIX file system. Also, increasing the total tile number imposes a slight performance penalty, since for each tile the files are separately loaded. Also, we found a performance penalty by using Minio and the *stfs* fuse to store and access the files.

The platform can be easily scaled with Spark, but for very high spatio-temporal resolutions we recommend implementing the TimeSat functionality within Spark, since we observed performance penalties during our platform scaling experiments, but also to take full advantage of the Spark RDDs and distributed storage. What is more, a big cluster infrastructure is required to handle fine-grained resolutions. Since we use Spark, one can simply add more nodes and be able to perform the same analysis at the cost of increasing the resources to reduce the processing time.

Our results highlight both the challenges and potential for integrating RS and ground observations. The USA-NPN network is making a step in the right direction, however, for a more complete and full-scale analysis, few more years are required for the ground measurement observations to increase enough to match (or at least approximate) the scale of the RS phenology. In our analysis, the observations were clustered on sites which gave at most 670 points for comparison (on a species level). Also, we recommend not only taking measurements at close proximities (observations close to each other) but also at increased distances in order to best approximate the RS scale.

Providing classifications or detection methods for filtering out outliers in the ground observation sets can be particularly useful. In addition, taking measurements frequently, or even daily, will reduce the uncertainty of the observed phenology state. Also, one can validate the accuracy of RS products in arid or semi-arid regions, where the SOS is problematic.

8.4 Learning Outcomes

There were several challenging aspects to this project, namely that there were areas in which I had little to no experience, and the phenology domain was certainly new to me. Although I had some experience working with Spark and Hadoop beforehand, cluster management, deployment, and optimization was something I was unfamiliar with. However, this gave me the opportunity to research and learn the technologies used (Vagrant, Minio, HDFS, Jupyter Notebooks, Spark, Geo Trellis, Scala), resulting in a great learning experience. In addition, working on an individual Computer Science project of such scale helped me build on my confidence and project management skills. Also, the planning and implementation was a real enjoyment, and moreover the evaluations uncovered a lot of exciting findings.

The phenological literature study gave me the fundamentals of phenology in general and remote sensing in particular and furthermore analyzing and computing the phenology results additionally increased my understanding in the field. The process of working with and analyzing data of such scale, pixel by pixel, significantly enhanced my analytical skills and showed me that to achieve a better understanding of the mechanisms, drilling down to the core fundamentals and specifics is required. Working with Big Data often requires working with the smallest possible chunk of data first.

Appendices

Appendix A

TimeSat Related

A.1 TimeSat Processing Steps

1. The first step is to prepare the input data. Both ASCII files or binary images are accepted. The binary files must be headerless (flat) binary files (two-dimensional spatial arrays). The values can be stored as 8-bit integers (0 - 255), 16-bit signed integer (-32767 to +32768), or 32-bit real. TimeSat is image oriented and does not consider geographical coordinate systems other than column and row. The columns and rows start from 1. To view image data the program "TSMimageview" can be used. To properly display the image one should know the image's dimensions, that is the number of rows and the number of columns. This module was only used during our initial exploratory analysis of the data.

2. With the module "TSM_GUI" a user can select pixels of interest, explore the quality of the input data and to select suitable fitting algorithms and settings for running the full data set. For the description of the algorithms and the available settings please refer to Section 3.1.1. We used this module to explore our data and easily tune the parameters, as it provides a mean of visualization between the time series and the fitting methods.

3. When the input parameters are decided we can use the "TSM_settings" tool to create and manage the settings-files. The text files are used to set the input parameters for the "TSM_process" which is the module to produce the seasonality data. Through the settings file we define the input parameters for the main processing module "TSF_process".

4. The core module is called "TSF_process". It is an executable Fortran program and TimeSat will start it from the command line. As input, the programme requires two parameters: path to the settings file and the number of processes to be use. Specifying more than 1 processor will run "TSF_process" in parallel. For example, if 7 parallel jobs are specified, the programme will divide the job into 7 equal parts, with 7 individual settings files. A script will be generated which will contain commands for starting the parallel jobs and merging back the results. See Section 3.1.2 for more information. The "TSF_process" will process data row by row. A user can choose whether to output the fitted or the original data as well. We are interested in the seasonality information which is delivered in binary (.tpa) files accompanied with index files (.ndx). For more information of the structure of the files and the seasonality parameters produced please refer to Figures 5.6a and 5.6b.

5. Several routines allow for generating output images or information from the "TSF_process" output files. For example, the "TSF_seas2img" allows for the creation of image files of seasonality and fitted data for a given time range. Using this module a user can produce binary images for a seasonality parameter of interest. However, one should know an interval for the season to output because the programme is capable of producing one season at a time. For pixels, where more than one season is found for the specified interval an error will be thrown. We decided to entirely rely on Spark for our image SOS generation and not use additional modules such as the "TSF_seas2img".

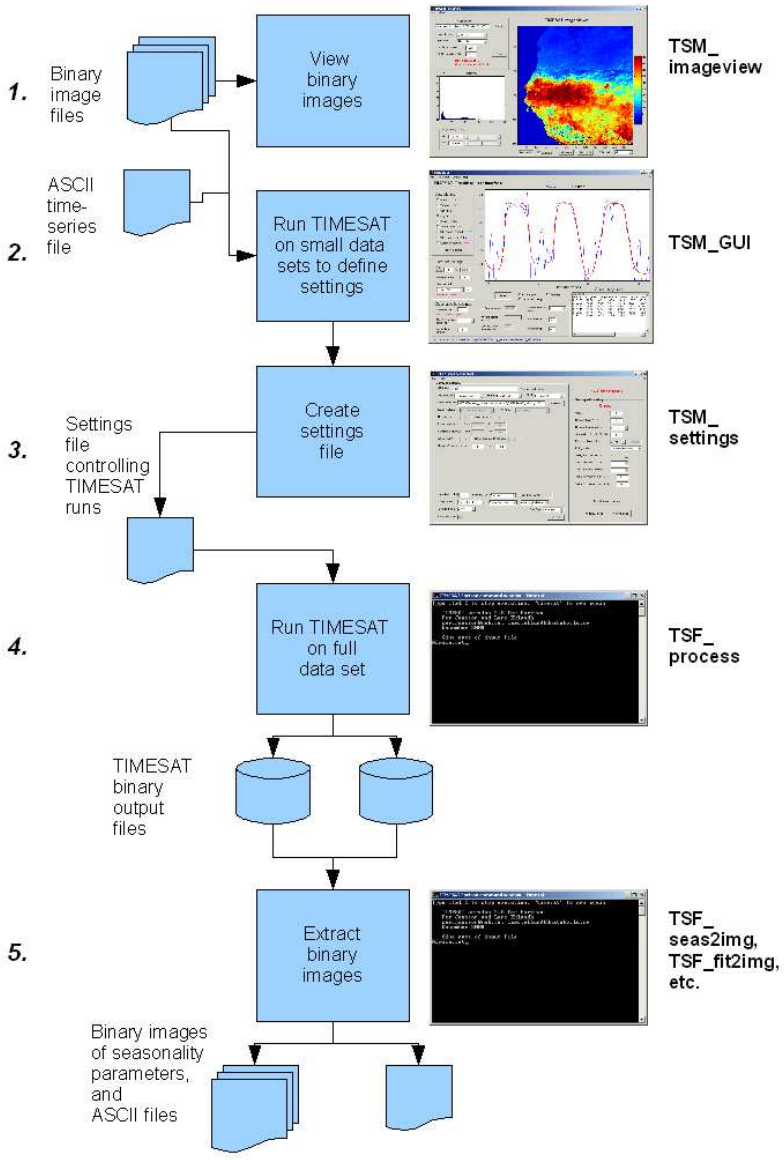


Figure A.1: TimeSat Processing Logic (Eklundha und Jönsson, 2017)

A.2 How we Ran TimeSat Standalone Mode

What we found is that the auto-generated TimeSat parallel bash script will not behave correctly if run from within another script, because it checks for running TimeSat processes

which execute in a *tty* terminal. As we are not using a terminal to launch the task, it will fail because the script does not pick the running processes. As a consequence, the script will clear the intermediate files before the processes finish. The solution was to change a single line in the provided parallel TimeSat script that will pick all running processes (not only running from tty terminal). Once all processes finish, we merged the results from each partition with TSF_merge. That will create a single 'tpa' file containing the seasonality information. A 16-bit signed integer binary image file was created with the "TSF_seas2img", for each season found. We decided to use only the module producing the seasonality parameters (TSF_process) and leave the rest to be handled by Spark.

A.3 High Standard Deviation Region

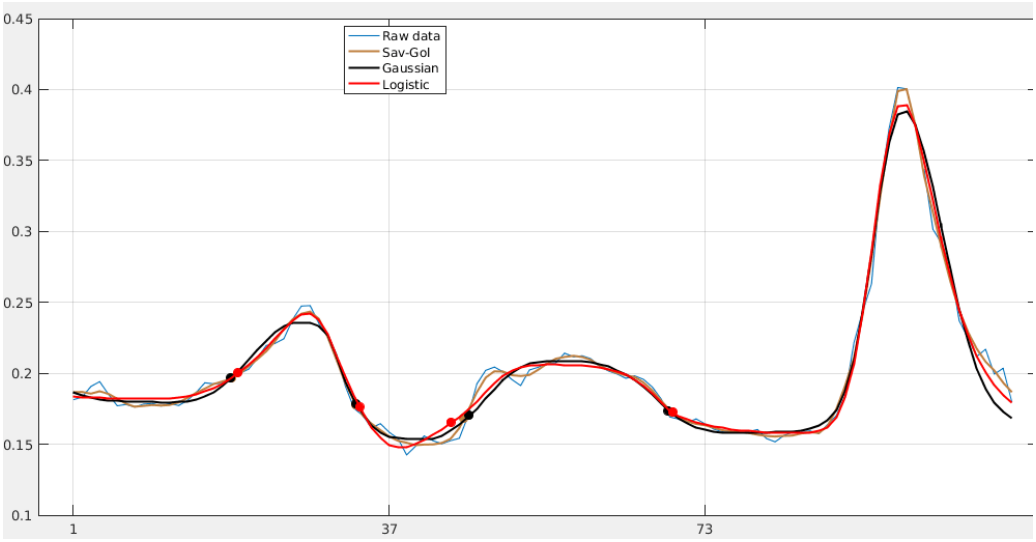


Figure A.2: Time-Series in Regions with High Standard Deviation, (Subset of 3 Years)

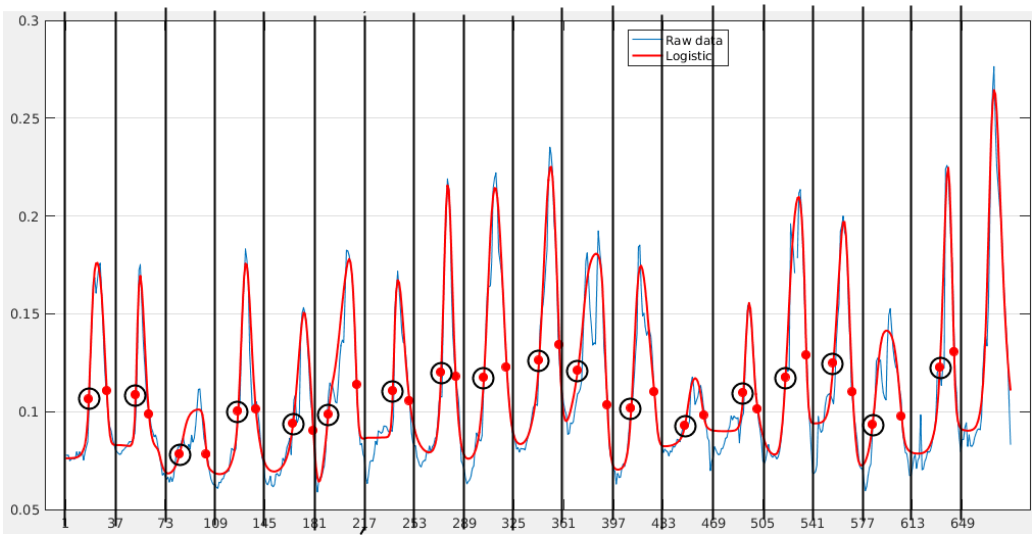


Figure A.3: Time-Series in Regions with High Standard Deviation (Full Temporal Range)

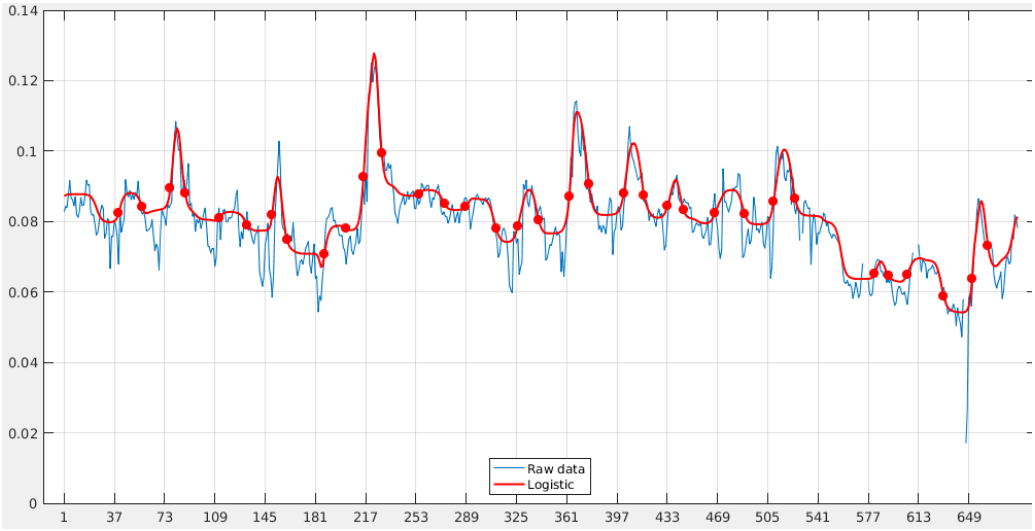


Figure A.4: Region with Minimal Seasonal Amplitude (0.03)

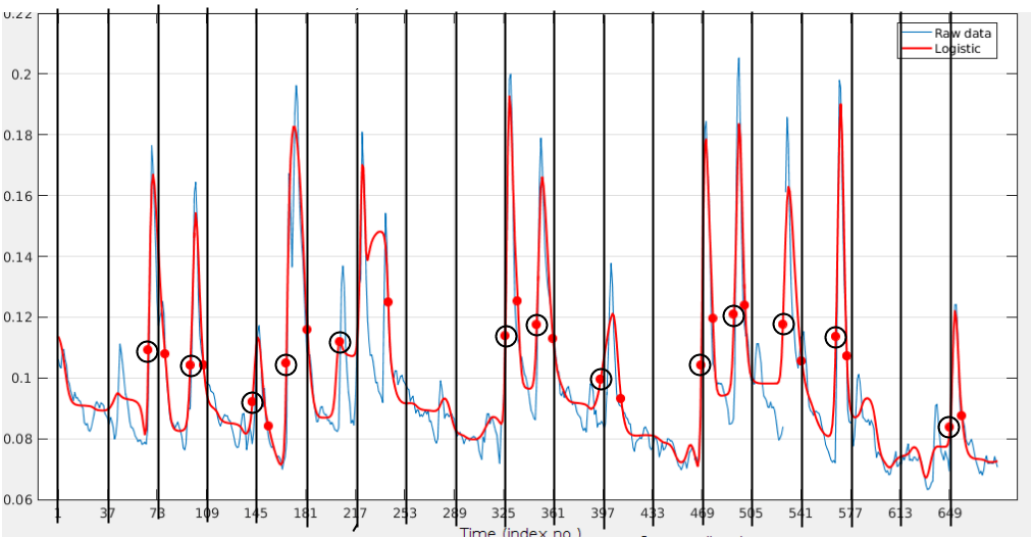


Figure A.5: Region on the two sides of the Early/Late Spring border (the 270th day) (Full Temporal Range)

Appendix B

Sen2-Agri: Products generated by the Vegetation Processor

Properties	Value
Available vegetation indicators	NDVI, LAI, NDVI metrics
Spatial extent	Global (not only crop area)
Spatial resolution	10 m
Temporal resolution	**NDVI and LAI**: 10 days (7 days with Sentinel-2A and -2B) **NDVI metrics**: at the middle of the season (start date) and at the end of the season (length and date of maximum growth)
Geometric accuracy	Same than L1C input data
Format	GeoTIFF raster images
Projection	UTM / WGS84
Metadata	XML file

Table B.1: Vegetation status indicators specifications

Appendix C

Remote Sensed Products

Sensor	Satellite	Overpass	Data Source	Data Record	Spatial R-n	Time Step
AVHRR	NOAA	Daily	USGS/EROS	1989-present	1 km	1-week, 2-week
AVHRR	NOAA	Daily	NASA Ecocast3	1982-2013	8 km	Twice monthly
ETM+	Landsat 7	16 days	USGS/EROS2	1999-present	30 m	by scene
Vegetation	SPOT	1-2 days	VITO	1999-present	100m, 300m, 1km	daily, 5-day, 10-day
Vegetation	SPOT + PROBA-V	1-2 days	Copernicus Global	1999-present	1 km	7-day 10-days 15-day
MODIS	Terra	1-2 days	LPDAAC5	2000-present	250 m, 500 m, 1 km	8-day , 16-day
MODIS	Aqua	1-2 days	LPDAAC5	2002-present	250 m, 500 m, 1 km	8-day, 16-day
eMODIS	Terra/ Aqua	1-2 days	USGS/EROS6	2000-present	250 m, 500 m, 1 km	7-day
AVHRR eMODIS	NOAA Terra/ Aqua	varying	NASA	1981-2014	5.6km	daily, 7-day, 15-day, monthly

Table C.1: Computer Vegetation Products

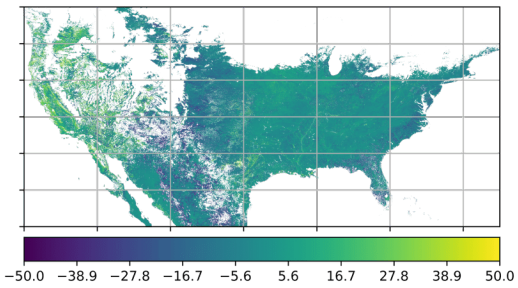
Appendix D

Phenology

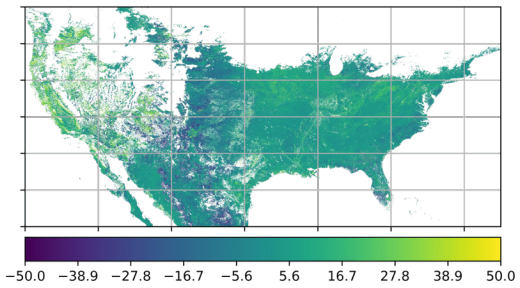
D.1 SOS Difference for the 2016 Year

Figure D.1: Difference between SOS experiments for the year 2016

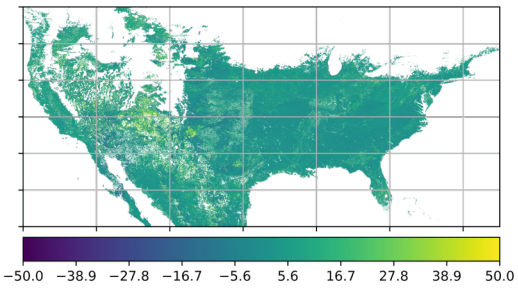
(a) EVI-AG minus NDVI- AG



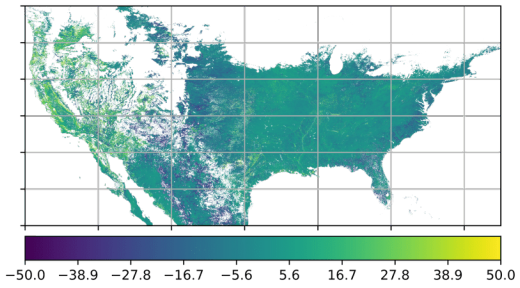
(b) EVI-AG minus NDVI-SG



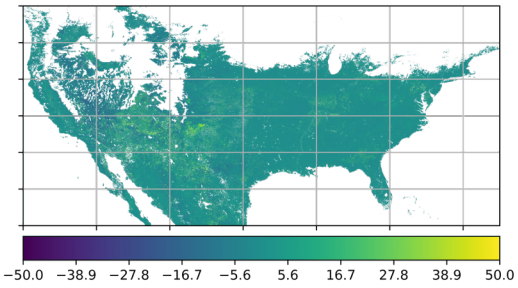
(c) EVI-AG minus EVI- SG



(d) EVI-AG minus NDVI - DL



(e) EVI-AG minus EVI - DL



D.2 Elevation Map

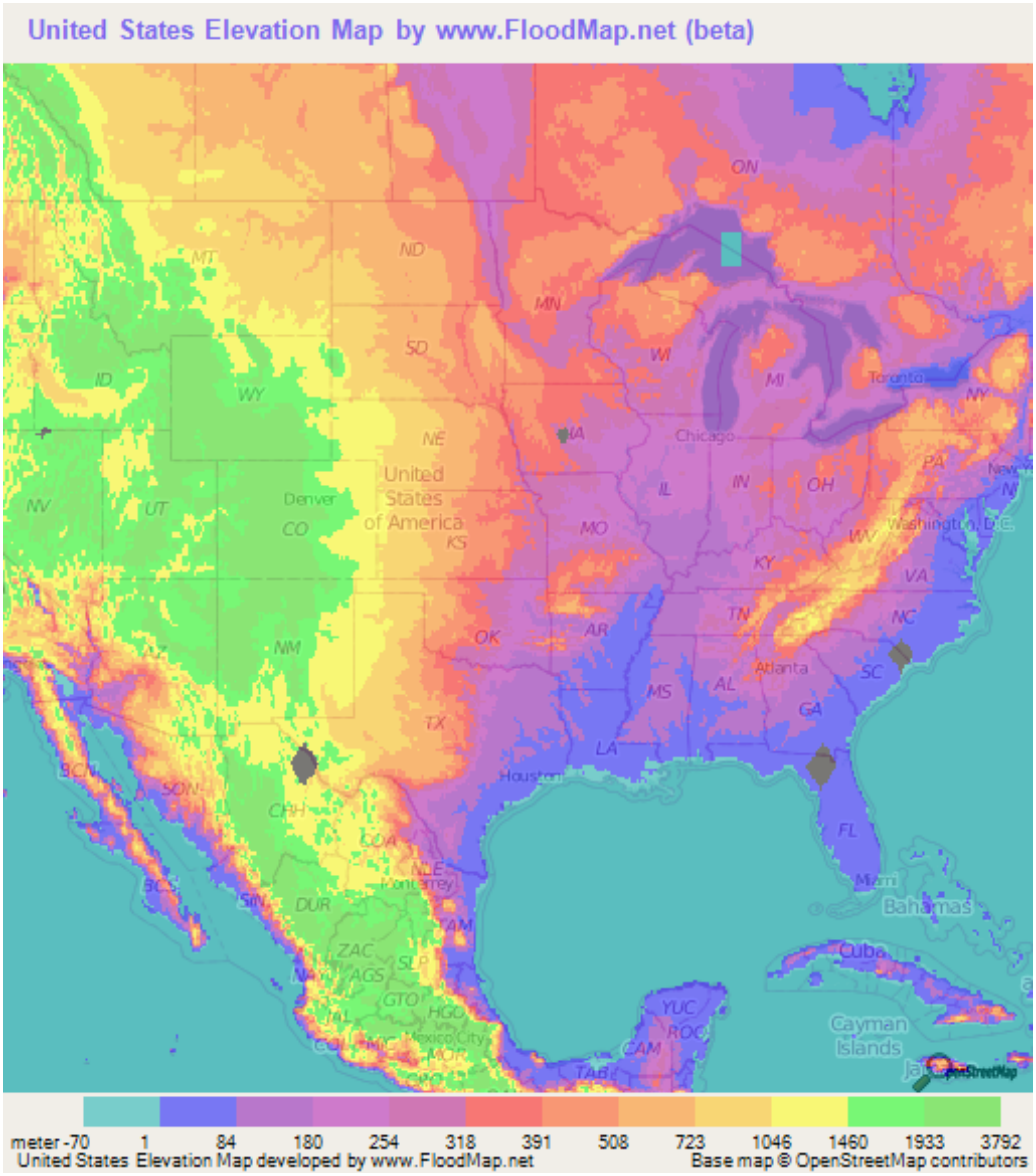


Figure D.2: USA Elevation Map

D.3 US Regions

.

Figure D.3: Further Separation on US Regions

D.4 Ecological Names

Figure D.4: Ecological Names: l1 names

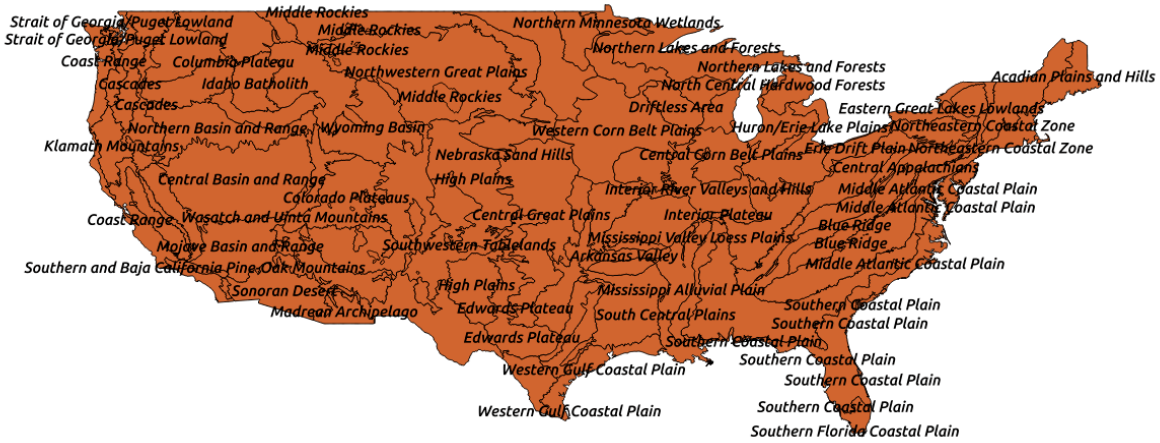


Figure D.5: Ecological Names: l3 names

Bibliography

C. Atzberger, A. Klisch, M. Mattiuzzi, and F. Vuolo. Phenological metrics derived over the european continent from ndvi3g data and modis time series. *Remote Sensing*, 6(1):257–284, 2013.

AWS. Registry of open data on aws (landsat-8 data). 2018. URL https://registry.opendata.aws/landsat-8/.

M. Azure. Optimize spark jobs, 2018. URL https://docs.microsoft.com/en-us/azure/hdinsight/spark/apache-spark-perf.

S. Azzali and M. Menenti. Mapping vegetation-soil-climate complexes in southern africa using temporal fourier analysis of noaa-avhrr ndvi data. *International Journal of Remote Sensing*, 21(5):973–996, 2000.

F.-W. Badeck, A. Bondeau, K. Böttcher, D. Doktor, W. Lucht, J. Schaber, and S. Sitch. Responses of spring phenology to climate change. *New Phytologist*, 162(2):295–309, 2004.

B. A. Bradley, R. W. Jacob, J. F. Hermance, and J. F. Mustard. A curve fitting procedure to derive inter-annual phenologies from time series of noisy satellite ndvi data. *Remote sensing of environment*, 106(2):137–145, 2007.

N. Burgess, J. Hales, E. Underwood, E. Dinerstein, D. Olson, I. Itoua, J. Schipper, T. Ricketts, K. Newman, et al. *Terrestrial ecoregions of Africa and Madagascar: a conservation assessment*. Island Press, 2004.

B. Butt, M. D. Turner, A. Singh, and L. Brottem. Use of modis ndvi to evaluate changing latitudinal gradients of rangeland phenology in sudano-sahelian west africa. *Remote Sensing of Environment*, 115(12):3367–3376, 2011.

M. Coffin and M. J. Saltzman. Statistical analysis of computational tests of algorithms and heuristics. *INFORMS Journal on Computing*, 12(1):24–44, 2000.

S. Consortium. Sentinel-2 for agriculture system - github, 2018. URL https://github.com/Sen2Agri/Sen2Agri-System. Accessed: 2018-15-03.

H. Q. Crick, C. Dudley, D. E. Glue, and D. L. Thomson. Uk birds are laying eggs earlier. *Nature*, 388(6642):526, 1997.

F. Dahms. Writing efficient spark jobs, 2018. URL http://fdahms.com/2015/10/04/writing-efficient-spark-jobs/1.

K. M. De Beurs and G. M. Henebry. Land surface phenology and temperature variation in the international geosphere–biosphere program high-latitude transects. *Global Change Biology*, 11(5):779–790, 2005.

J. Delegido, J. Verrelst, L. Alonso, and J. Moreno. Evaluation of sentinel-2 red-edge bands for empirical estimation of green lai and chlorophyll content. *Sensors*, 11(7):7063–7081, 2011.

G. R. Demarée. From "periodical observations" to "anthochronology" and "phenology"–the scientific debate between adolphe quetelet and charles morren on the origin of the word "phenology". *International journal of biometeorology*, 55(6):753–761, 2011.

S. Docs. Tuning spark. 2018. URL https://spark.apache.org/docs/1.2.0/tuning.html#data-locality.

D. Doktor, A. Bondeau, D. Koslowski, and F.-W. Badeck. Influence of heterogeneous landscapes on computed green-up dates based on daily avhrr ndvi observations. *Remote Sensing of Environment*, 113(12):2618–2632, 2009.

L. Eklundha and P. Jönsson. Timesat 3.3 with seasonal trend decomposition and parallel processing. 2017. URL http://web.nateko.lu.se/timesat/docs/TIMESAT33_SoftwareManual.pdf.

esa. Sentinel-2 spetial resolutons. 2018a. URL https://earth.esa.int/web/sentinel/user-guides/sentinel-2-msi/resolutions/spatial.

esa. Sentinel-2 products. 2018b. URL https://sentinel.esa.int/web/sentinel/missions/sentinel-2/data-products.

S. Farooqui. Advanced apache spark training, 2015. URL https://www.youtube.com/watch?v=7ooZ4S7Ay6Y.

A. Fitter and R. Fitter. Rapid changes in flowering time in british plants. *Science*, 296(5573):1689–1691, 2002.

GISExchange. Evi range. 2016. URL https://gis.stackexchange.com/questions/90655/formula-for-enhanced-vegetation-index-evi-in-erdas-modelbuilder.

GISGeography. What is ndvi(normalized difference vegetation index), 2018. URL https://gisgeography.com/ndvi-normalized-difference-vegetation-index/.

gitHub. S3fs codebase for cache. 2018a. URL https://github.com/s3fs-fuse/s3fs-fuse/blob/master/src/cache.cpp.

gitHub. S3fs github documentation. 2018b. URL https://github.com/s3fs-fuse/s3fs-fuse/wiki/Fuse-Over-Amazon.

GitHub. Big data platform to derive large scale phenological products, 2018. URL https://github.com/phenology/platform-for-pheno-products/.

jaceklaskowsk. Spark transformations. 2018. URL https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-rdd-transformations.html.

jaceklaskowski. Data locality / placement. 2016a. URL https://jaceklaskowski.gitbooks.io/mastering-apache-spark/content/spark-data-locality.html.

jaceklaskowski. Transformations, 2016b. URL https://jaceklaskowski.gitbooks.io/mastering-apache-spark/spark-rdd-transformations.html.

J. R. Jensen. *Remote sensing of the environment: An earth resource perspective 2/e*. Pearson Education India, 2009.

L. Ji and J. F. Brown. Effect of noaa satellite orbital drift on avhrr-derived phenological metrics. *International Journal of Applied Earth Observation and Geoinformation*, 62:215–223, 2017.

P. Jönsson and L. Eklundh. Timesat—a program for analyzing time-series of satellite sensor data. *Computers & Geosciences*, 30(8):833–845, 2004.

C. O. Justice, J. Townshend, B. Holben, and e. C. Tucker. Analysis of the phenology of global vegetation using meteorological satellite data. *International Journal of Remote Sensing*, 6 (8):1271–1318, 1985.

S. Klosterman, K. Hufkens, J. Gray, E. Melaas, O. Sonnentag, I. Lavine, L. Mitchell, R. Norman, M. Friedl, and A. Richardson. Evaluating remote sensing of deciduous forest phenology at multiple spatial scales using phenocam imagery. 2014.

H. J. Kramer. *Observation of the Earth and its Environment: Survey of Missions and Sensors*. Springer Science & Business Media, 2002.

K. LaLiberte. Bloom time chart for spring and summer bulbs, 2016. URL https://blog.longfield-gardens.com/bloom-time-chart-for-spring-and-summer-bulbs/.

Y. Le Page, D. Oom, J. Silva, P. Jönsson, and J. Pereira. Seasonality of vegetation fires as modified by human action: observing the deviation from eco-climatic fire regimes. *Global Ecology and Biogeography*, 19(4):575–588, 2010.

L. Y. L. C. Z. K. LI Hongjun, ZHENG Li. Comparison of ndvi and evi based on eos/modis data. *PROGRESS IN GEOGRAPHY*, 26(1):26, 2007. doi: 10.11820/dlkxjz.2007.01.003. URL http://www.progressingeography.com/EN/abstract/article_12120.shtml.

L. Liang, M. D. Schwartz, and S. Fei. Validating satellite phenology through intensive ground observation and landscape scaling in a mixed seasonal forest. *Remote Sensing of Environment*, 115(1):143–157, 2011.

H. Q. Liu and A. Huete. A feedback based modification of the ndvi to minimize canopy background and atmospheric noise. *IEEE Transactions on Geoscience and Remote Sensing*, 33(2):457–465, 1995.

MACCS. Maccs, 2018. URL https://logiciels.cnes.fr/en/content/maccs. Accessed: 2018-18-03.

A. Mack, E. R. Choffnes, et al. *The Influence of Global Environmental Change on Infectious Disease Dynamics: Workshop Summary*. National Academies Press, 2014.

Medium. Mae and rmse—which metric is better?, 2016. URL https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d.

A. M. Melesse, Q. Weng, P. S. Thenkabail, and G. B. Senay. Remote sensing sensors and applications in environmental resources mapping and modelling. *Sensors*, 7(12):3209–3241, 2007.

S. Moulin, L. Kergoat, N. Viovy, and G. Dedieu. Global-scale assessment of vegetation phenology using noaa/avhrr satellite measurements. *Journal of Climate*, 10(6):1154–1170, 1997.

U. NASS. Usual planting and harvesting dates for us field crops. *Agricultural Handbook*, 628, 1997.

K. Nguyen. Understand rdd operations: Transformations and actions, 2014. URL https://trongkhoanguyen.com/spark/understand-rdd-operations-transformations-and-actions/.

E. Observatory. Measuring vegetation ndvi and evi. URL https://earthobservatory.nasa.gov/Features/MeasuringVegetation/measuring_vegetation_4.php. Accessed: 2018-10-03.

A. Or. Understanding your apache spark application through visualization, 2015. URL https://databricks.com/blog/2015/06/22/understanding-your-spark-application-through-visualization.html.

J. M. Peña-Barragán, M. K. Ngugi, R. E. Plant, and J. Six. Object-based crop identification using multiple vegetation indices, textural features and crop phenology. *Remote Sensing of Environment*, 115(6):1301–1316, 2011.

S. Piao, J. Fang, L. Zhou, P. Ciais, and B. Zhu. Variations in satellite-derived phenology in china's temperate vegetation. *Global change biology*, 12(4):672–685, 2006.

R. B. Primack and A. J. Miller-Rushing. Uncovering, collecting, and analyzing records to investigate the ecological impacts of climate change: a template from thoreau's concord. *Bioscience*, 62(2):170–181, 2012.

QGIS. Qgis raster calculator. 2018a. URL https://docs.qgis.org/2.8/en/docs/user_manual/working_with_raster/raster_calculator.html.

QGIS. Qgis zonal statistics plugin. 2018b. URL https://docs.qgis.org/2.18/en/docs/user_manual/plugins/plugins_zonal_statistics.html.

B. C. Reed and K. Sayler. A method for deriving phenological metrics from satellite data, colorado 1991–1995. In *Proceedings of the Northern Great Plains Regional Workshop on Climate Change and Climate Variability*, pages 47–55, 1998.

A. D. Richardson, K. Hufkens, T. Milliman, D. M. Aubrecht, M. Chen, J. M. Gray, M. R. Johnston, T. F. Keenan, S. T. Klosterman, M. Kosmala, et al. Tracking vegetation phenology across diverse north american biomes using phenocam imagery. *Scientific data*, 5:180028, 2018.

K. M. Robbirt, A. J. Davy, M. J. Hutchings, and D. L. Roberts. Validation of biological collections as a source of phenological data for use in climate change studies: a case study with the orchid ophrys sphegodes. *Journal of Ecology*, 99(1):235–241, 2011.

A. H. Rosemartin, E. G. Denny, K. L. Gerst, R. L. Marsh, E. E. Posthumus, T. M. Crimmins, and J. Weltzin. Usa national phenology network observational data documentation. 2018.

S. Ryza. How-to: Tune your apache spark jobs (part 2), 2018. URL https://blog.cloudera.com/blog/2015/03/how-to-tune-your-apache-spark-jobs-part-2/.

Scala. Scala library for executing external commands. 2018. URL https://www.scala-lang.org/api/current/scala/sys/process/index.html.

M. D. Schwartz. Phenology: an integrative environmental science. 2003.

M. D. Schwartz and J. M. Hanes. Intercomparing multiple measures of the onset of spring in eastern north america. *International Journal of Climatology*, 30(11):1614–1626, 2010.

M. D. Schwartz and B. C. Reed. Surface phenology and satellite sensor-derived onset of greenness: an initial comparison. *International Journal of Remote Sensing*, 20(17):3451–3457, 1999.

M. D. Schwartz, J. L. Betancourt, and J. F. Weltzin. From caprio's lilacs to the usa national phenology network. *Frontiers in Ecology and the Environment*, 10(6):324–327, 2012.

J. S. Sedinger and D. G. Raveling. Timing of nesting by canada geese in relation to the phenology and availability of their food plants. *The Journal of Animal Ecology*, pages 1083–1102, 1986.

Sen2Agri. Sentinel-2 agriculture - software user manual, 2018. URL http://www.esa-sen2agri.org/wp-content/uploads/docs/Sen2Agri%20Software%20User%20Manual%202.2.pdf. Accessed: 2018-15-03.

C. G. L. Service. Top of canopy reflectances, 2018. URL https://land.copernicus.eu/global/products/toc-r.

D. A. Slayback, J. E. Pinzon, S. O. Los, and C. J. Tucker. Northern hemisphere photosynthetic trends 1982–99. *Global Change Biology*, 9(1):1–15, 2003.

K. Soudani, G. Le Maire, E. Dufrêne, C. François, N. Delpierre, E. Ulrich, and S. Cecchini. Evaluation of the onset of green-up in temperate deciduous broadleaf forests derived from moderate resolution imaging spectroradiometer (modis) data. *Remote Sensing of Environment*, 112(5):2643–2655, 2008.

Spark. Tuning spark, 2018. URL https://spark.apache.org/docs/latest/tuning.html.

tutorialspoint. Hadoop - hdfs overview. 2018. URL https://www.tutorialspoint.com/hadoop/hadoop_hdfs_overview.htm.

usanpn. Why phenology, 2018. URL https://www.usanpn.org/about/why-phenology. Accessed: 2018-10-04.

usgs. Ndvi, the foundation for remote sensing phenology, 2000. URL https://phenology.cr.usgs.gov/ndvi_foundation.php. Accessed: 2018-10-01.

USGS. Landsat 8, 2016. URL https://landsat.usgs.gov/landsat-8. Accessed: 2018-04-04.

J. Verbesselt, P. Jonsson, S. Lhermitte, J. Van Aardt, and P. Coppin. Evaluating satellite and climate data-derived indices as fire risk indicators in savanna ecosystems. *IEEE Transactions on Geoscience and Remote Sensing*, 44(6):1622–1632, 2006.

A. Verhegghen, S. Bontemps, and P. Defourny. A global ndvi and evi reference data set for land-surface phenology using 13 years of daily spot-vegetation observations. *International journal of remote sensing*, 35(7):2440–2471, 2014.

Vito. Vgt1 vs vgt2, 2016. URL http://www.vito-eodata.be/PDF/image/faq_help/Faq.html#PROBA-V.

Vito. Quality assessment report toc, 2018a. URL https://land.copernicus.eu/global/sites/cgls.vito.be/files/products/CGLOPS1_QAR_TOCR1km-PROBAV-V1.5_I2.10.pdfV.

Vito. Spirits tutorial, 2018b. URL http://spirits.jrc.ec.europa.eu/files/SpiritsTutorial.pdf. Accessed: 2018-08-03.

VITO. S10 radiometric - continental extracts, 2018a. URL https://www.vito-eodata.be/PDF/portal/Application.html#Browse;Root=2;Time=NORMAL,NORMAL,-1,,,-1,,.

VITO. Vito's earth observation collection catalogue. 2018b. URL https://www.vito-eodata.be/collections/srv/eng/main.home.

G.-R. Walther, E. Post, P. Convey, A. Menzel, C. Parmesan, T. J. Beebee, J.-M. Fromentin, O. Hoegh-Guldberg, and F. Bairlein. Ecological responses to recent climate change. *Nature*, 416(6879):389, 2002.

M. A. White, K. M. de Beurs, K. Didan, D. W. Inouye, A. D. Richardson, O. P. Jensen, J. O'KEEFE, G. Zhang, R. R. Nemani, W. J. van Leeuwen, et al. Intercomparison, interpretation, and assessment of spring phenology in north america estimated from remote sensing for 1982–2006. *Global Change Biology*, 15(10):2335–2359, 2009.

WHO. Phenology and human health:allergetic disorders. 2003. URL http://apps.who.int/iris/bitstream/handle/10665/107479/e79129.pdf.

Wikipedia. Decimal degrees of latitude longitude coordinates, 2018. URL https://en.wikipedia.org/wiki/Decimal_degrees.

Wine. Run windows applications on linux, 2018. URL https://www.winehq.org/. Accessed: 2018-04-03.

wrcc. Climat of arizona. 2014. URL https://wrcc.dri.edu/narratives/ARIZONA.htm.

X. Zhang, M. A. Friedl, and C. B. Schaaf. Sensitivity of vegetation phenology detection to the temporal resolution of satellite data. *International Journal of Remote Sensing*, 30(8): 2061–2074, 2009.

R. Zurita-Milla, R. Goncalves, E. Izquierdo-Verdiguier, and F. Ostermann. Exploring vegetation phenology at continental scales: Linking temperature-based indices and land surface phenological metrics. 2017.

Figures Citations

[Figure1] [Bharadwaj 2016] Bharadwaj, Suman: *What is the equivalent of MapReduce reducer in spark?* 2016. – URL https://www.quora.com/What-is-the-equivalent-of-MapReduce-reducer-in-spark

[Figure2] [Eklundha und Jönsson 2017] Eklundha, Lars ; Jönsson, Per: TIMESAT 3.3 with seasonal trend decomposition and parallel processing. (2017). – URL http://web.nateko.lu.se/timesat/docs/TIMESAT33_SoftwareManual.pdf

[Figure3] [eoPortal 2016] eoPortal: PROBA-V (Project for On-Board Autonomy - Vegetation). (2016). – URL https://directory.eoportal.org/web/eoportal/satellite-missions/p/proba-v

[Figure4] [GISGeography 2018] GISGeography: *What is NDVI(Normalized Difference Vegetation Index).* 2018. – URL https://gisgeography.com/ndvi-normalized-difference-vegetation-index/

[Figure5] [Nguyen 2014] Nguyen, Khoa: *A gentle introduction to Apache Spark.* 2014. – URL https://trongkhoanguyen.com/spark/a-gentle-introduction-to-apache-spark/

[Figure6] [quangnhathoang 2014] quangnhathoang: Spark job submission breakdown. (2014). – URL https://hxquangnhat.com/2015/04/03/arch-spark-job-submission-breakdown/

[Figure7] [Richardson u. a. 2018] Richardson, Andrew D. ; Hufkens, Koen ; Milliman, Tom ; Aubrecht, Donald M. ; Chen, Min ; Gray, Josh M. ; Johnston, Miriam R. ; Keenan, Trevor F. ; Klosterman, Stephen T. ; Kosmala, Margaret u. a.: Tracking vegetation phenology across diverse North American biomes using PhenoCam imagery. In: *Scientific data* 5 (2018), S. 180028

[Figure8] [Sen2Agri 2018] Sen2Agri: *Sentinel-2 Agriculture - Software User Manual.* 2018. – URL http://www.esa-sen2agri.org/wp-content/uploads/docs/Sen2Agri%20Software%20User%20Manual%202.2.pdf. – Accessed: 2018-15-03

[Figure9] [Service 2018] Service, Copernicous Global L.: *Top Of Canopy Reflectances.* 2018. – URL https://land.copernicus.eu/global/products/toc-r

[Figure10] [Vito 2018] Vito: *Quality Assessment Report TOC.* 2018. – URL https://land.copernicus.eu/global/sites/cgls.vito.be/files/products/CGLOPS1_QAR_TOCR1km-PROBAV-V1.5_I2.10.pdfV