Student thesis series INES nr 387

Developing a web-based system to visualize vegetation trends by a nonlinear regression algorithm

Yufei Wei

2016
Department of
Physical Geography and Ecosystem Science
Lund University
Sölvegatan 12
S-223 62 Lund
Sweden

Yufei Wei (2016).
Developing a web-based system to visualize vegetation trends by a nonlinear regression algorithm
Master degree thesis, 30 credits in *Geomatics*
Department of Physical Geography and Ecosystem Science, Lund University

Level: Master of Science (MSc)

Course duration: *January* 2016 until *June* 2016

Disclaimer

Developing a web-based system to visualize vegetation trends by a nonlinear regression algorithm

---

Yufei Wei

Master thesis, 30 credits, in *Geomatics*

Supervisors:
Lars Eklundh, Sadegh Jamali,
Roger Groth and Ali Mansourian
Department of Physical Geography and Ecosystem Science,
Lund University

Exam committee:
Micael Runnström and Per-Ola Olsson
Department of Physical Geography and Ecosystem Science,
Lund University

Abstract

Comparing with traditional linear regression methods that used to monitor vegetation trends, a nonlinear regression algorithm (PolyTrend) developed by Jamali et al. (2014) can provide more accurate information of vegetation trends by fitting a polynomial line with a degree of up to three for ASCII-formed time-series NDVI (Normalized Difference Vegetation Index**)** dataset of a single pixel. To extend the ability of the PolyTrend algorithm for processing time-series NDVI satellite imagery and to increase its accessibility, a web-based system for visualizing vegetation trends by the PolyTrend algorithm has been developed. The PolyTrend web-based system allows users to define the value of statistical significance of the PolyTrend algorithm, the nominal range of the input data, and the range of desired NDVI input to be processed. It applies the PolyTrend algorithm to each pixel of the uploaded time-series NDVI satellite imagery dataset. It returns the types of vegetation changes, the slope of the changes of NDVI values in the whole time span, and whether the net change of NDVI increases or decreases during this period, in the forms of ASCII files (i.e. text files) and binary files (i.e. images). By refining the existing PolyTrend algorithm written in MATLAB and embedding it in a web environment, the PolyTrend web-based system has proved its ability in monitoring global vegetation trends using raw time-series NDVI satellite imagery.

Keywords: Physical Geography and Ecosystem Analysis, NDVI, Vegetation Trends, Nonlinear Regression Algorithm, PolyTrend, Web Development, Django, MATLAB, Python.

Acknowledgements

I am very grateful to all of my supervisors, Ali Mansourian, who supported me with his knowledge and patience; Roger Groth, who provided me with help and suggestions on technical problems; Sadegh Jamali, who gave me help with regard to the PolyTrend algorithm and the time-series NDVI dataset, and Lars Eklundh, who corrected my writing and encouraged me a lot.

I would like to thank all participants of the survey.

I would like to thank Haiyi, Xilu, and Wenxiu, for your understanding and supports during the past six years.

I would like to thank Jane and Istvan, who helped me a lot after I started my study here.

At last, I am very grateful to my family who always support me.

Contents

Abbreviations

Ajax	Asynchronous JavaScript and XML
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
AVHRR	Advanced Very High Resolution Radiometer
BFAST	Breaks For Additive Season and Trend project
CC	Creative Commons
CSS	Cascading Style Sheets
ETM+	Landsat 7's Enhanced Thematic Mapper Plus
GDAL	Geospatial Data Abstraction Library
GIMMS-NDVI	Global Inventory Modelling and Mapping Studies- NDVI
GIS	Geographical Information System
GPLv2	GNU General Public License v2.0
GPLv3	GNU General Public License v3.0
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IE	Internet Explorer
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
JS	JavaScript
LandTrendr	Landsat-based Detection of Trends in Disturbance and Recovery
MATLAB API	MATLAB Engine API for Python
MATLAB SDK	MATLAB Compiler SDK
MODIS	Moderate Resolution Imaging Spectroradiometer
NDVI	Normalized Difference Vegetation Index
OGC	Open Geospatial Consortium
OLS	Ordinary Least Squares
OS	Operating System
OSS	Open Source Software
RDBMS	Relational Database Management System
RS	Remote Sensing
SDK	Software Development Kit
SVG	Scalable Vector Graphics
WCS	Web Coverage Service
WFS	Web Feature Service
WMS	Web Map Service
WWW	World Wide Web
W3C	World Wide Web Consortium
XML	eXtensible Markup Language

1. Introduction

Vegetation is an important component of ecosystems (e.g. Heinsch et al. 2006; Zhou et al. 2015). For example, vegetation presence has an impact on deciding the quantity of rainwater absorbed by the soil and preventing land erosion and degradation (Javier et al. 2015); vegetation photosynthesis is a measurement of quantifying carbon sinks and sources in carbon cycle (Pang et al. 2016). Consequently, monitoring vegetation trends is a convenient way of describing and predicting the development of ecosystem (Chapin et al. 1993; Fensholt et al. 2015). Furthermore, nowadays, because of the development of remote sensing, monitoring vegetation trends can be accomplished by analysing time series of satellite imagery.

For analysing vegetation trends in satellite imagery, the normalized difference vegetation index (NDVI) can be used to estimate vegetation growth status and biomass due to its sensitivity to the appearance and the density of vegetation (Herrmann et al. 2005; Liu et al. 2013). It is also one of the most prevalent vegetation indices (e.g. Liu et al. 2013; Tsai et al. 2016). By measuring changes of NDVI values derived from daily or regular interval satellite imagery, the temporal and the spatial dynamics of vegetation can be known to scientists and other users (Weier and Herring 2000).

Linear regression methods, such as Ordinary Least Squares (OLS) and the Theil-Sen estimator, have been widely used to detect changes of NDVI values in a time span (e.g. Hou et al. 2015; Liu et al. 2015a). This is because these methods are easy to implement and the resulting data is easy to interpret (Chambers and Dinsmore 2014). However, they are insufficient in trend analysis due to their characteristics. For example, linear regression methods use a straight line to describe the relationship between independent variables and dependent variables whereas a straight-line relationship cannot correspond to all real-world situations (Chambers and Dinsmore 2014). A study carried out by Liu et al. (2015b) showed the insufficiency of linear trend analysis due to the high level of species diversity and the complex food chains in forests. Furthermore, short-term variations of vegetation changes can be undetected by linear regression methods such as OLS (de Jong et al. 2012; Jamali et al. 2014).

To solve the shortcoming of linear regression methods of using a straight-line relationship to evaluate all natural phenomena, Jamali et al. (2014) has developed a nonlinear regression algorithm (PolyTrend) for monitoring vegetation trends. By stacking NDVI values of a single pixel according to their chronological order, this algorithm is capable of describing the character of changes of NDVI values by a cubic polynomial. To the knowledge of the author, except PolyTrend, currently there is no published nonlinear regression algorithm used to monitor vegetation trends derived from satellite imagery although there are other software packages for detecting vegetation trends, such as BFAST (Breaks For Additive Season and Trend project) (BFAST 2016) and LandTrendr (Landsat-based Detection of Trends in Disturbance and Recovery) (LandTrendr 2016), which process the non-linearity trends by dividing them into portions (e.g. Jamali et al. 2014). BFAST identifies the changes in vegetation trends by finding

candidate alignment locations (Homer et al. 2009) while LandTrendr detects the critical characteristics of the trends by straight line sections (Kennedy et al. 2010).

PolyTrend was originally programmed in MATLAB (Jamali et al. 2014). By accepting a vector of data containing NDVI values for a single pixel during a certain time period and a value of statistical significance, it will return the types of vegetation changes, the slope of the changes of NDVI values in the whole time span and whether the net change of NDVI increases or decreases during this period.

Nevertheless, the existence of two problems prevents PolyTrend from becoming a widely-used algorithm to monitor vegetation trends. The first problem is this algorithm is confined to process text-formed NDVI values of a single pixel, thus preventing its application to most commonly used satellite imagery, which are normally in binary format. If this algorithm can return results in the form of imagery and ASCII (American Standard Code for Information Interchange) files, then the resulting data can, not only acts as graphical results, but also be reused from the perspective of generating new raster imagery and revealing statistical results. The second problem is this algorithm has not been disseminated widely. A possible solution is embedding PolyTrend in a web-based system since the Internet is a good way of increasing the accessibility of this algorithm. In addition, if commercial software such as MATLAB (MathWorks 2016d) can situate in a web-based system, then its constrained accessibility caused by its high price can be solved. Furthermore, a web-based system does not require upgrades carried out by users, which ensures only the latest version of the PolyTrend web-based system is available to users at any time.

For solving the two problems above two modifications should be made. The first one is to modify the PolyTrend algorithm so that it can compute vegetation trends at image level. The second one is to develop a web-based system to let the PolyTrend algorithm (image level) be widely accessible.

For ensuring the modifications can be carried out successfully, a set of requirements should be formulated at the beginning. In general, these requirements can be classified into two categories: functional requirements and non-functional requirements. Functional requirements indicate what the system should do (Eriksson 2012). Non-functional requirements regulate all the remaining parts which are not indicated by the former (Eriksson 2012). As a part of non-functional requirements (e.g. Cohn 2008; Eriksson 2012), the usability of the PolyTrend web-based system will also be assessed.

1.1 Aims

In order to solve the problems of preventing PolyTrend from becoming a prevalent algorithm the aims of this project are

1. modify the existing PolyTrend algorithm for processing and visualizing satellite imagery containing NDVI values,

2. develop a web-based system to visualize the results produced by the PolyTrend algorithm that works for full images, and
3. evaluate this system by a time-series NDVI satellite imagery of a pilot area and evaluate this system in terms of requirements. The requirements comprise functional requirements, nonfunctional requirements with a focus on usability.

2. Background

2.1 Algorithm related

2.1.1 NDVI

In the field of remote sensing, the reflectance characteristics of land features can be quantified by spectral reflectance (Lillesand et al. 2008). However, vegetation indices are used more frequently than the former because of their convenience in calculation and data processing (Hatfield et al. 2010; Huang et al. 2012; Wang et al. 2016). Among them, one widely used vegetation index is NDVI (e.g. Tsai et al. 2016; Wang et al. 2016). It not only relates to chlorophyll at micro scale (Wang et al. 2016), but also sensitive to the leaf area and biomass at large scale (Ma et al. 1996; Shanahan et al. 2001; Shanahan et al. 2003; Solari et al. 2008). This index is computed from the equation

$$\text{NDVI} = \frac{NIR-RED}{NIR+RED} \qquad (1)$$ (Rouse et al. 1973)

where NIR represents spectral reflectance acquired from the near-infrared channel and RED represents spectral reflectance acquired from the red channel of a satellite. The value of NDVI ranges from -1 to 1. High, positive values (around 0.6 to 0.9) of NDVI indicate large amount of vegetation (USGS 2015). This is because for vegetation the spectral reflectance in the red channel is far lower than that in the near-infrared channel. The absorption of chlorophyll in the red band contributes to this phenomenon. For regions covered by sparse plants, their NDVI values are lower but still larger than 0 (around 0.2 to 0.5) (USGS 2015). In contrast, water, snow and clouds have higher reflectance in the red band than in the near-infrared band. As a result, they have negative NDVI values. Bare lands and rocks yield similar spectral reflectance in both the red channel and the near-infrared channel, thus their NDVI values are zero or near zero.

Large amounts of satellite sensors are capable of producing NDVI products, such as the Advanced Very High Resolution Radiometer (AVHRR), the Landsat 7's Enhanced Thematic Mapper Plus (ETM+) and the Moderate Resolution Imaging Spectroradiometer (MODIS). The application fields of NDVI includes agriculture (e.g. Bacchini and Miguez 2015; Meroni et al. 2016) and forestry (e.g. Mallegowda et al. 2015; Lin and Liu 2016).

2.1.2 Principles of linear regression methods

Linear regression methods have been widely used in the studies of trend analysis of time-series NDVI satellite imagery (e.g. Li et al. 2014; An et al. 2015). For the studies of trend analysis using NDVI, the main idea is to find a straight line to fit the data points that used to plot the relationship between each independent variable (i.e. time) and its according dependent variable (i.e. NDVI value) although different linear regression methods have different rules on defining the exact placement of the straight line among the data points. For example, for OLS, the rule of placing the fitted straight line is to ensure the sum of the squares of the vertical distances between each data point and the straight line is a minimum value; for the Theil-Sen estimator, another popular method in trend analysis (e.g. Cao et al. 2014; Liu et al. 2015b), the rule of placing the fitted straight line is choosing the line with the median slope among all lines through all pairs of sample points.

In addition, because the existence of residuals, a statistical significance test is required to prove the relationship between independent variable and dependent variable is coefficient significant. A kind of statistical significance test is t-test. It is often used for OLS method (e.g. Qin et al. 2009; Bodzin and Fu 2014). At the beginning a null hypothesis that supposes there is no coefficient significance between the independent variable and the dependent variable is made. A value of statistical significance level (α) that defines the chances of making mistakes is also required. It represents the probability of rejecting the null hypothesis when the null hypothesis is actually true. For example, when the value of α is 0.05, it means a 5% chance exists of making a mistake of concluding a difference obtains when there is no difference actually (Frost 2015). Under most circumstances the value of α is 0.01 or 0.05 (Sproull 2002; Craparo 2007) although its value ranges from 0 to 1 theoretically. After that, a t-test will be used to calculate the value of ρ. If the value of ρ is less than the value of α, then it can be concluded that the relationship between the independent variable and the dependent variable is coefficient significant.

However, a straight-line relationship cannot correspond to all natural phenomena. For example, the changes of NDVI may affected by human factors and wildfires (e.g. Liu et al. 2015a; Mishra et al. 2015). As a result, to solve the shortcoming of linear regression methods of using a straight-line relationship to evaluate all natural phenomena, a nonlinear regression algorithm (PolyTrend) developed by Jamali et al. (2014) has been used for monitoring vegetation trends.

2.1.3 Nonlinear regression algorithm (PolyTrend) overview

The PolyTrend algorithm is a three-phase algorithm used for detecting types of vegetation trends in time-series NDVI data. It was explained in detail by Jamali et al. (2014). The classification scheme was concluded in Figure 1 (Jamali et al. 2014). In this method, the type of significance test is a t-test and the value of statistical significance is 0.05.

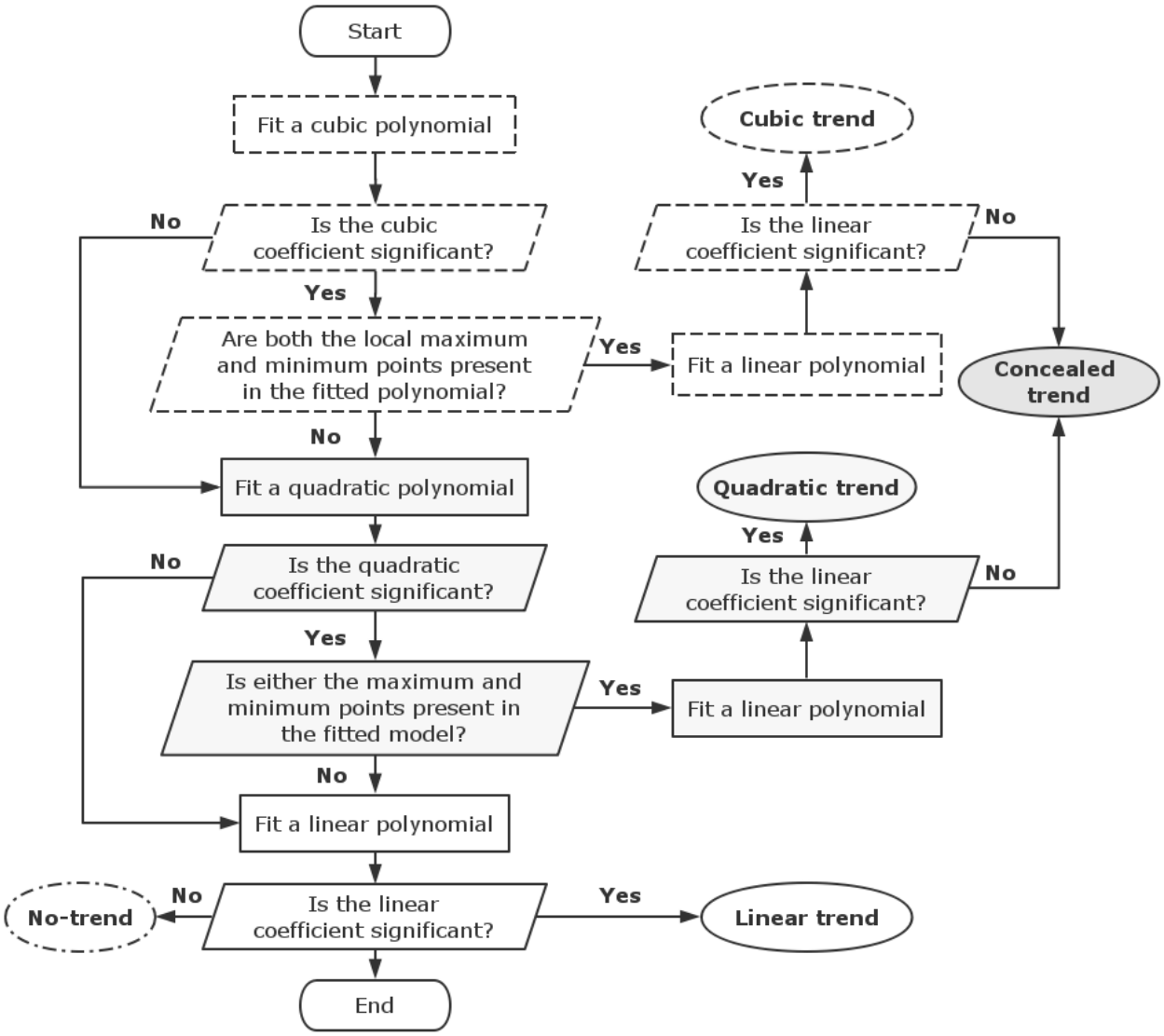

Figure 1 Classification scheme of the PolyTrend algorithm (Jamali et al. 2014). The statistical significance is 0.05 in all the statistical hypothesis tests of the scheme.

Two input arguments are required when using this algorithm. The first one is time-series NDVI values of a single pixel in the form of a column vector. The second one is a predefined statistical significance (α). A single value of α is used in all the statistical hypothesis tests during one classification procedure.

The PolyTrend algorithm returns one of five types of vegetation trends: cubic, quadratic, linear, no-trend and concealed. The explanation of these types is shown in Table 1. Figure 2 shows examples of the plots of these trend types.

Table 1 Types of vegetation trends in the PolyTrend algorithm

Trend Types	Explanation
cubic (Figure2a)	● The pattern of vegetation trends can be fitted by a cubic polynomial. The net change of NDVI can be detected when the statistical significance in a t-test is α.
quadratic (Figure2b)	● The pattern of vegetation trends can be fitted by a quadratic polynomial. The net change of NDVI can be detected when the statistical significance in a t-test is α.
linear (Figure2c)	● The pattern of vegetation trends can be fitted by a linear polynomial. The net change of NDVI can be detected when the statistical significance in a t-test is α.
no-trend (Figure2d)	● The pattern of vegetation trends can neither be fitted by a cubic, quadratic, nor linear polynomial. No net change of NDVI can be detected when the statistical significance in a t-test is α.
concealed (Figure2e&2f)	● The pattern of vegetation trends can be fitted by a cubic polynomial or a quadratic polynomial. No net change of NDVI can be detected when the statistical significance in a t-test is α.

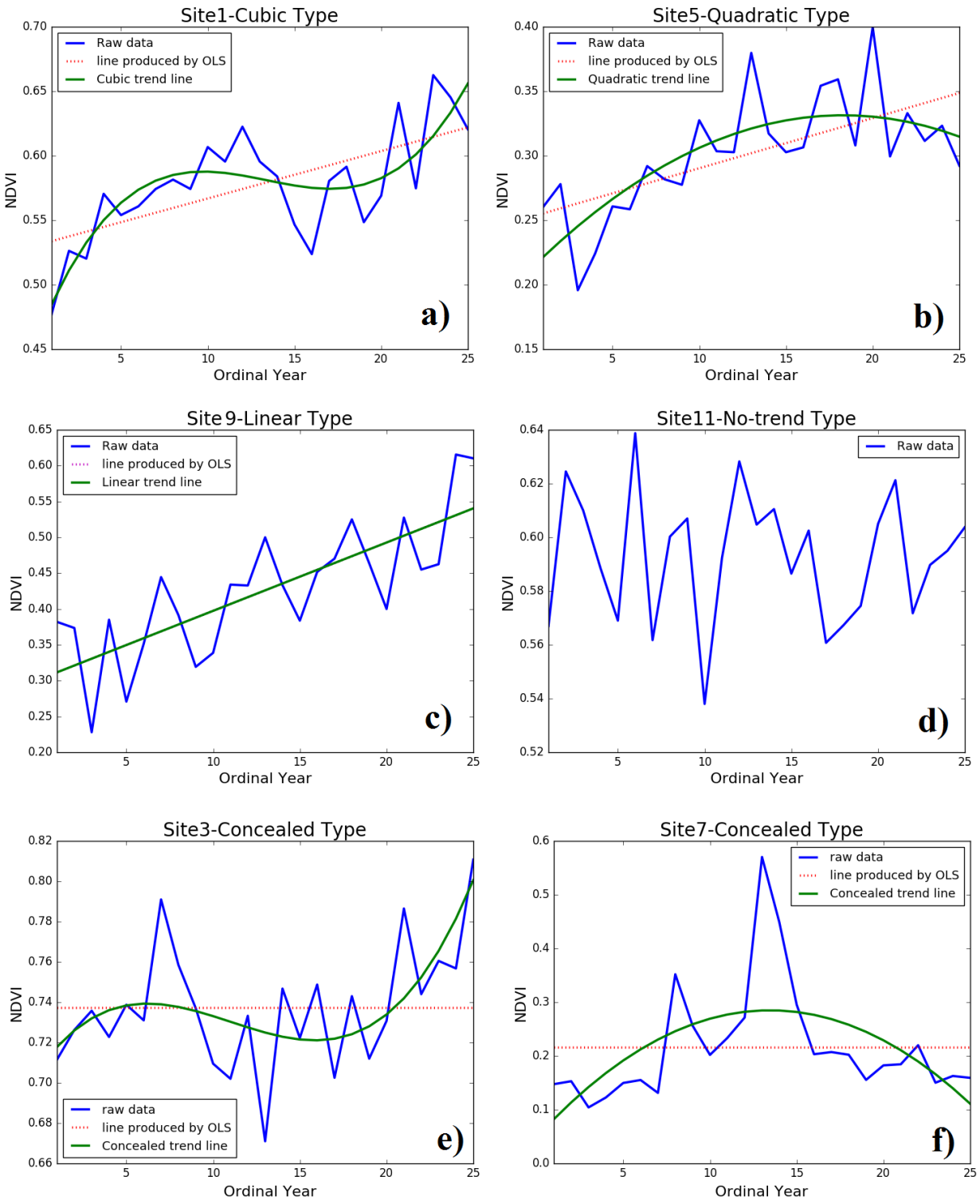

Figure 2 Examples of all types of vegetation trends plotted by the PolyTrend algorithm showing the trend types of cubic (a), quadratic (b), linear (c), no-trend (d) and concealed (e and f) using the sample data provided by Jamali. The data is validated annual NDVI data gathered at the sample sites of the Sahel. The details of sample data and the locations of the sample sites are described in Jamali et al. (2014).

The advantage of the PolyTrend algorithm over a traditional linear regression method from the perspective of sensitivity was shown when both of them were applied on processing the same validated GIMMS (Global Inventory Modelling and Mapping Studies)-NDVI time-series dataset over China during the time period from 1982 to 2012. In that study of Liu et al. (2015b), the Theil-Sen estimator, a more robust linear regression method than OLS (Fernandes and Leblanc 2005) and the PolyTrend algorithm were applied side by side for monitoring vegetation trends. Comparing with the Theil-Sen estimator which could only indicate increase and significant increase in the forests of Northeast China, the PolyTrend algorithm revealed not only two kinds of increase, but also cubic fluctuation and concealed fluctuation existed due to the various plant species in the forests (Liu et al. 2015b).

The PolyTrend algorithm was initially applied on observing vegetation changes over the Sahel from July 1981 to December 2006 using validated GIMMS-NDVI time-series dataset (Jamali et al. 2014). In the study, both the evident, long-term vegetation trends existed in the east-west zone across the Sahel and the weaker, short-term trends of vegetation appearing at the peripheries of the zone could be well detected (Jamali et al. 2014).

2.2 Concepts of developing the web-based system

2.2.1 Web development

Web development refers to the process of developing a web-based system hosted on the Internet or Intranet. A three-tier architecture is used for developing a web-based system. They are presentation tier, logic tier and data tier.

The presentation tier is accessible for users by using web browsers. The contents of the web pages are generated statically or dynamically by client[1] side development. During the process of client side development, HyperText Markup Language (HTML) (W3C 2016c), Cascading Style Sheets (CSS) (W3C 2016a) and JavaScript (JS) (ECMA 2016) are used to make webpages interact with users (Long 2012). An interaction generates a request and the request is sent to a server for further processing via the Hypertext Transfer Protocol (HTTP) (W3C 2016d) or other protocols depending on the type of the server, the purpose of the system and security constraints, etc.

The logic tier is a software framework used for processing and generating responses. A software framework comprises programs for fulfilling the request and a web server environment to run the programs (Ottinger 2008). The programs can be written in programming languages such as Java (Oracle 2016a), Python (2016) and PHP (2016). The software framework includes Spring (2016), Django (2016b) and CakePHP (2016).

The data tier refers to a database or file system. It also comprises the data for processing the request. A typical choice is using a relational database management system (RDBMS) (Yoshitaka

[1] The client refers to web browsers. Users use the client to surf the Internet.

2014). Popular RDBMS choices include Oracle Database (Oracle 2016b), MySQL (2016) and PostgreSQL (2016).

2.2.2 Web servers

The communication between the client side and the server side is linked by a web server. A web server is a kind of technology to handle requests sent from the client side via HTTP. This term either refers to the computer that has the software used to process HTTP requests, or specifically refers to that software. Web servers have a common character: Every web server receives HTTP requests from the client side through the network and provides HTTP responses to the client side. Usually a HTTP response contains an HTML document, but it can also include text files and images.

The HTML files are stored in a database or a file system. Web addresses and names of the HTML files are arranged in a hierarchal order. Once a web server is installed and correctly configured, the administrator will assign a local path from the place that stores files on the server as the root directory. The web server will link web addresses to the according file system path that has the requested file.

Table 2 shows the four most popular web servers, by market share in February 2016 (Netcraft 2016).

Table 2 Market share of four most popular web servers in Feb., 2016

Product	Vendor	Percent
Apache	Apache	32.80%
IIS	Microsoft	29.83%
Nginx	NGINX, Inc.	14.72%
GWS	Google	2.21%

Apache (2016) is free, open source software and supports a wide range of operating systems (OSs) including Windows, Linux and Unix. It has been the most prevalent web server for almost two decades (e.g. Jackson 2014; Netcraft 2014). A study from Lakka et al. (2012) shows that the Apache market saturation levels has not reached its maximum value. They found many factors influence the popularity of Apache, such as level of skills and education of potential users, institutional quality and rules and laws of an organized society. The various functionalities of Apache web server made it a popular research object. For example, in the research carried out by Andersson et al. (2003), Apache was used to build a model so that the performance of network traffic can be measured. In addition, Apache showed its potential to become a dynamic cloud-ready load balancer in the field of cloud computing (Heinzl and Metz 2013). Popular websites use Apache as web server includes: Apple (2016; e.g. W3Techs 2016a), Adobe (2016; e.g. W3Techs 2016a) and Paypal (2016; e.g. W3Techs 2016a).

The second most popular, free and open source software is Nginx (2016). Comparing with Apache, one difference between them is the way to handle connection and traffic. Nginx is good at processing concurrent requests in particular. This characteristic has been validated in the researches of Chi et al. (2012) and Tobias (2014), etc. Popular websites that use Nginx as web server includes: Wikipedia (2016c; e.g. W3Techs 2016d) and Mozilla (2016; e.g. W3Techs 2016d).

IIS (Microsoft 2016) and GWS (Google 2016) are not open source software. IIS only supports Windows OS, while GWS built on Linux OS and mainly used by Google for its online services. They are more suitable for developers working with specific platforms. IIS is used by Microsoft-related online services such as Bing (2016; e.g. W3Techs 2016c) and Skype (2016; e.g. W3Techs 2016c) and GWS is used by, such as Google Search (2016; e.g. W3Techs 2016b).

2.2.3 Web standards

Web standards refer to open, formal standards and technical specifications of every aspect of the World Wide Web (WWW). The standards depend on each other and have extended to the Internet. Once the Internet is built on standard-based protocols, web services can be provided over it fluently (Chappell and Jewell 2002). They mainly concern the interoperability, the accessibility and the usability of web sites.

Nowadays standards are made by consortiums and organizations. For example, to design the appearance of webpages, representative standards include recommendations drawn up by the World Wide Web Consortium (W3C 2016f); the format of web addresses is regulated by the Internet Engineering Task Force (IETF 2016) (Lee 1998) and the rules for displaying text in a multi-language environment are defined by the Unicode Consortium (2016).

W3C devotes to become an open platform with technical standards (Dardailler 2010). Many occurring problems when creating a web-based system before have now been solved by W3C. For example, a battery status application programming interface (API) is available in HTML5 standard. This API can measure the condition of the device's battery. If the battery is low then developers can decide to slow down or stop Ajax (Asynchronous JavaScript and XML) requests that may occur automatically when a certain page is loaded (de Rosa 2014). In addition, even some solutions made elsewhere have become a part of W3C standards (Sikos 2011). In this project, several standards from W3C are used, such as: HTML, a standard for defining infrastructures of webpages; CSS, a standard for depicting the presentation of documents written in markup languages, and Scalable Vector Graphics (SVG) (W3C 2016e), a standard for defining the graphics in XML (eXtensible Markup Language) (W3C 2016b) format.

Another, non-profit, standards organization made the ECMAScript Language Specification. It is used to regulate JavaScript, which is a programming language that adds dynamic functionalities to HTML. Together with HTML and CSS, they form the basic technologies of the Internet (W3C 2014).

In addition, the standards of geospatial data and services, such as the Web Map Service (WMS) (OGC 2016d), the Web Feature Service (WFS) (OGC 2016c) and the Web Coverage Service (WCS) (OGC 2016b), are regulated by the Open Geospatial Consortium (OGC 2016a). The standards from OGC were not used when developing this system.

2.2.4 Licenses

Licenses for computer code in open source software

Open source software (OSS) encourages the propagation of emerging technologies (Kapitsaki et al. 2015). Its developers' communities have become a feasible substitute to commercial-based cooperation (Awazu and Desouza 2004). During these years, OSS projects increased rapidly (Câmara and Fonseca 2008); some even became competitors of proprietary commercial software (Sen 2007), and among them, the best known OSS projects includes the Apache web server and the Linux OS (Colazo and Fang 2009). Thus, for OSS, it is important to know the terms of use and the conditions where these terms are applicable (Lawrence 2004) since software is closely related to copyright and intellectual property (Kapitsaki et al. 2015). As a result, various kinds of licenses appear to fit different conditions.

According to the Black Duck Knowledgebase (Black Duck 2016), the three most common OSS licenses are the MIT License (used by 26% OOS), the GNU General Public License v2.0 (GPLv2) (used by 21% OOS) and the Apache License 2.0 (used by 16% OOS).

The MIT License allows users to modify and/or redistribute the original software in the form of commercial or open source software, as long as the copyright notice and the permission notice are included in all copies or substantial portions of the software. The original author does not have any responsibility under any circumstances (Open Source Initiative 2016c).

The main difference between GPLv2 and the MIT License lies in that commercial software can keep its proprietary after it contains software that uses the MIT License. However, if the original software uses GPLv2, then any derived software of the original one, or uses the original software as a part of the new work should also use GPLv2, which means the whole part of code should be open source (Open Source Initiative 2016b). This requirement is not suitable for commercial development or for programs that related to national security.

The Apache License 2.0 have an advantage over GPLv2 from this perspective: If the original software uses the Apache License 2.0 then the derivative products do not need to present the original code, as long as users cite the source and fulfill the requirements made by the original user, such as quotation of trademark and agreement (Open Source Initiative 2016a).

Licenses for text, images and data

When designing a web-based system that contains a lot of information, the license used by text, images and other creative material should be taken into consideration. Traditional philosophy of copyright has two extremes, one is "all rights reserved" and the other is "public domain" (Wikipedia 2016a). However, this is not realistic with the development of the Internet. A non-

profit organization named the Creative Commons (CC) (Creative Commons 2016b) is devoted to finding a balance between the rights of the authors and the flow of information (Snijder 2015).

By using different combinations of its core, six right modules, licenses with various restrictions are available. Among the 11 kinds of valid combinations, the six most popular kinds are CC BY, CC BY-SA, CC BY-ND, CC BY-NC, CC BY-NC-SA and CC BY-NC-ND. "CC" is a prefix. The more modules a license has, the more restrictions users should follow. Table 3 shows the meaning of each module.

Table 3 CC license modules (Creative Commons 2016a)

Name	Explanation
BY	The name of the author should be mentioned.
SA	The derived works can only be released under the same law where the original work was created.
NC	The work cannot be used for commercial purpose.
ND	The work cannot be changed.

2.2.5 Web usability

Usability refers to the degree to which a used object has accomplished the predefined goals. According to the standards made by the International Organization for Standardization (ISO), the level of usability can be judged in terms of effectiveness, efficiency and satisfaction (ISO 1998).

Usability is also important in web development. When scanning web pages, people are less likely to wait and they have to grasp the theme in a few seconds at most (Nielsen and Norman 2000). If unintelligible contents or unhandy functionalities exist, people leave (Nielsen 2012).

To study how well the criteria of the usability of a web-based system are met, a basic method called the user testing can be used (Nielsen 2012). To carry out the testing, representative users can be invited to perform representative tasks of a web-based system and express their feelings and viewpoints (Nielsen 2012). On the other hand, the person who invites these users should observe carefully to find out where they encounter difficulty and success during the task and records the feedback (Nielsen 2012).

3. Data

3.1 PolyTrend function (pixel level)

The PolyTrend function (pixel level) refers to the PolyTrend algorithm written in MATLAB programming language. The input arguments are the same as the ones of the PolyTrend algorithm, i.e. time-series NDVI values of a single pixel in the form of a column vector, and a value ranging from 0 to 1 that acts as a predefined statistical significance (α).

The output arguments of the PolyTrend function not only indicate the type of vegetation trends, but also include information about net change of NDVI. Table 4 shows the input and the output arguments of the PolyTrend function (pixel level).

Table 4 Input and output arguments of the PolyTrend function (pixel level)

	Names	Data Types	Values	Explanation
Input	Y	vector (.txt)	[-1,1]	The time-series NDVI values of a single pixel.
	alpha	float	[0,1]	A predefined statistical significance for all hypothesis tests.
Output	Trend_type	integer	3,2,1,0 or -1	3: cubic trend type 2: quadratic trend type 1: linear trend type 0: no-trend type -1: concealed trend type
	Slope	float	any	The slope of vegetation trends of the elapsed time period.
	Direction	integer	1 or -1	1: In general, the net change of NDVI increases. -1: In general, the net change of NDVI decreases.
	Significance	integer	1 or -1	1: The trend type is cubic, quadratic or linear. -1: The trend type is no-trend or concealed.

3.2 Sample input data for validating the PolyTrend function (image level)

The sample data used to draw Figure 2 (in Section 2.1.3) is available in the form of six ASCII files with the TXT file extension. In each ASCII file, there are 25 lines of NDVI values representing a known, specific vegetation trend type of a time-series NDVI dataset of a single pixel in a 25-year time span. After each ASCII file is imported into MATLAB, a column vector of the size 25 x 1 will be formed. By concatenating these six column vectors, a matrix of the size of more than one column and more than one row can be formed to represent pixels with time-series NDVI data in a 25-year time span, where each element of the matrix represents a known vegetation trend type.

The method of validating the correctness of the PolyTrend function (image level) by using the sample input data is shown in Section 5.1. The general information of the sample input data is shown in Table 5. The content of each ASCII file is shown in Appendix B.

Table 5 General information of the sample input data

File Name	Trend Type
time_series_cubic1.txt	3
time_series_quadratci1.txt	2
time_series_linear1.txt	1
time_series_no_trend.txt	0
time_series_conceald_cubic1.txt	-1
time_series_conceald_quadratic1.txt	-1

3.3 Time-series NDVI satellite imagery

Once the web-based system is built, users can upload raw time-series NDVI remotely sensed imagery as the input dataset. In this project, imagery with the TIF file extension will be used as the input time-series NDVI satellite imagery. Among them, MODIS MOD13Q1 data is a kind of widely used satellite imagery in the studies of NDVI (e.g. Bhandari et al. 2011; Pandey et al. 2015). Because of its popularity, MODIS MOD13Q1 data will be used in the testing and the demonstration of the PolyTrend web-based system.

MOD13Q1 data is available on the website of MODIS Subsets (NASA 2016b). This website provides global time-series NDVI data from 2000 to this year with an interval of 16 days. By specifying coordinates for the center of area of interest and the number of kilometers encompassing the center location, a customized NDVI time-series dataset can be created for users.

The steps of creating a NDVI time-series dataset are listed below.

1. Go to the website of MODIS Subsets (NASA 2016b).
2. Input coordinates for the center of area of interest.
3. Input the number of kilometers encompassing the center location.
4. Choose MOD13Q1 products.
5. Select the starting date and the ending date of the data.
6. Enter an email address.
7. After the GeoTIFF folder is downloaded and unzipped, select files with filenames contain "NDVI".
8. Copy these files to a new folder; the new folder is the time-series NDVI dataset.

MOD13Q1 contains composite imagery gathered with 16 days' interval with 250m spatial resolution (Guo et al. 2015). Because the data is generated by a compositing method to decrease impacts from atmosphere, clouds, and viewing angles (e.g. Huete et al. 2002; Lambert et al. 2015; Shao et al. 2016), it is regarded as containing reliable pixel values (Huete et al. 1999). Another reason for using raw MOD13Q1 data lies in, although many smoothing and fitting techniques have been applied on these raw data due to the irregularities of noise, such as applying filters (Sakamoto et al. 2005), moving average (Reed et al. 1994; Hill and Donald 2003) and curve fitting (Jönsson and Eklundh 2002; Zhang et al. 2003; Lu and Guo 2008), these techniques might

decrease the peak value of the data (Reed et al. 1994) and cause the loss of the originality of the data. As a result, minor variations in vegetation trends may not be detected by the PolyTrend algorithm.

4. Design and development

4.1 Requirements

The aim of this project is to develop a web-based system to visualize vegetation trends by the PolyTrend algorithm. Based on the studied references (e.g. Eriksson 2012; Usabilityfirst 2016), a set of requirements is defined to keep the web-based system focused on the aim.

Functional requirements

Functional requirements define the behavior of a system. In this project, the PolyTrend web-based system should be able to

1. receive a time-series NDVI datasets and the value of statistical significance (α),
2. receive the nominal range of the NDVI dataset and the range of desired NDVI input to be processed,
3. provide correct results in the forms of binary files and ASCII files,
4. provide the concept of the PolyTrend algorithm,
5. provide downloadable resulting data, and
6. provide explanation of the results.

Nonfunctional requirements

Non-functional requirements define specific criteria and constraints used to assess the operation of a system (e.g. Burk 2010; Wikipedia 2016b). In this project, the nonfunctional requirements of the PolyTrend web-based system are listed below.

7. The value of α belongs to [0, 1].
8. NDVI values representing the vegetation and the soil (i.e. [0, 1]) are viewed as *qualified* input data.
9. NDVI values having evident changes[2] over time are viewed as *qualified* input data.
10. Only files with the TIF file extension can be uploaded.
11. At least four files should be uploaded for fitting a cubic polynomial (i.e. four equations can solve a system of four-variable polynomial equations).
12. The order of the files should allow customization so that they can be arranged in

[2] For a single pixel in the results, during the whole time span, the number of NDVI pixels that have a difference of less than 0.000001 comparing to the ones that represent adjacent years is larger than or equal to three times the number of NDVI pixels of the whole time span divided by four (Jönsson, P. and L. Eklundh).

chronological order before they are processed by the PolyTrend algorithm.

13. The web-based interface should support at least one kind of web browsers.
14. The criteria that used to judge the usability of the system are
    - easy-to-use functionalities,
    - easy-to-understand information, and
    - easy-to-identify text under different screen resolutions.

4.2 Choosing a language to be used in the logic tier

The selected programming language used in the logic tier is Python for a number of reasons.

Python is designed to be easy-to-read and consistent (Lutz 2009a). For example, the code should be aligned in accordance with the logic order of the program. Although the various forms of the code are limited, the code is easy to understand for people who are not the author.

Python is succinct. Usually, codes written in Python can be up to five times shorter than correspondent Java program and it may be up to ten times shorter than comparable program written in C++ (Van Rossum 1997). This enhances the efficiency of development.

Python is a cross-platform language. This means code written in a certain OS can be implemented in multiple computer platforms.

As Ruby and JavaScript, Python is also prevalent for web development (Glaser 2012). There are several frameworks and a community for supporting web development in Python (Glaser 2012).

4.3 Choosing a web framework and a development environment

A web framework is required for development after the language to be used in the logic tier is defined. Django is an ideal choice as it supports quick development and succinct design (e.g. Lutz 2011; Django 2016b). It is recommended by Sheena (2015) for people without previous experience of web development. Comparing with other Python web framework such as Pyramid, it is in Django straight-forward to plug in self-contained code from other sources by a mechanism called *application*. This brings convenience for equipping the project with ready-made applications (Sheena 2015). In addition, Django has a built-in lightweight and standalone web server for developer to see the effects instantly after each change of the code is made during development and testing.

For creating a Django web framework/project, a development environment should be built to incorporate and develop the web framework/project. Regarding the tools for building the environment, Eclipse (2016) and PyDev (2016) are the choices. This combination is recommended by Lutz (2009b). Eclipse is an open-source and free integrated development environment (IDE) that allows extensible plug-ins for customizing the environment. PyDev is a

plug-in for programming in Python. After PyDev is installed in Eclipse, a Django project can be created for web development in Python.

4.4 Choosing a way of calling MATLAB functions from Python

After the programming language to be used in the logic tier and the development environment are defined, it is necessary to find a way of calling MATLAB functions from Python, since the PolyTrend algorithm is written in MATLAB. Many free and open source tools are available, such as mlabwrap (2011), pymat2 (2016), and pymatlab (2016). However, an official tool released by MATLAB or Python is preferred. This is because the version of MATLAB (R2015b) and Python (3.4) used for this development are relatively new and the tools listed above may not fully support this task. For example, the documents of pymat2 (pymat2 2016) and pymatlab (pymatlab 2016) do not indicate the support for Python3 and the latest updated year of mlabwrap webpage was 2011 (mlabwrap 2011) when this thesis began in February, 2016.

No tools released by Python that support the interaction with MATLAB have been found by the author. On the other hand, two tools released by MATLAB are available. They are the MATLAB Compiler SDK (Software Development Kit) (MATLAB SDK) and the MATLAB Engine API for Python (MATLAB API). However, the steps of calling MATLAB functions from Python by the MATLAB SDK are far more than using the MATLAB API (e.g. MathWorks 2016a; MathWorks 2016b; MathWorks 2016c). As a result, the MATLAB API is the choice for calling MATLAB functions from Python.

4.5 Choosing the licenses

Licenses for computer code

Except MATLAB, all the software and tools used in this project are open-source. To disseminate the PolyTrend algorithm easily and to refine the PolyTrend web-based system in the future, the PolyTrend web-based system was decided to be released under the GPL 3.0 License (GPLv3) or any later version. GPLv3 contains the basic intent of GPLv2 which has been described in Section 2.2.4. Comparing with GPLv2, GPLv3 gives more rights to end users. For example, when computers contain a program that uses GPLv2, they can shut down if they detect modified software (Stallman 2007). This is not allowed when the program uses GPLv3. Code uses the MIT License from external sources is also included. GPLv3 is compatible with the MIT License. The way of gaining the source code of the PolyTrend web-based system is listed in Appendix E.

License for text, images and data

To disseminate the concept of the PolyTrend algorithm easily, for text, images and data of the PolyTrend web-based system, the license is CC BY-NC-SA. The explanation of this license is shown in Table 3.

4.6 Architecture and workflow

Based on the requirements and all the above choices, the architecture in Figure 3 is used for the PolyTrend web-based system.

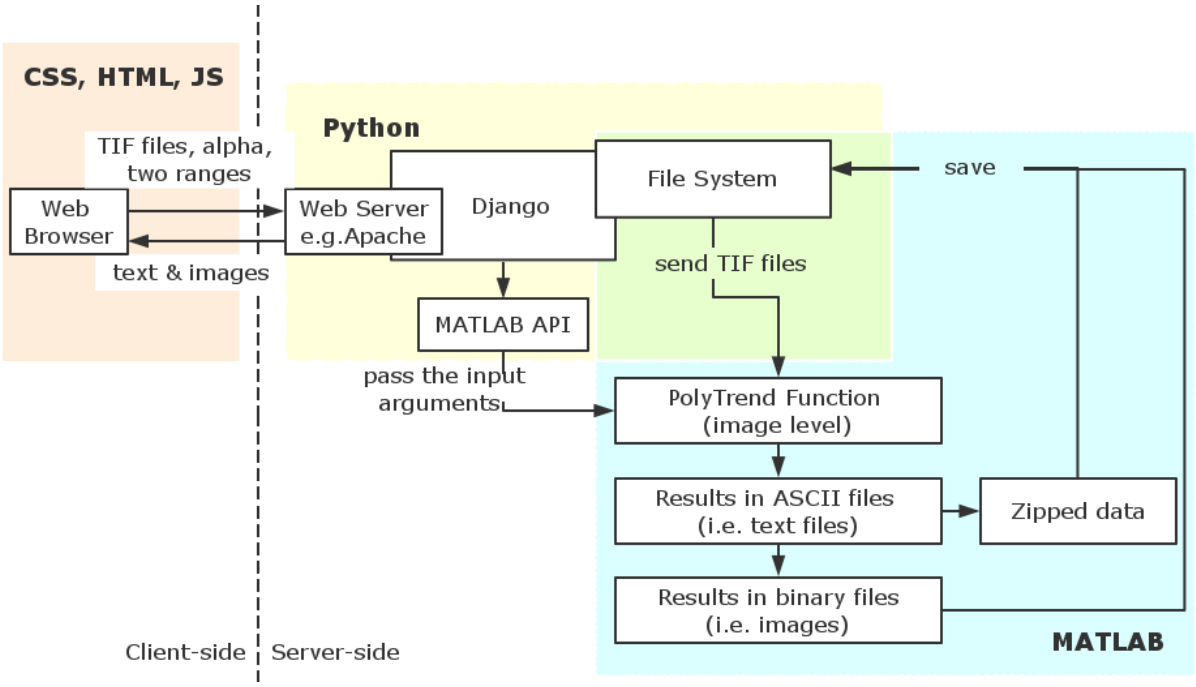

Figure 3 Architecture and workflow of the PolyTrend web-based system

At the beginning, a time-series NDVI dataset with the TIF file extension is uploaded through a web browser. A value of statistical significance (α), the nominal range of the NDVI dataset and the range of desired NDVI input to be processed are also required in this step. The appearance of the web browser and the interaction between the user and the browser are controlled by CSS, HTML and JavaScript.

These arguments are sent to the web framework (i.e. Django) by a web server. Before this web-based system is published, the built-in web server of Django can be a substitute of the real sever, e.g. the Apache HTTP Server.

The uploaded files are handled by a ready-made application in Django and they are sent to a predefined location in the file system. The input arguments are passed to the PolyTrend function (image level) by the MATLAB API.

The PolyTrend function (image level) will read the uploaded files in the file system and convert them to matrices. The PolyTrend function (image level) judges whether the time-series NDVI values of a pixel are qualified data and whether they are in the range of desired NDVI input to be processed defined by the user. If they are, they will be handled by the original PolyTrend function (pixel level), which is a subfunction[3] of the PolyTrend function (image level). If they are not, the unqualified values and the out-of-range values will be replaced by a fixed value respectively.

The resulting data of the PolyTrend function (image level) are matrices initially. For visualizing the results, they will be plotted and saved as images. On the other hand, the matrices will also be saved in the form of text files and zipped. Both the images and the text files are saved in another location in the file system by MATLAB. It is obvious that both MATLAB and Python can control the file system.

Finally, Django uses Python functions to return results in the form of a web page. In the web page showing the final results, images are displayed by the management of static files[4] of Django. The zipped folder contains text files is available in the form of a hyperlink. Other text-formed information, such as explanation, is hardcoded on the web page.

4.7 Main file structure

Before developing the PolyTrend web-based system, the development environment should be initialized. The steps of initializing the development environment and the steps of testing whether this environment is correctly configured are shown in Appendix A.

After the development environment is initialized, a Django project for developing the PolyTrend web-based system was created. The main file structure of the Django project is shown in Figure 4. Only parts that need major editing and modification during the development are shown. The entire file structure of the Django project is shown in Appendix D.

[3] In MATLAB, if one function calls another, the former is called the primary function and the latter is called a subfunction (Attaway 2013).

[4] In Django, the images are called static files (Django 2016a).

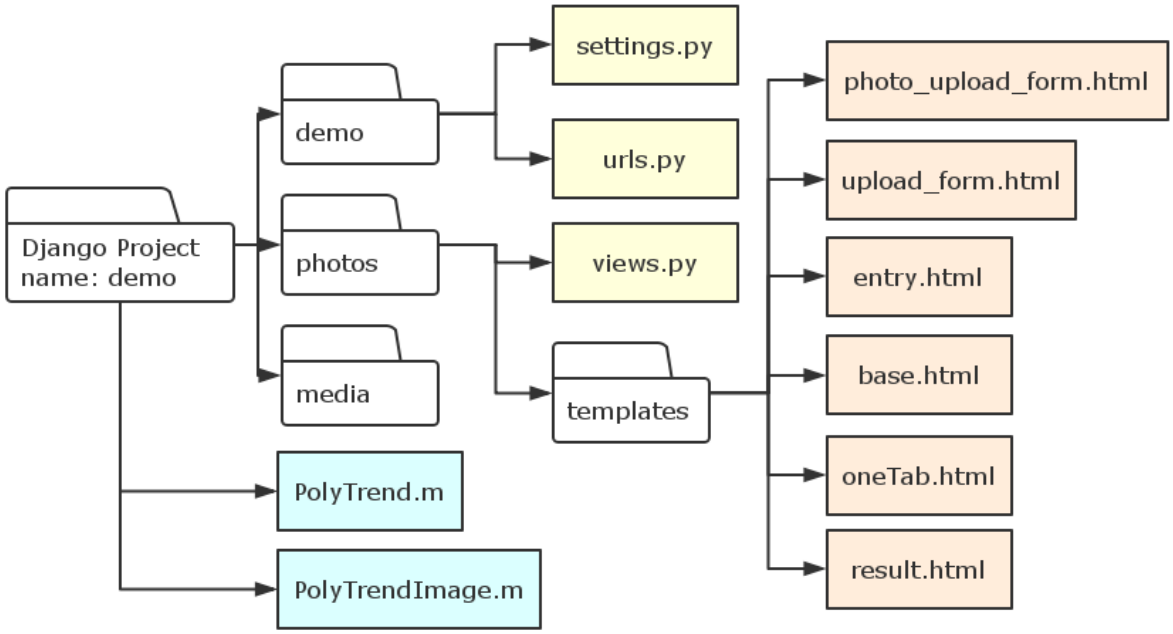

Figure 4 Main file structure of the PolyTrend web-based system. The files written in Python are yellow. The files written in CSS, HTML and JS are pink. The files written in MATLAB are blue. White graphics are folders.

At the beginning, a Django project called *demo* was created. It always contains a subfolder *demo* that has the same name as the created project. In the subfolder *demo*, *settings.py* contains the main configuration parameters, such as the names of the applications used in the project and the path of the static files. *urls.py* contains names of the Python functions and it indicates which web address will be used when a certain Python function in *views.py* is called to return a result. The result can either be a webpage or a response.

The folder *media* is created by a ready-made application *Django-jfu* (Github 2016) every time the web-based system runs. All the uploaded files will be sent to this folder for further processing.

photos is the name of the *Django-jfu* application in this Django project. In short, it provides a table that only allows one or more predefined type(s) of files to upload. Some characteristics of the table, such as the allowed file type(s) and the appearance of buttons in the table, are partly defined by *photo_upload_form.html* and *upload_form.html*. The remaining characteristics, such as the ways of interactions with users are defined by the files which are located under the preinstalled *Django-jfu* Python library. For the remaining four HTML files, the index *base.html* takes the table as a part of its contents, receives all the input values and passes them to the Python functions located in *views.py*; *entry.html* controls the accessibility of users, and if the web-based system has been opened in a tab, the webpage *oneTab.html* will be displayed; the resulting data is shown in *result.html*.

views.py contains all the Python functions that send responses to the web browser, the responses can either be visible (e.g. a webpage) or invisible (e.g. a response). It also controls the MATLAB API.

PolyTrend.m is the original PolyTrend function (pixel level) and *PolyTrendImage.m* is the PolyTrend function (image level) that calls the former when it runs. It is not mandatory that all MATLAB files are located under the directory of a Django project. However, for the convenience of management, all of them are under the root directory of this Django project. In addition, to let the MATLAB API identify all MATLAB files, the path of the MATLAB files should be added to the settings of MATLAB.

4.8 Steps and components

4.8.1 Controlling the input parameters

Value of statistical significance (α)

A statistical significance is a number ranging from 0 to 1. Before the value is submitted by a form in the index page *base.html*, a number of criteria are used to judge whether the value meets Requirements 1 and 7. The criteria of the value of the statistical significance are listed below.

1. It cannot be void.
2. It should be a number.
3. It belongs to [0, 1].
4. It cannot contain any space.

The above criteria are written in JavaScript functions. If the value of α meets all of them, it will be passed to a Python function *result* located in *views.py* by the form and used by the PolyTrend function (image level) after other input arguments also meet their respective criteria.

Nominal range of the NDVI dataset

MODIS MOD13Q1 data is the main kind of time-series NDVI satellite imagery used in this project. The range of NDVI values of the raw MOD13Q1 data belongs to [-10000, 10000]. However, it is also possible that the dataset is not from MODIS. In this case, the range of the dataset may belong to [-1, 1] because this range corresponds to the range of NDVI values.

To meet Requirement 2, the PolyTrend web-based system allows users to choose the range of the input time-series NDVI dataset. It can either be [-10000, 10000] or [-1, 1]. The chosen range will be used along with other input parameters by the PolyTrend function (image level).

Range of desired NDVI input to be processed

According to Requirements 8, NDVI values ranges from 0 to 1 will be processed by the PolyTrend algorithm. Therefore, the default to-be-processed range of NDVI input is [0, 1]. However, to fulfill Requirement 2, users can also customize the range by entering a minimum value and a maximum value. If they choose the customized range, then the values should meet the following criteria.

1. They cannot be void.
2. They should be numbers.
3. Both of them belong to [0, 1].
4. They cannot contain any space.
5. The minimum value is smaller than the maximum value.

The above criteria are written in JavaScript functions. If all the criteria are met, the values will be passed to a Python function *result* located in *views.py* by the form. They will be used by the PolyTrend function (image level) after other input arguments also meet their respective criteria.

Time-series NDVI dataset

A time-series NDVI dataset with equal interval can be uploaded by a ready-made application *Django-jfu* (Github 2016). The key functionalities of the application are listed below.

1. One or more predefined type(s) of files can be uploaded.
2. Users can remove files before they are sent for final calculation.
3. The order of the files shown in the table is decided by the order they are added. The order cannot be changed once they form the table.

In this web-based system, the name of the *Django-jfu* application is *photos*. It should be installed as a Python library before a development begins. In addition, to fulfill Requirements 1, 3, 10 and 12 the customization of *photos* should be carried on. The modifications of the original application are listed below.

1. A parameter in *photo_upload_form.html* was changed for uploading TIF files.
2. Buttons that used to change the order of the rows were added.
3. The function of single file upload was deleted for eliminating chances of mixing the order of upload.
4. For fulfilling the two items above, the modification of *upload_form.html* and the according JavaScript files in Python library were also carried out, the way of gaining the modified code is shown in Appendix E.

However, to meet Requirements 3 and 12 to make sure the files can be uploaded in chronological order and to gain correct results, two problems still exist. The first one is many tests show alphabetical order is used by MATLAB when reading files under a directory although this is not documented officially (MATLAB Answers 2016a). This means filenames affect the order of processing.

The solution to this problem is renaming files to let them have the same length of digital letters while remaining differences between the names for keeping the desired order (MATLAB Answers 2016a). In this project, all the uploaded files will be renamed according to their creation

time in the folder *media*. The earliest-created file will be renamed as *tif10000000001.tif* and the second-earliest-created file will be renamed as *tif10000000002.tif*, and so on.

The second problem is, although the creation times of the uploaded files are different, they cannot be distinguished in metadata information by Windows Explorer due to the high speed of uploading.

To solve this problem, a Python clause called *os.stat("a filename with its full path").st_ctime_ns* can return the creation time of a file in the form of an integer in unit of nanoseconds (Python 3.4.4 Documentation 2016). The following example (Table 6) shows the creation time expressed using the above Python clause.

Table 6 Creation times of files expressed in Windows Explorer and Python

File Name	Windows Explorer	Python
MOD13Q1.A2000049_NDVI.tif	3 April 2016, 1:22:12 PM	1459682532620664600
MOD13Q1.A2000065_NDVI.tif	3 April 2016, 1:22:12 PM	1459682532678667900
MOD13Q1.A2000081_NDVI.tif	3 April 2016, 1:22:12 PM	1459682532704669400
MOD13Q1.A2000097_NDVI.tif	3 April 2016, 1:22:12 PM	1459682532726670600
MOD13Q1.A2000113_NDVI.tif	3 April 2016, 1:22:12 PM	1459682532747671800

To meet Requirement 11, an Ajax request will be performed each time the mouse hovers over the button that sends the input arguments for processing. It checks the number of uploaded files and returns the value to the function that assesses whether the submission of all the input arguments should be performed.

In general, the workflow of controlling the input arguments is listed below.

1. A user inputs a value of α, the nominal range of the NDVI dataset, the range of desired NDVI input to be processed and uploads a time-series NDVI dataset.
2. When the user clicks the button for resulting data, an Ajax request and some JavaScript functions will run to check whether Requirements 1, 2, 7, 10 and 11 are fulfilled. If not, all the input parameters will not be processed. Otherwise they are sent to the Python function *result* in *views.py*.
3. The Python function *result* renames all uploaded files according to their creation time in the file system.
4. The Python function *result* calls the PolyTrend function (image level) by the MATLAB API (see Section 4.8.2).

4.8.2 Calling the PolyTrend function (image level) from Python

For calling MATLAB functions from Python, *eng = matlab.engine.start_matlab()* is used to start the MATLAB API. After that *eng.PolyTrendImage("names of all the input arguments except the dataset", nargout=0)* is used to call *PolyTrendImage*, the name of the PolyTrend function (image level). *nargout=0* means the PolyTrend function (image level) will not return any values but to

complete a series of tasks.

4.8.3 Creating the PolyTrend function (image level)

The function *PolyTrendImage* is to apply the PolyTrend algorithm at image level. It not only provides the results of trend type, slope, direction and significance of the time-series NDVI values, but also the mean, minimum, maximum NDVI values and the differences of NDVI values.

The general workflow of the PolyTrend function (image level) is listed below.

1. Reading all TIF files of the folder *image* in sequence, each file is a matrix and it will be stacked under the earlier one when it is read by MATLAB. During this process, if the data is MOD13Q1 data then each matrix will be divided by 10000. Otherwise, their values keep unchanged. Finally, a big matrix comprises all the values of time-series data are formed, with the earliest NDVI matrix located at the top, and so on.
2. Eight same-sizes zero matrices are created for storing results. Their sizes are as same as each of uploaded NDVI satellite imagery. It is presumed that the uploaded files have the same size because all of them have a same source.
3. Each time the PolyTrend function (image level) picks out a series of elements from the big matrix and stacks them to form a column vector. The element from the earliest data located at the top, and so on. All the elements represent the same location of the study area.
4. For assessing whether all the elements are qualified data. The column vector is inverted into a row vector and judged by two subfunctions. The first subfunction checks
   - whether any element has a negative value that represents water, snow and clouds (Requirement 8), and
   - whether the values of this vector do not change evidently (Requirement 9).

   If this vector fulfills at least one criterion then the pixel value will be classified as ***unqualified*** data. If the row vector passes this test, it will be sent to the second subfunction.
5. The second subfunction checks whether all elements of the row vector are in the range of desired NDVI input to be processed (Requirement 2). The range can be either the default range (i.e. [0, 1]) or the customized range made by the user (Section 4.8.1). If the row vector does not satisfy this criterion, it will be classified as ***excluded*** data. Otherwise it will be inverted to a column vector again and processed by the PolyTrend function (pixel level).
6. The qualified column vector's mean, maximum, minimum NDVI values and differences of NDVI values are computed by the MATLAB built-in functions. The values of trend type, slope, significance and direction are computed by the original PolyTrend function (pixel level). The computation is carried out on-the-fly and no new vectors will be formed. As soon as the computation results come out, the eight matrices created in step 2 will capture their corresponding elements and assign them with a location that is as same as they used to be in the original data. If the locations of the resulting matrices contain *unqualified* or *excluded* data, then these locations will be filled with distinctive negative values (i.e. -12000 for *unqualified* data and -11000 for *excluded* data).

7. After all elements in the big matrix are traversed, the eight matrices are full. They are plotted and saved as images. Matrices per se are also saved as eight text files and zipped as a folder. Both the images and the text files are saved in the file system.
8. Finally, all the uploaded NDVI time-series data is deleted by MATLAB for preparing receiving a new time-series NDVI dataset.

4.8.4 Returning results of the PolyTrend function (image level)

After the execution of PolyTrend function (image level), the Python function *result* returns a webpage to display the results along with explanation. The image files will be displayed by Django. Their path should be configured in *settings.py* in advance. The zipped folder containing matrices in text form is available by a traditional hyperlink of HTML. Thus Requirements 4, 5 and 6 are satisfied.

4.8.5 Controlling accessibility

Due to the limitation of time, currently this web-based system cannot process concurrent requests. This means for a single web browser only one tab is allowed to open the PolyTrend web-based system. For reaching this goal, *entry.html* is added at the top of the index page *base.html* to control accessibility by incorporating the index page and the resulting page as its inner frame. If the accessibility of the outer frame *entry.html* changes the status of the whole web-based system changes.

To record the accessible status of the PolyTrend web-based system, *HTML local storage* is used. Comparing with cookies, it can store more information (≥ 5MB) and the stored information is never sent to the server (W3schools 2016). The status will be changed from free to busy once a certain tab is showing any content of the system. During this time, if other tabs try to open this system, they will be redirected to an information page *busy.html*. Once no tabs open the system, the status will change from busy to free again.

5 Testing and evaluation

5.1 Validating the PolyTrend function (image level)

For fulfilling Requirement 3 the correctness of the PolyTrend function (image level) should be validated. To carry out the validation, it was sufficient to show that the matrix of trend types could be produced correctly. This is because for the output arguments of the original PolyTrend function (pixel level) the trend type is one element of the output arguments. In addition, MATLAB has built-in functions to calculate mean, minimum, maximum values and difference of a certain vector.

The sample input data (Section 3.2) could be used for validation. Each of them is a 25 x 1 column vectors so they could be viewed as six NDVI datasets in a 25-year time span. To form 25

matrices representing NDVI data, these six vectors were merged into a big matrix by duplicating and concatenating for gaining an area larger than 1 pixel. After that, they were divided for representing multiple years. In addition, because the looping mechanism also needed testing, for each matrix both the size of row and the size of the column would be larger than 1. As a result, the size 2 x 6 was used. For the first row of each matrix, from left to right, the elements represented cubic, quadratic, linear, no-trend, concealed (of cubic) and concealed (of quadratic) trend type. For the second row of each matrix, the order of the elements was reversed.

If the PolyTrend function (image level) was correct, according to the description of the PolyTrend function (pixel level) (Section 3.1) the validation result would be same as Figure 5. The MATLAB code for validating the algorithm is in Appendix C. The real result of running the validation code is shown in Figure 6. Thus the correctness of the PolyTrend function (image level) was proved.

3	2	1	0	−1	−1
−1	−1	0	1	2	3

3	2	1	0	−1	−1
−1	−1	0	1	2	3

Figure 5(left) Presumed result of validating the correctness of the PolyTrend function (image level). Figure 6(right) Real result of running the validation code.

5.2 Evaluating the usability

The usability of the PolyTrend web-based system is evaluated in the form of handing out questionnaires. Before carrying out the evaluation, the web-based system was published using the Apache HTTP Server. The PolyTrend web-based system is accessible by entering http://polytrend.gis.lu.se/ in web browsers.

A questionnaire (shown in Appendix F) was designed according to Requirement 14. It has nine questions and can be divided into two parts. The first part aims at gathering basic information of the background knowledge of the respondents and the information of the devices they used to access this system. The second part of the questionnaire aims at gathering users' experience in using this system by asking them to accomplish a task.

Seven respondents were invited to the survey. All of them could use the PolyTrend web-based system successfully and return valid answers. Among them, four of them have geo-related main study fields, while others have a main study field in economics. Table 7 and Table 8 list part of answers of the respondents.

Table 7 Basic information of the respondents and answers of Question 2 to 4

Respondents	Main study field	Knowledge on GIS/RS/NDVI/ vegetation trends?	Screen resolution (px)	Browser
No 1	GIS	All	2880 x 1800	Chrome
No 2	GIS	GIS, RS, NDVI	1280 x 800	Chrome
No 3	GIS	GIS, RS, NDVI	1920 x 1200	Chrome
No 4	hydrology	GIS	1366 x 768	Chrome
No 5	economics	RS	1366 x 768	Chrome
No 6	economics	GIS	1366 x 768	Chrome
No 7	economics	None	2500 x 1600	Chrome

Table 8 Answers of Question 5 to 8

Respondents	Time to fulfill the task?	Easy-to-understand information of the index?*	Easy-to-understand information of the results?*	Easy-to–use functionalities?*
No 1	3 min 13 sec	2	2	2
No 2	2 min 26 sec	1	1	1
No 3	4 min 5 sec	4	4	2
No 4	3 min 35 sec	3	2	1
No 5	5 min 4 sec	3	3	4
No 6	5 min 13 sec	4	3	3
No 7	3 min 36 sec	3	3	1

* 1= very easy, 5 = very hard, and so on.

Question 9 asked the participants to give comments on the PolyTrend web-based system. Four of them thought this system was easy-to-use. Three of them suggested inserting graphs on introducing the theory of the PolyTrend algorithm. One suggested the button *Run PolyTrend* should be larger and should be close to the other three buttons in the interface.

5.3 Pilot area

After the PolyTrend web-based system was developed, a time-series NDVI satellite imagery representing a pilot area in the Yesanpo National Park was used to evaluate the PolyTrend web-based system.

The Yesanpo National Park is located in Hebei Province, China and is about 100km west of Beijing (e.g. Yesanpo 2011; NationalParkOfChina 2016). It is a mountainous tourist site with karst topography. The area of the park is 520km^2 and the elevation of the highest peak is 1983m (e.g. NationalParkOfChina 2016). The park is situated in a region with a humid continental climate. The average temperature of a year is 12 °C and the frost free period is 165 days (e.g. Tao

2016). The average precipitation of a year is 600mm (e.g. Tao 2016). There are various kinds of vegetation in the Yesanpo Nation Park. The types of vegetation are distinguishable according to the altitude. Regions above the sea level of 800m have deciduous trees such as elms and willows (Wang et al. 2013). Regions above the sea level of 1000m have shrubs (Wang et al. 2013). Birches and populous exist in the regions that are 1300m above the sea level (Wang et al. 2013).

Figure 7 shows the general location of the pilot area in the MODIS Sinusoidal Tile Grid by a red cross. The coordinates of the borders of the area under WGS 84 Web Mercator coordinate system are shown in Figure 8.

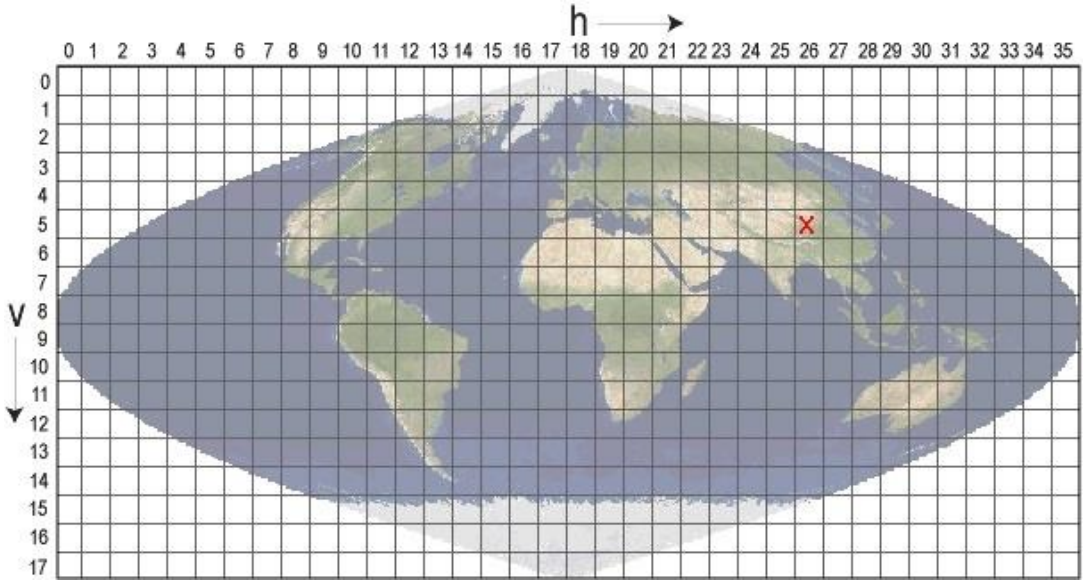

Figure 7 General location of the test area in the MODIS Sinusoidal Tile Grid (NASA 2016a)

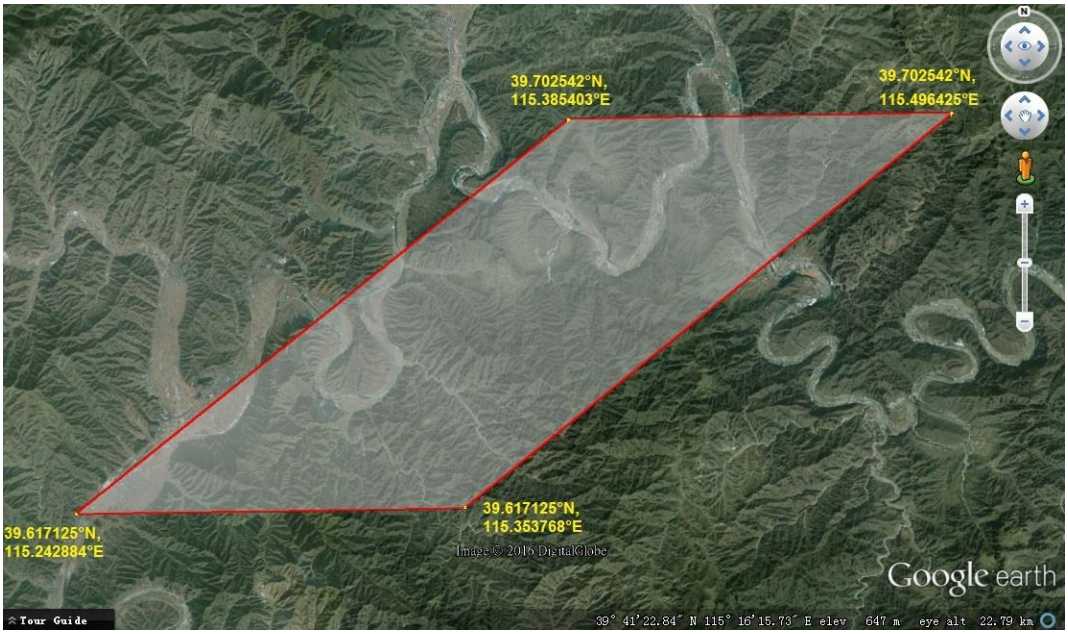

Figure 8 Coordinates of the borders of the pilot area under WGS 84 Web Mercator coordinate system

6 Results and discussion

6.1 Results

This system is capable of producing information of vegetation trends in any parts of the world. For demonstrating the system, the pilot area described in Section 5.3 was selected to produce the resulting data. All the input arguments are listed in Table 9.

The screenshots of the index and the resulting webpage of the PolyTrend web-based system are shown in Figure 9 and Figure 10. The resulting images are shown from Figure 11 to Figure 18. The ASCII files with the TXT file extension containing the resulting matrices are downloadable by clicking the hyperlink in the resulting page.

For Figure 11 to Figure 18, the pixels in the results are viewed as *unqualified* if they fulfill at least one of the following criteria.

1. For a single pixel in the results, at least one NDVI value in the whole time span is smaller than 0, which indicates the existence of water, snow and clouds (Step 4 in Section 4.8.3).
2. For a single pixel in the results, during the whole time span, the changes of NDVI values are not evident (Step 4 in Section 4.8.3).

For Figure 11 to Figure 18, the pixels in the results are viewed as *excluded* if they are qualified pixels but are out of the desired range defined by the user (Step 5 in Section 4.8.3).

Table 9 Input for the pilot area

Items	Values and description
Statistical significance	0.05
Input range of the dataset	[-10000, 10000]
Desired output range of the dataset	[0.02, 0.98]
Time-series NDVI dataset	Product: MODIS MOD13Q1 Location centered on: 39.659792 N, 115.369436 E Size: 10.25 km x 10.25 km Time period: February 18, 2000 to April 06, 2016 Temporal resolution: 16 days

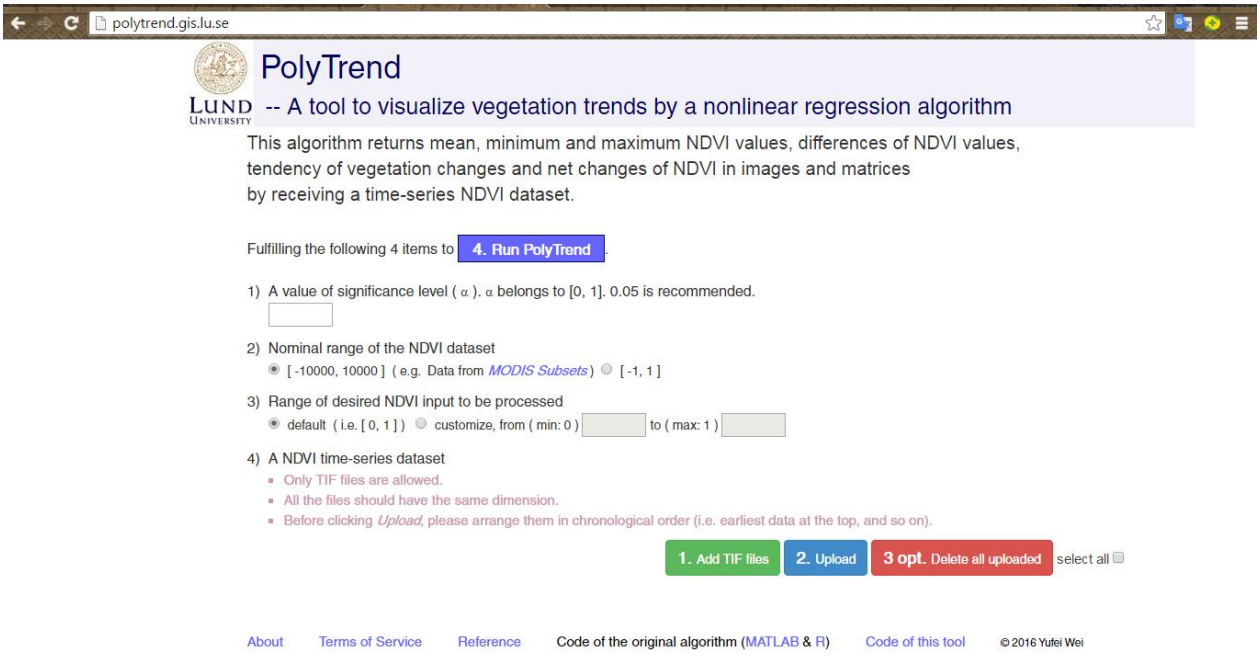

Figure 9 Screenshot of the index of the PolyTrend web-based system

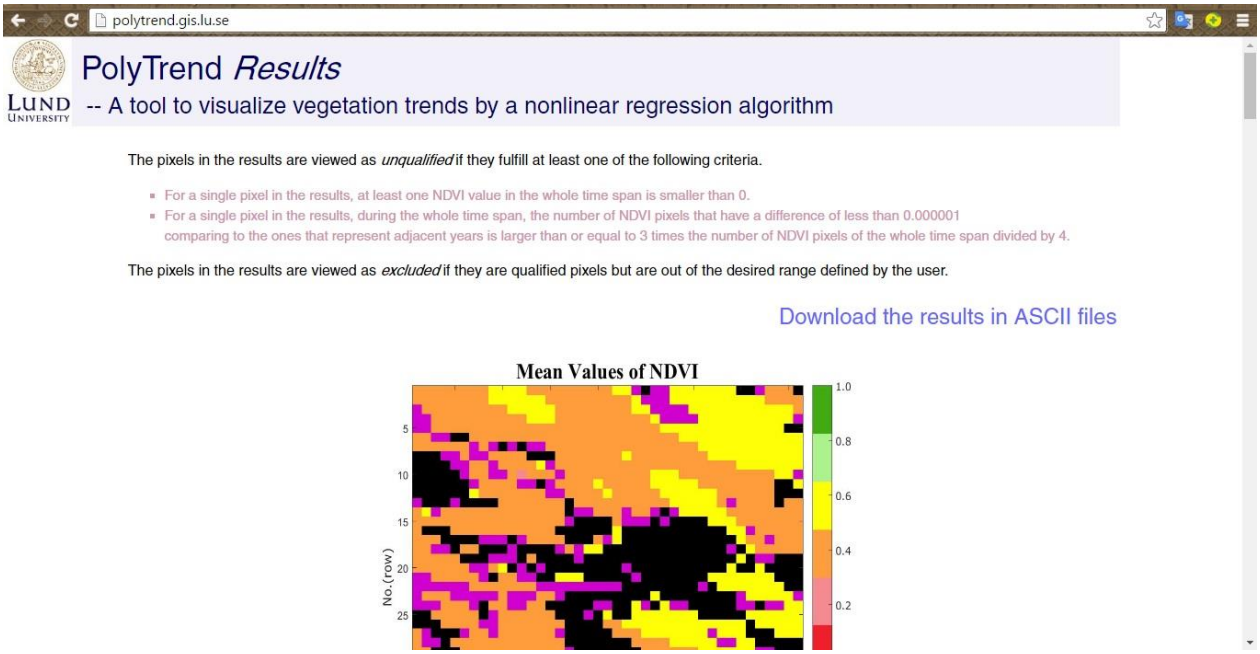

Figure 10 Screenshot of the resulting page of the PolyTrend web-based system

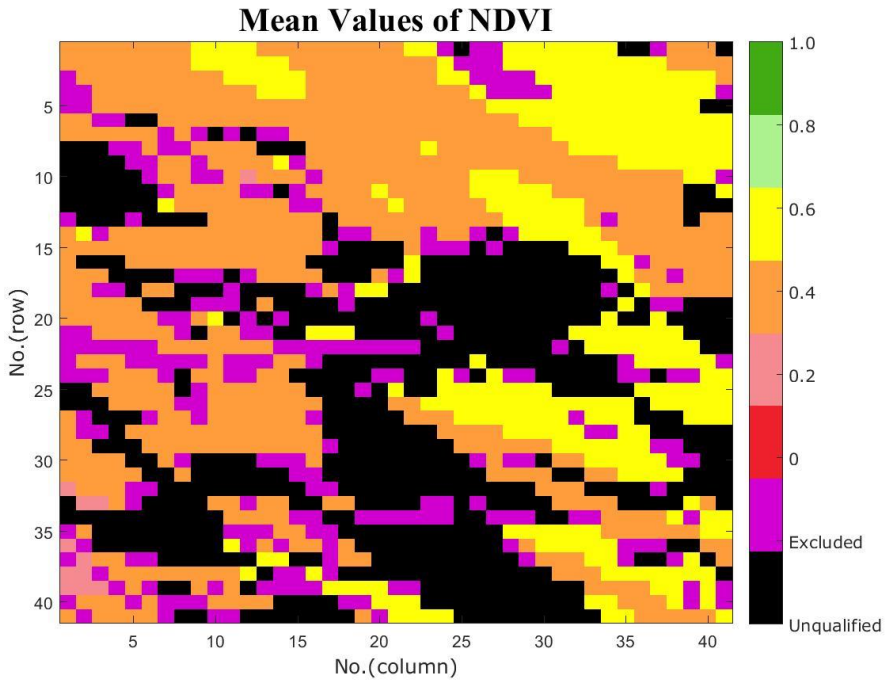

Figure 11 Mean values of NDVI of the time-series dataset

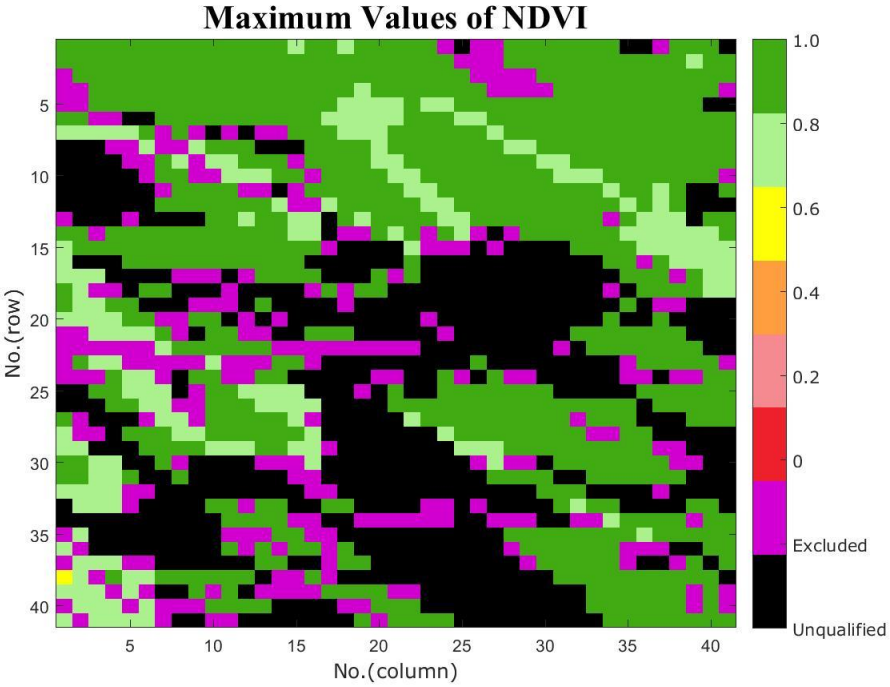

Figure 12 Maximum values of NDVI of the time-series dataset

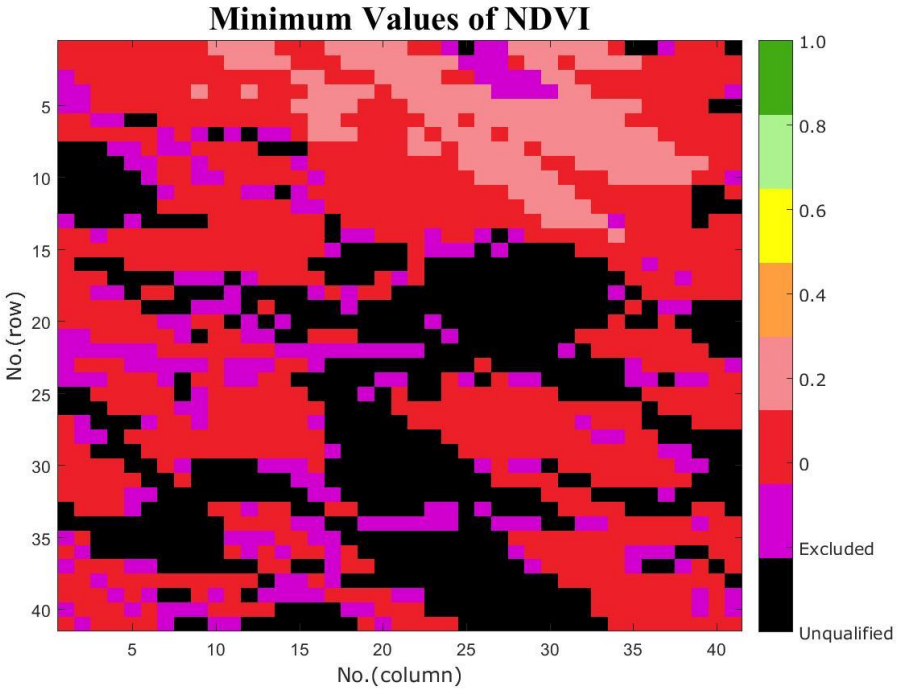

Figure 13 Minimum values of NDVI of the time-series dataset

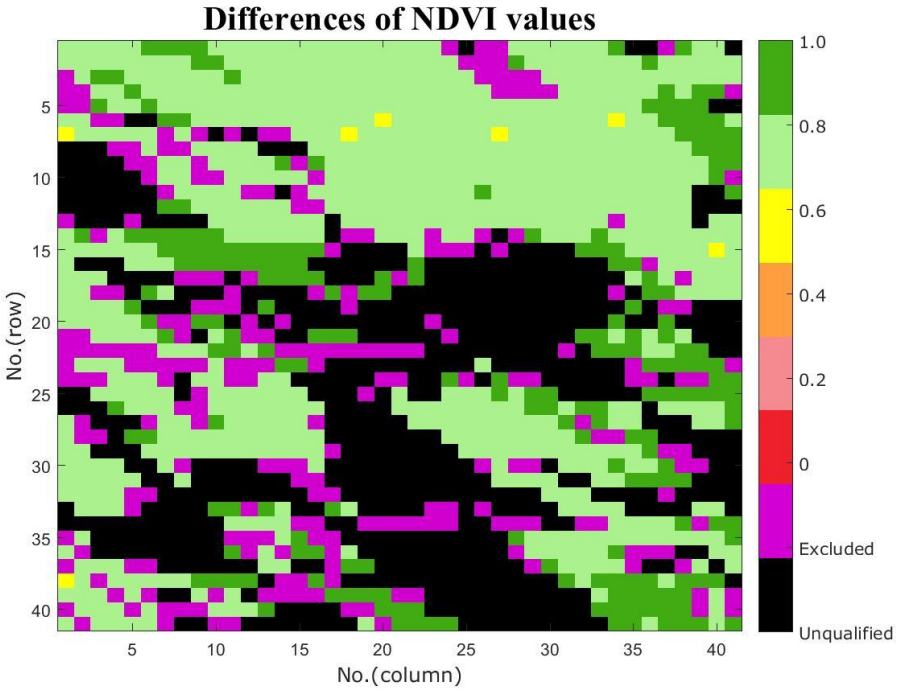

Figure 14 Differences of NDVI values of the time-series dataset

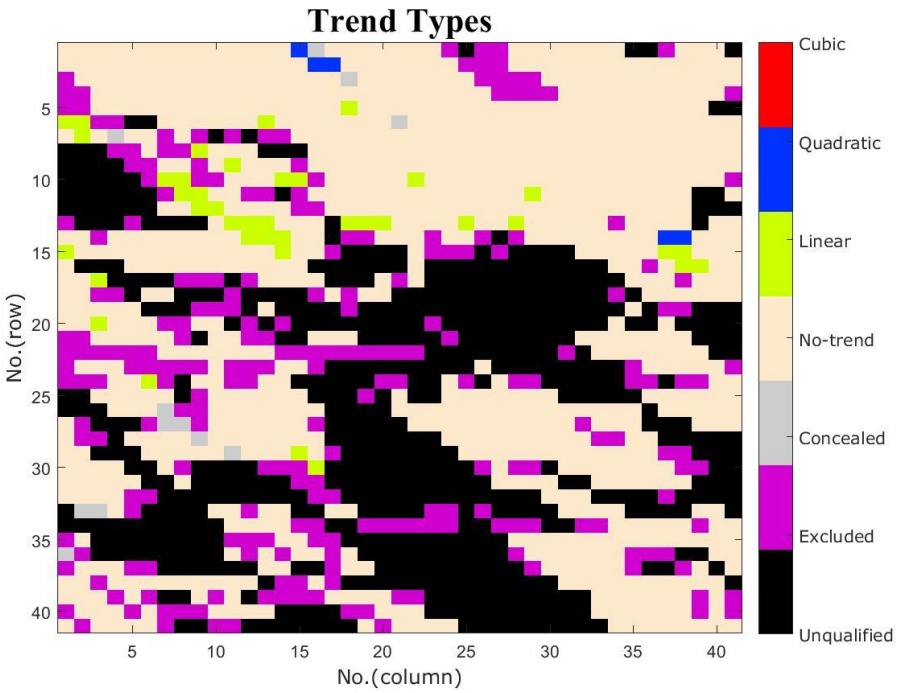

Figure 15 Trend types of the time-series NDVI dataset

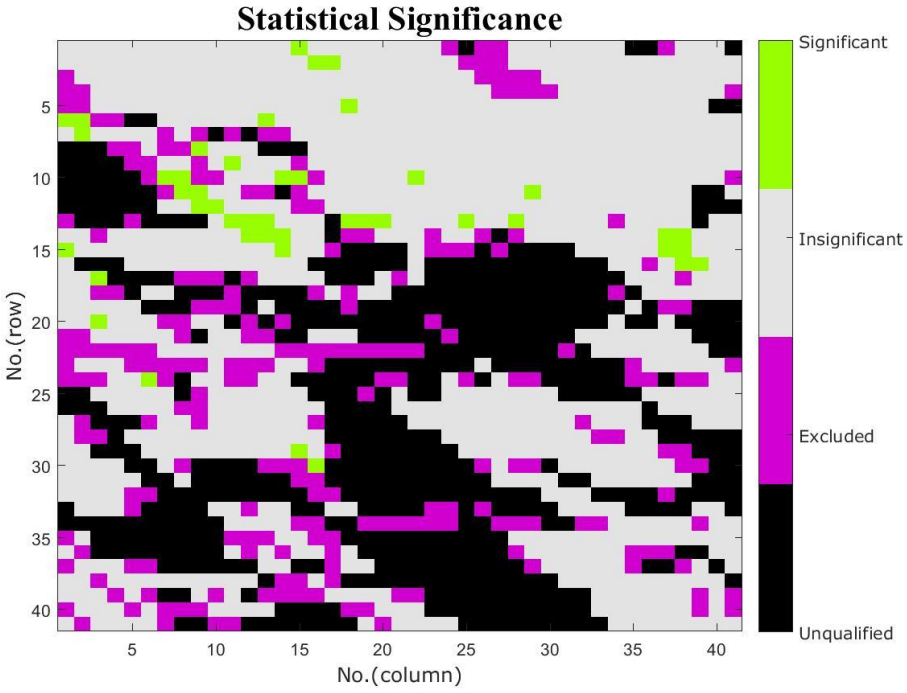

Figure 16 Statistical significance of the time-series NDVI dataset

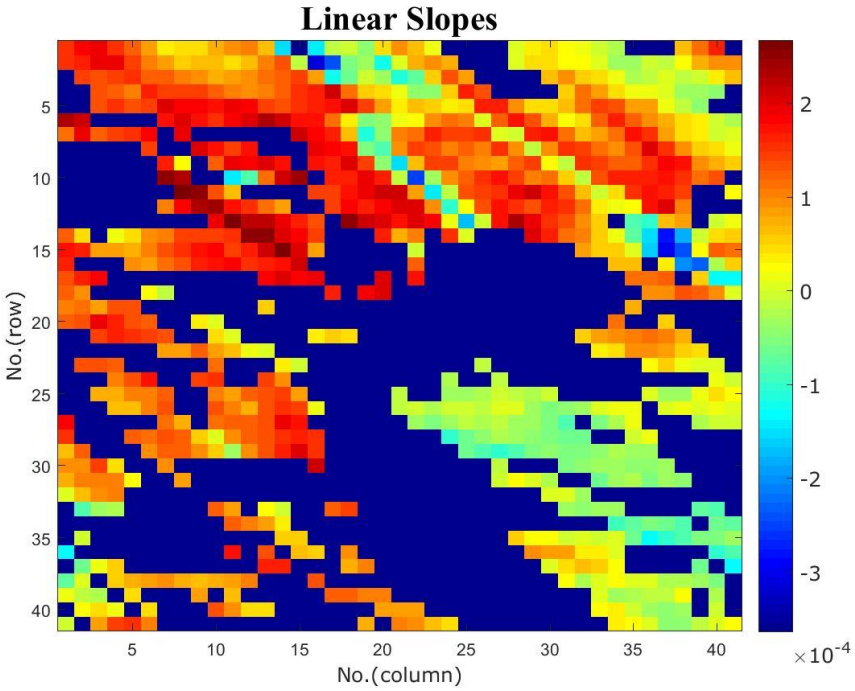

Figure 17 Linear slopes of the time-series NDVI dataset

The extreme dark-blue pixels refer to *unqualified* and/or *excluded* pixels.

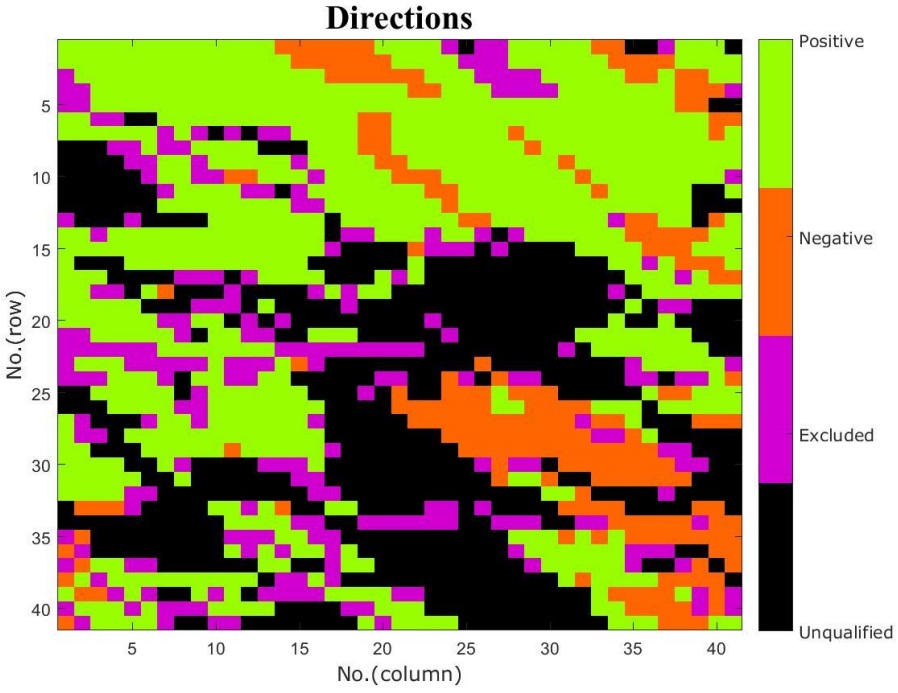

Figure 18 Directions of the time-series NDVI dataset

6.2 Discussion

The first aim of this project is to modify the existing PolyTrend algorithm for processing and visualizing satellite imagery containing NDVI values. However, currently this system can only process time-series NDVI dataset with the TIF file extension. This is because the PolyTrend function (image level) can only read a certain kind of file and convert them to matrices in sequence. A possible solution of solving this problem is using the Geospatial Data Abstraction Library (GDAL) (GDAL 2016c). This open source library can read, write and convert different kinds of raster geospatial data including GeoTIFF, Erdas Imagine and JPEG2000 (GDAL 2016c). It is a cross-platform library that supports Linux, MacOS X and Windows (GDAL 2016d) and can be installed as a library of programming languages such as C, C++, Python and Java (GDAL 2016a; GDAL 2016b). By using GDAL to convert multiple types of uploaded raster images to a single format (e.g. TIF), the PolyTrend web-based system can process time-series NDVI data from more sources and thus improve its usability.

The second aim is to visualize the results produced by the PolyTrend algorithm (image level). The architecture of the PolyTrend system is designed to return the binary files produced by MATLAB by "pasting" them in the webpage. The way of generating the ASCII files is similar. However, this way has a disadvantage. It cannot handle concurrent requests because the resulting data stored in the folder will be overwritten by MATLAB every time the PolyTrend function (image level) runs. A possible solution is to enhance the *Django-jfu* application to store each group of uploaded files into separate folders. These separate folders will be arranged according to their creation time by Python. After that, each time Python will collect all the files of the foremost folder and send them to MATLAB to process. The results could then reach users in the form of email.

The third aim is to evaluate this system by a time-series NDVI satellite imagery of a pilot area and evaluate the PolyTrend web-based system in terms of functional requirements, nonfunctional requirements and usability. Section 6.1 shows the PolyTrend web-based system functions well by receiving the time-series NDVI dataset of the pilot area. Requirements 1 to 12 are fulfilled during the development. Requirement 13 has been reached since both the used web browser during development and the web browsers used by all respondents are Chrome. In addition, several tests have confirmed that the system also functions well using Firefox and Opera. However, the number of the uploaded files cannot be returned by Internet Explorer (IE). This is because IE does not support Ajax requests well (e.g. Mombrea 2012; jQuery 2013). Possible solutions may include extending the existing Ajax function or creating a separate function for Internet Explorer (Stackoverflow 2016). Requirement 14 was assessed during the survey. Data in Table 8 shows most of the respondents thought this system has easy-to-use functionalities. However, for respondents whose main study fields were not geo-related, the information on the system is hard to understand. They also spent more time on using this system. A possible solution is adding brief introduction of NDVI and a graph of the main idea of the PolyTrend algorithm. It can be concluded that the PolyTrend web-based system provides easy-to-

identify text under different screen resolutions because the data in Table 7 and Table 8 does not indicate that the screen resolutions influence the respondents' time spent on fulfilling the task.

When counting the time that each respondent spent on the system during the survey, the time of producing the results were also included. However, this part of time is hard to study further. This is because the time for generating the resulting data is mainly decided by MATLAB. There are many factors that influence the running speed of MATLAB, such as the version of MATLAB (MATLAB Answers 2016b), environment (MathWorks 2016e), MATLAB code (MathWorks 2016e) and the performance of the computer where MATLAB was installed. Other factors also influence the speed, such as the code structure of the webpages, the way of referring images, the performance of the server (Aragon 2013) and the geodesic distances between the client and the server.

6.2.1 Future improvements

According to the architecture of the PolyTrend web-based system, the results of the evaluation and the practice of using and testing the system, some future improvements can be applied on the PolyTrend web-based system. They are listed below.

1. Increasing its capability to process multi-format dataset by using GDAL.
2. Adding the function of handling concurrent requests by changing its architecture.
3. Refining buttons to make them easy-to-identify.
4. Adding user authentication to improve the level of security.
5. To increase the interest of users, a counter that records how many visitors have visited this system can be added on the index.

7 Conclusions

The aim of this project is developing a web-based system to visualize vegetation trends by the PolyTrend algorithm based on functional and nonfunctional requirements with a focus on usability.

The original PolyTrend algorithm has been modified to receive time-series NDVI satellite imagery, filter unqualified data, allow users to define the range of desired NDVI input to be processed and return the information of vegetation trends at image level.

The enhanced PolyTrend algorithm has been integrated into web environment using the web framework Django and the programming language Python. The PolyTrend web-based system has proved its accessibility during the evaluation of the usability.

The system functions well by receiving a time-series NDVI dataset of a pilot area. During development, all the functional and nonfunctional requirements have been reached. According to the feedback of the evaluation, the PolyTrend web-based system has easy-to-use functionalities although its contents were more difficult to understand for people whose main study fields were

not geo-related. The system also needs to improve its adaptability, robustness and attraction in the future.

Equipped with a nonlinear regression algorithm that can provide more accurate information of vegetation changes than regular linear regression algorithms, the PolyTrend web-based system provides an accessible way of monitoring global vegetation trends through the Internet.

References

Adobe. 2016. Adobe website. Retrieved 16 March, 2016, from http://www.adobe.com/

An, Y.Z., W. Gao, Z.Q. Gao, C.S. Liu, and R.H. Shi. 2015. Trend analysis for evaluating the consistency of Terra MODIS and SPOT VGT NDVI time series products in China. *Frontiers of Earth Science* 9: 125-136. DOI: 10.1007/s11707-014-0428-9

Andersson, M., J. Cao, M. Kihl, and C. Nyberg. 2003. Performance modeling of an Apache web server with busty arrival traffic. In *IC03: proceedings of the international conference on internet computing*, 508-511. Georgia: CSREA Press.

Apache. 2016. Apache HTTP Server Project. Retrieved 9 June, 2016, from https://httpd.apache.org/

Apple. 2016. Apple website. Retrieved 16 March, 2016, from http://www.apple.com/

Aragon, K. 2013. 10 Ways to Speed Up Your Website - and Improve Conversion by 7%. Retrieved 1 May, 2016, from https://blog.crazyegg.com/2013/12/11/speed-up-your-website/

Attaway, S. 2013. *MATLAB: A Practical Introduction to Programming and Problem Solving (3rd Edition)*, 207 pp. Oxford: Butterworth-Heinemann.

Awazu,Y., and K.C. Desouza. 2004. Open knowledge management: Lessons from the open source revolution. *Journal of the American Society for Information Science and Technology* 55: 1016-1019. DOI: 10.1002/asi.20050

Bacchini, R.D., and D.F. Miguez. 2015. Agricultural risk management using NDVI pasture index-based insurance for livestock producers in south west Buenos Aires province. *Agricultural Finance Review* 75: 77-91. DOI: 10.1108/AFR-12-2014-0044

BFAST. 2016. Welcome to Breaks For Additive Season and Trend project! Retrieved 9 June, 2016, from http://bfast.r-forge.r-project.org/

Bhandari, S., P. Stuart, and G. Tony. 2011. Assessing viewing and illumination geometry effects on the MODIS vegetation index (MOD13Q1) time series: implications for monitoring phenology and disturbances in forest communities in Queensland, Australia. *International Journal of Remote Sensing* 32: 7513-7538. DOI: 10.1080/01431161.2010.524675

Bing. 2016. Bing website. Retrieved 16 March, 2016, from http://www.bing.com/

Black Duck. 2016. Top 20 open source licenses. Retrieved 17 March, 2016, from https://www.blackducksoftware.com/resources/data/top-20-open-source-licenses

Bodzin, A.M., and Q. Fu. 2014. The effectiveness of the geospatial curriculum approach on urban middle-level students' climate change understandings. *Journal of Science Education and Technology* 23: 575-590. DOI: 10.1007/s10956-013-9478-0

Burk, S. 2010. Functional requirements and nonfunctional requirements in software engineering explained. Retrieved 25 April, 2016, from http://searchsoftwarequality.techtarget.com/answer/Functional-requirements-and-nonfunctional-requirements-in-software-engineering-explained

CakePHP. 2016. CakePHP website. Retrieved 9 June, 2016, from http://cakephp.org/

Cao, R., W.G. Jiang, L.H. Yuan, W.J. Wang, Z.L. Lv, and Z.Chen. 2014. Inter-annual variations in vegetation and their response to climatic factors in the upper catchments of the Yellow River from 2000 to 2010. *Journal of Geographical Science* 24: 963-979. DOI: 10.1007/s11442-014-1131-1

Chambers, M., and T.W. Dinsmore. 2014. *Advanced Analytics Methodologies: Driving Business Value with Analytics*, 149 pp. New Jersey: Pearson FT Press.

Chapin, F.S., E. Rincon, and P. Huante. 1993. Environmental responses of plants and ecosystems as predictors of the impact of global change. *Journal of Biosciences* 18: 515-524. DOI: 10.1007/BF02703083

Chappell, D., and T. Jewell. 2002. *JAVA Web services*, 6 pp. California: O'reilly.

Chi, X.N., B.C. Liu, Q. Niu., and Q.X. Wu. 2012. Web load balance and cache optimization design based Nginx under high concurrency environment. In *2012 Third International Conference on Digital Manufacturing & Automation*, 1029-1032. New York: IEEE 2012.

Cohn, M. 2008. Non-functional Requirements as User Stories. Retrieved 22 April, 2016, from https://www.mountaingoatsoftware.com/blog/non-functional-requirements-as-user-stories

Colazo, J., and Y. Fang. 2009. Impact of License Choice on Open Source Software Development Activity. *Journal of the American society for information science and technology* 60: 997–1011. DOI: 10.1002/asi.21039.

Craparo, R.M. 2007. Significance level. In *Encyclopedia of Measurement and Statistics 3*, ed. Salkind, N.J., 889-891 pp. California: SAGE Publications.

Creative Commons. 2016a. About the Licenses. Retrieved 17 March, 2016, from https://creativecommons.org/licenses/

Creative Commons. 2016b. Creative Commons website. Retrieved 10 June, 2016, from https://creativecommons.org/

Câmara, G., and F. Fonseca. 2008. Information policies and open source software in developing countries. *Journal of the American Society for Information Science and Technology* 58: 121-131. DOI: 10.1002/asi.20444

Dardailler, D. 2010. W3C: An Open Platform for Web Standardization. Retrieved 17 March, 2016, from https://www.w3.org/2010/06/dd-diplo.html#(7)

de Jong, R., J. Verbesselt, M. E. Schaepman, and S. de Bruin. 2012. Trend changes in global greening and browning: Contribution of short-term trends to longer-term change. *Global Change Biology* 18: 642-655. DOI: 10.1111/j.1365-2486.2011.02578.x

de Rosa, A. 2014. 10 HTML5 APIs Worth Looking Into. Retrieved 29 March, 2016, from http://www.sitepoint.com/10-html5-apis-worth-looking/

Django. 2016a. Managing static files (e.g. images, JavaScript, CSS). Retrieved 31 March, 2016, from https://docs.djangoproject.com/es/1.9/howto/static-files/

Django. 2016b. Meet Django. Retrieved 31 March, 2016, from https://www.djangoproject.com/

Eclipse. 2016. Desktop IDEs. Retrieved 31 March, 2016, from https://eclipse.org/ide/

ECMA. 2016. ECMAScript® 2015 Language Specification. Retrieved 9 June, 2016, from http://www.ecma-international.org/ecma-262/6.0/ECMA-262.pdf

Eriksson, U. 2012. Functional vs Non Functional Requirements. Retrieved 31 March, 2016, from http://reqtest.com/requirements-blog/functional-vs-non-functional-requirements/

Fensholt, R., S. Horion, T. Tagesson, A. Ehammer, K. Rasmussen, and E. Ivits. 2015. Global-scale mapping of changes in ecosystem functioning from earth observation-based trends in total and recurrent vegetation. *Global Ecology and Biogeography* 24:1003-1017. DOI: 10.1111/geb.12338

Fernandes, R., and S.G. Leblanc. 2005. Parametric (modified least squares) and non-parametric (Theil-Sen) linear regressions for predicting biophysical parameters in the presence of measurement errors. *Remote Sensing of Environment* 95: 303-316. DOI: 10.1016/j.rse.2005.01.005

Frost, J. 2015. Understanding hypothesis tests: significance levels (alpha) and p values in statistics. Retrieved 28 March, 2016, from http://blog.minitab.com/blog/adventures-in-statistics/understanding-hypothesis-tests%3A-significance-levels-alpha-and-p-values-in-statistics

GDAL. 2016a. Documentation of the API of the Java bindings for GDAL/OGR. Retrieved 1 May, 2016, from http://gdal.org/java/

GDAL. 2016b. GDAL API Tutorial. Retrieved 1 May, 2016, from http://www.gdal.org/gdal_tutorial.html

GDAL. 2016c. GDAL - Geospatial Data Abstraction Library. Retrieved 27 April, 2016, from http://www.gdal.org/

GDAL. 2016d. GDAL/OGR Binaries. Retrieved 27 April, 2016, from https://trac.osgeo.org/gdal/wiki/DownloadingGdalBinaries

Github. 2016. A Django Library for jQuery File Upload. Retrieved 1 April, 2016, from https://github.com/Alem/django-jfu

Glaser, H. 2012. Specific advantages of Python. Retrieved 31 March, 2016, from https://www.quora.com/What-are-the-advantages-of-Python-over-C++#

Google. 2016. Google Cloud Platform website. Retrieved 9 June, 2016, from https://cloud.google.com/

Google Search. 2016. Google search website. Retrieved 16 March, 2016, from https://www.google.com/

Guo, W., D.S. Lu, Y.L. Wu, and J.X. Zhang. 2015. Mapping impervious surface distribution with integration of SNNP VIIRS-DNB and MODIS NDVI Data. *Remote Sensing* 7: 12459-12477. DOI:10.3390/rs70912459

Hatfield, J.L., and J.H. Prueger. 2010. Value of using different vegetative indices to quantify agrigultural crop characteristics at different growth stages under varying management practices. *Remote Sensing* 2: 562-578. DOI: 10.3390/rs2020562

Heinsch, F.A., M.S. Zhao, S.W. Running, J.S. Kimball, R.R. Nemani, K.J. Davis, P.V. Bolstad, B.D. Cook, et al. 2006. Evaluation of remote sensing based terrestrial productivity from MODIS using regional tower eddy flux network observations. *IEEE Transactions on Geoscience and Remote Sensing* 44: 1908-1925. DOI: 10.1109/TGRS.2005.853936

Heinzl, S., and C. Metz. 2013. Toward a Cloud-ready Dynamic Load Balancer based on the Apache Web Server. In *22nd IEEE international WETICE Conference*, 342-345. New York: IEEE 2013. DOI: 10.1109/WETICE.2013.63

Herrmann, S. M., A. Anyamba, and C. J. Tucker. 2005. Recent Trends in Vegetation Dynamics in the African Sahel and Their Relationship to Climate. *Global Environmental Change* 15: 394-404. DOI: 10.1016/j.gloenvcha.2005.08.004

Hill, M.J., and G.E. Donald. 2003. Estimating spatio-temporal patterns of agricultural productivity in fragmented landscapes using AVHRR NDVI time series. *Remote Sensing of Environment* 84, 367-384. DOI: 10.1016/S0034-4257(02)00128-1

Homer, N., B. Merriman, and S.F. Nelson. 2009. BFAST: An Alignment Tool for Large Scale Genome Resequencing. *PLoS ONE* 11: 1-12.

Hou, W.J., J.B. Gao, S.H. Wu, and E.F. Dai. 2015. Interannual variations in growing-season NDVI and its correlation with climate variables in the southwestern karst region of China. *Remote Sensing* 7: 11105-11124. DOI:10.3390/rs70911105

Huang, N., Z. Niu, Y. Zhan, S. Xu, M.C. Tappert, C. Wu, W. Huang, S. Gao, et al. 2012. Relationships between soil respiration and photosynthesis-related spectral vegetation indices in two cropland ecosystems. *Agricultural and Forest Meteorology* 15: 80-89. DOI: 10.1016/j.agrformet.2012.03.005

Huete, A., C. Justice, and W. Van Leeuwen. 1999. MODIS Vegetation Index: Algorithm Theoretical Basis Document. Retrieved 17 March, 2016, from http://modis.gsfc.nasa.gov/data/atbd/atbd_mod13.pdf

Huete, A., K.Didan, T. Miura, E.P. Rodriguez, X. Gao, and L.G. Ferriera. 2002. Overview of the radiometric and biophysical performance of the MODIS vegetation index. *Remote Sensing of Environment* 83: 195-213. DOI: 10.1016/S0034-4257(02)00096-2

IETF. 2016. The Internet Engineering Task Force. Retrieved 9 June, 2016, from https://www.ietf.org/

ISO. 1998. Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability. Retrieved 29 March, 2016, from https://www.iso.org/obp/ui/#iso:std:iso:9241:-11:ed-1:v1:en

Jackson, J. 2014. Microsoft closes in on Apache Web server lead. Retrieved 16 March, 2016, from http://www.computerworld.com/article/2489229/network-servers/microsoft-closes-in-on-apache-web-server-lead.html

Jamali, S., J. Seaquist, L. Eklundh, and J. Ardö. 2014. Automated mapping of vegetation trends with polynomials using NDVI imagery over the Sahel. *Remote Sensing of Environment* 141: 79-89. DOI: 10.1016/j.rse.2013.10.019

Javier, L.P., S. Susanne, and C.B. Antonio. 2015. The role of vegetation covers on soil wetting processes at rainfall event scale in scattered tree woodland of Mediterranean climate. *Journal of Hydrology* 529: 951-961. DOI:10.1016/j.jhydrol.2015.09.018

jQuery. 2013. IE11 crashes on ajax call. Retrieved 27 April, 2016, from https://bugs.jquery.com/ticket/14655

Jönsson, P., and L. Eklundh. 2002. Seasonality extraction by function fitting to time-series of satellite sensor data. *IEEE Transactions on Geoscience and Remote Sensing* 40: 1824-1832. DOI: 10.1109/TGRS.2002.802519

Kapitsakia, G.M., N.D. Tselikasb, and I.E. Foukarakisb. 2015. An insight into license tools for open source software systems. *The Journal of Systems and Software* 102: 72-87. DOI: 10.1016/j.jss.2014.12.050

Kennedy, R.E., Z.Q. Yang, W. B. Cohen. 2010. Detecting trends in forest disturbance and recovery using yearly Landsat time series: 1. LandTrendr-Temporal segmentation algorithms. *Remote Sensing of Environment* 114: 2897-2910. DOI: 10.1016/j.rse.2010.07.008

Lakka, S., C. Michalakelis, D. Varoutas, and D. Martakos. 2012. Exploring the determinants of the OSS market potential: The case of the Apache webserver. *Telecommunications Policy* 36: 51-68. DOI:10.1016/j.telpol.2011.11.018

Lambert, J., J.P. Denux, J. Verbesselt, G. Balent, and V. Cheret. 2015. Detecting clear-cuts and decreases in forest vitality using MODIS NDVI time series. *Remote Sensing* 7: 3588-3612. DOI: 10.3390/rs70403588

LandTrendr. 2016. LandTrendr: Capturing the changing land surface. Retrieved 9 June, 2016, from http://landtrendr.forestry.oregonstate.edu/

Lawrence, R. 2004. *Open source licensing: software freedom and intellectual property law*. New Jersey: Prentice Hall.

Lee, T.B. 1998. Uniform Resource Identifiers (URI): Generic Syntax. Retrieved 17 March, 2016, from https://tools.ietf.org/html/rfc2396

Li, B., L. Zhang, Q. Yan, and Y.J. Xue. 2014. Application of piecewise linear regression in the detection of vegetation greenness trends on the Tibetan Plateau. *International Journal of Remote Sensing* 35: 1526-1539. DOI: 10.1080/01431161.2013.878066

Lillesand, T.M., R.W. Kiefer, and J.W. Chipman. 2008. *Remote sensing and image interpretation (sixth edition)*, 13 pp. New Jersey: John Wiley & Sons.

Lin, S., and R. Liu. 2016. A simple method to extract tropical monsoon forests using NDVI based on MODIS data: a case study in South Asia and Peninsula Southeast Asia. *Chinese Geographical Science* 26: 22-34. DOI: 10.1007/s11769-015-0789-3

Liu, J.H., J.J. Wua, Z.T. Wua, and M. Liu. 2013. Response of NDVI dynamics to precipitation in the Beijing-Tianjin sandstorm source region. *International Journal of Remote Sensing* 34: 5331-5350. DOI: 10.1080/01431161.2013.787505

Liu, Y., Y. Li, S.C. Li, and S. Motesharrei. 2015a. Spatial and temporal patterns of global NDVI trends: correlations with climate and human factors. *Remote Sensing* 7: 13233-13250. DOI: 10.3390/rs71013233

Liu, Y.X., X.F. Liu, Y.N. Hu, S.S. Li, J. Peng, and Y.L. Wang. 2015b. Analyzing nonlinear variations in terrestrial vegetation in China during 1982-2012. *Environmental Monitoring and Assessment* 187: 722-735. DOI: 10.1007/s10661-015-4922-7

Long, J. 2012. I Don't Speak Your Language: Frontend vs. Backend. Retrieved 28 March, 2016, from http://blog.teamtreehouse.com/i-dont-speak-your-language-frontend-vs-backend

Lu, L., and H. Guo. 2008. Wheat phenology extraction from time-series of SPOT/VEGETATION data. In *Proceedings - 1st International Congress on Image and Signal Processing*. DOI: 10.1109/CISP.2008.104

Lutz, M. 2009a. *Learning Python (4th Edition)*, 4 pp. California: O'reilly.

Lutz, M. 2009b. *Learning Python (4th Edition)*, 63 pp. California: O'reilly.

Lutz, M. 2011. *Programming Python (4th Edition)*, 777 pp. California: O'reilly.

Ma, B.L., M.H. Morrision, and L.M. Dwyer. 1996. Canopy light reflectance and field greenness to assess nitrogen fertilization and yield of corn. *Agronomy Journal* 88: 915-920. DOI: 10.2134/agronj1996.00021962003600060011x

Mallegowda, P., G. Rengaian, J. Krishnan, and M. Niphadkar. 2015. Assessing habitat quality of forest-corridors through NDVI analysis in dry tropical forests of South India: implications for conservation. *Remote Sensing* 7: 1619-1639. DOI: 10.3390/rs70201619

MathWorks. 2016a. Call User Script and Function from Python. Retrieved 25 April, 2016, from http://se.mathworks.com/help/matlab/matlab_external/call-user-script-and-function-from-python.html

MathWorks. 2016b. Install MATLAB Engine API for Python. Retrieved 31 March, 2016, from http://se.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html

MathWorks. 2016c. Integrate a Python Package. Retrieved 31 March, 2016, from http://se.mathworks.com/help/compiler_sdk/python/integrate-a-python-package.html

MathWorks. 2016d. MATLAB - The Language of Technical Computing. Retrieved 9 June, 2016, from http://se.mathworks.com/products/matlab/index.html?s_tid=gn_loc_drop

MathWorks. 2016e. Techniques to Improve Performance. Retrieved 27 April, 2016, from http://www.mathworks.com/help/matlab/matlab_prog/techniques-for-improving-performance.html?s_tid=gn_loc_drop

MATLAB Answers. 2016a. List of files sorting. Retrieved 2 April, 2016, from http://www.mathworks.com/matlabcentral/answers/13978-list-of-files-sorting

MATLAB Answers. 2016b. What are the benefits of 64-bit MATLAB versus 32-bit MATLAB? Retrieved 27 April, 2016, from http://se.mathworks.com/matlabcentral/answers/92518-what-are-the-benefits-of-64-bit-matlab-versus-32-bit-matlab?requestedDomain=www.mathworks.com

Meroni, M., D. Fasbender, R. Balaghi, M. Dali, M. Haffani, I. Haythem, J. Hooker, M. Lahlou, et al. 2016. Evaluating NDVI data continuity between SPOT-VEGETATION and PROBA-V missions for operational yield forecasting in North African countries. *IEEE Transactions on Geoscience & Remote Sensing* 54: 795-804. DOI: 10.1109/TGRS.2015.2466438

Microsoft. 2016. IIS website. Retrieved 9 June, 2016, from http://www.iis.net/

Mishra, N.B., K.A. Crews, N. Neeti, T. Meyer, and K.R. Young. 2015. MODIS derived vegetation greenness trends in African Savanna: Deconstructing and localizing the role of

changing moisture availability, fire regime and anthropogenic impact. *Remote Sensing of Environment* 169: 192-204. DOI: 10.1016/j.rse.2015.08.008

mlabwrap. 2011. mlabwrap v1.1. Retrieved 31 March, 2016, from http://mlabwrap.sourceforge.net/

Mombrea, M. 2012. AJAX requests not executing or updating in Internet Explorer? Here's a solution. Retrieved 27 April, 2016, from http://www.itworld.com/article/2693447/ajax-requests-not-executing-or-updating-in-internet-explorer-solution.html

Mozilla. 2016. Mozilla website. Retrieved 16 March, 2016, from https://www.mozilla.org/en-US/firefox/new/#

MySQL. 2016. MySQL website. Retrieved 9 June, 2016, from https://www.mysql.com/

NASA. 2016a. Global Subsetting Tool: Data Visualization and Download. Retrieved 9 June, 2016, from http://daacmodis.ornl.gov/glb_viz_3/08Jun2016_16:03:52_414464726L39.659792L115.369436S41L41_MOD13Q1/index.html

NASA. 2016b. Global Subsetting Tool: MODIS Land Products. Retrieved 17 March, 2016, from http://daacmodis.ornl.gov/cgi-bin/MODIS/GLBVIZ_1_Glb/modis_subset_order_global_col5.pl

NationalParkOfChina. 2016. Yesanpo-National Park of China. Retrieved 10 June, 2016, from http://www.nationalparkofchina.com/yesanpo.html

Netcraft. 2014. May 2014 Web Server Survey. Retrieved 16 March, 2016, from http://news.netcraft.com/archives/2014/05/07/may-2014-web-server-survey.html

Netcraft. 2016. February 2016 Web Server Survey. Retrieved 16 March, 2016, from http://news.netcraft.com/archives/2016/02/22/february-2016-web-server-survey.html

Nginx. 2016. Nginx website. Retrieved 9 June, 2016, from https://nginx.org/en/

Nielsen, J. 2012. Usability 101: Introduction to Usability. Retrieved 29 March, 2016, from https://www.nngroup.com/articles/usability-101-introduction-to-usability/

Nielsen, J., and D. Norman. 2000. Usability Is Not a Luxury. Retrieved 29 March, 2016, from http://www.jnd.org/dn.mss/usability_is_not_a_l.html

OGC. 2016a. OGC website. Retrieved 9 June, 2016, from http://www.opengeospatial.org/

OGC. 2016b. Web Coverage Service. Retrieved 9 June, 2016, from http://www.opengeospatial.org/standards/wcs

OGC. 2016c. Web Feature Service. Retrieved 9 June, 2016, from http://www.opengeospatial.org/standards/wfs

OGC. 2016d. Web Map Service. Retrieved 9 June, 2016, from http://www.opengeospatial.org/standards/wms

Open Source Initiative. 2016a. Apache License, Version 2.0. Retrieved 17 March, 2016, from https://opensource.org/licenses/Apache-2.0

Open Source Initiative. 2016b. GNU General Public License, version 2 (GPL-2.0). Retrieved 17 March, 2016, from https://opensource.org/licenses/GPL-2.0

Open Source Initiative. 2016c. The MIT License (MIT). Retrieved 17 March, 2016, from https://opensource.org/licenses/MIT

Oracle. 2016a. Java website. Retrieved 9 June, 2016, from https://java.com/en/

Oracle. 2016b. Oracle database website. Retrieved 9 June, 2016, from http://www.oracle.com/technetwork/database/enterprise-edition/overview/index.html

Ottinger, J. 2008. What is an App Server? Retrieved 28 March, 2016, from http://www.theserverside.com/news/1363671/What-is-an-App-Server

Pandey, P C., V.P. Mandal, S. Katiyar, P. Kumar, V. Tomar, S. Patairiya, N. Ravisankar, and B. Gangwar. 2015. Geospatial approach to assess the impact of nutrients on rice equivalent yield using MODIS sensors'-based MOD13Q1-NDVI data. *IEEE Sensors Journal* 15: 6108-6115. DOI: 10.1109/JSEN.2015.2451113

Pang, J.P., X.F. Wen, and X.M. Sun. 2016. Mixing ratio and carbon isotopic composition investigation of atmospheric CO2 in Beijing, China. *Science of the Total Environment* 539: 322-330. DOI: 10.1016/j.scitotenv.2015.08.130

Paypal. 2016. Paypal website. Retrieved 16 March, 2016, from https://www.paypal.com/us/webapps/mpp/home

PHP. 2016. PHP website. Retrieved 9 June, 2016, from http://php.net/

PostgreSQL. 2016. PostgreSQL website. Retrieved 9 June, 2016, from https://www.postgresql.org/

PyDev. 2016. PyDev website. Retrieved 31 March, 2016, from http://www.pydev.org/

pymatlab. 2016. pymatlab 0.2.3. Retrieved 31 March, 2016, from https://pypi.python.org/pypi/pymatlab

pymat2. 2016. pymat2 website. Retrieved 31 March, 2016, from https://code.google.com/archive/p/pymat2/

Python. 2016. Python website. Retrieved 9 June, 2016, from https://www.python.org/

Python 3.4.4 Documentation. 2016. Miscellaneous operating system interfaces. Retrieved 2 April, 2016, from https://docs.python.org/3.4/library/os.html

Qin, H.Z., A.T.K. Wan, and G.H. Zou. 2009. On the sensitivity of the one-sided t test to covariance misspecification. *Journal of Multivariate Analysis* 100: 1593-1609. DOI: 10.1016/j.jmva.2009.01.003

Reed, B.C., J.F. Brown, D. VanderZee, T.R. Loveland, J.W. Merchant, and D.O. Ohlen. 1994. Measuring phenological variability from satellite imagery. *Journal of Vegetation Science* 5: 703-714.

Rouse, J.W., R.H. Haas, J.A. Schell, and D.W. Deering. 1973. Monitoring vegetation systems in the Great Plains with ERTS. *Third Earth Resources Technology Satellite - 1 Symposium - Volume I: Technical Presentations. NASA SP*-351: 309-317. Washington DC: NASA.

Sakamoto, T., M. Yokozawa, H. Toritani, M. Shibayama, N. Ishitsuka, and H. Ohno. 2005. A crop phenology detection method using time-series MODIS data. *Remote Sensing of Environment* 96: 366-374. DOI: 10.1016/j.rse.2005.03.008

Sen, R. 2007. A strategic analysis of competition between open source and proprietary software. *Journal of Management Information Systems* 24: 233-257. DOI: 10.2753/MIS0742-1222240107

Shanahan, J.F., J.S. Schepers, D.D. Francis, G.E. Varvel, W. Wilhelm, J.M. Tringe, M.R. Schlemmer, and D.J. Major. 2001. Use of remote-sensing imagery to estimate corn grain yield. *Agronomy Journal* 93: 583-589.

Shanahan, J.F., K.H. Holland, J.S. Schepers, D.D. Francis, M.R. Schlemmer, and R. Caldwell. 2003. Use of a crop canopy reflectance sensor to assess corn leaf chlorophyll content. *American Society of Agronomy special publication* 66: 135-150. DOI: 10.2134/asaspecpub66.c11

Shao, Y., R.S. Lunetta, B. Wheeler, J.S. Iiames, and J.B. Campbell. 2016. An evaluation of time-series smoothing algorithms for land-cover classifications using MODIS-NDVI multi-temporal data. *Remote Sensing of Environment* 174*:* 258-265. DOI:10.1016/j.rse.2015.12.023

Sheena. 2015. Python Framework Comparison: Django vs. Pyramid. Retrieved 31 March, 2016, from https://www.codementor.io/python/tutorial/django-vs-pyramid-python-framework-comparison

Sikos, L.F. 2011. *Web Standards - Mastering HTML5, CSS3, and XML*. New York: Apress.

Skype. 2016. Skype website. Retrieved 16 March, 2016, from http://www.skype.com/en/

Snijder, R. 2015. Better Sharing Through Licenses? Measuring the Influence of Creative Commons Licenses on the Usage of Open Access Monographs. *Journal of Librarianship & Scholarly Communication* 3: 1-21. DOI: 10.7710/2162-3309.1187

Solari, F., J. Shanahan, R.B. Ferguson, J.S. Schepers, and A.A. Gitelson. 2008. Active sensor reflectance measurements to corn nitrogen status and yield potential. *Agronomy Journal* 100: 571-579.

Spring. 2016. Spring website. Retrieved 9 June, 2016, from https://spring.io/

Sproull, N.L. 2002. *Handbook of Research Methods: A Guide for Practitioners and Students in the Social Sciences (2nd Edition)*, 49-64 pp. Maryland: Scarecrow Press.

Stackoverflow. 2016. IE not triggering jQuery Ajax success. Retrieved 27 April, 2016, from http://stackoverflow.com/questions/782532/ie-not-triggering-jquery-ajax-success

Stallman, R. 2007. Why upgrade to GPL Version 3. Retrieved 1 May, 2016, from http://gplv3.fsf.org/rms-why.html

Tao, Z. 2016. Backpacking in Yesanpo. Retrieved 10 June, 2016, from http://www.1bj.com/News/p/1997.html (in Chinese)

The Unicode Consortium. 2016. Unicode Technical Reports. Retrieved 17 March, 2016, from http://www.unicode.org/reports/index.html

Tobias, L.D. 2014. Caching HTTP: A comparative study of caching reverse proxies Varnish and Nginx. Bachelor Thesis. UPSALLA1, Sweden: Högskolan i Skövde.

Tsai, H.P., Y.H. Lin, and M.D. Yang. 2016. Exploring long term spatial vegetation trends in Taiwan from AVHRR NDVI3g dataset using RDA and HCA Analyses. *Remote Sensing* 8: 1-20. DOI: 10.3390/rs8040290

Usabilityfirst. 2016. Requirements Specification. Retrieved 31 March, 2016, from http://www.usabilityfirst.com/about-usability/requirements-specification/

USGS. 2015. NDVI, the Foundation for Remote Sensing Phenology. Retrieved 15 March, 2016, from http://phenology.cr.usgs.gov/ndvi_foundation.php

Van Rossum, G. 1997. Comparing Python to Other Languages. Retrieved 31 March, 2016, from https://www.python.org/doc/essays/comparisons/

W3C. 2014. The web standards model - HTML CSS and JavaScript. Retrieved 20 May, 2016, from https://www.w3.org/wiki/The_web_standards_model_-_HTML_CSS_and_JavaScript

W3C. 2016a. CSS homepage. Retrieved 9 June, 2016, from https://www.w3.org/Style/CSS/

W3C. 2016b. Extensible Markup Language (XML). Retrieved 9 June, 2016, from https://www.w3.org/XML/

W3C. 2016c. HTML homepage. Retrieved 9 June, 2016, from https://www.w3.org/html/

W3C. 2016d. HTT-Hypertext Transfer Protocol. Retrieved 9 June, 2016, from https://www.w3.org/Protocols/

W3C. 2016e. Scalable Vector Graphics (SVG) 1.1 (Second Edition). Retrieved 9 June, 2016, from https://www.w3.org/TR/SVG/

W3C. 2016f. W3C website. Retrieved 9 June, 2016, from https://www.w3.org/

W3schools. 2016. HTML5 Local Storage. Retrieved 4 April, 2016, from http://www.w3schools.com/html/html5_webstorage.asp

W3Techs. 2016a. Usage statistics and market share of Apache for websites. Retrieved 16 March, 2016, from http://w3techs.com/technologies/details/ws-apache/all/all

W3Techs. 2016b. Usage statistics and market share of Google Servers for websites. Retrieved 16 March, 2016, from http://w3techs.com/technologies/details/ws-google/all/all

W3Techs. 2016c. Usage statistics and market share of Microsoft-IIS for websites. Retrieved 16 March, 2016, from http://w3techs.com/technologies/details/ws-microsoftiis/all/all

W3Techs. 2016d. Usage statistics and market share of Nginx for websites. Retrieved 16 March, 2016, from http://w3techs.com/technologies/details/ws-nginx/all/all

Wang, R., K. Wang, and X.Q. Fu. 2013. The development and the protection of the tourism in Yesanpo forest. *Economic Research Guide* 8(2013): 86-87. Accession No: 1673-291X(2013)08-0086-02

Wang, R.Y., K. Cherkauer, and L. Bowling. 2016. Corn response to climate stress detected with satellite-based NDVI time series. *Remote Sensing* 8: 1-22. DOI: 10.3390/rs8040269

Weier, J., and D. Herring. 2000. Measuring vegetation (NDVI & EVI). Retrieved 13 March, 2016, from http://earthobservatory.nasa.gov/Features/MeasuringVegetation/

Wikipedia. 2016a. Creative Commons. Retrieved 17 March, 2016, from https://zh.wikipedia.org/wiki/%E5%88%9B%E4%BD%9C%E5%85%B1%E7%94%A8 (in Chinese)

Wikipedia. 2016b. Non-functional requirement. Retrieved 25 April, 2016, from https://en.wikipedia.org/wiki/Non-functional_requirement

Wikipedia. 2016c. Wikipedia website. Retrieved 16 March, 2016, from https://www.wikipedia.org/

Yesanpo. 2011. The Yesanpo National Park website. Retrieved 10 June, 2016, from http://www.ysp.gov.cn/geopark/asp/20100622/92.asp

Yoshitaka, S. 2014. Web Development 101: What is Web Development? Retrieved 15 March, 2016, from https://www.upwork.com/blog/2014/03/web-development-101-web-development/

Zhang, X., M.A. Friedl, C.B. Schaaf, A.H. Strahler, J.C.F. Hodges, F. Gao, B.C. Reed, and A. Huete. 2003. Monitoring vegetation phenology using MODIS. *Remote Sensing of Environment* 84: 471-475. DOI: 10.1016/S0034-4257(02)00135-9

Zhou, Y., X.L. Chang, S.X. Ye, Z.R. Zheng, and S.H. Lv. 2015. Analysis on regional vegetation changes in dust and sandstorms source area: a case study of Naiman Banner in the Horqin sandy region of Northern China. *Environmental Earth Sciences* 73: 2013-2025. DOI: 10.1007/s12665-014-3566-1

Appendix A – Steps of initializing and testing the development environment

The OS of the computer used to develop this web-based system is Windows7. To make sure tools and software can fulfill their functionalities, a full administrator access was used for each installation. The steps of initializing the development environment are

1. install MATLAB. For using the MATLAB API, the version of MATLAB should be R2014b or later (MathWorks, 2016a),
2. remove the Python that installed as a part of other software (e.g. ArcGIS) to make Django run stably,
3. install Python2.7, Python 3.3 or Python3.4 (MathWorks, 2016b),
4. install Django,
5. install Eclipse IDE,
6. install third-party plug-in PyDev, and
7. install the MATLAB API.

The steps of testing whether the initialization is successful are

1. create a Django project in Eclipse IDE,
2. create an void application in that project and run the whole project, and
3. go to http://127.0.0.1:8000/, a light blue welcome page will appear if the initialization is successful.

References

MathWorks. 2016a. Install MATLAB Engine API for Python. Retrieved 31 March, 2016, from http://se.mathworks.com/help/matlab/matlab_external/install-the-matlab-engine-for-python.html

MathWorks. 2016b. System Requirements for MATLAB Engine API for Python. Retrieved 25 April, 2016, from http://se.mathworks.com/help/matlab/matlab_external/system-requirements-for-matlab-engine-for-python.html

Appendix B - Sample input data for validating the PolyTrend function (image level)

time_series_cubic1.txt

```
#Copyright 2016 Sadegh Jamali
0.477250000000000
0.526250000000000
0.520250000000000
0.570500000000000
0.554000000000000
0.560750000000000
0.574250000000000
0.581500000000000
0.574250000000000
0.606750000000000
0.595500000000000
0.622500000000000
0.595500000000000
0.584000000000000
0.546750000000000
0.523750000000000
0.580500000000000
0.591500000000000
0.548500000000000
0.569000000000000
0.641000000000000
0.574750000000000
0.662500000000000
0.645250000000000
0.620500000000000
```

time_series_quadratci1.txt

```
#Copyright 2016 Sadegh Jamali
0.260250000000000
0.278000000000000
0.195750000000000
0.224250000000000
0.260750000000000
0.258500000000000
0.292000000000000
0.281750000000000
0.277500000000000
0.327500000000000
0.303500000000000
0.302750000000000
0.379750000000000
0.317250000000000
0.302750000000000
0.306500000000000
0.354250000000000
0.359250000000000
0.308000000000000
0.399750000000000
0.299500000000000
0.333000000000000
0.311500000000000
0.323250000000000
0.292250000000000
```

time_series_linear1.txt

```
#Copyright 2016 Sadegh Jamali
0.382000000000000
0.373500000000000
0.228000000000000
0.385000000000000
0.270750000000000
0.351500000000000
0.444500000000000
0.391500000000000
0.319250000000000
0.338750000000000
0.434000000000000
0.432750000000000
0.500000000000000
0.433250000000000
0.383750000000000
0.451750000000000
0.470250000000000
0.525000000000000
0.463750000000000
0.400000000000000
0.527500000000000
0.455000000000000
0.462500000000000
0.615500000000000
0.610250000000000
```

time_series_no_trend.txt

```
#Copyright 2016 Sadegh Jamali
0.567000000000000
0.624500000000000
0.610000000000000
0.588500000000000
0.569000000000000
0.638750000000000
0.561750000000000
0.600250000000000
0.607000000000000
0.538000000000000
0.592250000000000
0.628250000000000
0.604750000000000
0.610500000000000
0.586500000000000
0.602500000000000
0.560750000000000
0.567250000000000
0.574500000000000
0.605000000000000
0.621250000000000
0.571750000000000
0.589750000000000
0.595000000000000
0.603750000000000
```

time_series_conceald_cubic1.txt

```
#Copyright 2016 Sadegh Jamali
0.477250000000000
0.526250000000000
0.520250000000000
0.570500000000000
0.554000000000000
0.560750000000000
0.574250000000000
0.581500000000000
0.574250000000000
0.606750000000000
0.595500000000000
0.622500000000000
0.595500000000000
0.584000000000000
0.546750000000000
0.523750000000000
0.580500000000000
0.591500000000000
0.548500000000000
0.569000000000000
0.641000000000000
0.574750000000000
0.662500000000000
0.645250000000000
0.620500000000000
```

time_series_conceald_quadratic1.txt

```
#Copyright 2016 Sadegh Jamali
0.260250000000000
0.278000000000000
0.195750000000000
0.224250000000000
0.260750000000000
0.258500000000000
0.292000000000000
0.281750000000000
0.277500000000000
0.327500000000000
0.303500000000000
0.302750000000000
0.379750000000000
0.317250000000000
0.302750000000000
0.306500000000000
0.354250000000000
0.359250000000000
0.308000000000000
0.399750000000000
0.299500000000000
0.333000000000000
0.311500000000000
0.323250000000000
0.292250000000000
```

Appendix C - Code for validating the PolyTrend function (image level)

ValidatePolyTrendImage.m

```
%-------------------------------------------------------------------------------------------
%  This script validates the correctness of the PolyTrend function (image level) by forming
%  a big matrix representing 25-year time-series NDVI values of a area with a size of 2x6.
%  For the details of the theory, see Section 5.1
%  Author:
%  Yufei Wei,Lund University
%  Email: weiyufei2014@outlook.com
%  Copyright (C) 2016 Yufei Wei
%
%  This program is free software: you can redistribute it and/or modify
%  it under the terms of the GNU General Public License as published by
%  the Free Software Foundation, either version 3 of the License, or
%  (at your option) any later version.
%
%  This program is distributed in the hope that it will be useful,
%  but WITHOUT ANY WARRANTY; without even the implied warranty of
%  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%  GNU General Public License for more details.
%
%  For a copy of the GNU General Public License,
%  see <http://www.gnu.org/licenses/>.
%-------------------------------------------------------------------------------------------
clear;
clc;
load time_series_cubic1.txt
load time_series_quadratci1.txt
load time_series_linear1.txt
load time_series_no_trend.txt
load time_series_conceald_cubic1.txt
load time_series_conceald_quadratic1.txt
cubic=time_series_cubic1;
quadratc=time_series_quadratci1;
linear=time_series_linear1;
notrend=time_series_no_trend;
concealCubic=time_series_conceald_cubic1;
concealQuaratic=time_series_conceald_quadratic1;
FirstRow=[cubic,quadratc,linear,notrend,concealCubic,concealQuaratic];
SecondRow=fliplr(FirstRow);
ndviWhole=zeros(50,6);
j=0;
for i=1:2:50   % form a big matrix representing 25-year data of a area with a size of 2x6
    ndviWhole(i,:)=FirstRow(i-j,:);
    ndviWhole(i+1,:)=SecondRow(i-j,:);
    j=j+1;
end
rr=2;
cc=6;
ValiTytrMatrix=zeros(rr,cc);
for i=1:rr
    for j=1:cc
        tp=[ndviWhole(i,j)];
        for ii=1:25-1   % stack pixels that have same geographical locations
            ttp=ndviWhole(i+ii*rr,j); % but with different time periods
            tp=[tp;ttp];
        end
        invtp=tp';   % invert the vector for sending to the checking functions
        if dataCheck(invtp)==0 % if the vector cannot pass the judgement of
            ValiTytrMatrix(i,j)=-12000; % the dataCheck function then the resulting values
        else                            %  are replaced by -12000
            if rangeCheck(invtp,0,1)==0 % if the vector cannot pass the judgement of
                ValiTytrMatrix(i,j)=-11000; % the rangeCheck function then
                                            % the resulting values are replaced by -11000
            else % if the vector can pass both of the ckecking functions, then use the
```

```
                [TrTy,Slop,Dire,Signi]=PolyTrend(tp,0.05); % PolyTrend function(pixel level)
                ValiTytrMatrix(i,j)=TrTy;
            end
        end
    end
end
ValiTytrMatrix
```

dataCheck.m

```
%-----------------------------------------------------------------------------------------------
%  Check if the time-series should be processed. Time-series with
%  any negative data value or with a low amplitude, indicating
%  values over see or desert, should not be processed.
%  y: a vector contains time-series NDVI values
%  Authors:
%  Per J\"onsson, Malm\"o University, Sweden
%  Email: per.jonsson@ts.mah.se
%
%  Lars Eklundh, Lund University, Sweden
%  Email: lars.eklundh@nateko.lu.se
%
%  Yufei Wei, Lund University
%  Email: weiyufei2014@outlook.com
%  Copyright (C) 2016 Per J\"onsson, Lars Eklundh, Yufei Wei
%
%  This program is free software: you can redistribute it and/or modify
%  it under the terms of the GNU General Public License as published by
%  the Free Software Foundation, either version 3 of the License, or
%  (at your option) any later version.
%
%  This program is distributed in the hope that it will be useful,
%  but WITHOUT ANY WARRANTY; without even the implied warranty of
%  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%  GNU General Public License for more details.
%
%  For a copy of the GNU General Public License,
%  see <http://www.gnu.org/licenses/>.
%-----------------------------------------------------------------------------------------------
function process = dataCheck(y)
npt=size(y,2);% count the number of NDVI values
dif=zeros(1,npt-1);% create a zero vector to store differences between each values comparing
                   % with its neighbors
difpt=npt-1;% the number of the differences
signY=sign(y);% get the sign of each element
for i=1:difpt
    dif(i)=abs(y(i+1)-y(i));
end
if (sum(dif<1.e-6)>=3*npt/4)||any(signY==-1)% If at least one NDVI values in the whole
                                            % time span is smaller than 0,
    process=0; % or if during the whole time span, the number of NDVI pixels that have
else           % a difference of less than 0.000001 comparing to the ones that represent
    process=1; % adjacent years is larger than or equal to 3 times the number of years
end            % of the whole time span divided by 4, then the whole vector is viewed
end            % as unqualified and will not be processed.
```

rangeCheck.m

```
%-----------------------------------------------------------------------------------------------
%  Check if the time-series should be processed. Time-series with any data value that
%  is out of a predefined range should not be processed.
%  y: a vector contains time-series NDVI values
%  Author:
%  Yufei Wei, Lund University
%  Email: weiyufei2014@outlook.com
%  Copyright (C) 2016 Yufei Wei
%
%  This program is free software: you can redistribute it and/or modify
%  it under the terms of the GNU General Public License as published by
%  the Free Software Foundation, either version 3 of the License, or
%  (at your option) any later version.
%
%  This program is distributed in the hope that it will be useful,
%  but WITHOUT ANY WARRANTY; without even the implied warranty of
%  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%  GNU General Public License for more details.
%
%  For a copy of the GNU General Public License,
%  see <http://www.gnu.org/licenses/>.
%-----------------------------------------------------------------------------------------------
function process = rangeCheck(y,minRange,maxRange)
minValue=min(y);
maxValue=max(y);
if minValue<minRange || maxValue>maxRange
    process=0;
else
    process=1;
end
end
```

Pvalue.m

```
%-----------------------------------------------------------------------------------------------
%  This function computes p-value for testing statistical significance
%  of first coefficient of a polynomial (i.e. a) in ax^n+bx^(n-1)+...
%  Sadegh Jamali
%  Lund University, Sweden
%  E-mail: Sadegh.Jamali@nateko.lu.se
%  Copyright (C) 2016 Sadegh Jamali
%
%  This program is free software: you can redistribute it and/or modify
%  it under the terms of the GNU General Public License as published by
%  the Free Software Foundation, either version 3 of the License, or
%  (at your option) any later version.
%
%  This program is distributed in the hope that it will be useful,
%  but WITHOUT ANY WARRANTY; without even the implied warranty of
%  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%  GNU General Public License for more details.
%
%  For a copy of the GNU General Public License,
%  see <http://www.gnu.org/licenses/>.
%-----------------------------------------------------------------------------------------------
function p=Pvalue(coef,R,normr,df)
% coef: the first coefficient of the polynomial
% R, normr, and df are fields of the structure S (see help for the polyfit)
          VC=(inv(R))*(inv(R))'*(normr^2)/df; % Variance-Covariance Matrix (see
                                              % help for the polyfit), same as
                                              % VC=inv(A'*A)*(normr^2)/df,
                                              % if A is the design matrix (L=A.X)
          VC1=sqrt(VC(1,1));% Variance of the first coefficient
          statistic=(coef)/VC1;% t_ststistic to test significance of the first coefficient
          p=2*(1-tcdf(abs(statistic),df));% p value
end
```

PolyTrend.m

```
%-------------------------------------------------------------------------------------------------
%  This function detects nonlinear/linear trend in time series of a pixel data
%  and determines the trend type, slope, direction, and statistical significance.
%  Sadegh Jamali
%  Lund University, Sweden
%  E-mail: Sadegh.Jamali@nateko.lu.se
%  Copyright (C) 2016 Sadegh Jamali
%
%  This program is free software: you can redistribute it and/or modify
%  it under the terms of the GNU General Public License as published by
%  the Free Software Foundation, either version 3 of the License, or
%  (at your option) any later version.
%
%  This program is distributed in the hope that it will be useful,
%  but WITHOUT ANY WARRANTY; without even the implied warranty of
%  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
%  GNU General Public License for more details.
%
%  For a copy of the GNU General Public License,
%  see <http://www.gnu.org/licenses/>.
%-------------------------------------------------------------------------------------------------
function [Trend_type,Slope,Direction,Significance]=PolyTrend(Y,alpha)
% Y: time series pixel data (column vector)
% alpha: statistical significance level
% Trend_type: -1=concealed, 0=no trend, 1=linear,  2=quadratic, 3=cubic
% Slope: linear slope value
% Direction: 1=positive ; -1=negative
% Significance: 1=statistically significant ; -1=statistically insignificant
X=(1:length(Y))';
[p3,S3]=polyfit(X,Y,3); % Cubic fit
Roots3=roots([3*p3(1),2*p3(2),p3(3)]);
Roots3=sort(Roots3,'ascend');
Pcubic=Pvalue(p3(1),S3.R,S3.normr,S3.df);% Pvalue is a function to test significance
                                         % of the obtained fit (cubic here) coefficient
if imag(Roots3(1))==0  && imag(Roots3(2))==0  && Roots3(1)~=Roots3(2) && ...
X(1)<=Roots3(1) && Roots3(1)<=X(end) && X(1)<=Roots3(2) && Roots3(2)<=X(end) && Pcubic<alpha
    [p1,S1]=polyfit(X,Y,1);% linear fit
    Plin=Pvalue(p1(1),S1.R,S1.normr,S1.df);% P value for linear coefficient
    Slope=p1(1); % slope value
    Direction=sign(Slope); % slope direction (positive or negative)
    if Plin<alpha
        Trend_type=3;% 3 means cubic trend
        Significance=1; % 1 means significant linear trend
        Poly_degree=3; % polynomial degree
    else
        Trend_type=-1;% -1 means concealed trend
        Significance=-1; % -1 means in-significant linear trend
        Poly_degree=3; % polynomial degree
    end
else
    [p2,S2]=polyfit(X,Y,2);% Quadratic fit
    Roots2=roots([2*p2(1),p2(2)]);
    Pquadratic=Pvalue(p2(1),S2.R,S2.normr,S2.df);% P value for the quadratic coefficient
    if X(1)<=Roots2 && Roots2<=X(end) && Pquadratic<alpha
        [p1,S1]=polyfit(X,Y,1);% linear fit
        Plin=Pvalue(p1(1),S1.R,S1.normr,S1.df);% P value for the linear coefficient
        Slope=p1(1); % slope value
        Direction=sign(Slope); % slope direction (positive or negative)
        if Plin<alpha
            Trend_type=2;% 2 means quadratic trend
            Significance=1; % 1 means significant linear trend
            Poly_degree=2; % polynomial degree
        else
```

```
            Trend_type=-1;% -1 means concealed trend
            Significance=-1; % -1 means insignificant linear trend
            Poly_degree=2; % polynomial degree
        end
    else
        [p1,S1]=polyfit(X,Y,1);% Linear fit
        Plin=Pvalue(p1(1),S1.R,S1.normr,S1.df);% P value for the linear coefficient
        Slope=p1(1); % slope value
        Direction=sign(Slope); % slope direction (positive or negative)
        if Plin<alpha
            Trend_type=1;% 1 means linear trend
            Significance=1; % 1 means significant linear trend
            Poly_degree=1; % polynomial degree
        else
            Trend_type=0;% 0 means no-trend type
            Significance=-1; % -1 means in-significant linear trend
            Poly_degree=0; % polynomial degree
        end
    end
end
end
```

Appendix D - Complete file structure

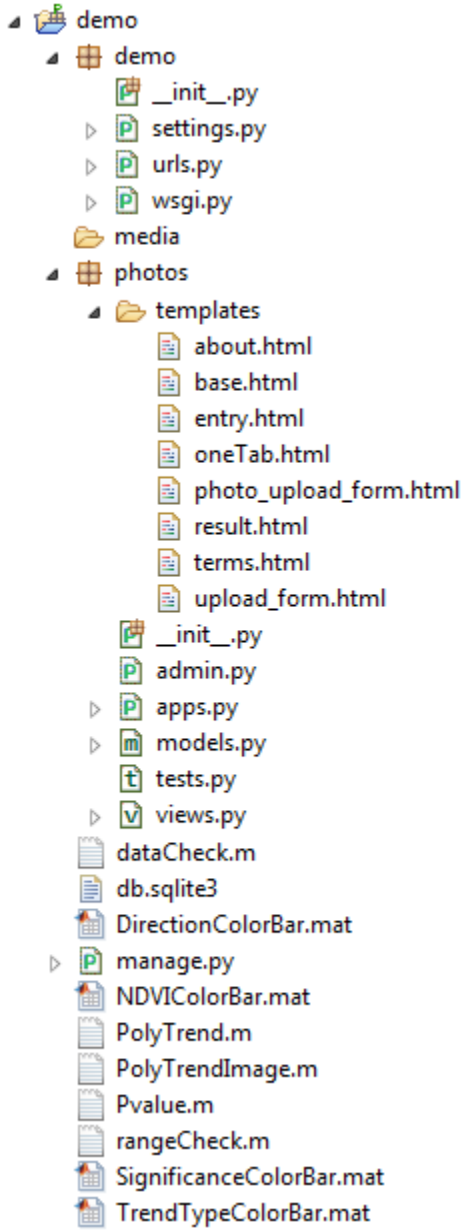

Figure D1 Complete file structure of the PolyTrend web-based system.

Appendix E - Code

The source code of the PolyTrend web-based system and the modified code of Django-jfu application are available at
https://github.com/weiyufei/polytrend

Appendix F – Questionnaire

Questionnaire of the PolyTrend web-based system

Before you carry out the following task please prepare a watch or other time measuring devices.

Please underline your choice(s).

1. What's the date when you fill out this questionnaire? dd-mm-yyyy

2. Do you have the background knowledge of the following topics? (Multiple choice)
   ①remote sensing ②geographical information system ③vegetation trends
   ④Normalized Difference Vegetation Index ⑤none of the above

3. What is your screen resolution? width:____px height:____px

4. Go to the index of the PolyTrend web-based system (http://polytrend.gis.lu.se/). What web browser are you using? *Do not use Internet Explorer!*
   ①Chrome ②Firefox ③Opera ④Others

5. Is the information on the index of the PolyTrend web-based system easy to understand?
   1=very easy to understand, 5=very hard to understand
   ①1 ②2 ③3 ④4 ⑤5

6. Now start the timer.
You are going to visualize the changing patterns of vegetation in a tourist attraction from the 49th day of 2000 to 353th day of 2001. Your task is following the instructions of the PolyTrend web-based system to generate the resulting page.

In this task, the value of significance level is 0.05 and the range of desired NDVI input to be processed is from 0.15 to 0.85. The NDVI time-series dataset that describes the vegetation growth status comes from *MODIS Subsets* and it has been sent to your mailbox along with this questionnaire. The names of the TIF files indicate their creation times. For example, *MOD13Q1.A2000049.tif* means this file was generated on the 49th day of 2000.

Stop the timer when you come to the page named *PolyTrend Results*.

How long did it take you to fulfill the above task? _____min_____sec

7. Is the information on the page *PolyTrend Results* easy to understand?
   1=very easy to understand, 5=very hard to understand
   ①1 ②2 ③3 ④4 ⑤5

8. During the whole process, do you think the functionalities of this site are easy to use?
   1=very easy to use, 5=very hard to use
   ①1 ②2 ③3 ④4 ⑤5

9. Do you have any comments on any aspects of this system? (Optional)

Thank you very much for your participation!

Institutionen för naturgeografi och ekosystemvetenskap, Lunds Universitet.

Student examensarbete (Seminarieuppsatser). Uppsatserna finns tillgängliga på institutionens geobibliotek, Sölvegatan 12, 223 62 LUND. Serien startade 1985. Hela listan och själva uppsatserna är även tillgängliga på LUP student papers (https://lup.lub.lu.se/student-papers/search/) och via Geobiblioteket (www.geobib.lu.se)

The student thesis reports are available at the Geo-Library, Department of Physical Geography and Ecosystem Science, University of Lund, Sölvegatan 12, S-223 62 Lund, Sweden. Report series started 1985. The complete list and electronic versions are also electronic available at the LUP student papers (https://lup.lub.lu.se/student-papers/search/) and through the Geo-library (www.geobib.lu.se)

350 Mihaela – Mariana Tudoran (2015) Occurrences of insect outbreaks in Sweden in relation to climatic parameters since 1850

351 Maria Gatzouras (2015) Assessment of trampling impact in Icelandic natural areas in experimental plots with focus on image analysis of digital photographs

352 Gustav Wallner (2015) Estimating and evaluating GPP in the Sahel using MSG/SEVIRI and MODIS satellite data

353 Luisa Teixeira (2015) Exploring the relationships between biodiversity and benthic habitat in the Primeiras and Segundas Protected Area, Mozambique

354 Iris Behrens & Linn Gardell (2015) Water quality in Apac-, Mbale- & Lira district, Uganda - A field study evaluating problems and suitable solutions

355 Viktoria Björklund (2015) Water quality in rivers affected by urbanization: A Case Study in Minas Gerais, Brazil

356 Tara Mellquist (2015) Hållbar dagvattenhantering i Stockholms stad - En riskhanteringsanalys med avseende på långsiktig hållbarhet av Stockholms stads dagvattenhantering i urban miljö

357 Jenny Hansson (2015) Trafikrelaterade luftföroreningar vid förskolor – En studie om kvävedioxidhalter vid förskolor i Malmö

358 Laura Reinelt (2015) Modelling vegetation dynamics and carbon fluxes in a high Arctic mire

359 Emelie Linnéa Graham (2015) Atmospheric reactivity of cyclic ethers of relevance to biofuel combustion

360 Filippo Gualla (2015) Sun position and PV panels: a model to determine the best orientation

361 Joakim Lindberg (2015) Locating potential flood areas in an urban environment using remote sensing and GIS, case study Lund, Sweden

362 Georgios-Konstantinos Lagkas (2015) Analysis of NDVI variation and snowmelt around Zackenberg station, Greenland with comparison of ground data and remote sensing.

363 Carlos Arellano (2015) Production and Biodegradability of Dissolved Organic Carbon from Different Litter Sources

364 Sofia Valentin (2015) Do-It-Yourself Helium Balloon Aerial Photography - Developing a method in an agroforestry plantation, Lao PDR

365 Shirin Danehpash (2015) Evaluation of Standards and Techniques for Retrieval of Geospatial Raster Data - A study for the ICOS Carbon Portal

366 Linnea Jonsson (2015) Evaluation of pixel based and object based classification methods for land cover mapping with high spatial resolution satellite imagery, in the Amazonas, Brazil.

367 Johan Westin (2015) Quantification of a continuous-cover forest in Sweden using remote sensing techniques

368 Dahlia Mudzaffar Ali (2015) Quantifying Terrain Factor Using GIS Applications for Real Estate Property Valuation

369 Ulrika Belsing (2015) The survival of moth larvae feeding on different plant species in northern Fennoscandia

370 Isabella Grönfeldt (2015) Snow and sea ice temperature profiles from satellite data and ice mass balance buoys

371 Karolina D. Pantazatou (2015) Issues of Geographic Context Variable Calculation Methods applied at different Geographic Levels in Spatial Historical Demographic Research -A case study over four parishes in Southern Sweden

372 Andreas Dahlbom (2016) The impact of permafrost degradation on methane fluxes - a field study in Abisko

373 Hanna Modin (2016) Higher temperatures increase nutrient availability in the High Arctic, causing elevated competitive pressure and a decline in *Papaver radicatum*

374 Elsa Lindevall (2016) Assessment of the relationship between the Photochemical Reflectance Index and Light Use Efficiency: A study of its seasonal and diurnal variation in a sub-arctic birch forest, Abisko, Sweden

375 Henrik Hagelin and Matthieu Cluzel (2016) Applying FARSITE and Prometheus on the Västmanland Fire, Sweden (2014): Fire Growth Simulation as a Measure Against Forest Fire Spread – A Model Suitability Study –

376 Pontus Cederholm (2016) Californian Drought: The Processes and Factors Controlling the 2011-2016 Drought and Winter Precipitation in California

377 Johannes Loer (2016) Modelling nitrogen balance in two Southern Swedish spruce plantations

378 Hanna Angel (2016) Water and carbon footprints of mining and producing Cu, Mg and Zn: A comparative study of primary and secondary sources

379 Gusten Brodin (2016) Organic farming's role in adaptation to and mitigation of climate change - an overview of ecological resilience and a model case study

380 Veränika Trollblad (2016) Odling av *Cucumis Sativus* L. med aska från träd som näringstillägg i ett urinbaserat hydroponiskt system

381 Susanne De Bourg (2016) Tillväxteffekter för andra generationens granskog efter tidigare genomförd kalkning

382 Katarina Crafoord (2016) Placering av energiskog i Sverige - en GIS analys

383 Simon Nåfält (2016) Assessing avalanche risk by terrain analysis An experimental GIS-approach to The Avalanche Terrain Exposure Scale (ATES)

384 Vide Hellgren (2016) Asteroid Mining - A Review of Methods and Aspects

385 Tina Truedsson (2016) How does the amount of snow and wind conditions effect water pressure measurements during winter in a lake in western Greenland?

386 Chloe Näslund (2016) Prompt Pediatric Care Pediatric patients' estimated travel times to surgically-equipped hospitals in Sweden's Scania County

387 Yufei Wei (2016) Developing a web-based system to visualize vegetation trends by a nonlinear regression algorithm