SEMANTIC DATA CUBES UTILISING FREE AND OPEN-ACCESS EO-DATA TO GENERATE SPATIALLY-EXPLICIT EVIDENCE FOR ENVIRONMENTAL MONITORING

APPLIED USE-CASE IN SYRIA BASED ON SENTINEL-2 DATA

Master Thesis

by
Hannah Augustin

supervised by
Dr. Dirk Tiede
Martin Sudmanns, M.Sc

Interfaculty Department of Geoinformatics

Salzburg, July 2018

Submitted to the Interfaculty Department of Geoinformatics in partial fulfilment of the requirements for the degree of Master of Science (MSc) in Applied Geoinformatics at the Paris-Lodron-University of Salzburg.

ABSTRACT

Free and open-access Earth observation (EO) data with global coverage are being increasingly generated at higher spatial resolutions and temporal frequencies than ever before. Imagery with high spatial resolution (5-30m) enables multi-temporal and persistent monitoring of large spatial extents for various applications. It does this in an unbiased, consistent way, independent of political borders, even if it is not always independent from political processes. Free and open EO is a growing collection of data that is one of the few globally consistent sources available for generating information in support of international initiatives. However, this data requires automated workflows for handling, processing and analysis, including methods to convert data into valid information. Optical EO data can not directly measure most objects, processes or events on Earth (e.g. digital numbers contain no semantics and many different surfaces can be represented by similar values). Indicator extraction is one way to translate this data into meaningful information to support actions towards meeting global agreements.

This thesis presents an overview of the current state of open and free EO data in a larger framework of international initiatives, such as the UN Sustainable Development Goals (SDGs), big Earth data and data cube technologies. Based on this information, a proof-of-concept implementation of a semantic EO data cube was created that is automatically updated with the newest Sentinel-2 data on a daily basis, including automatically generated generic semantic enrichment. It continuously incorporates all of the available Sentinel-2 data covering the study area located in north-western Syria (30,000km^2), which, at the time of writing, encompasses nearly 600 scenes.

This work demonstrates one example of how free and open EO data can be efficiently and automatically used to generate information products. The output of semantic queries based on vegetation- and water-like semi-concepts from the generic semantic information layers are presented, including challenges faced in their interpretation. These outputs could be seen as documenting surface water and vegetation dynamics over time. Their intended purpose is to move in the direction of generating ad-hoc reproducible, scalable, repeatable and spatially-explicit information as indicators to support specific environment related targets of the SDGs.

Keywords: remote sensing | big Earth data | data cube | semantic enrichment | reproducible research | sustainable development goals

ZUSAMMENFASSUNG

Kostenlose und frei zugängliche Erdbeobachtungsdaten (engl. Earth observation, EO) mit globaler Abdeckung werden zunehmend mit höherer räumlicher Auflösungen und zeitlichen Wiederholungsraten als je zuvor aufgenommen. Bilder mit hoher räumlicher Auflösung (5-30m) ermöglichen die multi-temporale und dauerhafte Überwachung großer räumlicher Ausdehnung für unterschiedliche Anwendungen. Dies geschieht auf eine unvoreingenommene, konsistente Weise, unabhängig von politischen Grenzen aber nicht unbedingt unabhängig von politischen Entscheidungsprozessen. Die Erdbeobachtung basiert auf einer wachsenden Sammlung von Daten, die eine der wenigen beständigen Quellen globaler Abdeckung sind, und kann damit zur Generierung von Informationen zur Unterstützung internationaler Initiativen genutzt werden. Diese Daten erfordern jedoch automatisierte Workflows zur Handhabung, Verarbeitung und Analyse, einschließlich Methoden um aus den Daten Informationen zu produzieren. Da solche optischen EO-Daten keine direkten Messungen der meisten Objekte, Prozesse oder Ereignisse auf der Erde erlauben (z.B. Digitalzahlen von EO-Daten haben keine Semantik), ist die Indikatorextraktion eine Möglichkeit, diese Daten in aussagekräftige Informationen zu übersetzen (z.B. für internationale Initiative).

Diese Arbeit gibt einen Überblick über den aktuellen Stand der kostenlosen und frei zugänglichen optischen EO-Daten in einem größeren Rahmen internationaler Initiativen wie den Nachhaltigen Entwicklungszielen (engl. Sustainable Development Goals, SDGs) der Vereinten Nationen (engl. United Nations, UN), *big Earth data* und *data cube* Technologien. Basierend auf diesen Informationen wurde eine Proof-of-Concept-Implementierung eines semantischen EO *data cubes* erstellt, der täglich automatisch mit den neuesten Sentinel-2-Daten aktualisiert wird, einschließlich generischer semantischer Anreicherung der Daten. Der data cube enthält mit kontinuierlicher Aktualisierung alle verfügbaren Sentinel-2 Daten aber auch automatisch generierten generischen semantischen Informationsschichten, die sich in der Untersuchungsgebiet im Nordwesten Syriens befinden (30,000km²). Die Ergebnisse semantischer Abfragen, die auf vegetations- und wasserähnlichen Semi-Konzepten aus dem generischen semantischen Informationsschichten basieren, wird vorgestellt, einschließlich der Herausforderungen, denen sie bei ihrer Interpretation gegenüberstehen. Die Arbeit zeigt am Beispiel von Sentinel-2, wie Erdbeobachungsdaten effizient ausgewertet werden können, um sie zur Erstellung von Informationsprodukten im Kontext der SDGs zu nutzen. Dies wurde beispielhaft für Oberflächenwasser und Vegetationsdynamik mit semantischen Abfragen umgesetzt. Ihr beabsichtigter Zweck ist es, ad-hoc reproduzierbare, räumlich skalierbare, wiederholbare und räumlich-explizite Informationen als Indikatoren zu generieren.

STATEMENTS

DECLARATION OF ACADEMIC INTEGRITY

I hereby confirm that this thesis is my own work and that if any text passages or diagrams from books, papers, the web or other sources have been copied or in any other way used, all references, including those found in electronic media, have been acknowledged and fully cited.

The thesis at hand has not yet been submitted as a thesis in this or a similar form.

Salzburg, July 2018

Hannah Augustin

DISCLOSURE STATEMENT

No potential conflict of interest was reported by the author.

This research was supported by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" within the project SemEO (contract number: 855467).

Being taken seriously means missing out on the chance
to be frivolous, promiscuous, and irrelevant.
The desire to be taken seriously is precisely what compels people
to follow the tried and true paths of knowledge production
around which I would like to map a few detours.

— J. Jack Halberstam, *The Queer Art of Failure*

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Dr. Dirk Tiede for supportive and attentive supervision, insightful discussions, constructive feedback, patience and unprecedented support – more generally, for believing in the people he works with and being driven by a much larger vision.

Martin Sudmanns, for his input and direction, both in shaping the topic of this work, but also for technical advice, resources, support and friendship – I cannot wait to witness what you will accomplish in the years to come.

Andrea Baraldi, for sharing his knowledge and wisdom, and for generously allowing access to and use of part of his life's work.

I would like to thank the other members of the SemEO team, Dr. Stefan Lang and Christian Werner, for valuable input, as well as the rest of the Z_GIS team for constructive discussions over the past many months, especially Thomas Strasser and Barbara Riedler.

The GIStammtisch crew in Salzburg, for celebrating all of the victories (especially the not so victorious) but also, and perhaps more importantly, commemorating failures.

"Broomhildas", for their tenacious drive to keep improving and for getting back up, especially when we fall together.

My simultaneous sidekick and hero, for being grounded, and, in moments when it seems the sky is falling, reminding me to look up.

Finally, my parents, for all of their unconditional love and support, including my decision not to return to the place I grew-up.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

AGDC Australian Geoscience Data Cube

API Application Programming Interface

AQL Array Query Language

ARD analysis-ready data

ASTER-GDEM Advanced Spaceborne Thermal Emission and Reflection Radiometer Global Digital Elevation Map

AVHRR Advanced Very High Resolution Radiometer

BMVIT Austrian Federal Ministry of Transport, Innovation and Technology

BoA bottom-of-atmosphere

C3S Copernicus Climate Change Service

CAMS Copernicus Atmosphere Monitoring Service

CEOS Committee on Earth Observation Satellites

CLI command line interface

CLMS Copernicus Land Monitoring Service

CMEMS Copernicus Marine Environment Monitoring Service

CPU central processing unit

CSIRO Commonwealth Scientific and Industrial Research Organisation

DEM Digital Elevation Model

DIAS Data and Information Access Services

DMSP Defence Meteorological Satellite Program

DNB Day/Night Band

DRR Sendai Framework for Disaster Risk Reduction

DSM digital surface model

EC European Commission

EMS Emergency Management Service

ENVI Environment for Visualizing Images

EO Earth observation

EO4SDG EO for Sustainable Development in Service of the 2030 Agenda

EPSG European Petroleum Survey Group

ESA European Space Agency

EU European Union

FP7 Seventh Framework Programme

G-SEXTANT Geospatial Services in Support of EU External Action

GDAL Geospatial Data Abstraction Library

GDP gross domestic product

GEE Google Earth Engine

GEO Group on Earth Observations

GEOGLAM Group on Earth Observations Global Agricultural Monitoring

GHS S-MOD Global Human Settlement Model

GHS-BUILT Global Human Settlement built-up layer

GHS-POP Global Human Settlement population layer

GHSL Global Human Settlement Layer

GIS Geographic Information System

GMES Global Monitoring for Environment and Security

GUI graphical user interface

HR high resolution

http Hypertext Transfer Protocol

IDP internally displaced people

IMF International Monetary Fund

INPE National Institute for Space Research

IQ ImageQuerying

IT information technology

JEODPP JRC Earth Observation Data and Processing Platform

JPSS Joint Polar Satellite System

JRC Joint Research Centre

L1C level-1C

L2A level-2A

LiMES Live Monitoring of the Earth's Surface

MDG Millennium Development Goal

MIR mid-infrared

MODIS Moderate Resolution Imaging Spectroradiometer

MSI multi-spectral imager

NASA National Aeronautics and Space Administration

NDBI Normalized Difference Built-Up Index

NDSI Normalized Difference Snow Index

NDVI Normalised Difference Vegetation Index

NDWI Normalised Difference Water Index

netCDF Network Common Data Format

NIR near-infrared

NOAA National Oceanic and Atmospheric Administration

NTL night-time light

ODC Open Data Cube

OECD Organisation for Economic Co-operation and Development

OGC Open Geospatial Consortium

OKF Open Knowledge Foundation

OLAP Online Analytical Processing

OLS Operational Line Scan System

OS operating system

OSF Open Science Framework

QGIS Quantum GIS

RAM random-access memory

RUS Research and User Support

SAS Statistical Analysis System

SCM scene classification map

SDC Swiss Data Cube

SDG Sustainable Development Goal

SIAM Satellite Image Automatic Mapper™

SML Symbolic Machine Learning

SRTM Shuttle Radar Topography Mission

SURF surface reflectance

ToA top-of-atmosphere

UN-GGIM United Nations Committee of Experts on Global Geospatial Information Management

UN United Nations

UNHCR UN Refugee Agency

USGS United States Geological Survey

UTM Universal Transverse Mercator coordinate system

VHR very high resolution

VIIRS Visible Infrared Imaging Radiometer Suite

WOfS Water Observation from Space

ZAMG Austrian Meteorological Service

ZGIS University of Salzburg Department of Geoinformatics

GHz gigahertz

nm nanometer

m meter

km	kilometer
MB	megabyte
GB	gigabyte
TB	terabyte

Part I

CONTEXT

1 INTRODUCTION

1.1 MOTIVATION

Technological advances have driven many changes in Earth observation (EO) from space, including new, innovative sensors and ways to handle, store and process exponentially growing data archives. In 1972, the launch of the Landsat programme, co-managed by the United States Geological Survey (USGS) and National Aeronautics and Space Administration (NASA), began what is now the longest record of Earth's status and dynamics. When the archive was opened to public access in 2008, it set the stage for free and open EO data (Wulder, Masek, Cohen, Loveland, & Woodcock, 2012). The launch of the Sentinel satellites starting in 2014 from the EU's EO programme, Copernicus, has massively increased the spatial resolution and temporal frequency of freely and openly available EO data from a variety of sensors, including radar and multi-spectral imagers (MSIs).

This kind of data is often referred to as big Earth data, but the challenge that faces researchers is not merely technological, in terms of data storage, access and processing, but rather in developing methods to distill *information* from this wealth of data. Free and open EO data with global coverage offer potential for large-scale, multi-temporal and persistent monitoring and analysis, especially with a promise of continued observation for years to come (Drusch et al., 2012).

Manual or visual analysis can in no way keep up with the continuous flow of data. It is necessary to develop higher degrees of automation for information extraction (e.g. automated-prior-knowledge based or machine learning classification procedures), especially when conducting analysis over large-scale areas or long, dense time-series. Without such automated methods for information extraction, much data is not accessed or utilised, taking up space on some server, and remains as relatively meaningless digital numbers even if the images are interesting to look at.

We are currently witnessing a shift towards an alternative approach for big data storage and analysis, enabling access to massive geo-spatio-temporal data ready for analysis. This is achieved by using multi-dimensional data cubes for massive, gridded data (Baumann, 2017). This approach contrasts to file-based systems, where, instead of an infrastructure that can serve data to users, it brings users to data.

However, data is still merely numbers and not information, even if the data is considered ready for analysis, however defined. With so much data and so many domains where they may be relevant, it makes sense to avoid application-specific (pre-)processing.

Optical EO data provide digital numbers of measurements associated with spectral wavelengths to represent that which is located between the sensor and Earth at different moments in time. Objects cannot be directly measured since pixels with similar reflectance values can represent different objects or surfaces, but these reflectance values can indicate certain land cover types or surfaces. Non-physical entities cannot be directly measured (e.g. political boundaries), and the spatial resolution and acquisition frequency also limit direct measurements of many objects or events on Earth (i.e. mixed pixels or relatively slow events).

Semantic enrichment of EO data (i.e. generating meaningful information) is not only important for conducting meaningful analysis, but is necessary for turning data into understandable information products. Information extracted from EO data provides spatially-explicit evidence that can complement existing indicators or reports from other in-situ sources in a variety of thematic domains. For example, looking at agricultural exports, consumption, malnutrition, or other statistics, it could be deduced that agricultural production has changed within a region or country, but it is often still unknown exactly where these changes have occurred. EO data might be able to offer useful spatially-explicit evidence for further inquiry or research. Information from EO sources are particularly relevant for areas that are difficult, dangerous or expensive to reach in the field (e.g. war zones, landslide-prone areas), and where consistent, reliable and unbiased data collection over longer periods of time is necessary for the application at hand.

Due to the consistent global coverage, independent of political or other human-imposed borders, free and open high resolution (HR) EO data are ideal sources of evidence for generating meaningful information products to support decision-makers in an international context. They are constantly gathered in an unbiased way, without requiring tasking or direct acquisition costs for those that utilise them. Combined or integrated with additional data sources (EO or otherwise), extracted information becomes increasingly meaningful through consilience. Their open and free nature could reduce costs for gaining certain kinds of information, as well as improve the timeliness of actionable information. Information derived from this kind of EO data has the potential to be of great service to global initiatives, whether it be monitoring the United Nations (UN) Sustainable Development Goals (SDGs) (United Nations, 2015b), the Sendai Framework for Sendai Framework for Disaster Risk Reduction (DRR) (United Nations, 2015a), or otherwise.

One existing transferable method for initial, generic semantic enrichment is automatic spectral categorisation of EO data (i.e. preliminary classification). This work explores the utility of such enrichment within an implementation of the Open Data Cube (ODC), demonstrating its usefulness as the basis for semantic queries in service of generating meaningful information perhaps relevant to the SDGs. This moves away from application-based algorithms (e.g. water detection) and sample-based classifiers, which are often not transferable among multiple images at different spatio-temporal locations, much less different sensors. Completely automated understanding of remotely sensed images is something for the future, but preliminary classification can be understood as a first, fully automated step towards automated land cover classification (A. Baraldi & Boschetti, 2012b).

This work focuses on the current and potential contributions of free and open HR EO imagery as sources of spatially-explicit evidence for a variety of domains through transferable, semi-automated, multi-temporal information extraction. It applies a highly automated, scalable workflow using Sentinel-2 data that stores pre-classified information layers (i.e. semi-concepts) in an implementation of the ODC, enabling a variety of multi-temporal, spatial and semantic queries. Northern Syria was chosen as the use-case location based on a plethora of existing EO-based research, climate suitability for optical time-series analysis, low average cloud cover in the Sentinel-2 archive, and the currently on-going conflict, which makes other methods of data acquisition challenging or impossible. Conflict on a country scale is difficult to assess or monitor, but is assumed to drive detectible land cover changes that are not homogenous. This contribution is an example of an automated, reproducible framework for handling and analysing massive EO data, and demonstrates the benefits of automated, knowledge-based semantic enrichment for environmental change detection and EO-based information extraction framed in the scope of the SDGs.

1.2 GENERAL OBJECTIVES

This thesis was motivated by a few broad objectives. They can be summarised by the following:

- review existing indicators for the SDGs and other global initiatives
- explore ways EO-data can contribute to global initiative indicators
- develop a highly automated, transferable, scalable workflow
- utilise *all* Sentinel-2 data available for the study area (i.e. large data handling)
- gain familiarity with the ODC
- improve knowledge and experience with Python, Linux and Jupyter notebooks

- investigate repeatable, reliable, semi-automated information extraction using pre-classified semi-concepts
- build a transferable and reproducible analysis environment

1.3 RESEARCH QUESTIONS

This thesis aims towards answering the following three main research questions and related sub-questions:

CONTEXT

How can EO be used to generate indicators various domains?

- What are some ways that EO contributes or can be envisioned as contributing towards spatially-explicit evidence for SDGs indicators?
- What are some current examples of free and open EO-based indicators or evidence for indicators?

IMPLEMENTATION

Can optical EO data be automatically transformed into information?

- Is it possible to automatically download Sentinel-2 data and automatically enrich it semantically?
- Can all Sentinel-2 data available for a region be automatically incorporated in a semantically enriched data cube?
- Is the available hardware sufficient for such an automated workflow and queries within a reasonable execution time?
- Is the ODC software conducive to handling relatively simple semantic queries based on semi-concepts?

RESULTS

Are semantic queries of EO data possible?

- Are semi-concepts sufficient for ad-hoc semi-automated monitoring of vegetation and water dynamics over time?
- Can information generated from querying vegetation-like or water-like semi-concepts utilising time be used in a meaningful way in the context of existing indicators?
- How does this information characterise changes to water and vegetation cover for the temporal extent of the implemented data cube?

- What sort of information is needed to better assess the quality and confidence of aggregated indicator-like results?
- Can differences in vegetation dynamics in agricultural areas be detected between Syria and Turkey using this implementation?

2

THEORY & PRACTICE

This section aims to lay the groundwork and context in which the applied example, a semantic data cube implemented for north-western Syria, was conceived, clarifying its intended purpose. Various relevant terms are explained, if not defined, and broader, on-going global initiatives are described, for which this work hopes to be relevant.

2.1 FRAMING TERMS

Before digging into anything further, a few terms and concepts need to be clarified based on intended use throughout this thesis. Some of these terms are still evolving, so the baseline for their usage needs to be explored or established, but it is by no means definitive.

2.1.1 *Open Data*

one definition

The Open Knowledge Foundation (OKF) defines knowledge as being *open* when anyone is able to freely access, use, modify, and share it (Molloy, 2011; Open Knowledge Foundation, 2018). This definition of openness is also applied to data, as long as they have an open license, are provided in a machine-readable way including necessary metadata, are user-friendly and provided in an open format. Simply publishing data on the Web and making them available at no financial cost is not sufficient for them to be considered open.

"Open data", as a term, does not merely refer to data that are free for anyone to access, use, modify and share, but also encompasses a philosophy frequently applied by public organisations in our increasingly knowledge- and service-based global economy (Hossain, Dwivedi, & Rana, 2016). Data that are produced by publicly funded institutions or initiatives ought to be made available at no additional cost because they are a public investment and doing so facilitates greater (potential) returns (Janssen, Charalabidis, & Zuiderwijk, 2012). This, however, only applies to data that cannot be traced back to specific individuals. Opening data to the public can increase accountability, reproducibility and transparency of the research and decisions made based on them and foster innovation in domains where a lack of data was previously a limiting factor. This assumes that the data are, however, actively being

used towards increasing these factors, because simply making data free and open to the public does not ensure this.

There is no intrinsic value in open data, rather the value of open data is created by using them (Janssen et al., 2012). Public institutions create and collect a lot of data. Opening data sources to the public that are not linked to individuals has the potential to increase benefits for society, the economy and environment. For example, EO images from the publicly funded Landsat mission have no intrinsic value, but a drastic increase in their use is precisely what happened when the archive was opened, leading to much research, information extraction, innovation and applications relevant to many domains (Wulder et al., 2012).

The Copernicus programme offers a plethora of free, full and open data, but the openness of these data can be challenged if they are not being used or offered in a user-friendly way. This is perhaps why the European Commission (EC) has established many programs to utilise Copernicus data and encourage user uptake and skills. Efforts include the development of operational services for monitoring the atmosphere (CAMS), marine environment (CMEMS), land (CLMS), climate change (C3S), emergency management (EMS) and security, as well as programmes geared towards users and scalable computing environments, such as Research and User Support (RUS) (Copernicus, 2017a) and Data and Information Access Services (DIAS) (Copernicus, 2017b). The more that Copernicus' open data, or any data, is not only being downloaded, but used, the higher the value and greater the benefit, hopefully for all.

2.1.2 *Big Earth Data*

Big data is an abstract, dynamic concept that poses challenges to researchers, and is defined by various unique qualities that describe the "bigness" of big data in comparison to more traditional data sources. Typically, big data refers to a massive amount of unstructured data that cannot be handled, stored or processed by traditional IT, software or hardware within a reasonable time or scope, generally requiring analysis in (near-)real-time (Chen, Mao, & Liu, 2014). These qualities are discussed in literature often using any number of terms, depending on the source, starting with the letter "V". What is considered big data will differ between application domains and will change over time as technology advances and storage, management, processing and analysis methods and strategies are developed to better handle and generate information from complex, big data sources.

The Vs of big data.

The 3 Vs, *volume*, *velocity* and *variety*, were first mentioned by Laney (2001) and later applied to characterising big data. Additional Vs have been used to describe big data over the years, including *veracity*, *vari-*

ability, *value*, and *visualisation* (Gandomi & Haider, 2015; Nativi et al., 2015).

VOLUME relates not only to the overall space that data take up on hardware – size and scale – but also the notion that the more data that is collected, the less valuable each individual observation becomes. Inferential statistics were intended to make inferences about a larger population based on sample observations, whereas big data sources generally encompass significantly more than a random sample of what they represent (Gandomi & Haider, 2015). There are limited methods that can utilise so much data in a meaningful way, while also avoiding redundancy and the threat of spurious correlations.

VELOCITY speaks to the rate at which data is generated – timeliness – in that data generation outpaces the speed at which data can be managed from acquisition to storage and utilised in a meaningful way. Big data are in motion (e.g. constant stream) and dynamic, therefore require methods for high-velocity capture, storage, discovery and analysis (Chen et al., 2014).

VARIETY has to do with the numerous formats and structures, differing semantics and many dimensions that data can take. This includes data originating from multiple, heterogeneous sources. Different types of data require different handling, thus may apply to different definitions of being big.

VERACITY of data addresses the aspect of uncertainty and unreliability that accompanies unstructured, often untrusted and messy data sources.

VARIABILITY was introduced by SAS as an additional dimension of big data (Gandomi & Haider, 2015), and refers to dynamic generation of data, meaning potentially inconsistent flow with peaks and lulls, such as is common with social media or clickstream data.

VALUE references the potential information that can be extracted from big data, transforming data to information for decision-making.

VISUALISATION speaks to the difficulty of displaying data or exploratory and analysis results from huge, heterogeneous datasets in ways that can be informative and relatively quick to understand. This is particularly relevant when it comes to multi-dimensional data, including those with a geospatial component.

Big Earth Data

Big data related to Earth sciences, including data about the Earth's atmosphere, surface and interior, are considered big Earth data, but they are more than just EO data. Big Earth data as a concept stems from integrating the concept of big data and the vision of digital Earth proposed by Gore (1998). Guo (2017) characterises big Earth data as being "massive, multi-source, heterogeneous, multi-temporal, multi-scalar, highly dimensional, highly complex, non-stationary and

unstructured." Observations are also generally not repeatable, depending on the object or process being observed, because each observation is particular to a geospatial location and moment in time.

As of 7 November 2017, there are an estimated 1,738 active, operating satellites in orbit with around 25% dedicated to Earth observation, regardless if government, military, commercial or civilian (Union of Concerned Scientists, 2017). Up until now, the majority of EO satellites have been operated by national governments, but the expected future trend is an increase in the number of civilian and commercial satellites (McCabe et al., 2017).

The Landsat-1 mission, launched in 1972, carried the first digital multi-spectral sensors intended for civilian use. Big EO data is a subset of big Earth data that really came into existence through the USGS's decision to open the Landsat archive and all new Landsat data to the public in 2008 under a free and open policy (Wulder et al., 2012). This presented new opportunities for many researchers to access and use Landsat data, stretching back until 1972. Before the archive was opened, a mere 53 Landsat scenes were downloaded on average per day, but now daily averages are around 5,700 scenes (Anderson, Ryan, Sonntag, Kavvada, & Friedl, 2017).

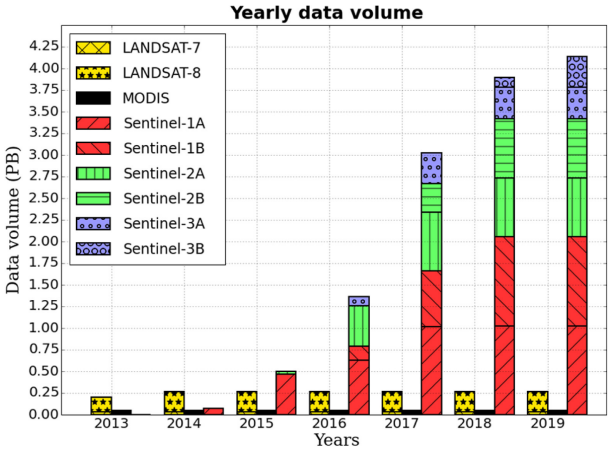

Figure 2.1: Estimates of annual data volumes from open and free data from Landsat, MODIS's Terra and Aqua units and the first three Sentinel missions (P. Soille et al., 2018)

The ECs Copernicus programme has decided to follow a similar free and open data policy and is now providing EO data comprised of multiple Sentinel satellite missions operated by European Space Agency (ESA) and *in situ* observations with global coverage. These data are being provided at an unprecedented frequency and spatial resolution for free, full and open data. There has never been so much EO data freely and openly available to the public. Daily data volumes from the Sentinel missions alone are expected to exceed 10TB (Figure 2.1) (P. Soille et al., 2018).

There is a need for automated methods.

Remotely-sensed big Earth data could continue to be analysed using methods that were applied to remotely-sensed images before Landsat opened its archive in 2008 (i.e. before they became a big open data source) but most EO data available would never be accessed, much less used. This means moving beyond file-based analysis, where each analyst must download each scene one by one, and minimising the transfer of data between data archives/stores and users.

The sheer amount of data is the main challenge in other kinds of big data domains, but satellite EO data requires conversion into value-added information to harness their potential. Many other spatial analysis routines can be chained together and automated to a large degree, yet analysis of EO images often requires more user interaction to produce meaningful results.

Developing large-scale, automated (i.e. repeatable and reliable) methods for extracting information from huge amounts of EO data is not only the current trend, but the only foreseeable way to derive meaning from a rapidly growing, free and open global data source. EO images are considered unstructured data, since they generally lack necessary structural organisation for machine-readability and automated analysis. They demand new and different automated methods to leverage their potential, whether related to storage, processing, access or analysis – algorithms that can handle many images, multiple sensors and many geographical areas and moments in time with heterogeneous conditions (e.g. cloud cover, climate, land cover).

Big Earth data is basically destined to serve international initiatives!

The benefits of big Earth data transcend the domain of Earth sciences Guo (2017). Free and open EO data are the only data sources that can continuously provide consistent information about the current and past state of the Earth's surface with global coverage that is, theoretically, the same everywhere. However, the quality and fitness of data for analysis may be unevenly distributed and differ depending on sensor and the geospatio-temporal location. Nevertheless, information extracted from big Earth data is a promising source of spatially-explicit evidence to serve a variety of application domains.

2.1.3 *Analysis Ready Data*

There has been a lot of hype recently around the term analysis-ready data (ARD), but it lacks a common, widespread definition. ARD has mostly been used in the realm of satellite imagery and seems to be related to concepts of openness (*see* Section 2.1.1), in that the intention is to make data more user-friendly, removing necessary pre-processing steps (Holmes, 2018a). Data preparation can include re-projection, organisation of data into a regular grid, co-registration, atmospheric correction, various kinds of calibration (e.g. ToA, BoA), cloud detection

or masking, image stacking to create a time-series for analysis, and more . However, data that is "analysis-ready" depends on the intended analysis, and can mean different things to different people in different domains, depending on the purpose or application.

Providers of satellite imagery are much better equipped to conduct routine pre-processing than most users. Until recently, satellite data was almost exclusively pre-processed by users, which often led to different pre-processing results even before analysis was conducted. These differences in initial data conditions for analysis, even when using the same fundamental datasets, has made comparability and reproducibility of analysis and results challenging.

Removing the user from pre-processing removes many design decisions making data-use simpler, but obscures the assumptions made by routine processes that may have an impact on analysis results. Many of these pre-processing algorithms, including those used to process open EO data, are proprietary and closed to users, providing a barrier to better understanding the data being analysed. Even if it is convenient to have data pre-processed to a certain level and may improve comparability of analysis conducted by different people, as long as closed algorithms are used, scientific research will be negatively affected.

Initiatives providing their definition of ARD.

Data provided as ARD seems to attempt to close the gaps between data availability, access and use, reducing the processing burden on users. Some current examples of EO data currently being provided as ARD include products provided by the USGS, Australian Geoscience Data Cube (AGDC) and Swiss Data Cube (SDC). The AGDC, whose data cube technology has evolved into the ODC, provides Landsat data calibrated to surface reflectance and in regular grids projected to suit Australian-specific needs (A. Lewis et al., 2017). Using similar technology (i.e. the ODC), the SDC is offering Landsat ARD for all of Switzerland (Giuliani et al., 2017a). Landsat ARD are provided by the USGS for the conterminous United States in a common tiling scheme different from the traditional path and row system of the Landsat mission (USGS, 2018). Multiple pre-processing levels and products are provided by the USGS, including data calibrated to top-of-atmosphere (ToA) reflectance, ToA brightness temperature, surface reflectance (SURF) (i.e. atmospherically corrected) and a sort of pixel quality assessment.

Semantic enrichment, information, on-demand access...

While ARD may not yet have a cohesive definition (especially when considering multiple types of sensors), in the near future, it is likely to encompass much more than it has up until now. Some suggestions include: cross-sensor and cross-provider standardisation to improve use of various data sources in analysis; improved unusable data masks and data quality metrics; methods to take into account that different sensors use different atmospheric models for atmospheric correction (e.g. Moderate Resolution Imaging Spectroradiometer (MODIS) vs. Sentinel-

2); on-demand, user-defined data selection, projection and gridding; semantic enrichment to allow content-based image retrieval; user-defined composites and mosaics; and cross-sensor alignment (Holmes, 2018a, 2018b). In particular, the term might include not only the original data, but also automated and routinely generated information products that can be used as inputs for further analysis, monitoring, or even on-demand Web-based online processing. These could include information layers or products such as land cover, land cover change, burned areas, dynamic surface water extents and fractional snow covered areas (USGS, 2018). Provision of cloud processing for users to conduct their own algorithms on data predominantly considered analysis-ready exist or are being developed, such as Google Earth Engine (GEE) (Gorelick et al., 2017) and at least four Copernicus DIASs (Copernicus, 2017b).

2.1.4 *Semantic Enrichment*

Applied to satellite imagery, semantic enrichment is about transforming pixel values (i.e. digital numbers) into meaningful information. Initial, generic semantic information is a prerequisite for content-based image retrieval and semantic queries. It also enables comparison of different images based on their content and content-related fitness for given analysis in an automated way rather than relying on general characteristics already provided in the metadata (e.g. acquisition time, geospatial extent) (Tiede, Baraldi, Sudmanns, Belgiu, & Lang, 2017).

automatic knowledge-based spectral categorisation

One existing transferable method for initial, generic semantic enrichment is automatic knowledge-based spectral categorisation, otherwise known as preliminary classification, which increases automation of EO-based indicator extraction. Image pre-classification is an initial classification of remotely-sensed images for use in more comprehensive image understanding workflows.

According to Marr (1982), human vision begins with a pre-attentive first stage and is what pre-classifiaction is intended to offer. The output of this pre-attentive first stage is a symbolic primal sketch, including both a raw and final version. The raw primal sketch is made of pure spectral differentiation of grey shades and colour tones, and the final primal sketch groups similar shades and tones. Pre-classification is a primal sketch as defined by Marr, where semi-concepts are groups of spectrally similar pixels.

Pre-classification is the assignment of each pixel to a spectral category using *a priori* knowledge-based spectral rules (i.e. a physical model) according to the pixel's spectral signature. Semi-concepts reduce the dimensionality of data in a repeatable way, transferable to multiple sensors (A. Baraldi & Boschetti, 2012a). They are considered semi-symbolic in that they are an initial step in connecting sensory data

(i.e. pixel values) to symbolic, semantic classes equal or inferior to land cover classes.

Semi-concepts require further context, analysis, or additional information to classify pixels into symbolic classes, such as land cover classes. For example, the spectral category AVHNIR stands for *Average Vegetation with High Near-Infrared* values, which could be a mixed forest, vegetated cropland, an urban park or something else entirely. More general to more detailed semi-concepts may be considered a sort of multi-scale segmentation (A. Baraldi & Boschetti, 2012a).

Applying generic, semantic enrichment moves away from application-specific algorithms (e.g. water classifiers) and sample-based classifiers, which are often not transferable among multiple images at different spatio-temporal locations. Completely automated remotely-sensed image understanding is something for the future, but pre-classification can be understood as a first, fully automated step towards image understanding, which is envisioned to include land cover classification (A. Baraldi & Boschetti, 2012b). Automatically generated, generic semantic enrichment transforms EO images into meaningful information in an automated way.

2.1.5 *Data Cube*

The data cube concept and technologies were developed independent of the EO domain (Nativi, Mazzetti, & Craglia, 2017). Also known as data hyper-cubes, they were originally designed to analyse statistics and business data using Online Analytical Processing (OLAP) (Strobl et al., 2017).

Data cubes have recently been gaining more traction, especially in the EO domain, in order to provide access to ARD, however defined, and enable multi-dimensional analysis. Advances in cloud processing environments expand many opportunities for multi-dimensional analysis using data cube infrastructures, which is one way to bring users directly to data, rather than the more traditional means of providing file-based data to users for download.

Exactly what constitutes a data cube and related standards are currently under development, and a review is outside the scope of this thesis. A manifesto was created by Baumann (2017) in response to recent attention in the realm of big Earth data in order to clarify principles of data cube service requirements. They recognise that axes all need to be treated alike, irrespective of having spatial, temporal or other semantics, and that extraction, processing, filtering and fusion must be possible in an ad-hoc way. A data cube as defined by Baumann (2017) is:

> a massive multi-dimensional array, also called "raster data" or "gridded data"; "massive" entails that we talk about sizes significantly beyond the main memory resources of the server hardware. Data values, all of the same data type, sit at grid points as defined by the d axes of the d-dimensional datacube. Coordinates along these axes allow addressing data values unambiguously.

Data cubes support ARD.

Baumann (2017) claim that by following their identified requirements, massive, gridded, spatio-temporal data stored in data cubes becomes analysis-ready. In the case of EO data, this also requires necessary pre-processing, as mentioned in Section 2.1.3. Data cubes might be increasingly common in the future of EO data storage, access and analysis.

2.1.6 *Reproducibility*

Reproducibility is one of the fundamental meta-concepts in science, the basis of the scientific method. It is not just about reproducing an experiment or a result, but also increasing the transparency of processes leading to its outcome. At the time of writing, scientific reproducibility in EO analytics has not been comprehensively studied.

Across various scientific disciplines and in everyday language, there has long been confusion about what reproducibility means, leading to varying and often conflicting definitions. Plesser (2018) and Nüst et al. (2018) recently compared multiple definitions from different domains. In the context of EO, the following terms are understood in the context of this thesis:

REPLICABILITY: ability for a *different team* to obtain the *same results* through an array of *different methods* using *independently collected data*

REPRODUCIBILITY: ability for a *different team* to obtain the *same results* through the *same methods* using the *same data*

REPEATABILITY: ability for the *same team* to obtain the *same results* through the *same methods* using the *same data*

TRANSFERABILITY: ability for *any team* to obtain *comparable results* through the *same methods* using *different data*[1]

Due to widespread confusion around what reproducibility is, Goodman, Fanelli, & Ioannidis (2016) clarified reproducibility by breaking down different aspects within research. Space and time are fundamental to EO because every observation has a geospatio-temporal location that is

1 This can refer to different sensors, different levels of pre-processing, different geospatial locations or moments in time, or other factors.

unique and cannot be repeated. This differs from some other scientific disciplines, where data to test a hypothesis can generally be assumed to be independently collected at any time or geospatial location given the necessary materials and knowledge (e.g. testing the law of gravity). Listed below is how specific kinds of reproducibility in EO data analytics based on Goodman et al. (2016) are understood in this work, recognising that independent EO data collection to test hypotheses is often not possible (e.g. historical land cover change detection), depending on the spatio-temporal location, scale and context of the events, processes, states, objects, etc. under investigation:

Different dimensions of reproducibility.

METHODS REPRODUCIBILITY: ability to exactly repeat the *same methods* by providing sufficient detail about procedures and data

RESULTS REPRODUCIBILITY: ability to obtain the *same results* through the *same methods* using the *same data*

INFERENTIAL REPRODUCIBILITY: ability to draw qualitatively *similar conclusions* from results obtained by using *different methods*, *different or independently collected data* or by reproducing the original study

2.1.7 *UN Sustainable Development Goals*

Many goals have been identified in the scope of various international agreements, with the expressed purpose of improving the lives of people across the world and mitigating potential or inevitable risks and vulnerabilities. Multiple targets have been identified for each of these goals. In this context, indicators exist or are being developed in order to monitor targets and report on progress over time using increasingly standardised and reproducible methods. Many of the developed indicators are based on official statistics at a regional, national or provincial level, but are not necessarily spatially-explicit. In order to keep the scope manageable, the focus here is looking more closely at the UN SDGs, which pertain to goals set in the year 2015 to achieve by 2030.[2]

The UN's 193 Member States have identified and agreed to work towards 17 interconnected goals with many targets and a related indicator framework for the 2030 Agenda for Sustainable Development (United Nations, 2015b). These goals are known as the SDGs (Figure 2.2), and replace what were the 8 Millennium Development Goals (MDGs), presented by the UN to be achieved by 2015. Development is a charged concept (e.g. What does it mean to be "developed" as a country? Is it a state-of-being that has even been achieved by any country?), with

2 Other international agreements with identified indicators that EO data could serve include the Sendai Framework for Disaster Risk Reduction (DRR) (United Nations, 2015a), the Paris Agreement on Climate Change, and the New Urban agenda (Corbane et al., 2017).

many different connotations. That is beyond the scope of this work and will not be discussed, but any reference to development refers to how it is characterised through the SDGs.

Figure 2.2: United Nations Sustainable Development Goals (United Nations, 2018)

Goals pertaining to all.

The SDGs are different from the MDGs not only in their content, but also in how they were conceived and to whom they apply. The MDGs were explicitly and unfairly geared towards poorer countries and were developed primarily by stakeholders from the United States, Europe and Japan, co-sponsored by financially motivated international stakeholders including the International Monetary Fund (IMF), World Bank and Organisation for Economic Co-operation and Development (OECD) (Fehling, Nelson, & Venkatapuram, 2013). A distinction between "developed" and "developing" countries was made in the creation and implementation of the MDGs framework. In contrast, the currently active SDGs are more expansive, overarching ideals that all countries ought to make steps towards achieving. These goals are interrelated, including mitigating climate change, reducing poverty and hunger, improving gender equality and education.

In order to foster international support towards reaching these goals, concrete targets have been identified to track the progress of each goal, resulting in a total of 169 targets (United Nations, 2017). Multiple, measurable indicators for monitoring progress towards each target have also been identified, and are intended to be as standardised as possible, even if standardisation has not yet been achieved.

2.2 EO-BASED INDICATORS

Much analysis in the domain of EO images was previously limited to manually selected, hopefully representative images for the purpose of a given study. In terms of detecting changes, bi-temporal change analysis was, and still is, commonly employed, due to the prohibitive cost of images, and limited technological tools and hardware to handle, store and process large data. Big, free and open EO data have made access to images no longer a limiting factor, and transferability of existing methods to multiple images at the same geographic location, with different seasonal conditions, different temporal locations, or different geographic locations etc. has become increasingly important. This is especially important in the development of EO-derived information and indicators that can be transferred to different places on Earth and still be somehow comparable.

When it comes to EO images, indicator extraction is necessary because the reflectance observed by a sensor is only a proxy for detecting, identifying and monitoring objects, events and processes. Pixels representing similar reflectance values can represent different objects, surfaces, etc. Optical EO data does not contain direct measurements of most objects or events on Earth (i.e. mixed pixels or relatively slow events). Non-physical entities (e.g. political boundaries) also cannot be directly measured.

Indicator development is imperative for leveraging the potential of EO data and transforming them into meaningful and actionable information. As big, open and free data sources are collected over a longer timespan, standardised indicators transferable to multiple data sources and geospatio-temopral locations will be increasingly useful for interpreting a variety of EO data. It will also be important to develop new methods to appropriately analyse and interpret indicators derived from big EO sources, which may differ in assumptions to previously limited, sample-based analysis.

Information derived from EO data are based on images that have somehow been semantically enriched or classified, which, even if automated and validated, always includes subjective decisions. One step is often to reduce the dimensionality of the data in a meaningful way. This can be done by employing various classification techniques, calculating indexes (e.g. greenness index, Normalised Difference Vegetation Index (NDVI), Normalised Difference Water Index (NDWI), Normalized Difference Snow Index (NDSI), Normalized Difference Built-Up Index (NDBI)}), or computing aggregative statistics over the temporal stack.

EO-based indicators are categorically different from other statistical measures, such as population counts, disease prevalence rates, gross domestic product (GDP), etc., where things in focus already have a

commonly understood semantic association (e.g. a person or currency) and are less ambiguously delineated. This contrasts to some concepts observed in EO images, such as something as simple as a forest, where definitions vary depending on location, policy, language and culture. EO-based indicators can serve as spatially-explicit evidence to support other already identified indicators, and should be interpreted with care, especially when used in isolation.

Be wary of implicit assumptions in all things.

As Maini, Clarke, Blanchard, & Murray (2017) points out, indicators have the potential to be misleading if the data, assumptions or analyses that they are based on are biased or incorrect, not to mention interpretation. While this seems obvious, big EO analysis currently lacks best-practice metrics to quantify or characterise the ambiguity, and spatio-temporal distribution, variability and uncertainty it might hold. In the case of global free and open EO data, this applies less to the data, and more to the transformation into information, analysis and interpretation. Aggregated data of water observations over time, for example, may mask specific events of high or low water (e.g. floods or droughts), or the ephemerality of some water bodies that do not exist all of the time. Depending on the purpose of analysis, these events may or may not be relevant. It is extremely important to be critical of the assumptions made when transforming EO data into information, and the analysis and interpretation that follows. Interdisciplinary teams with knowledge of and experience in the places and domains being assessed will be required to keep these assumptions in check and in perspective, especially in service of global initiatives.

The cause of visible or detected changes cannot be derived from EO-based information or images, but paired with additional information about the area, the information can serve as a spatially-explicit indicator to assess the overall situation. For example, a reduction in the area covered by vegetation could be determined by comparing EO data from various years. However, other data, information and knowledge are necessary in order to understand why. The change could have been caused by an ongoing drought, changes in policy, damage to irrigation infrastructure, a seed shortage, re-location of people, or other drivers of land-use change.

Satellite-based EO has access to any area of the world, and since free and open EO data are independent of political boundaries, if not global in coverage (even if not the same everywhere), indicators derived from them will be especially useful in supporting international initiatives and agreements in various thematic domains. Utilising information based on free and open EO data can reduce costs for monitoring some of an initiative's goals in a consistent and standardised way independent of existing national borders. They can offer dynamic, regionalised information that can be generated more frequently than traditional statistical survey methods. This is true assuming the EO data have sufficient quality, are

suitable for the intended purpose or indicator, and are generated using automated information extraction methods. Replicable extraction of generic EO-based indicators can complement indicators and information from other *in situ* sources as evidence for consilience to support decision-makers. Incorporating information derived from an objective base of constantly and continuously collected EO data with existing indicators can offer spatially-explicit, reliable and unbiased evidence in a timely manner to better facilitate and inform action and monitor progress.

Information alone will not enact changes, nor further progress towards achieving goals like the SDGs. People *do* those things and through a complex nexus of dedicated collaboration over time. However, timely, meaningful information from a common, unbiased, free and open global data source has potential to support and inform those taking action and writing policy in ways previously not possible. Hopefully the generation, but more importantly the use, of EO-based information will increase the transparency of decision-making processes, lead to better stewardship of where, what and when to focus energy and resources on, and improve accountability of those invested in making change.

2.2.1 *SDG Indicators and EO*

The concept of having global goals with accompanying indicator-based approaches to support efforts was first proposed by Guatemalan and Columbian governments and introduced at the Rio+20 Earth Summit (UN Conference on Sustainable Development) in 2012 (Hák, Janoušková, & Moldan, 2016). In UN Resolution 66, (United Nations, 2012, p. 47), it was stated that:

> We recognize that progress towards the achievement of the goals needs to be assessed and accompanied by targets and indicators, while taking into account different national circumstances, capacities and levels of development.

A multitude of quantitative indicators for measuring, monitoring and assessing sustainable development have been created over the years, including various indices, indicator sets, integrated, aggregated and composite indicators, and more. However, development, sustainability and well-being are complex topics with a lack of consensus on how to define, much less measure them. While not perfect, the SDGs provide an internationally accepted policy framework to work within that offers a more comprehensive understanding of development than simply measuring economic phenomena (e.g GDP), taking multiple factors into account. The UN and other governments and organisations will use the SDGs in order to shape and inform their policies and agenda until 2030.

Indicators are important tools for communicating a problem, making it visible, and sensitising the public and decision-makers to something that needs change (Janoušková, Hák, & Moldan, 2018). Hák et al. (2016) identified the need for relevant indicators to support the SDGs based on more intensive conceptual and methodological work instead of simply producing new statistics. The SDGs and targets are part of a policy framework, but indicators are how these idealised, policy parts are actually assessed.

A collection of quantitative indicators for each of the identified targets has been produced, but exactly how these indicators will be measured, especially in a universal, standardised and operational way, is still a point of discussion. Janoušková et al. (2018) even claim that the existing indicators were developed to support evidence-based policy making, largely driven by data availability rather than conceptual or methodological reasons, without a clear consensus on how to define and operationalise sustainable development. They claim that relying on this set of indicators may lead to distortions in policy if used as the basis for decisions, since they are not a conceptually strong basis for communicating a comprehensive understanding of global progress towards sustainable development. There are over 243 unique indicators for SDG targets proposed, which make it difficult to produce a meaningful overview of the status of the SDGs, much less operationalise all of the indicators in a meaningful way. Those points aside, challenging or assessing validity of the developed SDG indicators is beyond the scope of this work, but it is important to note that there are critiques about their overall utility for their intended purpose.

Some potential EO-based contributions to the SDGs.

Some have decided to turn to free and open EO data to realise regular and timely information generation for some indicators. In the last few years, multiple initiatives have surfaced evaluating the potential for free and open EO data to support monitoring and information generation for the SDGs. These initiatives should not come as a surprise, since, as Anderson et al. (2017) points out, Article 76 of UN Resolution 70 (United Nations, 2015b, p. 32) explicitly states:

> We will promote transparent and accountable scaling-up of appropriate public-private cooperation to exploit the contribution to be made by a wide range of data, including earth observation and geospatial information, while ensuring national ownership in supporting and tracking progress.

The Group on Earth Observations (GEO) is explicitly linked to efforts towards sustainable development. It was first described at the 2002 World summit on the Sustainable Development Implementation Plan, and launched in 2005 (Anderson et al., 2017). Its purpose is to coordinate observations globally that have to do with the state of the Earth, but

Target Contribute to progress on the Target, not necessarily the Indicator									Goal	Indicator Direct measure or indirect support to the Indicator					
							1.4	1.5	1 No poverty	1.4.2					
						2.3	2.4	2.c	2 Zero hunger	2.4.1					
					3.3	3.4	3.9	3.d	3 Good health and well-being	3.9.1					
									4 Quality education						
								5.a	5 Gender equality	5.a.1					
		6.1	6.3	6.4	6.5	6.6	6.a	6.b	6 Clean water and sanitation	6.3.1	6.3.2	6.4.2	6.5.1	6.6.1	
					7.2	7.3	7.a	7.b	7 Affordable and clean energy	7.1.1					
								8.4	8 Decent work and economic growth						
					9.1	9.4	9.5	9.a	9 Industry, innovation and infrastructure	9.1.1	9.4.1				
						10.6	10.7	10.a	10 Reduced inequalities						
	11.1	11.3	11.4	11.5	11.6	11.7	11.b	11.c	11 Sustainable cities and communities	11.1.1	11.2.1	11.3.1	11.6.2	11.7.1	
				12.2	12.4	12.8	12.a	12.b	12 Responsible consumption and production	12.a.1					
					13.1	13.2	13.3	13.b	13 Climate action	13.1.1					
		14.1	14.2	14.3	14.4	14.6	14.7	14.a	14 Life below water	14.3.1	14.4.1	14.5.1			
	15.1	15.2	15.3	15.4	15.5	15.7	15.8	15.9	15 Life on land	15.1.1	15.2.1	15.3.1	15.4.1	15.4.2	
								16.8	16 Peace, justice and strong institutions						
17.2	17.3	17.6	17.7	17.8	17.9	17.16	17.17	17.18	17 Partnerships for the goals	17.6.1	17.18.1				

Figure 2.3: SDG targets and indicators that can be supported by EO data according to the GEO and their initiative EO4SDG (Group on Earth Observations, 2017)

also has a strong focus on how EO and geospatial information can serve international agreements and initiatives.

In 2015, GEO launched the initiative EO for Sustainable Development in Service of the 2030 Agenda (EO4SDG). More generally, UN-GGIM and GEO work closely with the statistical community and have evaluated the SDGs targets and indicator framework summarised in Figure 2.3. The identified targets in Figure 2.3 can benefit from EO-derived information towards progress, but cannot necessarily provide an indicator without additional information. The identified SDG indicators can be served by a direct measure or indirect support from EO. Having reviewed the goals, targets and indicators previously to accessing the results of this GEO evaluation, it seems to be quite comprehensive. It is also clear that EO-derived information can in some way contribute to assist decision-makers, nations, organisations and other stakeholders to better plan, monitor targets and track progress towards achieving nearly each and every one of the SDGs (Committee on Earth Observation Satellites, 2018).

2.3 STATE-OF-THE-ART

EO data are highly complex, rapidly increasing in volume and variety, and are unfortunately underutilised in terms of extracting their information potential. A lack of data can no longer be considered a limiting factor to generating meaningful information, rather a lack of reproducible, reliable and transferable methods. There have been many strides in technology, methods and analysis towards better leveraging the potential of free and open big EO data. They can offer long time-series with continuity, consistency and comparability that can be utilised complementarily with other existing, more traditional statistical methods.

2.3.1 *Free and Open EO-derived Information*

Automated, repeatable and reliable EO information extraction can theoretically be conducted over large areas and time-spans (assuming adequate hardware), but requires a combination of deductive (i.e. rule/expert-based) and inductive methods that are multi-dimensional and robust to redundant information. Many challenges exist beyond automatisation and improving processes (e.g. efficiency), including large data volumes, high repetition rates, and identifying significant indicators (e.g. advanced information extraction) for a given purpose.

Here are a few examples of transforming free and open EO images into information that may be relevant for the SDGs. To keep this scope manageable, the focus will be on Sentinel-2 and Landsat, including some MODIS and night-time light data applications, meaning predominantly optical MSIs.

Night-time Light Data

Global night-time lights (NTLs) data show the geospatial locations and brightness of light escaping into space. Most of these lights are electric and originate from human settlements, making NTL a useful data source for bridging social science and remote sensing. NTL data has been used as an indicator for various socio-economic factors, including energy consumption, population distribution, size and growth, urban extent and estimations of GDP (Q. Zhang, Levin, Chalkias, & Letu, 2015). They have also been used for crisis-related applications, such as estimating the number of affected or displaced people in the case of a crisis (Corbane, Kemper, Pesaresi, Freire, & Louvrier, 2016) or early damaged area estimation (Kohiyama et al., 2004). Most NTL studies up to now have been based on a few dates or annual image composites. Further research in this field would include focusing more on temporal dynamics in NTL, taking seasonal or hourly changes into consideration to better inform interpretations of results. For example, areas with limited electricity are sometimes limited to certain hours in the day/night or days of the week, which could impact interpretation of results regionally.

For 40 years, NTL data was collected using the Defence Meteorological Satellite Programs (DMSPs) Operational Line Scan System (OLS). The Joint Polar Satellite System (JPSS) from NASA and the National Oceanic and Atmospheric Administration (NOAA) includes the Visible Infrared Imaging Radiometer Suite (VIIRS) meant to replace the MODIS and Advanced Very High Resolution Radiometer (AVHRR) sensors for tasks such as night-time light analysis, active fire detection and climate change monitoring. Since 2011, NTL data are being captured by the VIIRS Day/Night Band (DNB). Methods exist for inter-calibrating DMSP with VIIRS data in order to gain longer time-series of images for detecting changes before VIIRS became operational in late 2011. (X. Li, Li, Xu, & Wu, 2017) inter-calibrated DMSP/OLS and VIIRS night-time light images in order to retrospectively analyse changes that occurred to human settlement areas during the course of the Syrian civil war. Pre-processing requires removal of background noise and solar or lunar light contamination, cloud cover screening and exclusion of non-electric light sources (e.g. volcanoes, fires) (C. D. Elvidge, Baugh, Zhizhin, Hsu, & Ghosh, 2017).

Corbane et al. (2016) integrated NTL EO data from the VIIRS with the JRC's Global Human Settlement Layer (GHSL) to assess the hu-

manitarian impact of the Syrian conflict by estimating the number of people impacted in a timely, consistent and objective manner. The GHSL built-up layer is based on analysis of Landsat imagery from 1975 until 2013-2014 (Pesaresi et al., 2016). VIIRS data from January 2013 until December 2015 have a spatial resolution of 750m and were masked using the GHSL built-up layer in order to separate city lights from other night-light emissions (e.g. oil and gas wells). Differences in the detected light intensities between each two consecutive months was calculated and used as a proxy for detecting affected or damaged areas. The GHSL built-up layer is based on Landsat data. The number of affected people for the identified areas was calculated using the GHSL and disaggregated population data, which the Joint Research Centre (JRC) used to produce depictions of global population distribution and densities in space and time (GHS-POP). Estimated numbers of affected people per month were aggregated by administrative region, or governorate, for reporting.

Landsat, Sentinel and MODIS

Free and open high resolution multi-spectral optical imagery is produced by Sentinel-2 and Landsat missions. Sentinel's multi-spectral satellites, 2A and 2B, run as part of the Copernicus programme (formerly known as GMES) led by the European Union (EU), together have a revisit time of 5 days over equatorial areas. They offer a relatively high spatial resolution (10-60m) with 13 spectral bands. Offering data already calibrated to ToA reflectance, with a collective total of around 3.4TB of data daily, the Sentinel-2 satellites are predominantly used to monitor water cover, vegetation, coastal areas, soils, natural disasters and other features of interest for land services (*see* Section 3.2.1 for more detailed information). Landsat 5/7/8 are other multi-spectral instruments with relatively lower spatial resolutions, less frequent re-visit times and are provided without pre-processed calibration to ToA reflectance. Landsat data can be exploited in similar ways, especially where historical data are relevant for comparison.

Landsat and Sentinel-2 sensors are limited in the sense that they can only detect features that are visible (e.g. built structures, vegetation, agricultural fields, roads). These data sources can be utilised to monitor security of livelihood assets (e.g. food or water security), land conflicts, post-crisis structural damage assessment, climate change effects, and more, especially when analysed multi-temporally. Due to the spatial resolution of both missions, pixels are generally mixed, which demands the use of indicator-based analysis methods.

These kinds of data are most applicable to geographic locations that experience low average cloud cover either annually or seasonally. Nearly 70% of the Earth is covered in clouds at any given time, and these clouds are not evenly distributed (King, Platnick, Menzel, Ackerman,

& Hubanks, 2013). One critical issue is that preliminary cloud masking and an accurate detection of haze conditions are required for analysis. In particular, clouds make optical data useless for land surface applications, while areas in the image affected by haze should be radiometrically corrected in order to avoid discarding potential information.

One valuable set of information products are the automatically generated GHSLs. Knowing where human settlements are located, their extent and how it changes over time is important for multiple SDGs. The GHSLs are based on analysis of Landsat imagery from 1975 until 2013-2014 and incorporate SRTM and ASTER-GDEM data using a method of supervised classification based on symbolic machine learning (Pesaresi et al., 2016). The different layers include a built-up layer (GHS-BUILT), a population grid layer (GHS-POP) and a global human settlement model (GHS S-MOD), which combines the two previous layers in a meaningful way.

As an example of multi-sensor analysis, Corbane et al. (2017) used Sentinel-1 data to generate more up-to-date information on human settlements than available in the first multi-temporal GHSL based on Landsat (Pesaresi et al., 2016). They used the results from the analysis of Sentinel-1 data to mitigate commission and omission errors produced by Pesaresi et al. (2016) by applying an Symbolic Machine Learning (SML) classifier designed for remote sensing big data analytics. The SML classifier used training sets derived from existing global land cover products in order to classify built-up areas from Sentinel-1 data for the GHS-BUILT.

Lefebvre, Sannier, & Corpetti (2016) utilised Sentinel-2 imagery to monitor urban areas using several separate image classifications, which are then fused using what is called the Dempster-Shafer theory. The classification used to detect urban areas is based on all spectral bands and a texture index called PANTEX, which ultimately integrates the spectral information with what could be considered a spatial component (i.e. texture). Lefebvre et al. (2016) also showed that Sentinel-2 and Landsat-8 data can be combined in order to improve the geometric accuracy of Landsat-8 classifications, or the repetitiveness and thematic accuracy of Sentinel-2-based analysis.

Mubareka & Ehrlich (2010) derived environmental indicators of conflict-induced land-use changes from field data and Landsat scenes in northern Iraq. They identified environmental indicators from EO data linked to population vulnerability, such as conversion of agricultural land to grassland, and calculated a risk index.

Broich et al. (2015) used a time-series of MODIS data to characterise land surface phenology and climate variability for all of Australia. By land surface phenology, they mean a spatio-temporal characterisation of vegetated land surface's different episodes of greening or browning in

order to serve multiple applications (e.g. crop monitoring, vegetation condition, resilience to climate variability).

The Group on Earth Observations Global Agricultural Monitoring (GEOGLAM) is a global initiative working closely with Committee on Earth Observation Satellites (CEOS) specifically towards providing information to support efforts towards achieving the SDGs second goal to reach zero hunger globally. It uses a plethora of satellite-based EO, including Sentinel 1-3, Landsat, and MODIS, as well as *in situ* data, crop calendars, crop masks, and more, to produce a suite of global information products. One of these includes a crop monitor for early warning, which, for 83 different countries, measures crop conditions to assess a level of food security risk. It is sometimes difficult to discern how products are produced or validated, and the spatio-temporal differences in quality and confidence.

M. C. Hansen et al. (2013) and P. Potapov et al. (2017) used Landsat data from 2000 until 2012 or 2013, respectively, to map global changes in forest cover. In particular, they focused on forest gain, but more closely on forest loss.

Frampton, Dash, Watmough, & Milton (2013) showed that Sentinel-2 is not only useful for detecting the presence of vegetation on land, but also for monitoring vegetation condition, and even estimating canopy chlorophyll content, but further validation is necessary. Sentinel-2 can also be used to measure water quality parameters, such as total suspended particulate matter, and other properties (H. Liu et al., 2017).

N. Mueller et al. (2016) mapped surface water in Australia using 25 years of Landsat imagery in the AGDC, software that is now known as the CEOS ODC, resulting in the product called Water Observation from Space (WOfS). It was produced using a water detection algorithm based on a decision tree classifier. An added bonus is that they also utilised a comparison methodology using logistic regression and the Shuttle Radar Topography Mission (SRTM) digital surface model (DSM) to indicate terrain shadow, mask steep slopes, etc., which allowed them to add a level of confidence to the water observation detection. They were able to compare results to an existing MODIS-based open water likelihood product valid from 1999 to 2010.

Scientists from the JRC collaborated with Google using GEE to quantify the extent, changes and various dynamics of global surface water based on Landsat imagery from 1984 until 2015 (Pekel, Cottam, Gorelick, & Belward, 2016). Products, accessible via the online Global Surface Water Explorer (European Commission Joint Research Centre, 2016), include maximum extent, percent of water occurrence and change intensity over the time-series, water seasonality by month, annual water recurrence and various types of water transitions between the first and last observation in the time-series. The results calculated in this product are not produced

in an automated way, so the most recent results available are from 2015. The results are, however, global.

M. G. Tulbure, Broich, Stehman, & Kommareddy (2016) calculated surface water extent dynamics in a semi-arid region of Australia based on three decades of seasonally continuous Landsat data using a random forest classifier. They were able to capture ephemeral water, floods, and more "permanent" water bodies, collectively known as surface water flooding dynamics.

In the scope of the EC Seventh Framework Programme (FP7) project, G-SEXTANT, Tiede, Lüthje, & Baraldi (2014) demonstrated a fully automated, parameter-free post-classification land cover change detection method based on a Landsat time-series. The focus was on changes to agricultural areas in the north-western Syrian-Turkish border region as a potential indicator for livelihood security, conflict-related changes or regional stability in areas where the regional climate requires irrigation to support crops. The information was used as part of a preliminary impact assessment of the Syrian conflict using data from August 2010. 2013 and 2014.

Other studies have used the same prior-knowledge-based classification to detect or characterise changes to land cover. Arvor et al. (2018) used a 30-year-long time-series of pre-classified Landsat scenes processed in order to monitor small water reservoirs in Brazil. Langer, Tiede, & Lüthje (2015) also used pre-classified multi-temporal Landsat scenes from 1994 until 2015 as input to an object-based, post-classification change comparison. This analysis was used to characterise environmental changes occurring around a refugee camp. Hagenlocher, Tiede, Wendt, & Lang (2015) used the technology to support semi-automated classification of refugee and internally displaced people (IDP) camps using very high resolution (VHR) and Landsat data.

Tiede, Baraldi, Sudmanns, Belgiu, & Lang (2016) utilised pre-classification in a parameter free and fully automated workflow that allowed semantic queries as part of a prototypical image understanding system called ImageQuerying (IQ). This system was used by Sudmanns, Tiede, Wendt, & Baraldi (2017) to automatically extract surface water in an area in Somalia based on 78 Landsat-8 scenes. IQ is based on a Rasdaman array database architecture with a Web-based GUI that allows users to create ad-hoc semantic queries.

2.3.2 *Relevant Data Cube Implementations*

Currently existing EO data cube implementations are geared towards providing access to ARD, and recent developments for EO data are numerous. Technical infrastructure implementations for instantiating

data cubes employed in the EO domain include the Rasdaman array database (Baumann, Dehmel, Furtado, Ritsch, & Widmann, 1998), SciDB array database (M. Stonebraker, Brown, Zhang, & Becla, 2013), free and open software provided by the CEOS ODC initiative and a data cube infrastructure based on the JRC Earth Observation Data and Processing Platform (JEODPP) (Nativi et al., 2017). What follows here does not intend to give a comprehensive review, but rather an idea of some of the EO data cube solutions that exist.

EarthServer (Baumann et al., 2016) is based on Rasdaman and developed by the same people behind Rasdaman and the data cube manifesto referenced in Section 2.1.5. It serves as an engine for big Earth data analytics of coverage-type datasets using Open Geospatial Consortium (OGC) big data standards and the Rasdaman query language for searching, filtering and extraction.

The SciDB data management and analytics system for multi-dimensional array data, primarily designed by the commercial company, Paradigm4. It uses an Array Query Language (AQL) for data access and analysis, and is the basis for what is known as EarthDB (Planthaber, Stonebraker, & Frew, 2012). It is also the basis of a data cube infrastructure by National Institute for Space Research (INPE) for producing land use and land cover classification maps in Brazil (Camara et al., 2017; INPE, 2016). SciDB is used to store and analyse EO data in an ad-hoc way, and offers interfaces to work with the data using R, Python and julia.

The AGDC (A. Lewis et al., 2017), currently implemented in a high performance computing environment (Evans et al., 2015), developed the basis for the free and open software that is now known more commonly as the Open Data Cube (ODC). The CEOS has set a goal of twenty operational national-scale data cubes by 2022 (Open Data Cube, 2017). The SDC (Giuliani et al., 2017a) and CDCol (Ariza-Porras et al., 2017) are similar operational implementations of ODC software in Switzerland and Columbia, respectively. After accessing the data using a Python Application Programming Interface (API), a Python N-dimensional *xarray* object is returned that meets the N-dimensional user-defined extent, and analysis then can continue in Python, or other languages (e.g. R).

The JEODPP of the EC has been in development since 2016, and is specifically geared to facilitate analysis of big Earth data in service of global initiatives (Nativi et al., 2017; P. Soille et al., 2018). It is being used primarily to develop, integrate and analyse a set of indicators based on satellite and *in situ* EO data.

Most current data cube implementations intend to provide their definition of ARD to users, which often lacks semantic enrichment necessary for turning data into understandable information products in a more automated way, leaving that to users to conduct. One notable frame-

work for monitoring the Earth's surface using a multi-dimensional data cube is Live Monitoring of the Earth's Surface (LiMES), proposed by Giuliani et al. (2017b) who are involved with the SDC (Giuliani et al., 2017a). LiMES identified various challenges building a monitoring framework, one of which was turning data into understandable information products.

One existing example combining fully automatic semantic enrichment of EO data in 3-dimensional data cube is the IQ system described in Section 2.3.1, which uses a Rasdaman array database (Tiede et al., 2016). This sort of system allows ad-hoc information extraction by combining declarative querying in array databases with access to generic semantic information layers (e.g. pre-classified or other layers), then known as semantic querying (Sudmanns et al., 2016). The applied example in this thesis applies a different sort of implementation using Sentinel-2 data in some ways similar to IQ, but without a sophisticated query language or graphical user interface (GUI).

2.4 FRAMING THE APPLIED EXAMPLE

Given all of this information at the current moment in time, it makes sense to situate the applied proof-of-concept implementation (Chapter 3) within it. In general, big EO data are highly complex, increasing in volume and lack efficient processing capabilities and quality indicators for workflows, methods and results.

This thesis deals with free and open Sentinel-2 data, a big Earth data source, that is provided to users already calibrated to ToA reflectance. Automated workflows are necessary for handling the Sentinel-2 mission's expected 3.4TB of daily data *volume* (European Space Agency, 2017). The data has a relatively high *velocity* due to global coverage on average every five days at the equator, and quite a data *variety* in terms of consistency and quality levels (e.g. cloud coverage that differs depending on the spatio-temporal location) (P. Soille et al., 2018).

Incorporating all of the Sentinel-2 data available for an area including information layers generated through pre-classification in an implementation of the ODC can currently be considered as providing the data in an analysis-ready way. Due to the fully automated preparation of data from acquisition to ODC ingestion, this implementation can be considered highly repeatable.

Concepts of reproducibility were strong drivers behind this implementation. The reproducibility of information extraction in this case is highly linked to the level of automation of information production, which is quite high. The methods and results ought to be reproducible, given access to the same Sentinel-2 data, versions of the Satellite Image

Automatic Mapper™ (SIAM) and ODC software, Python computing environments, code and queries.

The applied example presented here utilises similar automated data preparation as demonstrated by Tiede et al. (2014), Sudmanns et al. (2017) and others, including the pre-classification, but transfers it to Sentinel-2 imagery in an implementation of the ODC. It is inspired by analysis capabilities of IQ, the prototypical implementation described by Tiede et al. (2017) and the rest of the work conducted within the SemEO project at ZGIS. The SemEO project was funded by the Austrian Federal Ministry of Transport, Innovation and Technology (BMVIT) under the program "ICT of the Future" (contract number: 855467).

Multiple application areas can be covered based on user-generated queries without requiring re-processing of the original data. The generic initial semantic enrichment in conjunction with flexible queries through time allows inferring new information layers or higher semantic levels. This aims toward supporting interactive, ad-hoc analysis where users are not required to download any data, rather only the output they would like to keep.

Validation of the pre-classification information layers or information generated by this implementation is not part of this thesis, though comparison to some existing, externally generated information is made to give a rough idea of the plausibility of results. It is intended to be viewed as a proof-of-concept and a spring-board for further research leading towards more time-sensitive analysis as well as information to help monitor long-term goals.

The example results showcased in the scope of this thesis are intended to provide EO-based information that might eventually serve some SDGs, targets and indicators (United Nations, 2015b). Targets and indicators pertaining to 3 different goals are considered. Two of the indicators (2.4.1 and 6.6.1) are labeled as "tier 3", meaning that there is no internationally agreed methodology that exists for calculating them (Committee on Earth Observation Satellites, 2018). The last indicator (15.1.1) is considered "tier 1", meaning that established and acceptable methodology exists and data are already widely available:

GOAL 2 end hunger, archive food security and improved nutrition and promote sustainable agriculture

TARGET 2.4 by 2030, ensure sustainable food production systems and implement resilient agricultural practices that increase productivity and production, that help maintain ecosystems, that strengthen capacity for adaption to climate change, extreme weather, drought, flooding and other disasters and that progressively improve land and soil quality.

Figure 2.4: SDG Goal 2 icon (United Nations, 2018)

INDICATOR 2.4.1 proportion of agricultural area under productive and sustainable agriculture

---

Figure 2.5: SDG Goal 6 icon (United Nations, 2018)

GOAL 6 to ensure availability and sustainable management of water and sanitation for all.

TARGET 6.6 by 2020, protect and restore water-related ecosystems, including mountains, forests, wetlands, rivers, aquifers and lakes.

INDICATOR 6.6.1 change in the extent of water-related ecosystems over time

---

Figure 2.6: SDG Goal 15 icon (United Nations, 2018)

GOAL 15 protect, restore and promote sustainable use of terrestrial ecosystems, sustainably manage forests, combat desertification, and halt the reverse land degradation and halt biodiversity loss

TARGET 15.1 by 2020, ensure the conservation, restoration and sustainable use of terrestrial and inland freshwater ecosystems and their servces, in particular forests, wetlands, mountains and drylands, in line with obligations under national agreements.

INDICATOR 15.1.1 forest area as a proportion of total land area

Part II

APPLIED USE-CASE

3

IMPLEMENTATION

This applied example implementation covers a highly automated workflow for transforming Sentinel-2 data into meaningful information. Some of its potential utility is framed in Section 2.4, and preliminary ad-hoc proof-of-concept results for further discussion are provided in Chapter 4. This implementation was documented in less detail in H. Augustin, Sudmanns, Tiede, & Baraldi (2018).

3.1 USE-CASE SELECTION

An area located in what is currently known as north-western Syria was chosen as the use-case illustration and is covered by three adjacent Sentinel-2 granules with observations from June 2015 to July 2018. The data cube is, however, automatically updated once a day with the most recent Sentinel-2 data and derived information layers.

3.1.1 *Background*

The conflict in Syria has been going on since March-April 2011 ("Syria's civil war explained from the beginning," 2018). Tunisian and Egyptian uprisings in 2011, now referred to as the Arab Spring, inspired pro-democracy Syrian activists to organise protests. A harsh response to these protests by the Syrian government, led by Bashar al-Assad, resulted in hundreds of dead or imprisoned demonstrators. This violent response, paired with existing social unrest, motivated the establishment of the Free Syrian Army later in 2011, whose overall aim is to overthrow the Assad-led government. It was at this point that the conflict began shifting towards civil war.

Syria experienced a drought from around 2007 until at least 2010 (Fountain, 2018; "Syria's civil war explained from the beginning," 2018). While by no means the cause of Syria's civil war, some claim that the drought contributed to social unrest, a loss of livelihood security and an increased number of internally displaced people, fueling the conflict. According to B. I. Cook, Anchukaitis, Touchan, Meko, & Cook (2016), there is a 98% likelihood that this drought was drier than any comparable period in the last 500 years, if not the last 900 years (89% likelihood). This drought event contributed to already existing water

and agricultural insecurity, situated in a century-long trend towards drier, warmer conditions on average. It was a main factor in motivating some of the estimated 1.5 million Syrian people to relocate from rural areas to urban centres and peripheries (Kelley, Mohtadi, Cane, Seager, & Kushnir, 2015).

As of December 2015, an estimated 11 million Syrians have been displaced by the on-going civil war or events leading up to it, where 4.6 million are refugees and 6.6 million internally displaced (Corbane et al., 2016). The UN Refugee Agency (UNHCR) estimates from May 2018 identify around 5.6 million registered Syrian refugees and still 6.6 million internally displaced persons (UNHCR, 2018; UNHCR & Government of Turkey, 2018).

There are multiple application domains.

The framing of the illustrated proof-of-concept results in the context of this thesis do not directly deal with the on-going conflict in Syria. They are intended to provide preliminary information in service of the SDGs. However, the study area was chosen on the assumption that surface water and particularly vegetation dynamics might be impacted by the conflict (e.g. damage to agricultural irrigation infrastructure, internal displacement of people) and be able to be made visible using such a tool for documentation purposes. This kind of change detection was demonstrated in preliminary analysis by Tiede et al. (2014) and research on vegetation dynamics and erosion by Abdo (2018). It must also acknowledged that EO-derived information from such an implementation may be of use for developing indicators of humanitarian crisis, which deals with more immediate, time-sensitive situational awareness, rather than the monitoring of longer-term idealised goals.

3.1.2 *Spatio-Temporal Location*

This proof-of-concept implementation focuses on an area located in north-western Syria, along the border to Turkey from June 2015 until July 2018, the entire duration of Sentinel-2 observations at the time of writing. Three adjacent Sentinel-2 granules (37SBA, 37SCA, 37SDA) cover an area of more than 30,000km² (latitudes 36.01°-37.05°N; longitudes 35.67°-39.11°E), as depicted in Figure 3.1, and ultimately define the spatio-temporal extent of the study area. Based on maps referring to Syria before the start of the civil war, this area covers the entire northern stretch of the Aleppo Governorate, northwestern Raqqa and northern Idlib, in addition to parts of the following Turkish provinces bordering Syria: Hatay, Kilis, Gaziantep and Şanlıurfa.

Data characteristics of the study area based on their collective metadata indicate suitability for optical time-series analyses, primarily due to the low average annual cloud-cover reflected in the Sentinel-2 archive for the area. According to the Köppen-Geiger classification, the climate is

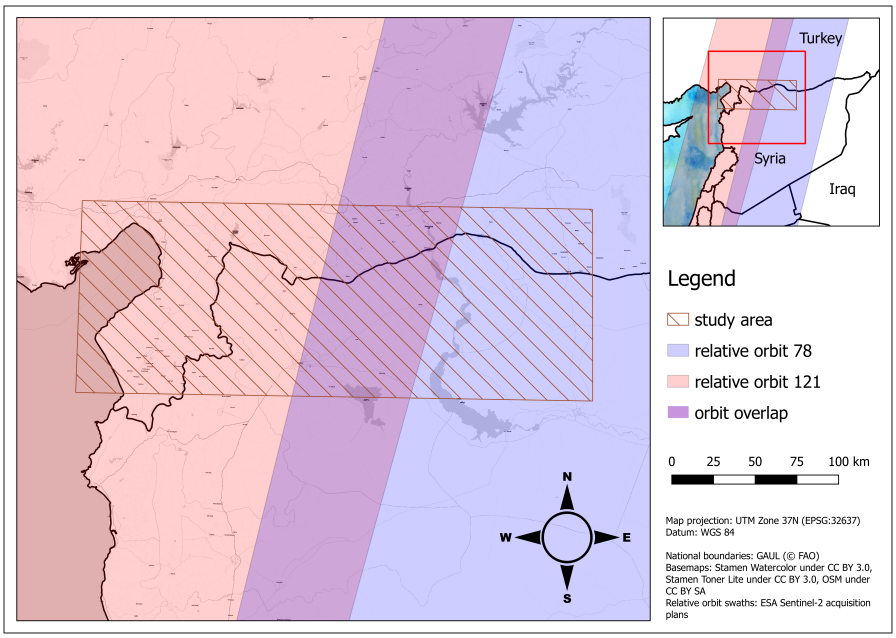

Figure 3.1: Overview of study area with Sentinel-2 relative orbits based on simplified acquisition swaths, showing an approximate orbit overlap in purple.

mostly warm Mediterranean (Csa) in the western part of the study area transitioning into warm and semi-arid (BSh) towards the east (Peel, Finlayson, & McMahon, 2007). The annual average cloud-cover percentage, extracted from ESA's level-1C (L1C) metadata, also decreases from west to east. The majority of scenes acquired from May to October have a cloud-cover percentage below 10%, while otherwise generally ranging between 20% and 40% from October to May. These summary metadata metrics were determined using the Sentinel-2 metadata analysis system developed at ZGIS, called EO-Compass (M. Sudmanns & Augustin, 2018; M. Sudmanns, Augustin, Cavallaro, Tiede, & Lang, 2017).

3.2 DATA

3.2.1 *Sentinel-2*

Specifications

Copernicus is a European Earth observation programme, previously known as GMES. It owns the fleet of Sentinel satellites, which is connected to other sensors called "contribution missions", and operates downstream services. The Sentinel-2 satellites are equipped with a multi-spectral imager (MSI) observing 13 spectral bands (443nm-2190nm). Data is captured with a swath width (i.e. field-of-view) of approximately 290km. Spatial resolution ranges from 10-60m: three visible bands and one

near-infrared band (10m); six red-edge/shortwave infrared bands (20m); and three atmospheric correction bands (60m). The spectral properties of the bands are somewhat similar to Landsat data, but lack thermal bands (Figure 3.2).

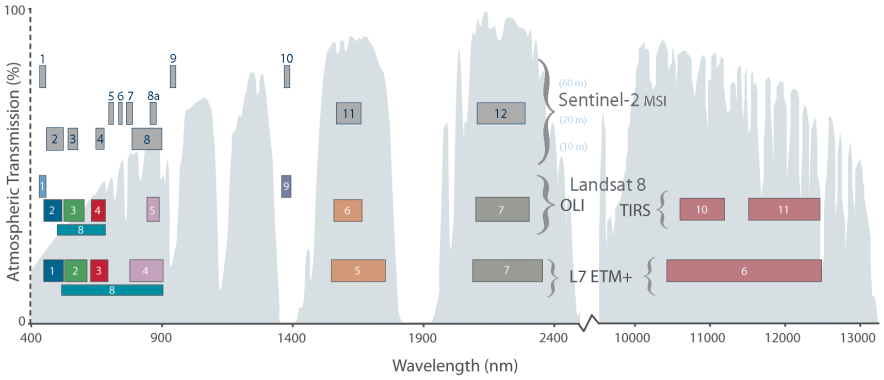

Figure 3.2: Spectral comparison of Landsat 7 and 8 bands with Sentinel-2 (retrieved on 25 April 2018 from https://landsat.gsfc.nasa.gov/sentinel-2a-launches-our-compliments-our-complements/)

Currently two Sentinel-2 satellites, known as Sentinel-2A and -2B, are continuously and systematically collecting observations globally. They were launched on 23 June 2015 and 7 March 2017, respectively. Sentinel-2C and -2D are already planned to extend the longevity of Sentinel-2 observations in the future. Considering observations from Sentinel-2A and -2B together, the nominal average revisit time at full operational capacity is 5 days at the equator, with more frequent data capture towards the poles. Data are processed and provided by ESA as level-1C (L1C), which includes radiometric calibration to ToA reflectance and geometric corrections (e.g. orthorectification, spatial registration). L1C scenes are available as granules (i.e. tiles). These granules each cover an area of approximately 100km by 100km and contain around 600MB of data, including all spectral bands, metadata and some quality indicators automatically generated by ESA.

Data Used

All Sentinel-2 L1C data available on the Copernicus Open Access Hub for these three granules are continuously included in the data cube, resulting in a dense time-series beginning 28 June 2015 until the most recent scene. All of the data used were acquired using the download script as part of the automated workflow described in Section 3.3.2. Including Sentinel-2 scenes, re-formatted Sentinel-2 data (which are redundant and could be deleted in the future after generating information layers) and the generated information layers, there is approximately 1.6TB of data related to this implementation at the time of writing. The vast majority of this volume is comprised of the indexed and ingested data

sources, but some data downloaded from the Copernicus Hub (e.g. ESA quality information layers) are counted in the 1.6TB, but not otherwise used in this implementation.

As of 22 June 2018, 591 Sentinel-2 granule-size scenes are incorporated in the data cube. This results in a range of approximately 127 to 258 observations at different moments in time throughout the study area for the temporal extent of all scenes.[1] These granules are captured by two Sentinel-2 relative orbits (78 and 121), resulting in temporally denser data where the orbits overlap (*see* Figure 3.1 for an estimation of the orbits). These three Sentinel-2 granules are provided by Copernicus in the same projection (UTM zone 37N, EPSG:32637), which means that the data do not require re-projection for collective storage or analysis.

3.3 METHODS

The workflow implemented here focuses on automation and big data, encompassing downloading Sentinel-2 data, re-formatting them, preliminarily classifying them with SIAM™ (i.e. generating multiple information layers), indexing Sentinel-2 scenes and information layers into an implementation of the ODC and ingesting information layers (Figure 3.3). This process runs automatically every day for each of the three study area granules. The result is daily incorporation of the most recently available Sentinel-2 data, ready for analysis, including semantic queries. Queries are facilitated using Jupyter notebooks by accessing the ingested information layers produced by SIAM™ via a Python API for ODC implementations.

3.3.1 *Computing Environment and Tools*

Before covering the workflow, it is important to disclose and document what hardware, software and computing environments were used for data acquisition, manipulation, processing, analysis and visualisation.

Hardware

The hardware used for this implementation is a Red Hat Enterprise Linux 7 virtual machine (if you can call it hardware), with 16 virtual

1 All of the Sentinel-2 products already indexed in the data cube implementation, whose ingested information layers are used in the showcased proof-of-concept results, are documented in Table A.1. This table is perhaps considered unnecessary to include, but I think it is important to offer some feeling of how much data, or at least how many Sentinel-2 data products, are actually being handled here.

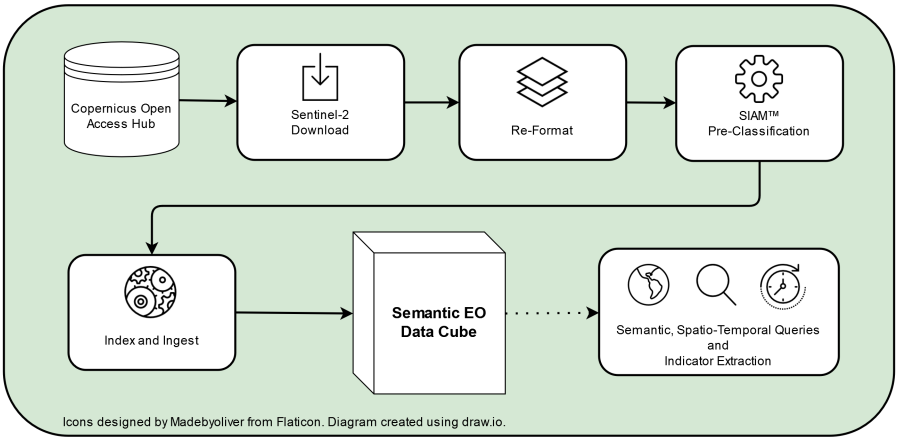

Figure 3.3: Automated workflow overview from download to queries and indicator extraction, which utilises the Python API. Author's illustration.

central processing units (CPUs) at 2.5GHz clocking, 31GB random-access memory (RAM) and 3TB of generic, all-use storage.

SIAM™

This implementation incorporates multiple semantic semi-concept granularities generated by the Satellite Image Automatic Mapper™ (SIAM™, release 88v7) (A. Baraldi, 2018). This generic semantic enrichment enables semantic queries based on multiple semantic granularities (i.e. number of semi-concepts). Spectral-based image pre-classification, as implemented by SIAM™, divides the feature space of a multi-spectral image into semantic semi-concepts using a knowledge-based approach, in contrast to data-driven approaches (e.g. supervised classification) (Baraldi, 2011; A. Baraldi, 2018; A. Baraldi et al., 2010). It could also be called descriptive colour naming, because the resulting semi-categories refer to the spectral information a pixel can offer.

Assuming scenes are calibrated to a minimum of ToA reflectance, semi-concepts generated by SIAM™ are comparable and transferable between multiple images and optical sensors without any additional user-defined parameterisation (i.e. fully automatic). This is the only software used in the workflow that is closed-source. Refer to Section 2.1.4 for a bit more theoretical information behind it.

Open-Source Software

This implementation had the overarching goal of using as much open-source software and tools as possible. This included working exclusively with Linux, Python, conda and virtual environments, git for versioning, the ODC, PostgreSQL, Jupyter notebooks for interactive analysis, GDAL for creating a mosaic of tiled analysis output and QGIS for any additional,

basic calculations, visualisation outside the Jupyter notebooks and cartographic layouts.

LINUX

The choice to work with Linux was a no-brainer, and a server with Red Hat Enterprise Linux 7 was what was available through the SemEO project. Becoming more proficient in Linux was something that I had wanted to do for a long time, having dealt enough with difficulties using GDAL, different Python environments on Windows, and, perhaps much more relevant, long pathnames associated with Sentinel-2 products. The long pathnames of the first Sentinel-2 products (which have since been shortened) are a nightmare on Windows. For these reasons, using Linux makes life easier and plays well together with most open-source software.

PYTHON, CONDA AND VIRTUALENV

Three reproducible environments for Python are used in the workflow. The first and third environments described here are managed by conda, an open-source environment and package management system that was originally designed for working with Python, but has since been implemented to accommodate various programming languages on Windows, macOS and Linux (Anaconda Inc., 2017). It installs, runs and updates dependencies and packages for a defined version of a programming language based on specific commands or environment definitions. This theoretically makes computing environments using a specific language increasingly reproducible, and allows users to have multiple configured environments at their disposal, depending on the project or purpose. The second environment is implemented using a Python package called *virtualenv* for creating isolated Python environments.

Never again hear: "Well, it works on my machine just fine..."

Searching the Copernicus Open Access Hub and downloading data to processing them with SIAM™ occurs in a Python 2.7 conda environment. The environment is equipped with *Scipy* used for resampling the required Sentinel-2 bands with a 20m spatial resolution (i.e. bands 11 and 12) to 10m. Geospatial Data Abstraction Library (GDAL) is used for various tasks and the Python package *requests* is used for handling http downloads. The complete environment file including dependencies is available at Listing A.1.

Indexing and ingesting datasets is conducted using a Python 3.4.5 *virtualenv* virtual environment recommended for ODC installations by CEOS (2018). A conda environment could also be used (Geoscience Australia, CSIRO, & NCI, 2017e). The list of the package requirements, according to the command `pip freeze` within the existing environment, is available at Listing A.6.

While interactive analysis could occur within the same environment as indexing and ingestion, another was created with conda to avoid any dependency conflicts between the automated workflow and analysis. This environment includes the *datacube*, *jupyter*, *matplotlib*, *Scipy*, *xarray*, *basemap*, *basemap-data-hires* and more for enabling access to data in the ODC implementation via the Python API and interactive analysis within Jupyter notebooks (CEOS-SEO, 2016/2017). The complete environment file including dependencies is available at Listing A.12.

GIT

While not of influence to the outcome of project development, the open-source versioning software, *git*, was used to document changes to the code base for the entirety of the project's conception and development. All code relevant to the automated workflow implementation (excluding ODC and other software) is public and freely and openly available under http://github.com/augustinh22/AIQ in the "thesis" directory.

OPEN DATA CUBE

This implementation uses the ODC version 1.5.1, known as Purpler Unicorn (13 July 2017), with access to data provided by a Python API. The ODC was chosen because it functions within the data cube paradigm, has been proven to be robust and scalable, and allows for transferable approaches through its open-source license.

The Open Data Cube (ODC) is much more than software. It is an international initiative based on software initially developed by AGDC, now managed by CEOS. The objective is to increase the impact of EO, build a community of users, and provide free and open software as a means to store, manage and analyse large volumes of EO data by providing an integrated gridded data analysis environment, including documentation (Committee on Earth Observation Satellites, 2017). CEOS is also aligned with overarching international agendas, including the SDGs, and has a mandate to improve the use of EO-derived information.

POSTGRESQL

The back-end of the ODC implementation is PostgreSQL version 9.2.21, an open-source relational database. This version is a bit out-of-date from the version recommended by the ODC installation documentation (i.e. 9.5+), but it has seemed to hold up fine. There are multiple back-end solutions that could have been employed, but this one was chosen also for its open-source license.

JUPYTER NOTEBOOKS

An open-source server-client Web-application, Jupyter notebooks allow interactive analysis, as well as a means to write, document and share code, visualisations, results, and more. The app is installed on the server where the data cube is located, allowing fairly uncomplicated access from anywhere via the Web. notebooks contain live code blocks that can be run sequentially, or not, depending on user needs. Output from each code block is printed below the block that generated it after execution, so output from analysis can be saved, if desired. Any code run in the Jupyter notebook occurs in the Python 3.4.2 data cube virtual environment. Markdown can also be used within the notebook to document and explain code blocks and results.

QGIS

Version 2.18 of the open-source GIS software, Quantum GIS (QGIS) was used to create cartographic representations of proof-of-concept results generated through interactive analysis in the Jupyter notebooks. Again, this software was chosen due to its open-source license.

3.3.2 *Automated Workflow*

Various scripts were developed to automate this workflow. The majority of the following tasks are automated on a daily basis using cron, a job scheduler on Linux, and bash scripts to run the Python scripts in their necessary environments with proper parameters. The majority of necessary scripts developed in the scope of this thesis are located in the appendix (*see* Listings A.2, A.3, A.4, A.5, A.8, A.9, and A.11), but are also freely, publicly and openly available at: http://github.com/augustinh22/AIQ

Accessing and Acquiring Sentinel-2 data

A command line interface (CLI) was implemented in Python to allow searching and downloading data from the Copernicus Open Access Hub's API, similar to the work of Hagolle (2015/2018). It is intended to be used on Windows, Mac or Linux OS. The temporal restraints for queries can be determined based on start and end times for data acquisition or ingestion to the hub. Spatial restraints can be set by defining a point, polygon or granule name. The last of those possibilities is facilitated by an in-house API at ZGIS developed by Martin Sudmanns that returns the centre point of any existing Sentinel-2 granule by name[2]. If this option is used, only the identified granule is downloaded from any

2 In the current implementation of the Copernicus Open Access Hub's API, using the centre point is the best way to limit results to a given granule rather than using the entire footprint extent of a granule. This is because each granule overlaps neighboring

matching results. This is especially important for targeted extraction of specific granules from older, multiple granule products in the archive from before 6 December 2016. Downloaded scenes are automatically unzipped in a designated target directory (in this case, granule-based directories), and any products already located in the target directory are not downloaded again.

EO-Compass also uses a modified version of this script to harvest Sentinel-2 metadata.

The structure of products and metadata has seen a few modifications since the first scenes offered to the public in 2015, which requires some more complex handling in the script. Most notably, Sentinel-2 products were served in packages of multiple granules prior to 6 December 2016, with sizes sometimes exceeding 6GB. The actual physical file structure has, however, remained mostly the same (*see* Figure 3.4). In the case of older multiple-granule products, the product name might appear more than once in Table A.1, but are saved in the appropriate granule-named directory in this implementation (i.e. a product could contain granules 37SBA and 37SCA, so is listed twice, but in each granule-specific directory, the product directory only contains the data extracted for that specific granule).

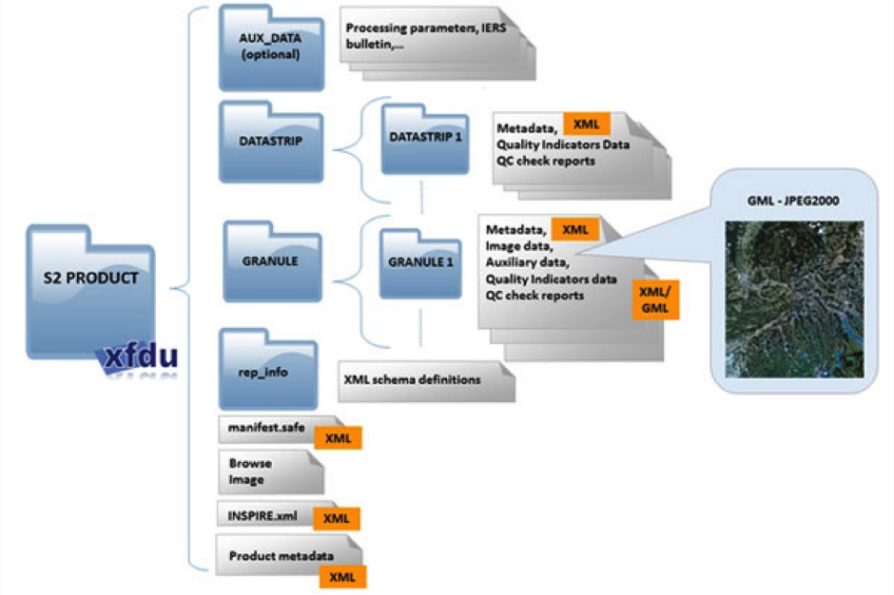

Figure 3.4: The Sentinel-2 product physical format (Source: European Space Agency (2015) accessed 26 January 2017)

For those large package products on the hub prior to 6 December 2016, the file structure is generated and each file related to a specific granule is iteratively extracted and downloaded. This contrasts to granule-specific products offered after 6 December 2016, which can be downloaded in the compressed format provided natively on the hub. It has since been announced that products provided prior to 6 December 2016 will be

granules by at least 200m on each side. Granule specific Sentinel-2 data retrieval at the time of writing is not a feature offered by the Copernicus Open Access Hub.

reprocessed to fit the current "Complete Single Tile" (i.e. single granule) format, but when this re-processing will be completed by the hub for the entire archive before 6 December 2016 is still unknown (European Space Agency, 2018a).

The time to download each scene varies.

The speed of downloads is highly dependent upon the Copernicus Open Access Hub and variables related to the Internet connection and University network. It can take anywhere from a few minutes to download one scene to upwards of 20 minutes. Based on operation during the month of June 2018, it took around 10 minutes on average per scene.

SENTINEL-2 DOWNLOADER

During the course of this work, some individuals at ZGIS expressed interest in a GUI for using this download script. The decision was made to implement the script as a QGIS plugin with an interface designed using Qt (*see* Figure 3.5). The download script used in this thesis was modified to search and download Sentinel-1 and -2 data from multiple hubs, allow manual manipulation of search results from the hub and enable individual granule extraction from older Sentinel-2 product file structures.

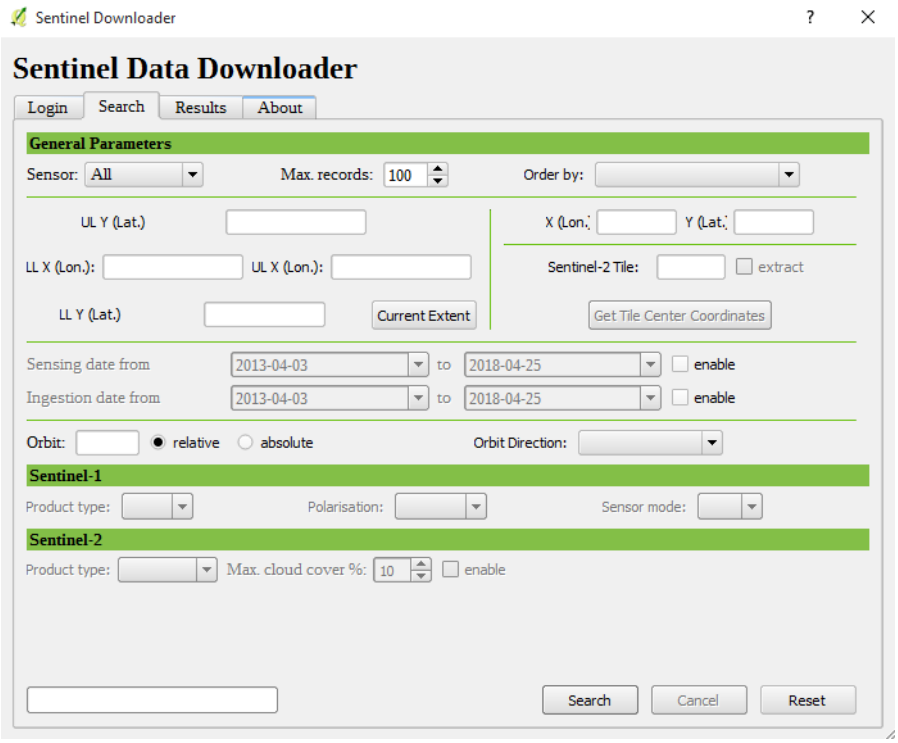

Figure 3.5: The query user-interface for the QGIS plugin to search and download Sentinel-1 and -2 data from multiple hubs. Author's illustration.

Any hub that uses a similar API and metadata structure as the Copernicus hub can be integrated with relatively low time investment. At the

time of writing, the current hubs include: Copernicus Open Access Hub (both API access, limited to 100 results, and archive access that the Web interface is based on, limited to downloading 10 results at a time); the Sentinels National Mirror Austria (https://data.sentinel.zamg.ac.at/) run by ZAMG; and the Hellenic National Sentinel Data Mirror (https://sentinels.space.noa.gr/) run by the National Observatory of Athens, Greece.

The intention is to keep improving the code base, adding more hubs, as well as integrating Sentinel-3 searches and downloads. The actual code is not provided here, but it is based on the code in Listing A.2. The code is, however, freely and openly available at: https://github.com/augustinh22/SentinelDownloader

Formatting data for SIAM™

Generate no-data mask, resample, convert to 8-bit and stack.

Following complete and successful download, the necessary bands from each newly acquired Sentinel-2 scene are automatically re-formatted. SIAM™ is sensor independent, but input data format requirements are based on Landsat for high-resolution data. Six bands (blue, green, red, near infrared and two medium infrared bands) are used (i.e. bands 2, 3, 4, 8, 11, 12). Scipy resamples bands 11 and 12 from 20m pixels to 10m. Sentinel-2's MSI does not capture any thermal bands, so a constant is used to ignore thermal decision rules in SIAM™. Based on the assumption that pixels with a value of 0 in any of the input bands contain no meaningful data and not a measured value of 0, a no-data mask is generated (*see* Section 5.3.2 for further discussion). Finally, using GDAL, the six Sentinel-2 bands of ToA reflectance values are converted to an 8-bit range, stacked in ascending order and saved in the ENVI data format required as input for SIAM™.

It takes 1 minute to re-format one scene.

Re-formatting each Sentinel-2 scene on the current set-up takes roughly 1 minute. This process is distributed over all cores, so might be faster with different hardware, but evaluating and improving the efficiency and speed is beyond the scope of this thesis.

SIAM™: generating information layers

A batch script is required to run SIAM™ is automatically generated that contains commands to process all newly acquired, re-formatted scenes. Four semi-concept granularities (i.e. 18, 33, 48 and 96 semi-concepts) and four additional information layers are automatically generated. An older example of 61 semi-concepts with broad descriptions and pseudo-colour representation can be seen in Figure 3.6 Those semi-concepts on the same line in Figure 3.6 stem from the same parent spectral category within the SIAM™ decision-tree.

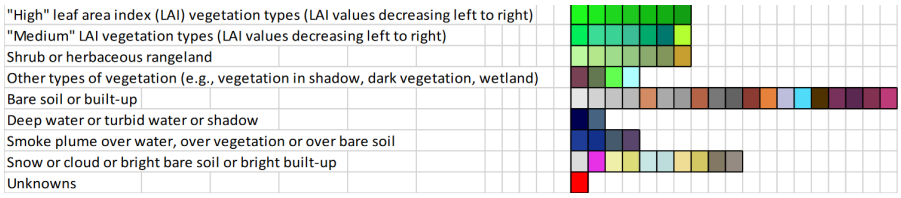

Figure 3.6: Semi-concept example with a granularity of 61, sorted in rows by parent category and each represented by a pseudo-colour suitable to the semi-symbolic associated semantics (A. Baraldi, 2018)

The additional information layers included in the applied example include a:

(1) binary vegetation mask based on vegetation-related semi-concepts;
(2) pentanary haze mask, a discretised continuous symbolic variable;
(3) ratio greenness index, i.e. (NIR / red) + (NIR / MIR1) – (red / MIR1) (A. Baraldi, 2018; A. Baraldi et al., 2010);
(4) panchromatic brightness image, a linear combination of all multi-spectral input bands.

It takes around 5 minutes to process one scene.

Processing of each Sentinel-2 scene takes roughly 4-5 minutes on the current set-up. SIAM™ in its current implementation runs on only one CPU, which makes it great to have multiple instances running in parallel. The initial processing of 450 Sentinel-2 scenes in December 2017 was completed within one evening by having 10 different instances run simultaneously. Finding other ways to improve the performance of this software is outside the scope of this thesis.

ODC: indexing scenes and information layers

Before getting into the specifics of indexing, the complete workflow for working with the ODC software can be seen in Figure 3.7. It offers a nice overview of all possible steps necessary to create an instance of a data cube within the ODC software.

In order to index data, a product description first needs to be defined. Indexing establishes links to externally stored data in a format defined in the product description and is backed by PostgreSQL. Product descriptions identify metadata common to all datasets of that product (Geoscience Australia, CSIRO, & NCI, 2017c) and only need to be defined once. Both Sentinel-2 data and SIAM™ information layers have been defined as products by modifying existing scripts provided by the ODC initiative. The product definition for information layers can be seen in Listing A.7.

Metadata for each Sentinel-2 scene and also for the information layers is necessary for indexing, and is automatically generated for each dataset. This has been implemented for Sentinel-2 and SIAM™ generated infor-

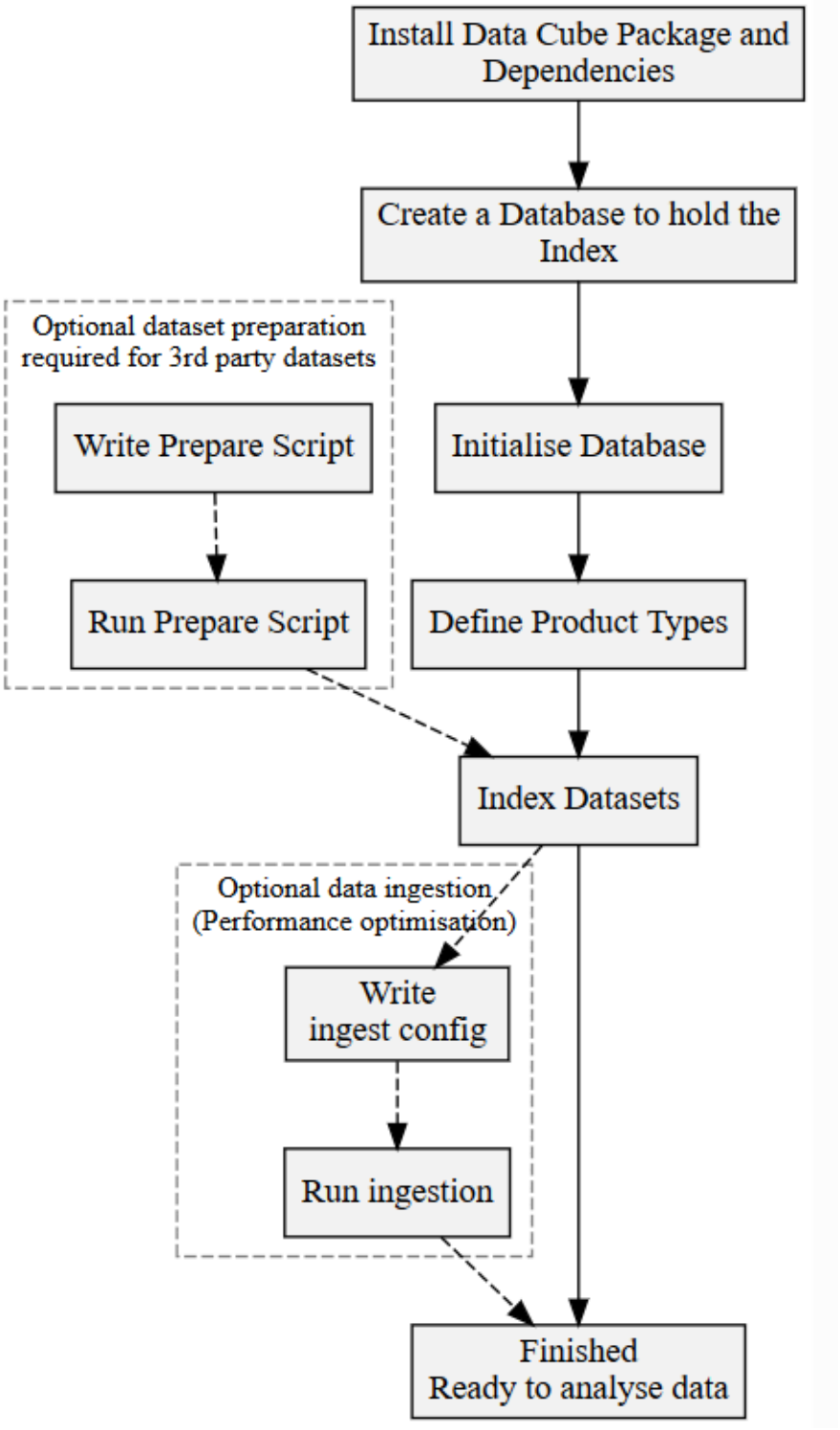

Figure 3.7: Workflow for working with the ODC. (Source: https://datacube-core.readthedocs.io/en/latest/ops/overview.html accessed on 22 June 2018)

mation layers by modifying existing Python scripts provided by the ODC initiative (*see* Listing A.8). This metadata includes spatio-temporal data extents, data format, projection, bands/layers, file paths relative to the metadata, and more. A copy of the source Sentinel-2 dataset's metadata is included in the information layer metadata to document provenance. Once metadata has been generated, indexing automatically follows using a Python script and the data cube's API (*see* Listing A.9).

Metadata generation takes a handful of seconds per scene.

Based on the initial ingestion of around 450 scenes in January 2018, it took on average 0.66 seconds to generate metadata for each scene, and around 1.5 seconds to index a scene.

ODC: ingesting information layers

Data that has been indexed can also be ingested, resulting in automated tiling of an indexed product into Network Common Data Format (netCDF) files for more efficient access, creating a gridded time-series data cube (Geoscience Australia, CSIRO, & NCI, 2017d)[3]. The user needs to select a successfully indexed product (e.g. indexed SIAM™ information layers), define the ingestion configuration and the ODC software takes care of the rest. It automatically creates a new product description, re-projects the data if necessary, tiles them according to the configuration, creates the necessary metadata and indexes them, with automatic checks to avoid duplication. This ingestion command simply needs to be run whenever new data is added to an indexed product that ought to be ingested.

In this implementation, automated ingestion of information layers in 100km^2 tiles (10km by 10km by one time-step) occurs, keeping the original projection (i.e. UTM zone 37N, EPSG:32637). As of 22 June 2018, 72,672 tiles of ingested information layers had been created based on 591 Sentinel-2 scenes, a total volume of approximately 180GB of ingested data. An evaluation of a more efficient tiling scheme is a much larger issue, and is outside the scope of this thesis.

It takes around 1 minute to ingest one scene.

Ingestion takes more computing power and time than indexing, because a bunch of netCDF tiles need to be generated. Based on the ingestion of 450 scenes in January 2018, this process takes around 1 minute per scene.

3 It is important to note that the logical view offered to the user is a multi-dimensional data cube regardless of whether or not a product has been indexed or ingested.

3.3.3 *Ad-hoc Queries*

Querying the data cube occurs using a Python API, which I have chosen to access using a Jupyter notebook for interactive generation and manipulation of data cube output.

ODC: Python API

Once data has been indexed and ingested, they can be accessed using a Python API (Geoscience Australia, CSIRO, & NCI, 2017a). This API retrieves data from a given indexed or ingested product for a defined spatio-temporal extent and returns it to the user as a *Dataset* object from the *xarray* Python package. This Python object is a multi-dimensional, in-memory array with dimension names, and is used for further analysis (e.g. in Jupyter notebooks). Indexed data sources can also be accessed using this same Python API and with the same commands, but will almost definitely take longer if not save as netCDF files optimised for access (i.e. ingested).

Any queries that might exceed memory ought to be run using the Gridworkflow class of the API to access and process data in smaller chunks (Geoscience Australia, CSIRO, & NCI, 2017b). This may not be desirable or even possible for certain types of queries or processes.

Jupyter notebook: ad-hoc queries

The Jupyter notebook used to query the cube and produce output is largely based on code blocks that exist in other ODC Jupyter notebook examples, freely and openly available from CEOS-SEO (2016/2017). See Figure 3.8 for a rough idea of what a notebook looks like.

Within the notebook, it takes around 90 seconds to connect to the ingested SIAM™ information layer product. Following that initial, one-time connection, it all depends on what the user would like to do with the data and information within the data cube in terms of how long processing may take.

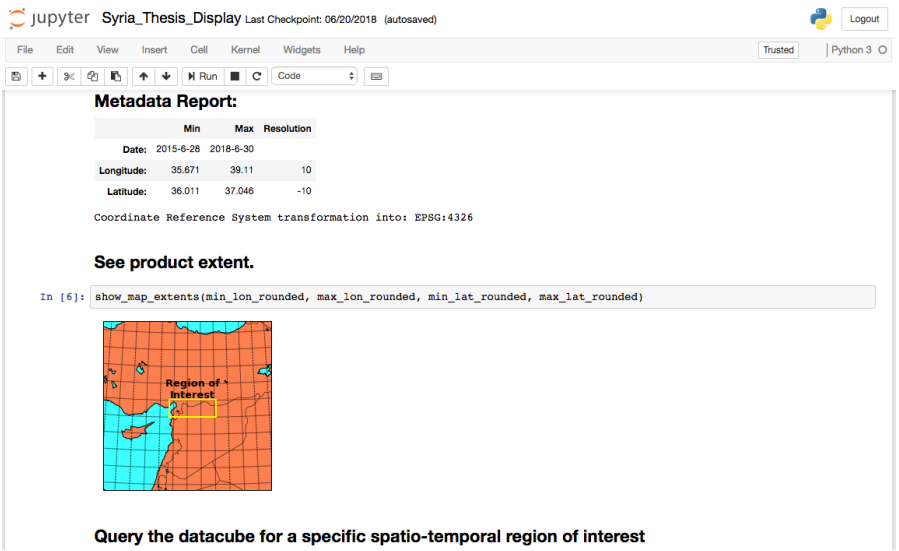

Figure 3.8: Screenshot of one of the Jupyter notebooks used, including some of the functions primarily developed by CEOS-SEO (2016/2017). This shows the spatio-temporal extent of the data cube in the metadata report, as well as the spatial extent in the small map. Author's illustration.

4

PROOF-OF-CONCEPT

The main result of this work is a constantly up-to-date semantically enriched data cube implementation for north-western Syria. This implementation may continue to be used for research of individual observations in time, monthly or seasonal analysis, up to the entire temporal extent of the growing data cube for the entire geospatial extent. In addition to the ingested semantically enriched layers, the original Sentinel-2 data and information layers in their original format can be accessed in a similar manner. Access to the original Sentinel-2 data that the information layers are generated from in a data cube infrastructure may be very useful for certain applications and questions, but is not addressed in this thesis.

In order to demonstrate its potential utility, a few proof-of-concept demonstrations of automated information extraction about water and vegetation extent and dynamics over time based on semi-concepts are provided. Each of these queries is also accompanied by a few additional geospatial layers to better provide insight about the quality of the output, and to aid interpretation. Such aggregated analysis produced from big Earth data currently have few methods or metrics available to assess quality, distribution and variability through space *and* time for meaningful interpretation and comparability. This kind of automated information extraction may be relevant to SDG targets 2.4, 6.6 and 15.1 (*see* Section 2.4), which is what is framing these examples, but also for indicators for DRR, livelihood security (*see* Section 6.6), broader situational awareness and more.

4.1 AGGREGATED TIME-SERIES OUTPUT

While other queries (e.g. post-classification change) could have been conducted, a conscious choice was made to produce output aggregated over time. This was decided in order to better explore some of the challenges faced in generating such output, but more importantly, challenges encountered in their interpretation.

These proof-of-concept results are inspired by similar aggregated geospatial time-series analysis conducted by N. Mueller et al. (2016) and M. G. Tulbure et al. (2016) of surface water bodies. Both of those examples were conducted within the AGDC, the basis for the ODC software. The Python functions used in this thesis to produce these aggregated time-

series results were developed by the ODC and are available on GitHub (CEOS-SEO, 2016/2017). It is assumed that very similar, if not the same, time-series functions used in those examples of aggregated water analysis over time have been used here, but using generic semi-concepts instead of the results of a sample-based surface water classifier.

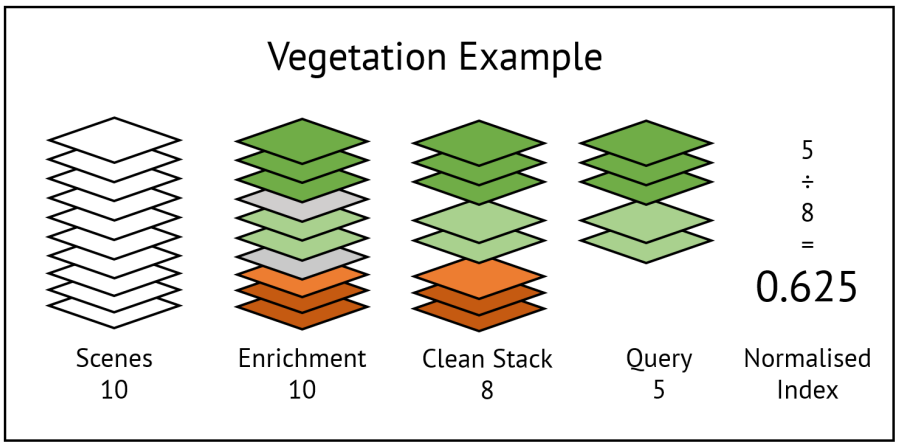

Figure 4.1: This demonstrates how vegetation-like semi-concept occurrence aggregated over time is calculated. The clean stack contains no "invalid" observations, such as no-data or cloud-like semi-concepts. In this case, the query demonstrated is for all vegetation semi-concepts (i.e. green in the figure), but it could be anything else. Author's illustration.

The aggregated time-series function delivers a single geospatial layer, where the time dimension is aggregated to one value per pixel, either a single value, count, index, or otherwise. It summarizes the percentage of per-pixel observations where the queried semi-concept(s) were detected based on the total number of "clean" pixels. In this case, "clean" refers to the exclusion of cloud-like and unknown semi-concepts, as well as pixels with a measured value of 0 in any of the 6 bands required for SIAM™, which are identified using the no-data mask generated during pre-processing (*see* Figure 3.3.2). An example for one pixel being queried for vegetation-like semi-concepts is illustrated in Figure 4.1.

This time-series function is implemented in Python and can deliver more than just the aggregated occurrence referenced above. It can also deliver the minimum or maximum value over the temporal extent, which, for semi-concept-based analysis is not really meaningful. However, a layer containing the total number of observations matching a query (e.g. the total number of vegetation-like semi-concepts detected in a pixel over time in Figure 4.1) and a layer containing the total number of clean observations can also be generated. These aid in assessing the differences in data through space that may affect your result. I also decided to run the time-series function on the no-data mask for the spatio-temporal query extents in order to also know exactly how many Sentinel-2 scenes were available. This allows one to better qualify the

total number of observations in the context of available data but also interpret the normalised index in the context of the initial data source.

4.2 MAPS

Below are visualised outputs from the implementation for two different spatial-extents with multiple temporal-extents. The first covers nearly 3 years of data over the whole study area looking at vegetation- and water-like semi-concepts, as well as the distribution of data through space. The second repeats a similar query for vegetation-like semi-concepts from 1 May to 15 June in three separate years (2016, 2017, 2018), and the difference between the normalised vegetation-like observation occurrence is calculated, but requires further discussion (*see* Section 5.1.2).

4.2.1 *Exploration: June 2015-2018*

An aggregated time-series analysis on the entire geo-spatio-temporal extent of the cube was conducted (i.e. from 28 June 2015 until 22 June 2018 over latitudes 36.01°-37.05°N and longitudes 35.67°-39.11°E in EPSG:4326), effectively covering 3 years of data in 591 Sentinel-2 scenes. The longest temporal axis reported when querying the data cube was 282 observations, but this was only outside of the valid data area where the no-data masks do not cover the entire tiling scheme of the ingested information layer product because the ingested grid is larger than the actual granule footprints. The highest number of observations at any valid pixel within the granule footprints is 258 and the shortest is 150.

Ranging from 4 to 5 hours per query.

Three queries were conducted. First, vegetation-like (i.e. using the vegetation mask with a value of 1) and water-like semi-concepts (i.e. categories 21-24 based on the SIAM™ granularity of 33 semi-concepts) were queried. Both queries took roughly 5 and 4 hours, respectively, to process all of the data, which was divided into 370 cells (i.e. tiles) of 10km by 10km by the maximum number of observations in time, and then mosaic them together. These queries are visualised in terms of the total number of matching observations, as well as the normalised occurrence index (i.e. value from 0 to 1, with 1 meaning an occurrence of 100% of the clean observations). The clean pixel layer is the same for both queries, and is what is used to normalise the values. "Unclean" pixels are defined as any pixels with a value of 1 in the no-date mask, and categories 25, 29, and 33 (i.e. cloud-like, ice-like and unclassified) based on the SIAM™ granularity of 33 semi-concepts. The total number of pixels not included in the no-data mask (i.e. number of total valid pixels) was also calculated in order to show the spatial distribution of aggregated acquisitions on a per-pixel basis, but this unfortunately

cannot take into account the variability distribution through time. This query took roughly 4 hours to complete and mosaic.

There is an issue in mosaicing output generated on a tile-by-tile basis. The strange grid lines on each of the resulting maps are one pixel wide and related to missing pixels separating each tile of the tiling scheme (i.e. 10km), meaning they have a value of 0 (*see* Section 5.3.4 for more information).

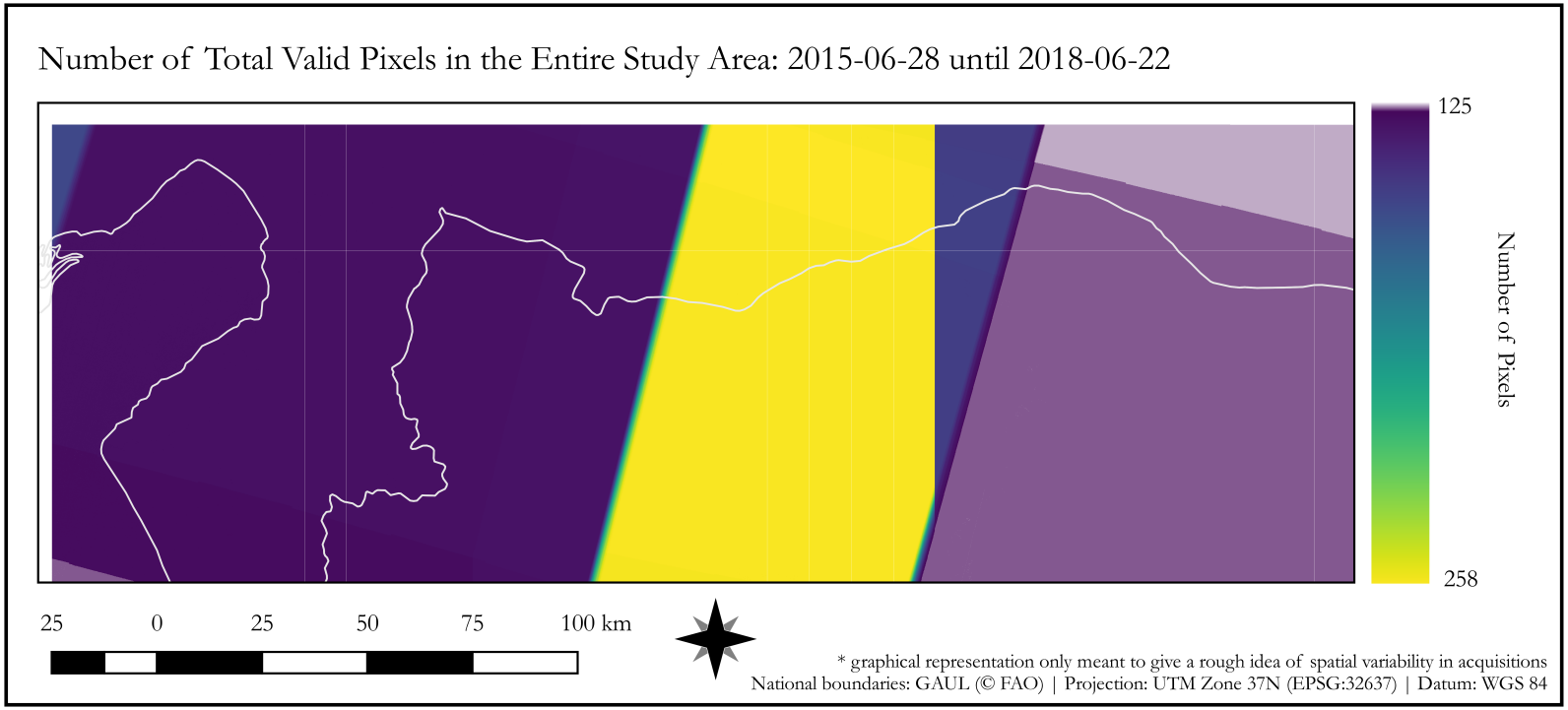

Figure 4.2: The number of total pixels containing measured values not equal to 0 in the original Sentinel-2 data based on nearly 3 years of data. Values 0-125 are white (because there are none) and the lilac purple colour begins at value 126, the lowest number of acquisitions.

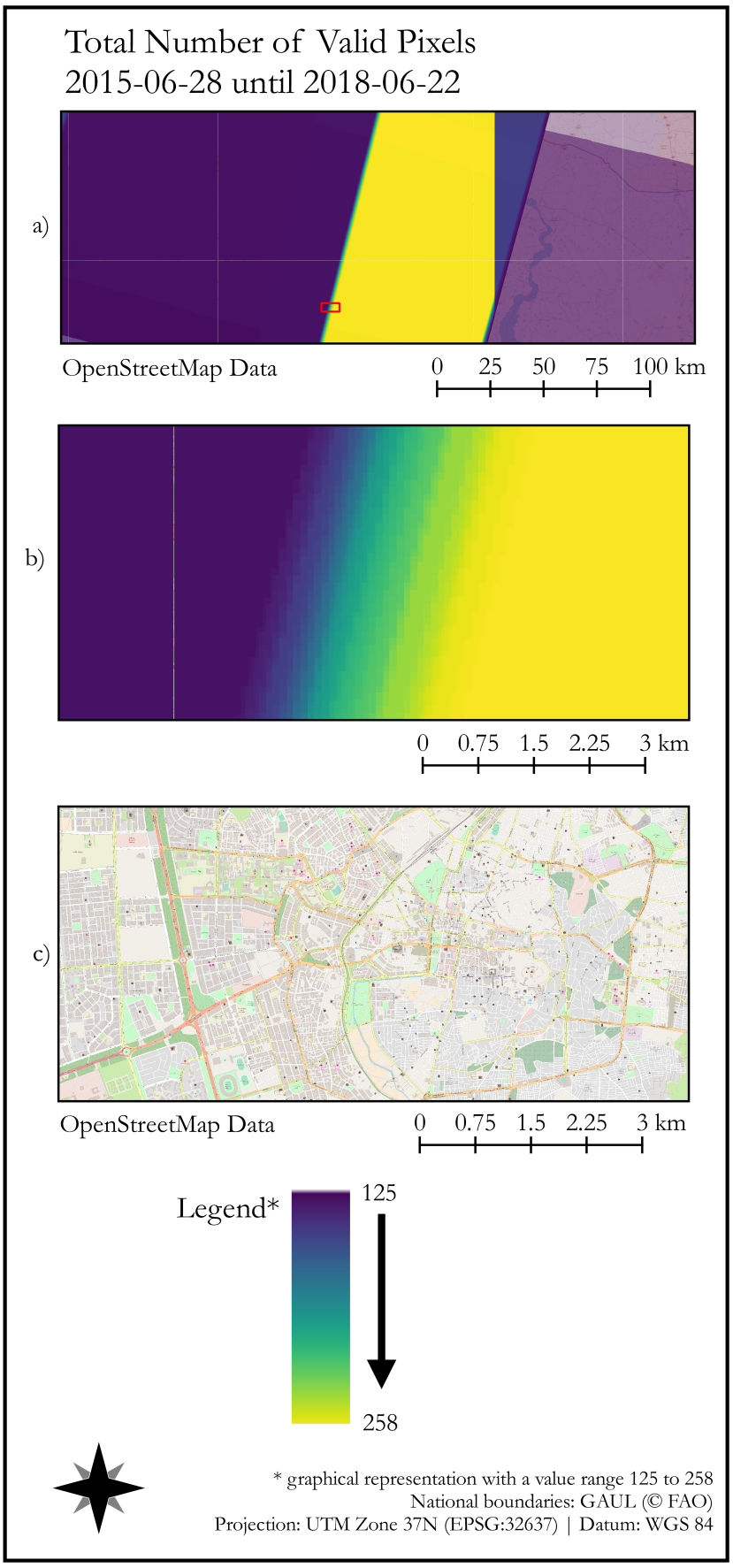

Figure 4.3: A closer look at the number of total pixels containing measured values not equal to 0 in the original Sentinel-2 data. Values 0-125 are white (because there are none) and the lilac purple colour begins at value 126, the lowest number of acquisitions. (c): same area as in (b), but with OpenStreetMap data for context.

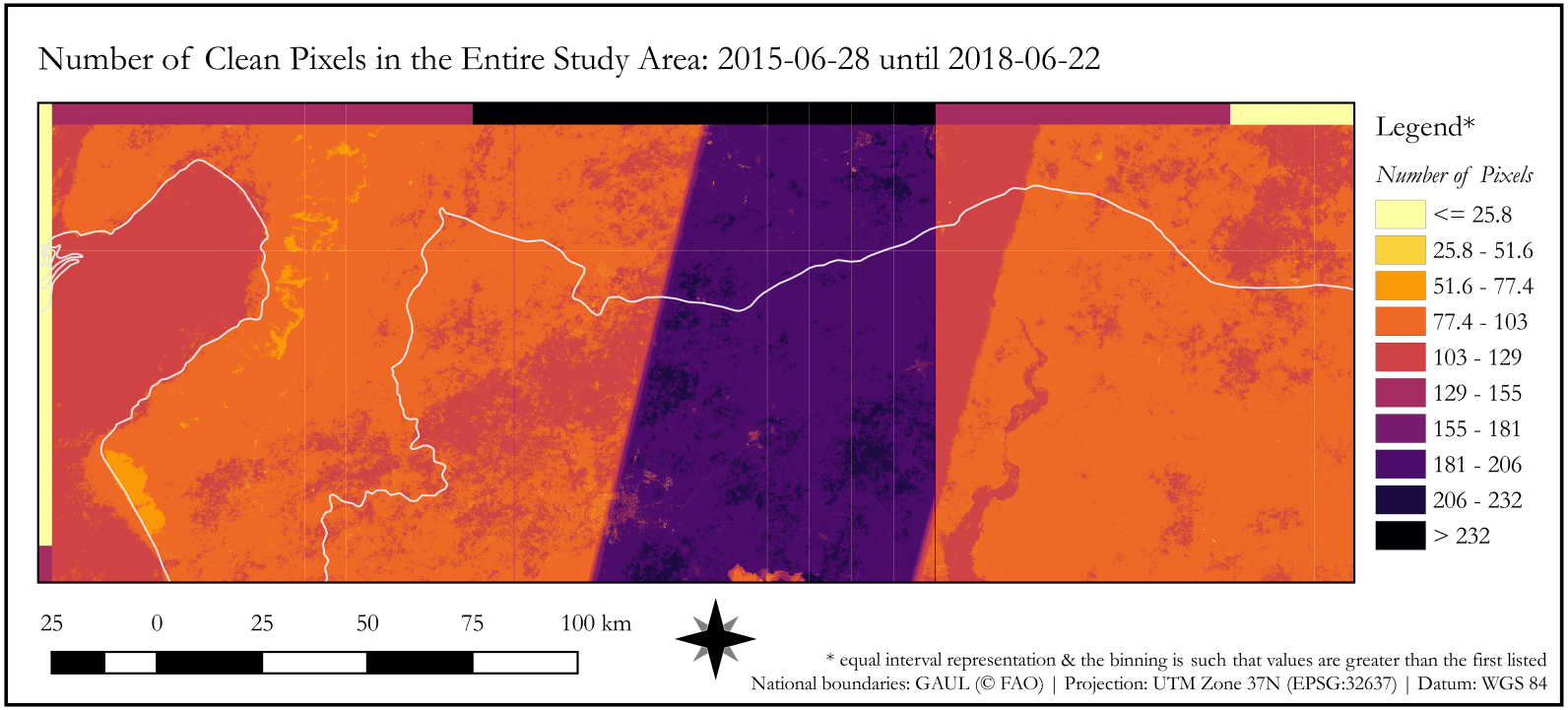

Figure 4.4: The number of clean pixels based on nearly 3 years of data visualised using an equal interval legend using 10 classes with values ranging from 0 to 258.

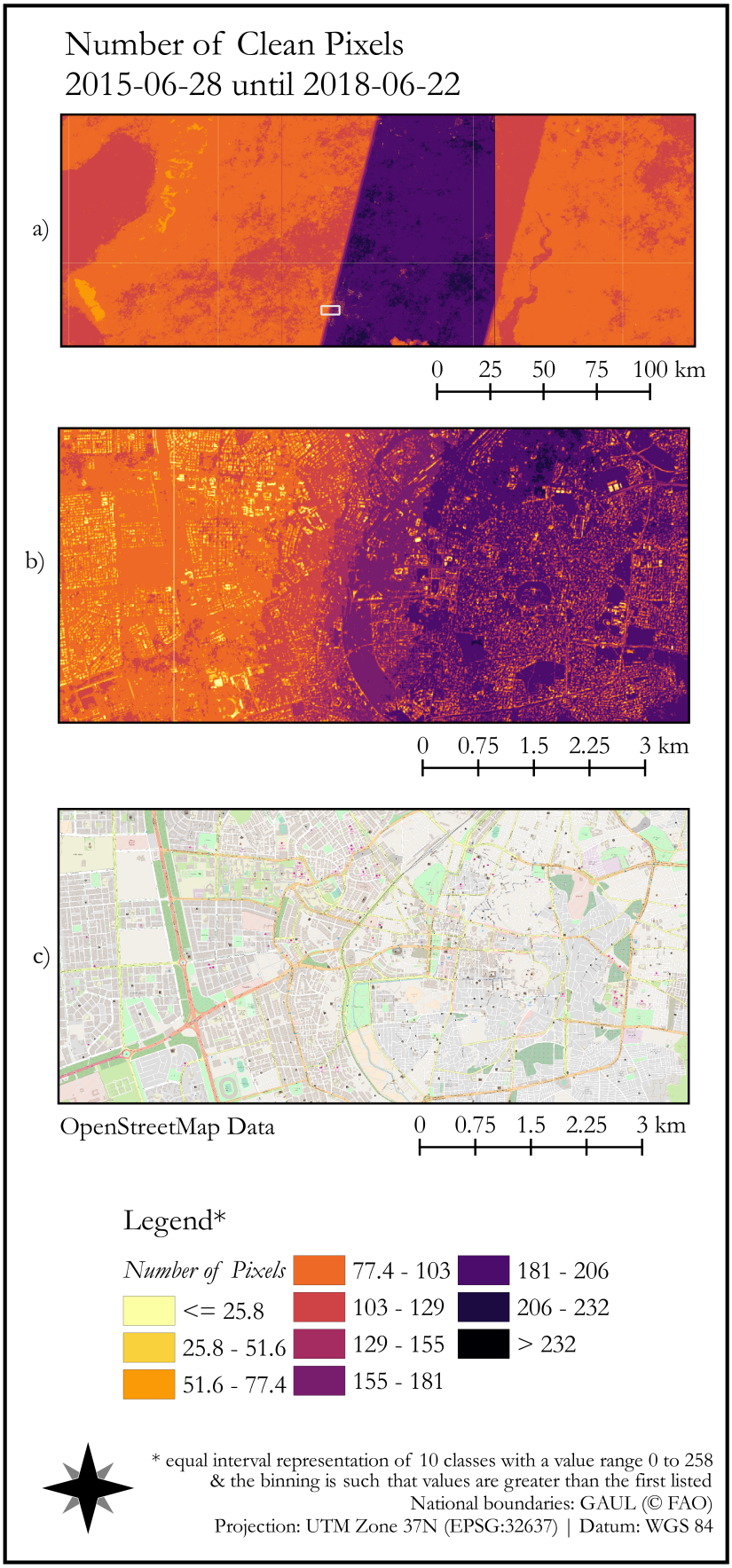

Figure 4.5: A closer look at the number of clean pixels based on nearly 3 years of data visualised using an equal interval legend using 10 classes with values ranging from 0 to 258. (c): same area as in (b), but with OpenStreetMap data for context.

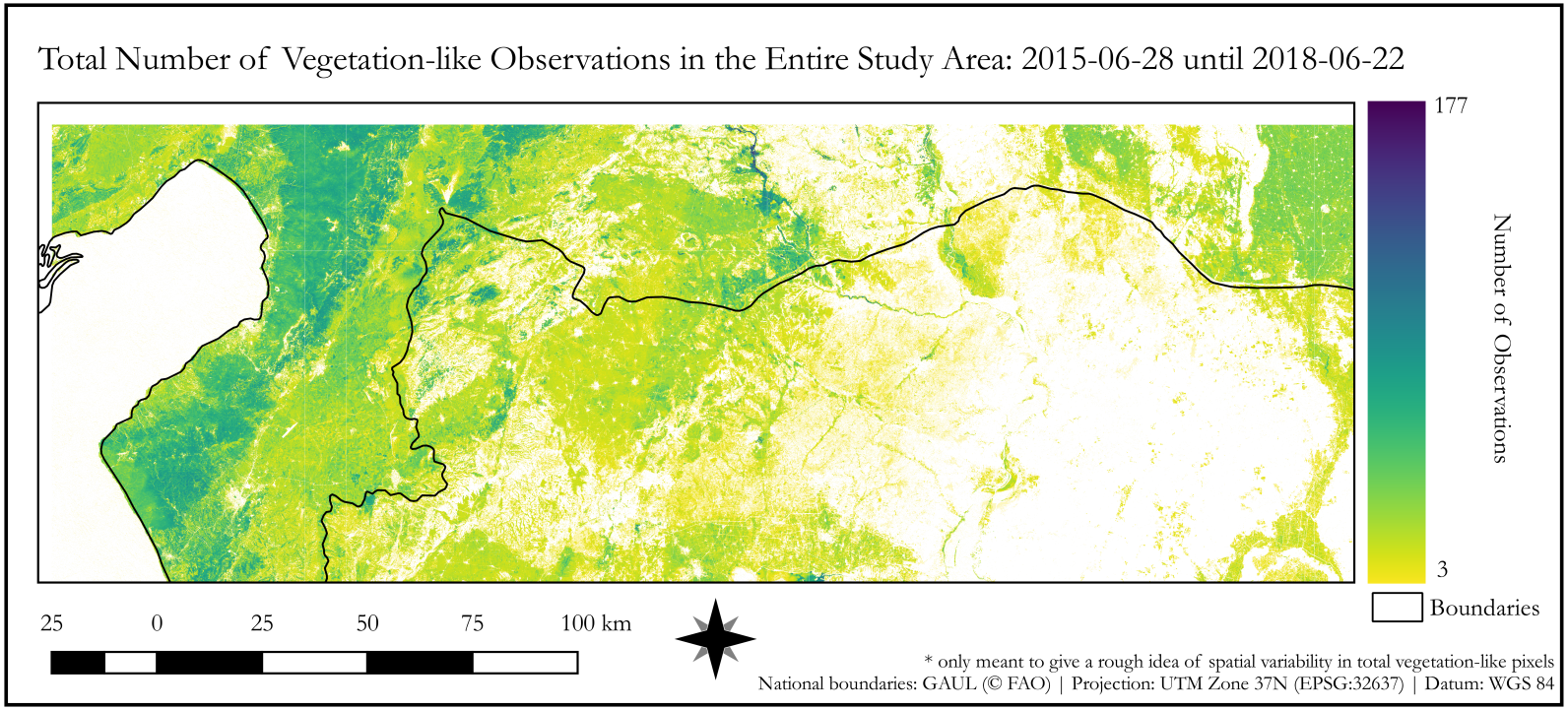

Figure 4.6: The number of total vegetation-like pixels observed based on nearly 3 years of data visualised. This is just meant to illustrate pixels having more than 3 observations up to 177 of vegetation-like semi-concepts over 3 years.

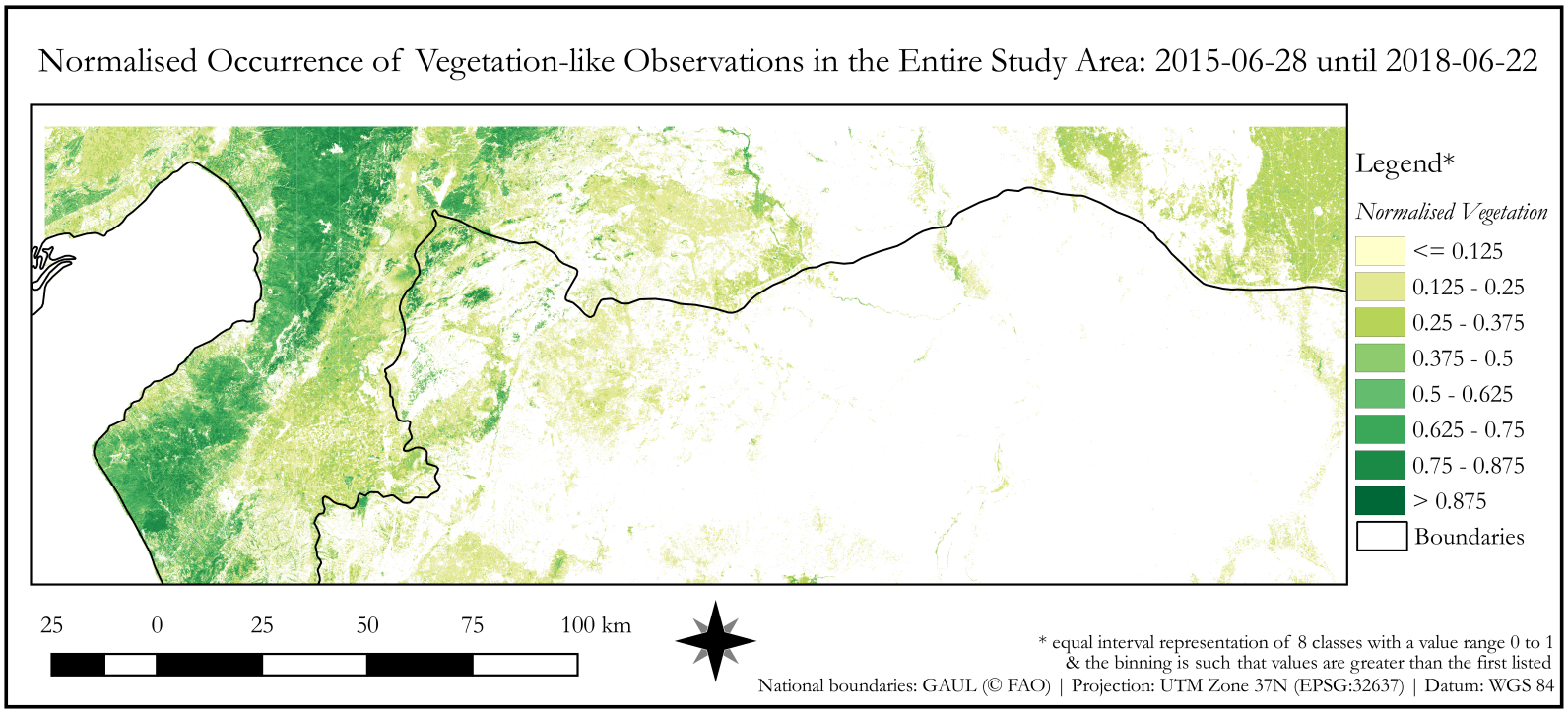

Figure 4.7: The normalised index of vegetation occurrence over nearly 3 years of data visualised using an equal interval legend using 8 classes. Values below or equal to 0.125, including values of 0 are coloured white.

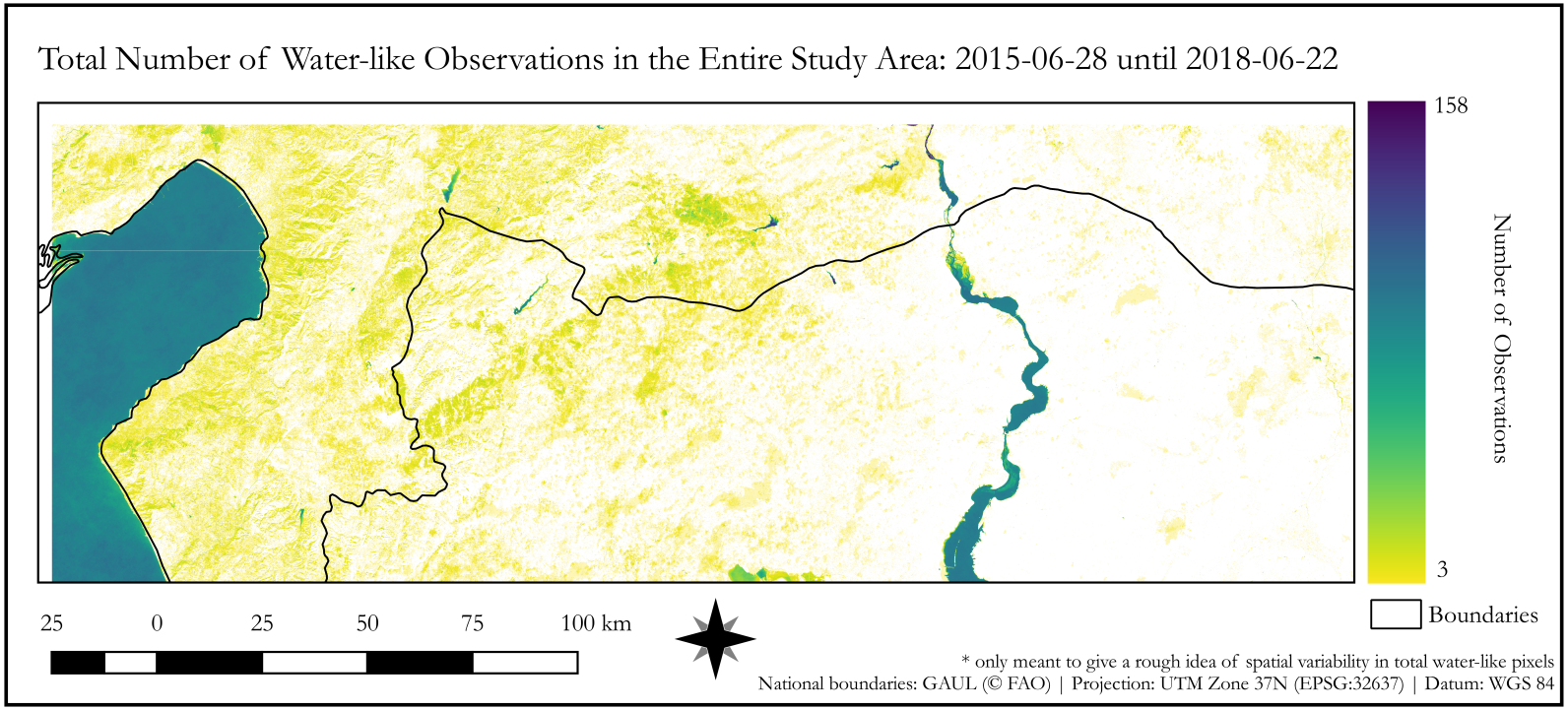

Figure 4.8: The number of total water-like pixels observed based on nearly 3 years of data visualised. This is just meant to illustrate pixels having more than 3 observations up to 158 of water-like semi-concepts over 3 years.

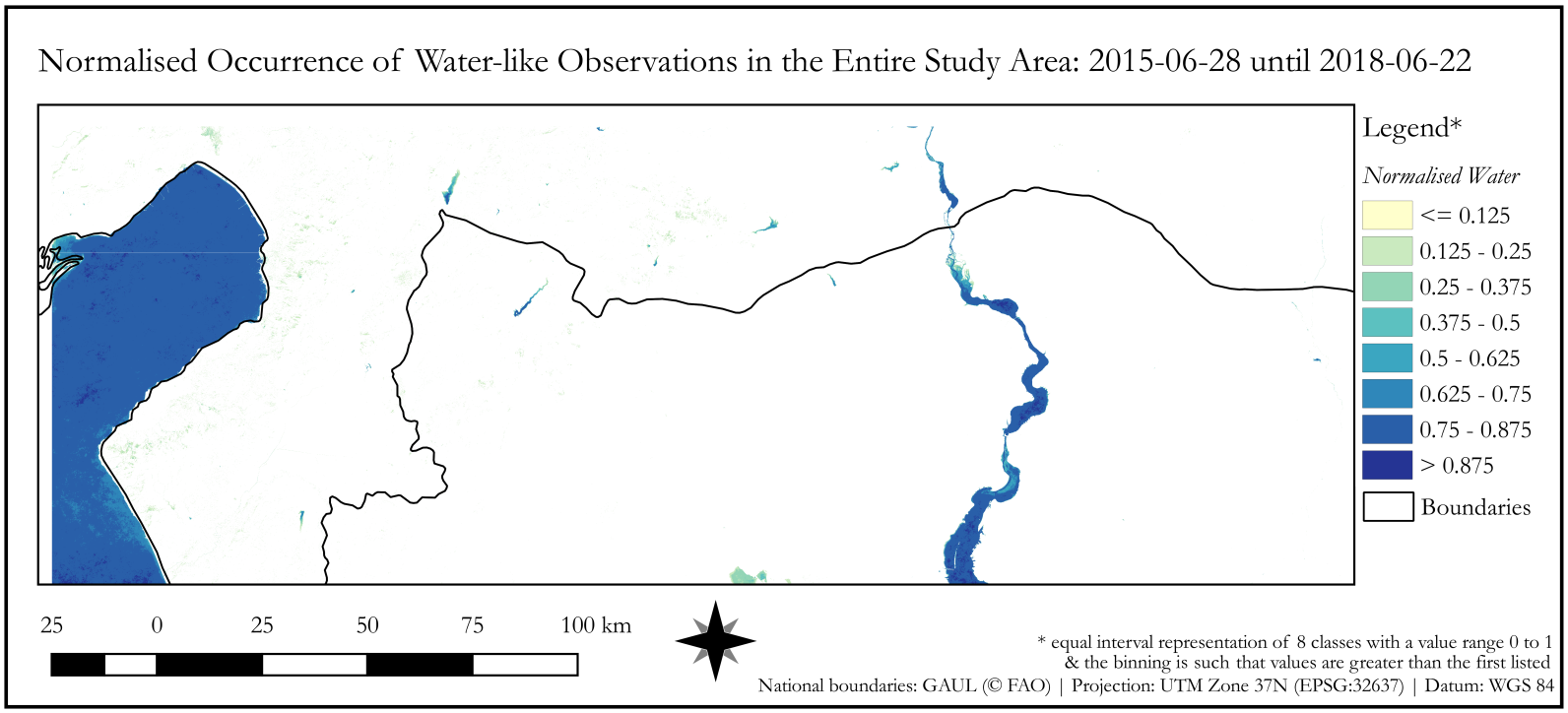

Figure 4.9: The normalised index of water occurrence over nearly 3 years of data visualised using an equal interval legend using 8 classes. Values below or equal to 0.125, including values of 0 are coloured white.

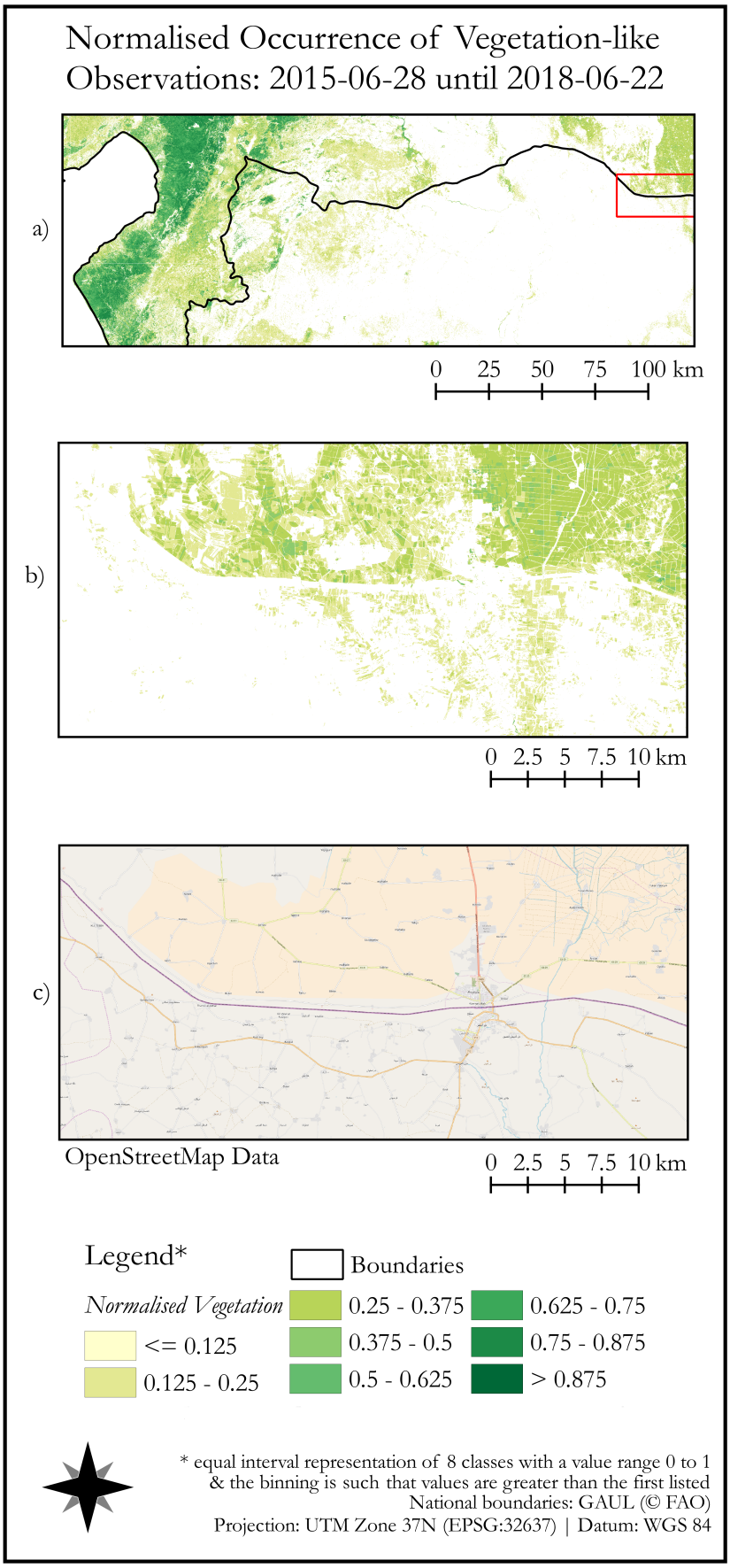

Figure 4.10: A closer look at the normalised index of vegetation occurrence along the border to Turkey. Values below or equal to 0.125, including values of 0 are coloured white. (c): same area as in (b), but with OpenStreetMap data for context.

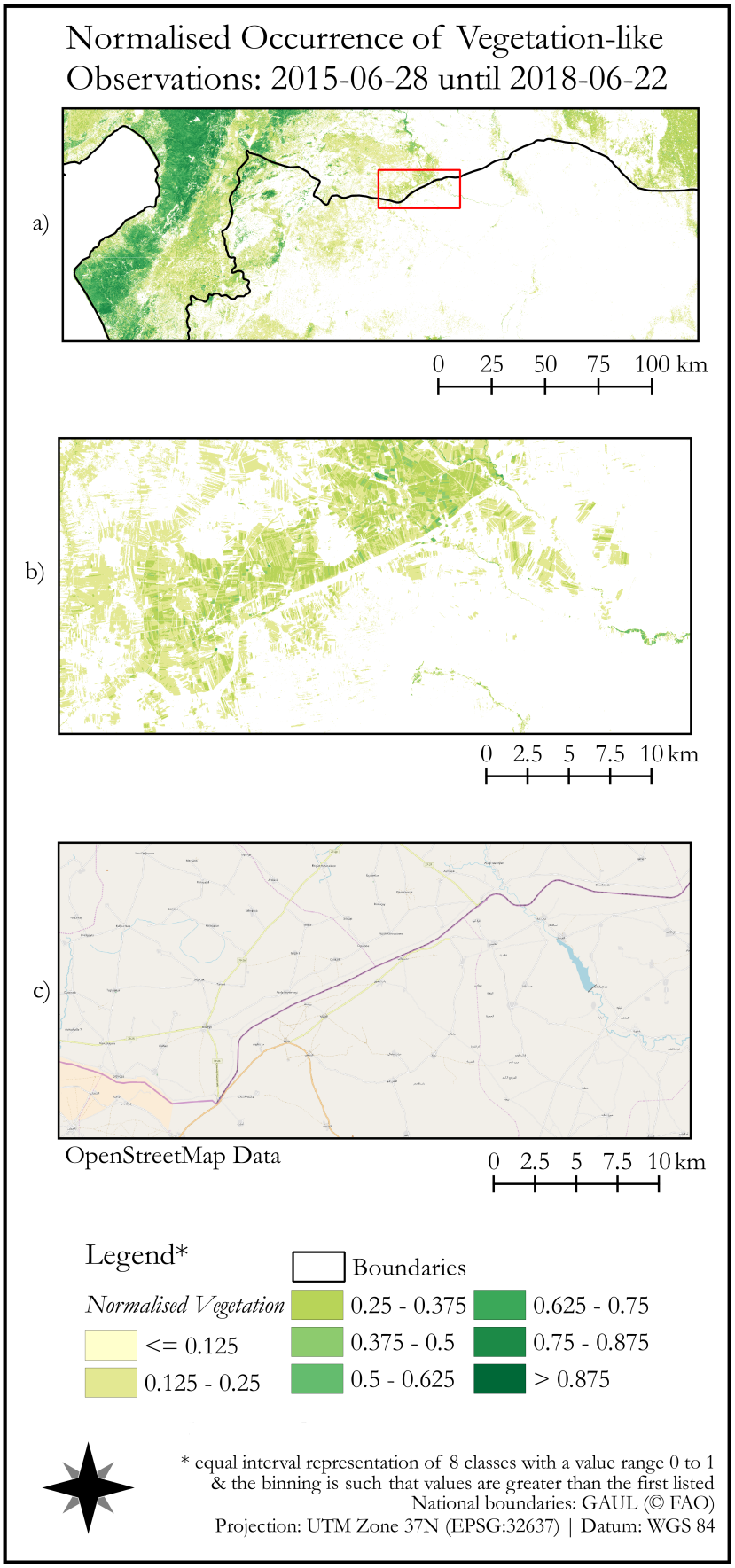

Figure 4.11: Another closer look at the normalised index of vegetation occurrence along the border to Turkey. Values below or equal to 0.125, including values of 0 are coloured white. (c): same area as in (b), but with OpenStreetMap data for context.

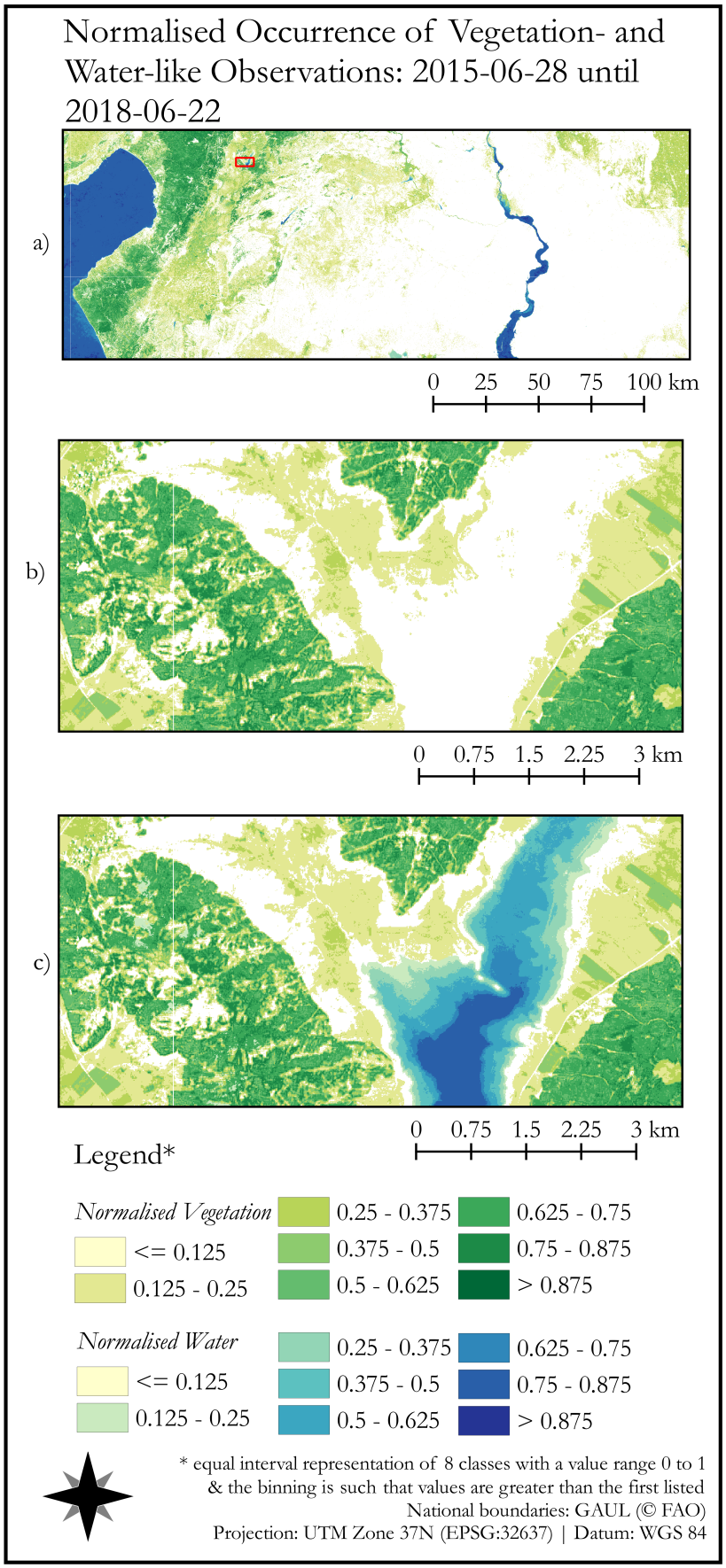

Figure 4.12: A closer look at the normalised index of vegetation but also water occurrence in Turkey. Values below or equal to 0.125, including values of 0 are coloured white. (c): same area as in (b), but with water occurrence overlaid on top of vegetation occurrence.

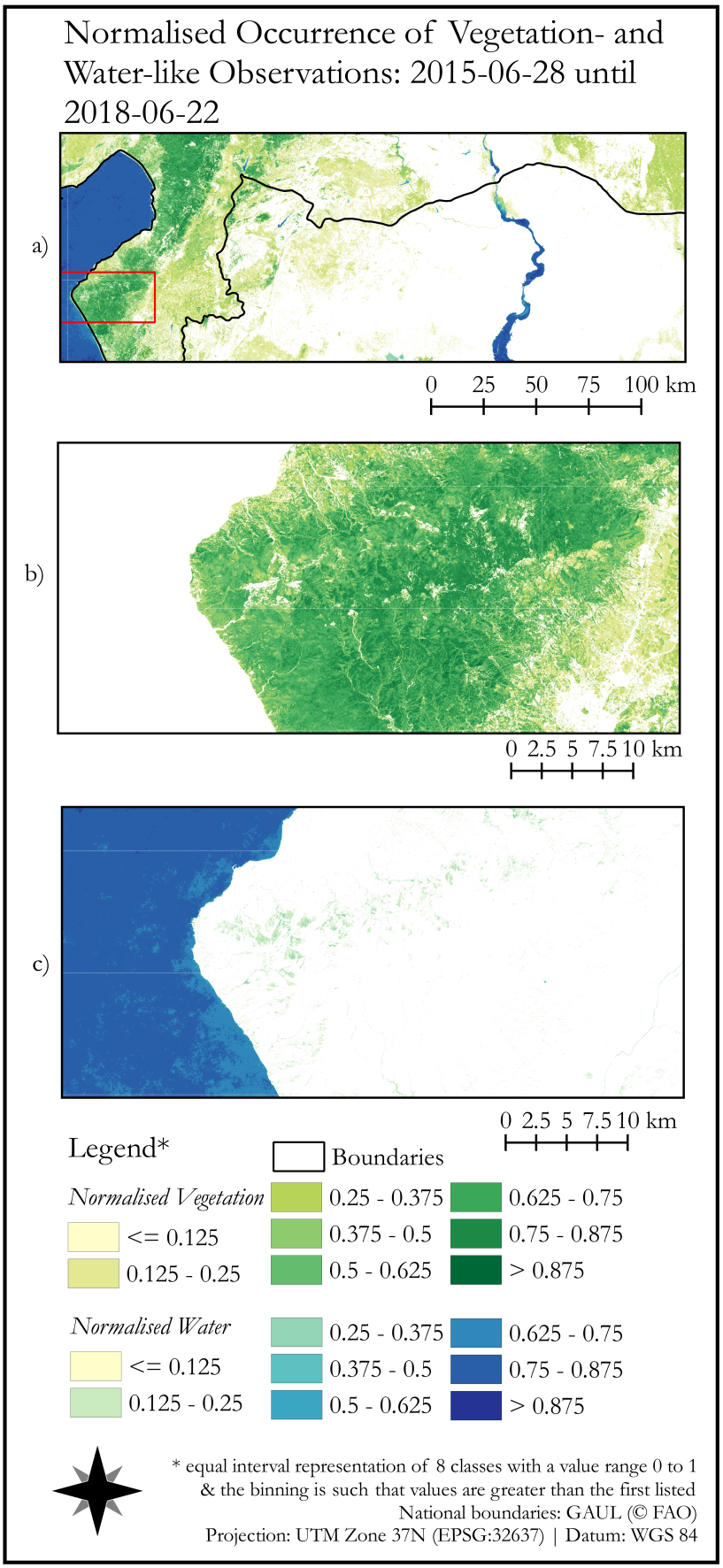

Figure 4.13: Another closer look at the normalised index of vegetation but also water occurrence in Turkey. Values below or equal to 0.125, including values of 0 are coloured white. (c): same area as in (b), but with water occurrence. Here some artefacts from terrain shadows and clouds are visible.

4.2.2 *Exploration: Multi-year May-June*

An aggregated time-series analysis for the same spatial extent in three different years was conducted (i.e. from 1 May to 15 June in 2016, 2017 and 2018) over the area in Syria known as Afrin (i.e. latitudes 36.2°-37.0°N and longitudes 36.3°-37.3°E in EPSG:4326). The area contained 100 different 10km by 10km cells with differing numbers of scenes available per year and spatial location.

The number of time-steps available per year were: at most 12, but generally 8 in 2016; at most 9 but otherwise 5 in 2017; and at most 17, but otherwise 9 in 2018. In 2016, a count of 12 was reported as the largest number of acquisitions, but the highest value to actually appear is 8. This is assumed to be due to edge effects of the tiling scheme, overlapping acquisition swaths or some of the granules containing predominantly no data offered earlier in the lifespan of Sentinel-2A. Further exploration is necessary to understand why this value is what it is.

Around 10 minutes to process per year.

These results took approximately 10 minutes for each year to cover Afrin, including aggregated time-series analysis of vegetation semi-concepts (~7 minutes) and the no-data mask to ascertain the number of underlying valid scenes per-pixel (~2 minutes).

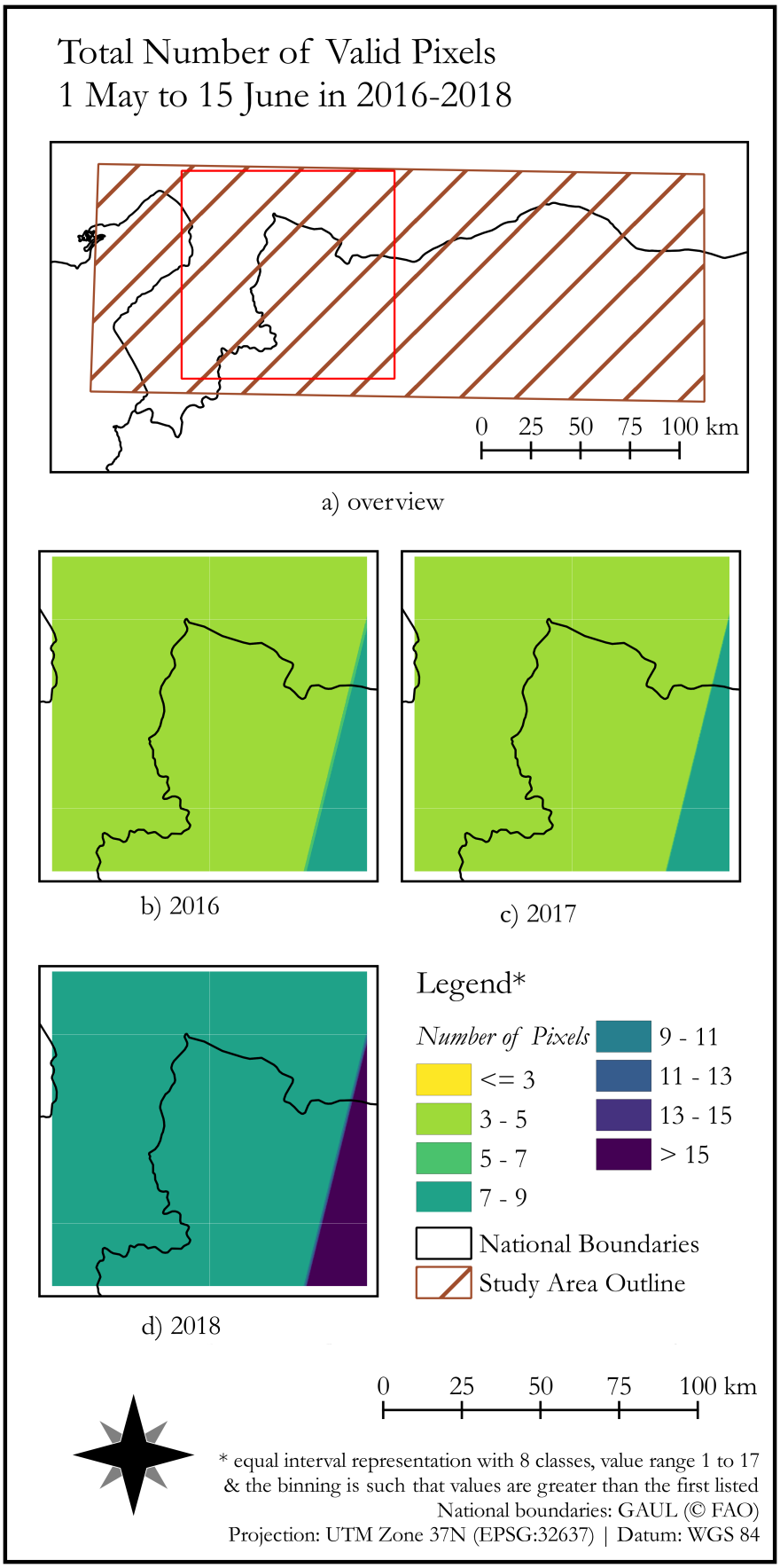

Figure 4.14: (a): the spatial extent of the query; (b, c and d): the number of acquisitions available from 1 May to June 15 in each of the years. 2018 has considerably more acquisitions because Sentinel-2B became operational in the months following June 2017.

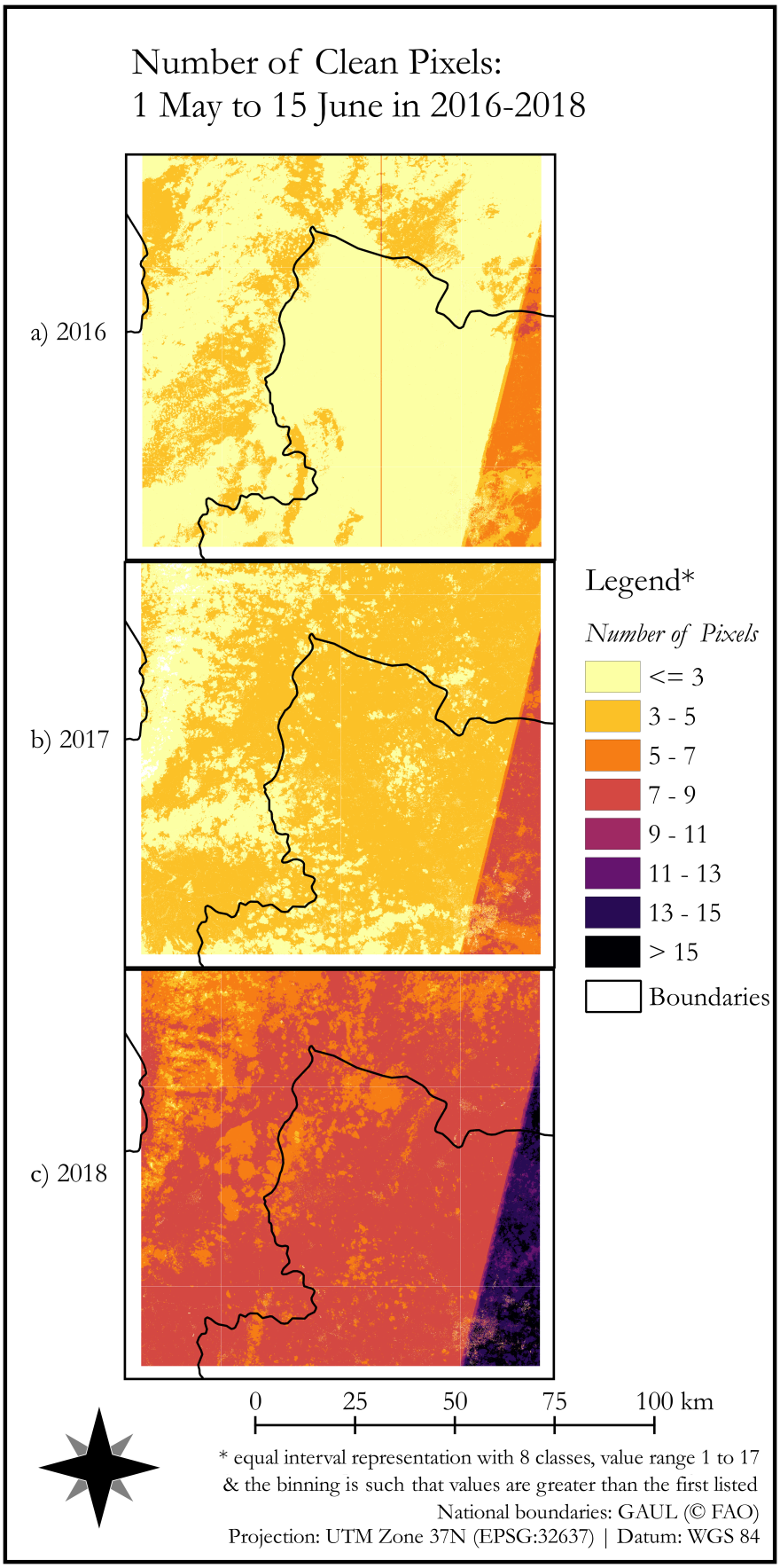

Figure 4.15: The number of clean pixels available for 1 May to 15 June in each year. Artefacts from cloud-like semi-concepts and surfaces with a higher reflectance can be seen, but also the overlapping swaths. The city of Aleppo is located in the lower right corner.

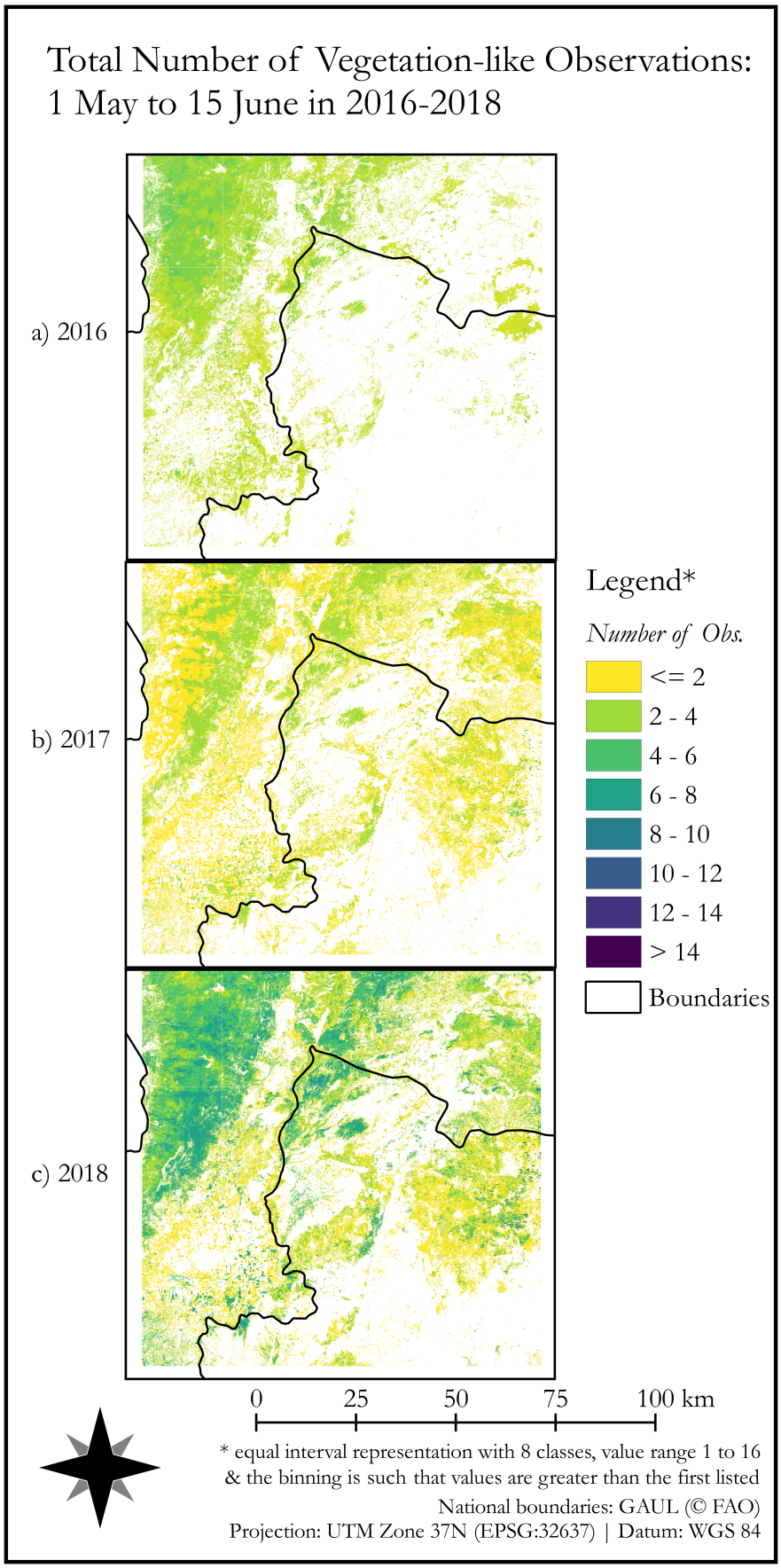

Figure 4.16: The total number of vegetation-like observations available for 1 May to 15 June in each year. Pixels with a value of 0 are white. Differences in the baseline total number of observations for each year is visible.

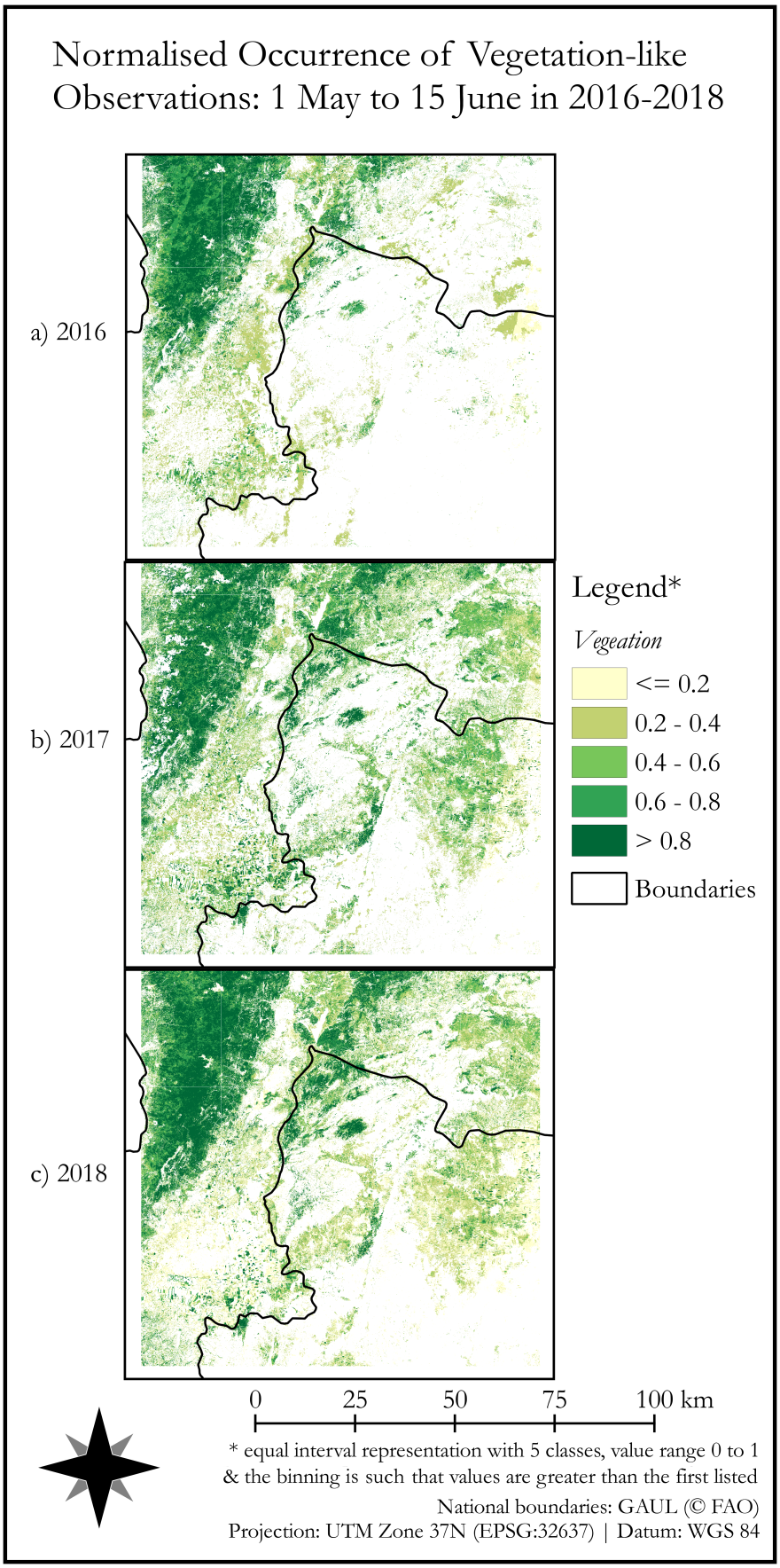

Figure 4.17: The normalised occurrence of vegetation-like observations available for 1 May to 15 June in each year. Be wary when interpreting, since some areas only have 6 moments in time to even work with (i.e. two vegetation-like observations automatically can mean at least 33%, and if clouds were observed once, 40%!)

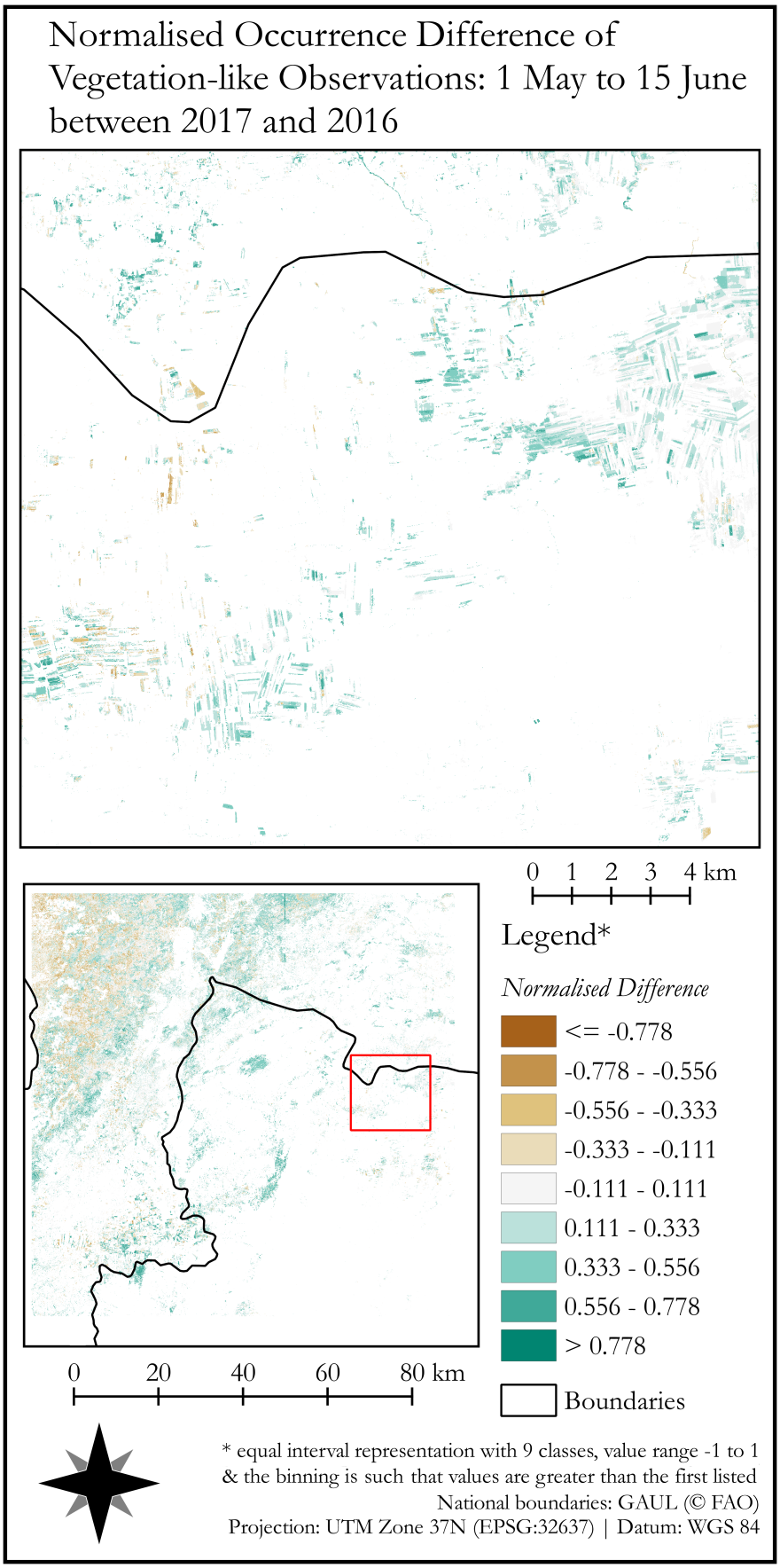

Figure 4.18: Difference between the normalised occurrence of vegetation-like observations from 1 May to 15 June between 2017 and 2016. The 2016 results are subtracted from 2017, meaning negative values had a higher observed vegetation occurrence in 2016 than in 2017 (reddish), values of 0 had no change, and positive values had higher observed vegetation occurrence in 2017 than 2016. Values ±0.111 are white.

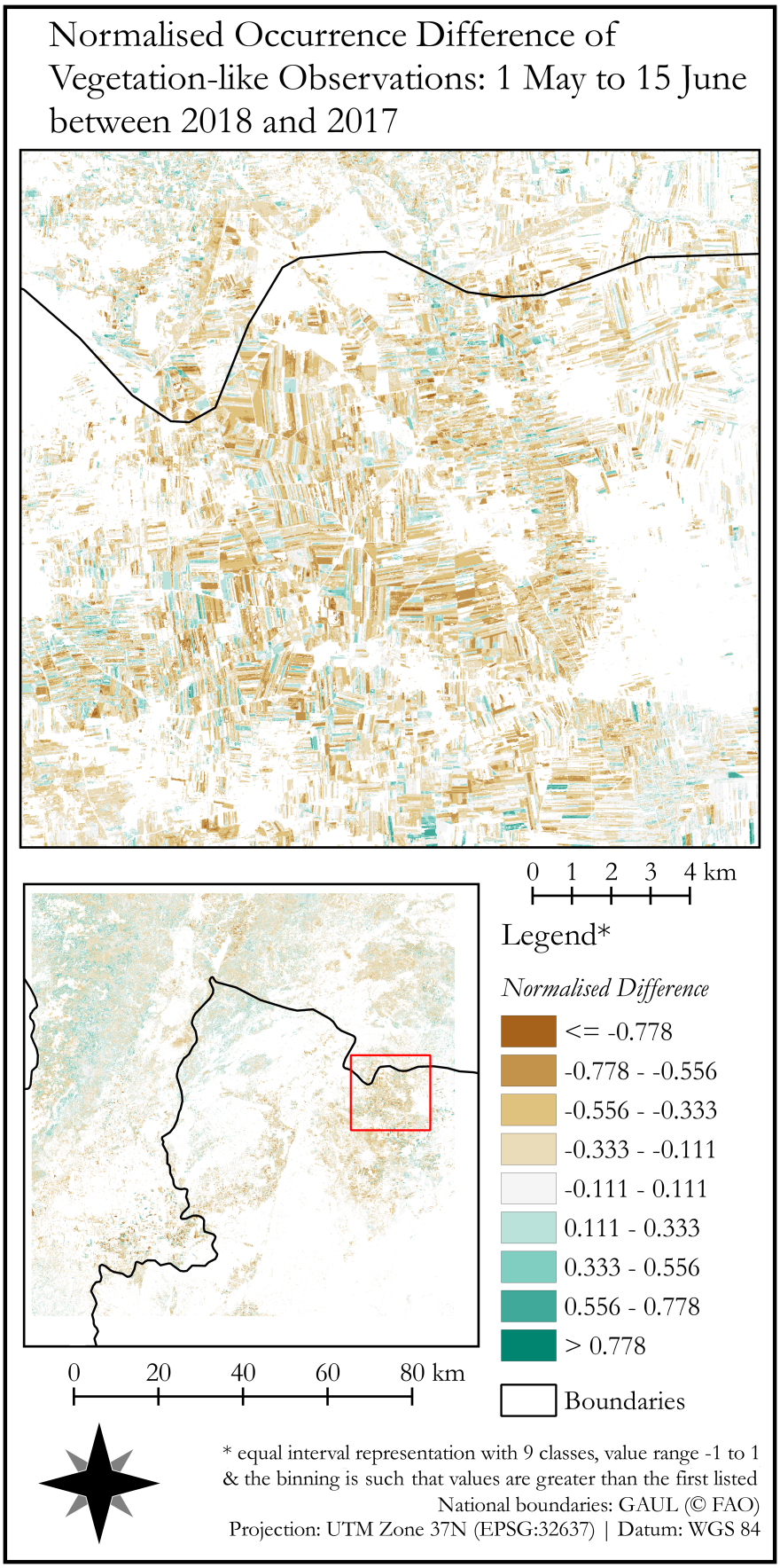

Figure 4.19: Difference between the normalised occurrence of vegetation-like observations from 1 May to 15 June between 2018 and 2017. The 2017 results are subtracted from 2018, meaning negative values had a higher observed vegetation occurrence in 2017 than in 2018 (reddish), values of 0 had no change, and positive values had higher observed vegetation occurrence in 2018 than 2017. Values ±0.111 are white.

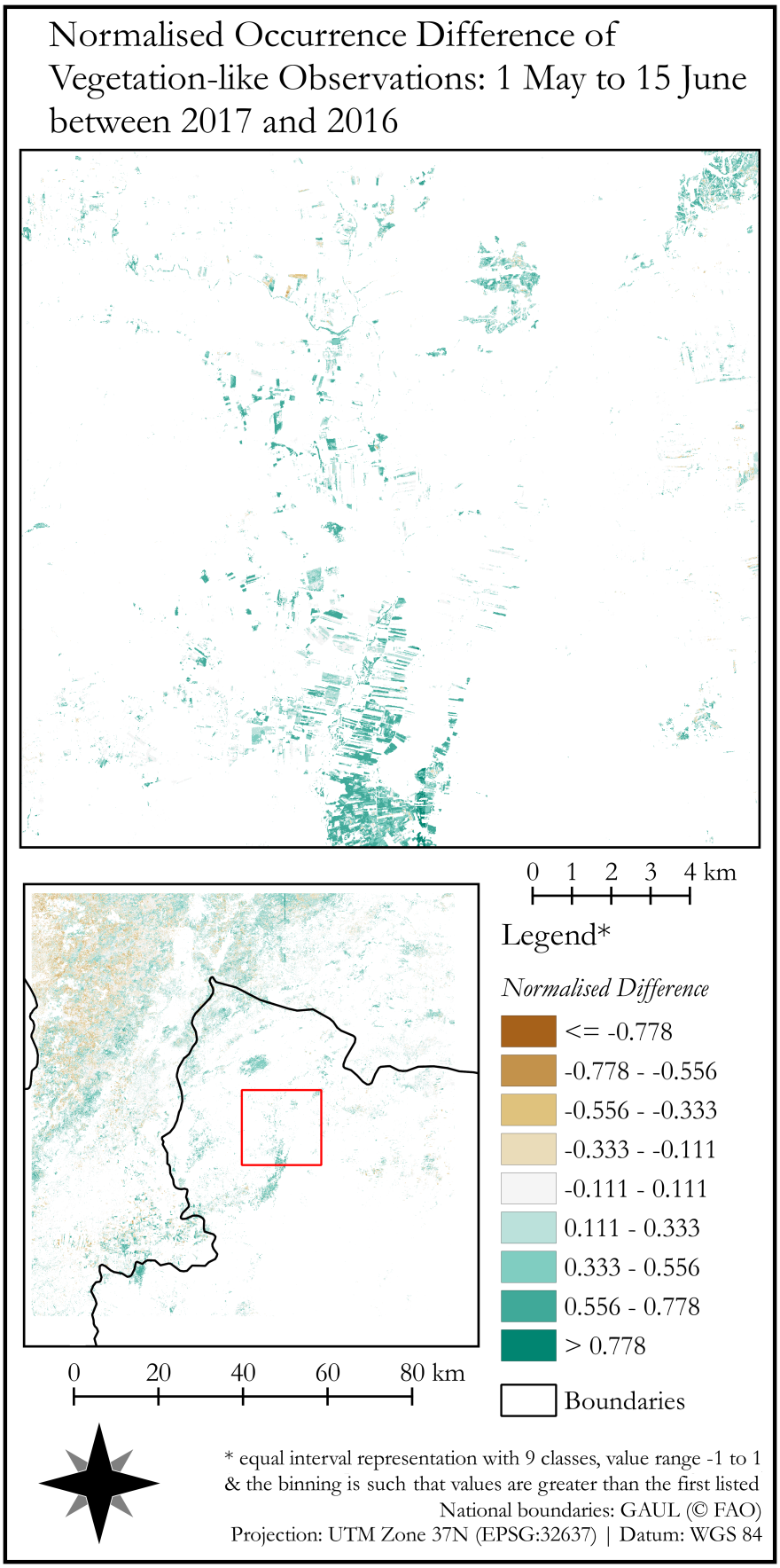

Figure 4.20: Difference between the normalised occurrence of vegetation-like observations from 1 May to 15 June between 2017 and 2016. The 2016 results are subtracted from 2017, meaning negative values had a higher observed vegetation occurrence in 2016 than in 2017 (reddish), values of 0 had no change, and positive values had higher observed vegetation occurrence in 2017 than 2016. Values ±0.111 are white.

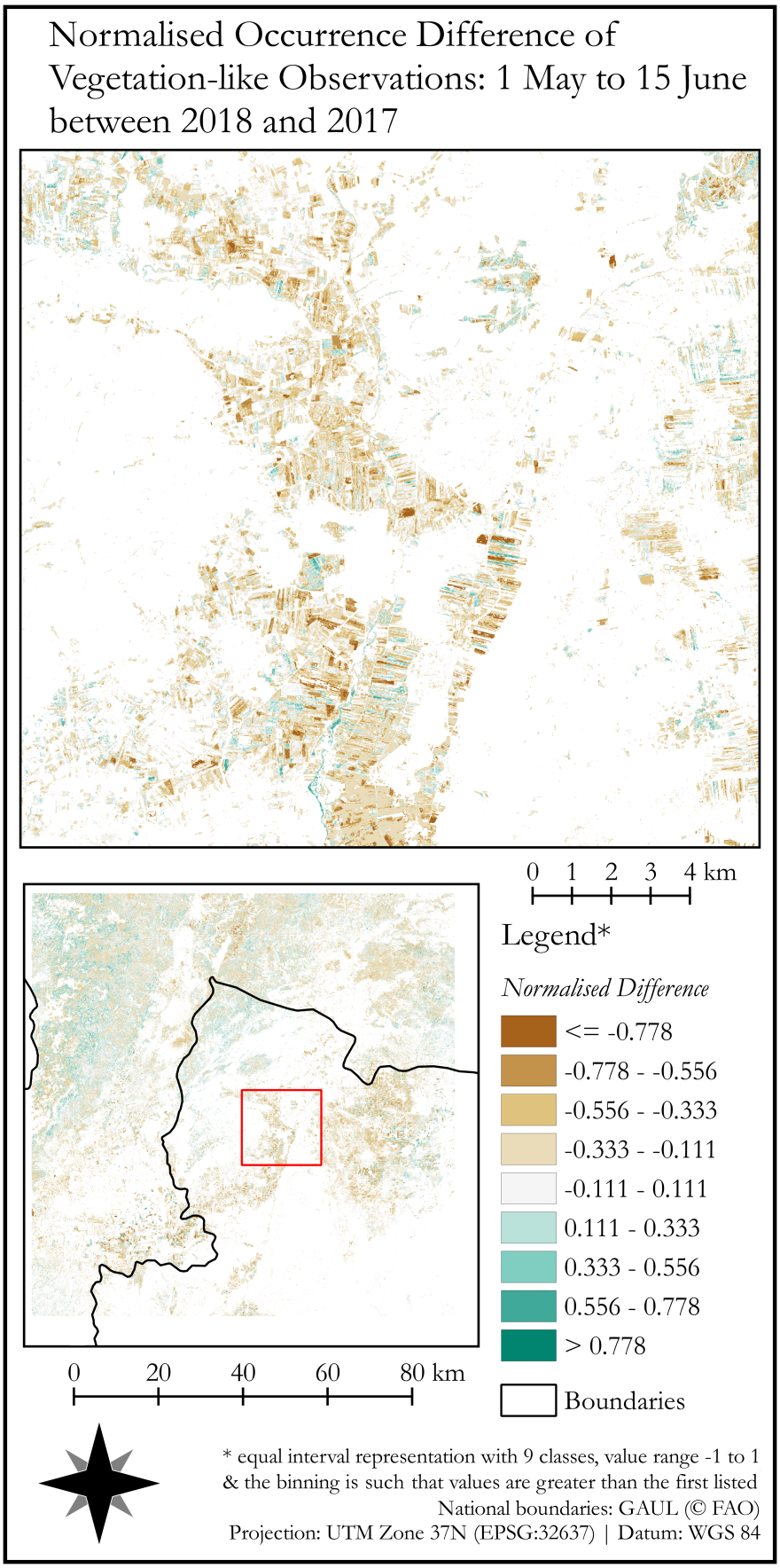

Figure 4.21: Difference between the normalised occurrence of vegetation-like observations from 1 May to 15 June between 2018 and 2017. The 2017 results are subtracted from 2018, meaning negative values had a higher observed vegetation occurrence in 2017 than in 2018 (reddish), values of 0 had no change, and positive values had higher observed vegetation occurrence in 2018 than 2017. Values ±0.111 are white.

5

DISCUSSION

This section looks more closely at the proof-of-concept results provided in Chapter 4 derived exclusively from a dense time-series of automatically semantically enriched Sentinel-2 scenes. Finer details pertaining to the data, design decisions and methods used, including challenges faced along the way are explored. It is structured in a way that separates the output of the system from the original Sentinel-2 data and implementation methods.

5.1 INTERPRETATION OF MAPS

Interpreting aggregated time-series output from this implementation is not as easy as one might think. The aggregated time-series results can be visualised in one layer, but are just one representation of complex processes, events, etc. Spatial statistical methods exist for assessing variability in space, but the output here has a temporal dimensions as well as at least one semantic or thematic dimension. Finding ways to visualise and incorporate different temporal variability and distribution, as well as different levels of confidence in semi-concept fitness for a given query or application are points that need to be explored further, and are ultimately outside the scope of this thesis.

It is clear that not very much geospatial information open to the public, not to mention an English-speaking public, has been coming out of Syria over the past many years. That aside, there are very few datasets that exist anywhere in the world that can reasonably be used to check for validity or agreement of results that have a 10m spatial resolution over 30,000km^2, not to mention with a similar density or temporal frequency.

Free and open vector datasets (e.g. OpenStreetMap data, national data) might be helpful when a query is about determining the extent of certain objects at moments in time, or on average (e.g. water bodies, national forests). However, when it comes to assessing the dynamics of a particular kind of land cover (e.g. surface water dynamics), there is very little that exists to rely on in much of the world. Not only the number of acquisitions that output is based on matters, but often it is incredibly important *when* the acquisitions with "clean" pixels were acquired.

That being said, the following exploratory outputs offer plenty of questions, some potential insights, and ideas for future work. A rough reference to existing monthly average rainfall is included in order to assist in contextualising vegetation and water dynamics (*see* Figures 5.1 and 5.2). Temperature and other additional information would also likely be very useful, depending on the application. Existing vector information about irrigated areas could also be used to check plausibility of vegetation-like observations, as done by Tiede et al. (2014), but the resolution of such data is based on a 250m by 250m grid. The conflict in Syria has likely impacted the actual area of actively irrigated agricultural production significantly within the last 8 years, due to displacement of people and damage to irrigation infrastructure. Without employing some form of object-based image analysis to process the output here in an object-based way, the agreement of the output at 10m spatial resolution is difficult to meaningfully assess. OpenStreetMap data has been included in some smaller insets to demonstrate that certain structures can be discerned based on the output from the data cube implementation in order to support establishing plausibility.

5.1.1 *Entire Cube, June 2015-2018*

The multiple exploratory visualisations of different output covering data observed from 28 June 2015 to 22 June 2018 (*see* Section 4.2.1) raise lots of interesting questions to be explored. The ability to compare information derived from 3 years of data with 10m spatial resolution for an area of 30,000km^2 overnight on the hardware and computing environment used here is something that likely would not have been possible a few years ago; it definitely was not possible with the same temporal density before Sentinel-2 existed.

The heterogeneous spatial variability in the number of available acquisitions can be seen in Figure 4.2. Not only are differences in average cloud cover (e.g. higher in mountainous regions to the West) important, but the actual number of available acquisitions. The differences in average cloud cover are somewhat made visible in Figure 4.4, but also differences in the reflectivity of certain surfaces (e.g. what are very likely built-up surfaces, shallow water). Figure 4.5 is the same output of "clean" pixels as shown in Figure 4.4, but one can see that some built-up, urban structures in the city of Aleppo are made visible. It seems to fit well with structures depicted in the inset (c) of the same area containing OpenStreetMap data. These structures are visible due to having a relatively high surface reflectance that are often attributed to an unknown semi-concept category and thus excluded from analysis. Water bodies, especially the northern part of the traditionally seasonal saline lake, Jabbūl, can also be discerned, since pixels containing shallow water (e.g. shorelines) are also often attributed to the unknown semi-concept

category. The same area of Aleppo can be seen in Figure 4.3, where the overlapping swaths are apparent. Perhaps calculating the percentage of "clean" pixels divided by the total number of possible pixels would offer a better idea of the spatial differences in available data an output is based on.

The total vegetation-like observations can be seen in Figure 4.6, which gives an idea of the possible absolute extent of vegetation over the three years, which, due to the very arid climate in the East, is strongly tied to precipitation events. It makes sense, then, that the number of total vegetation-like observations increases from East to West. The normalised values based on available "clean" pixels can be seen in Figure 4.7, but only pixels with an observed occurrence over 12.5% of the 3 years of data are visible. These are calculated exactly as described in Figure 4.1.

Two depictions of differences in the normalised index can be seen in Figures 4.10 and 4.11, showing what is most likely irrigated agricultural land on each side of the Turkish-Syrian border. Some of the articles cited in Section 3.1.1 talked about damage to irrigation infrastructure or reduction in irrigation practices due to the displacement of people. These two figures might be considered as providing evidence supporting that claim in a spatially-explicit way. Without similar data prior to the start of the Syrian conflict, such an assertion based on this output is more than risky and could be due to other causes. However, even if Sentinel-2 data was available before 2015, the presence of intense drought starting around 2007 in Syria definitely caused land cover change, but also limits the ability to attribute changes to possibly being related to conflict. Average monthly rainfall data for the duration of Sentinel-2 observations including 6 months before from the city of Akçakale, Turkey, shown in Figure 4.10, can be seen in Figure 5.1. This is not necessarily useful for interpreting output shown here, but might be useful if three different annual aggregations, or seasonal aggregations in different years are calculated, similar to what is done in Section 5.1.2.

While there are no metrics of statistically-based confidence to offer here, water-like semi-concepts sometimes look very similar to areas with shadow, whether from clouds or terrain. It is important to make clear that even in an exploratory way, water-like semi-concepts move through relatively more uncertainty than looking at vegetation-like semi-concepts without incorporating any additional knowledge or information (e.g. DEM, improved cloud and cloud-shadow masking). As becomes immediately evident when looking at Figure 4.8, the more mountainous region in the West contains observed water-like semi-concepts in areas that are very likely mostly shadows from mountainous terrain (*see* Figure 4.13 (c)). What are likely freshly irrigated (i.e. basically flooded) agricultural fields can also be assumed to be made visible based on their regular, rectangular shapes and very low counts of water-like observations (i.e. depicted in a yellowish tone).

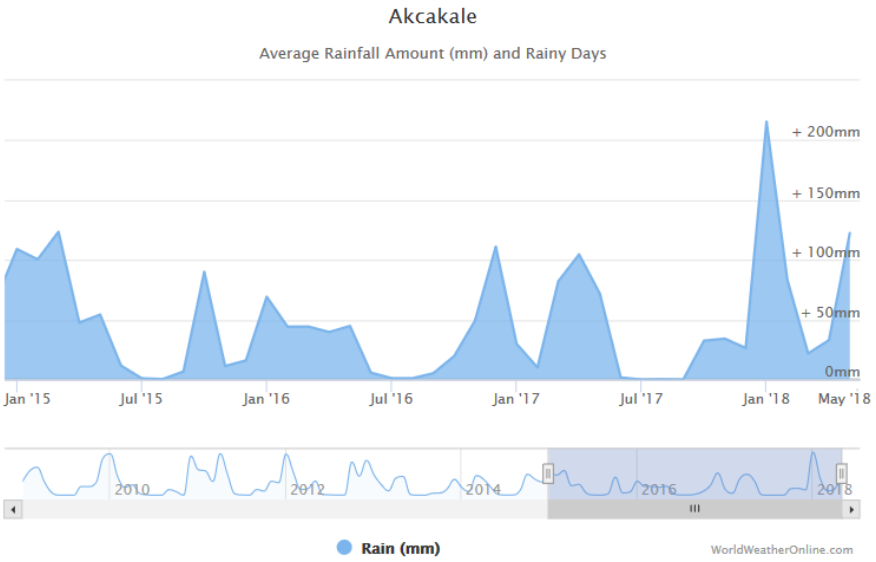

Figure 5.1: A rough idea of precipitation from December 2014 until May 2018 as recorded at Akçakale, Turkey, on the border to Syria. Data and visualisation provided by WorldWeatherOnline.com

The lack of a thermal band in Sentinel-2 imagery also makes dense clouds with relatively high reflectance difficult to discern as being cloud-like, especially when located over water bodies. Based on closer examination of the generated information layers, those pixels are often are attributed to a built-up semi-concept with very high reflectivity. This artefact can also be seen in the Mediterranean sea in Figures 4.8 and Figure 4.13, whereby the huge water body that ought to always be identified as looking like deep water has a range of total water-like observations despite having mostly the same number of available acquisitions (*see* Figure 4.2) and clean pixels (*see* Figure 4.4). M. G. Tulbure et al. (2016) also faced difficulties with shallow water due to high spectral variability when calculating surface water extent dynamics in Australia, so it is not a problem only faced when using generic semi-concepts.

The normalised occurrence of water-like semi-concepts can be seen in Figure 4.9. Pixels with an average occurrence of observed water-like semi-concepts below 12.5%, including values of 0, are coloured white, which leaves only a few artefacts from terrain shadow, that could likely be identified using the aspect and slope of a DEM. The changes in surface water extent can also be seen, again including lake Jabbūl in the South and in (c) of Figure 4.12. Because the spectral profile of pixels containing shallow water seem to sometimes be identified as an unknown semi-concept, ephemeral and seasonal likes may be made visible by this output, but requires further tests for agreement and validation with other methods and datasets before any real conclusions can be made.

5.1.2 *Multi-temporal Springtime in Afrin*

These exploratory visualisations of the same seasonal timeframe (1 May to 15 June) in 2016, 2017 and 2018 (*see* Section 4.2.2) offer different insights than for output aggregated over longer timespans, but also bring different challenges. The idea here is to explore a possible sort of method to move away from bi-temporal change detection and towards utilising temporally denser stacks of data to detect perhaps more meaningful change. Hopefully relying on data aggregated over time, or somehow otherwise including or accounting for time, removes some of the spuriousness that can sometimes occur when an image used in bi-temporal change detection is not actually representative of the process, event, etc. that the change detection is supposed to represent.

Another issue that can arise with bi-temporal change detection is when multiple time-steps are supposed to be compared (e.g. months or years). Different climate factors (e.g. precipitation, temperature) in different climate zones impact land cover differently from season to season, year to year, whether by shifting the phenological cycle (e.g. vegetative peak, lull or periodicity), completely reducing vegetative cover (e.g. a drought) or surface water extent (e.g. a flood), etc. These changes are visible in the few example outputs, especially since a fixed time-frame was chosen in each year (e.g. 1 May to 15 June). For example, depending on the purpose, using a data cube implementation to detect a suitable time-frame including the vegetative peak (i.e. perhaps defined as the largest vegetated extent, or "greenest" semi-concepts using a semantic query) and using that result to detect vegetation changes between years could produce more comparable and meaningful results.

In this case, the area queried is over 10,000km^2 and visible in the red box shown in (a) of Figure 4.14. This area covers what is known as Afrin, Syria. In that same figure, a substantial difference in the number of available acquisitions can be seen between years 2016 and 2017, before Sentinel-2B was launched and operational, and 2018, where almost twice as many acquisitions are available for parts of the queried area for the same duration of time (i.e. around 6 weeks). A similar pattern of acquisition distribution can also be seen in Figure 4.15, where it is also clear that 2017 has a few more acquisitions than 2016 for the same time-frame, but also on fewer cloud-like pixels in the acquisitions available. The more mountainous region in the West also is visible, especially in in 2018, due to lower numbers of "clean" pixels likely attributable to higher average cloud cover, even though more total acquisitions are available. As mentioned in Section 5.1.1, the city of Aleppo is also visible in the lower right corner, especially once more acquisitions are incorporated, such as in 2018. This is similarly visible in Figure 4.5 over the entire 3 years of data, but is interesting to see the pattern emerge even with, at most, 17 acquisitions in 2018.

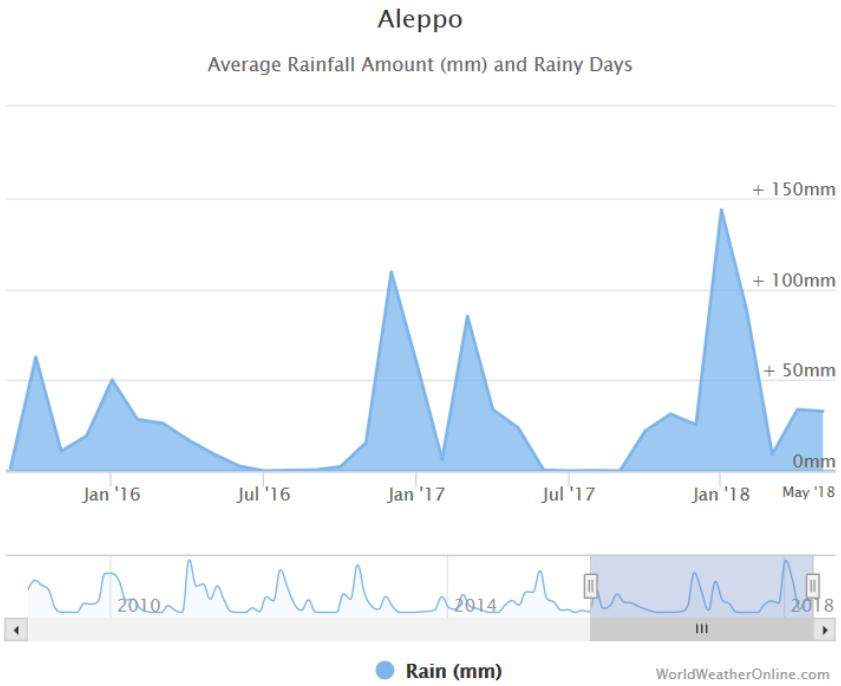

Figure 5.2: A rough idea of precipitation from September 2015 until May 2018 as recorded at the international airport in Aleppo, Syria. Data and visualisation provided by WorldWeatherOnline.com

Moving on to vegetation, in this case, it is important to take into account rainfall at this time of year, and the preceding months. Figure 5.2 shows average monthly rainfall in Aleppo for the years 2016-2018. It is clear that the end of 2015 and beginning of 2016 received relatively less average rainfall than 2017 or 2018. There is a visible difference in the total number of vegetation-like pixels shown in Figure 4.16, not only because of the lower number of acquisitions in 2016, but likely also because there was less average rainfall.

Once the normalised occurrence of observed vegetation-like pixels over time is calculated (*see* Figure 4.17), this difference becomes less evident in the more persistently vegetated areas in the more mountainous regions, which likely receive more precipitation on average than Aleppo. Areas that are likely agricultural fields, however, are almost non-existent in the output of 2016. The lower number of acquisitions available in 2016 and 2017 compared to 2018 must be taken into account. Some areas in 2016 have only 6 moments in time to even work with (i.e. two vegetation-like observations automatically can mean at least 33%, and if clouds were observed once, 40%). Also, cloud cover is important to consider, especially for shorter aggregated output. Figure 5.3 shows areas in white where there are no valid data to work with from 1 May to 15 June in 2017, despite having 5 acquisitions covering that area.

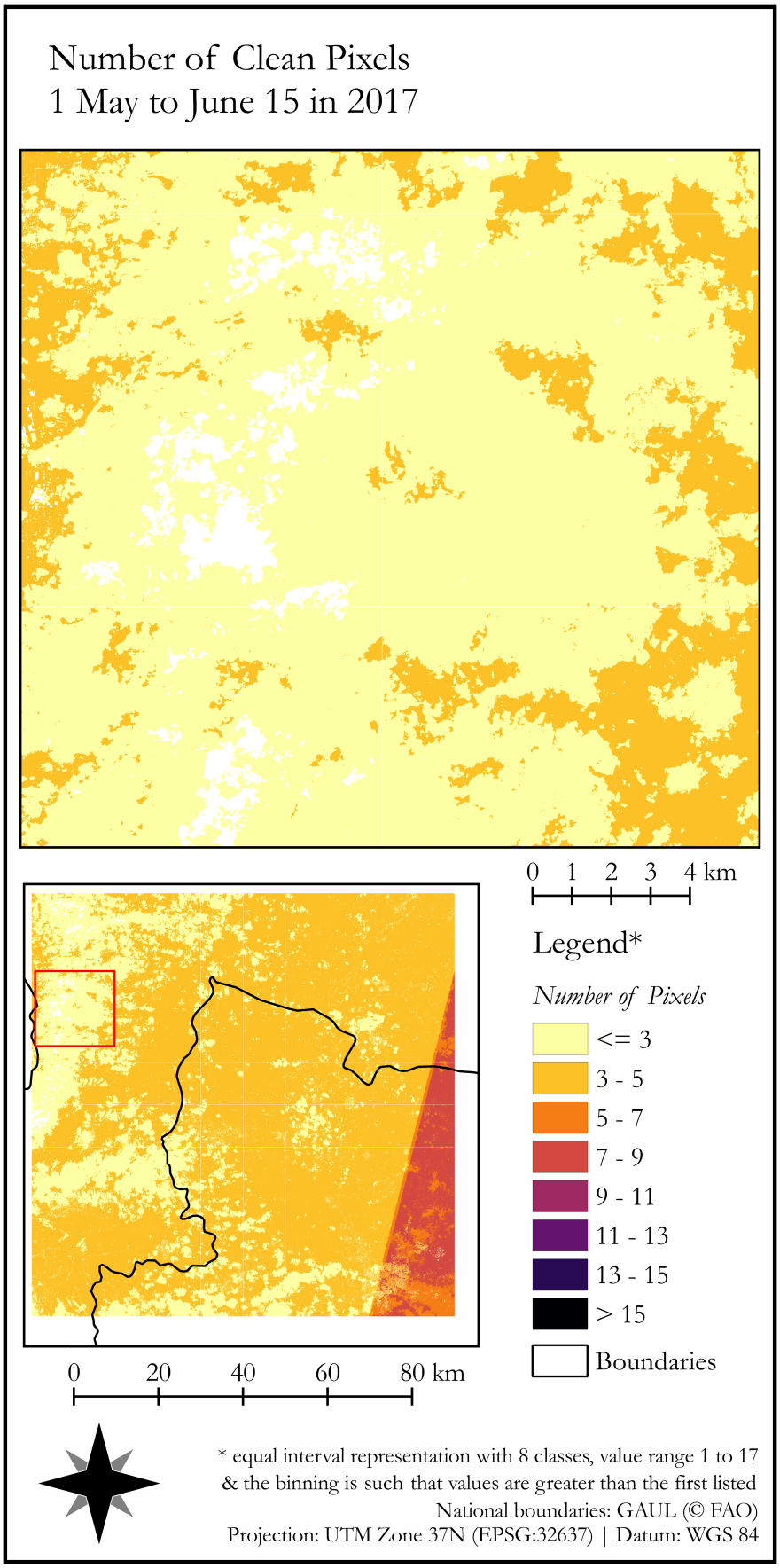

Figure 5.3: A closer look at the number of "clean" pixels from 1 May to 15 June 2017, showing areas with no valid data in white due to cloud cover, despite having 5 acquisitions in the time frame for this specific area.

Figure 4.18 and Figure 4.19 look at differences in the normalised occurrence of observed vegetation-like semi-concepts of the same area, between 2017 and 2016, and 2018 and 2017, respectively. In this case, it is, again, important to know that, while 2016 experienced lower average rainfall during and preceding the queried temporal-extent in comparison to 2017 and 2018, there are also fewer "clean" observations for this area. For almost all pixels in this smaller spatial extent, there are only 2 "clean" pixels in the stack, to be exact. It would be useful to know if the "clean" pixels for this area occurred from the same 2 scenes, and whether or not they were acquired earlier or later in the temporal extent from 1 May to 15 June 2015. The same type of difference was calculated for another area in Figures 4.20 and 4.21, also between 2017 and 2016, and 2018 and 2017, respectively.

In retrospect, it would makes sense not to have difference values ± 0.111 be white, but either only values of 0, or to mask results based on the combined vegetation-like extents detected in the two years being compared. Without such masking, only relatively larger changes in normalised occurrence can be discerned, and a difference between non-vegetated areas consistent in both temporal-extents versus vegetated areas that experienced no relative change cannot be made.

A question also arises as to whether or not the aggregated normalised occurrence information is comparable between two different temporal extents with different base data. This is perhaps less trivial when comparing 2018 with 2017, whereby each vegetation-like pixel in the relatively shorter stack in 2017 has a relatively higher influence on the output than an observation in 2018, which generally has double the acquisitions. It might make sense to figure out guidelines or metrics to assess whether the number of valid acquisitions and "clean" pixels over a temporal-extent, but also taking their variability and distribution through time, as a way to better judge comparability from a statistical standpoint, or even somehow assess statistical significance of detected changes.

5.1.3 *Potential Connection to SDGs*

Aggregated time-series output was generated in the hopes of moving towards an automated workflow to supply information in the scope of the SDGs mentioned in Section 2.4. Surface water bodies and various types of vegetated land cover (e.g. agricultural fields, forests) cannot be explicitly identified or classified with this output, but seem to be able to be made visible in some way using this generic implementation based on the initial exploratory proof-of-concept output (*see* Chapter 4). It seems plausible, that similar automatically generated output from this generic semantic data cube could be integrated with additional data to

offer spatially-explicit evidence to eventually support the SDGs, whether at 10m spatial resolution or based on larger, meaningful objects.

INDICATOR 2.4.1 proportion of agricultural area under productive and sustainable agriculture

While agricultural areas cannot directly be discerned from output, in arid climates, such as the eastern part of the study area, vegetated agricultural fields can be visually detected over a defined growing season. Looking at the dynamics of the greenness of these fields, once identified, could offer more information about the level or frequency of productivity over time. Changes relative to these identified agricultural field objects (*see* Figure 4.19 for a preliminary idea of what this could look like on a per-pixel basis), paired with information about drought, temperature, rainfall, or other factors, could be monitored and provide spatially-explicit information relevant to the food security of a region. EO-derived information in this context could provide more insight into where changes to agricultural fields are occurring in order to inform campaigns, such as building or repairing irrigation infrastructure, supplying seeds, etc.

INDICATOR 6.6.1 change in the extent of water-related ecosystems over time

Sentinel-2 data is limited to detecting objects or surfaces visible from space, which makes certain types of water-related ecosystems difficult to detect. Smaller waterways, for example, may not be visible, or surface water located under vegetation, not to mention ground-water-fed ecosystems. However, the extent of relatively permanent and seasonal surface water can be made visible in an ad-hoc way. Perhaps the annual average extent of a lake has not changed in the past 4 years, but the dynamics of when it grows and shrink have and are important to note. Temporally-specific changes in surface water extent could be queried using a generic implementation like this (*see* (c) in Figure 4.12 or Figure 4.9).

An existing set of proof-of-concept implementations for calculating this indicator using EO data already exists from the EO4SDG initiative run by GEO (Group on Earth Observations, UN Environment, & NASA, 2017). The proof-of-concept implementation used in producing Figure 4.9 is much more generic, but must be tested for transferability to other geographic locations and temporal extents. This is not because the semantic information is not transferable, rather due to potential limitations that arise when applied to different places on Earth with different data availability, quality and characteristics (e.g. cloud cover). Sentinel-2 images have a higher temporal frequency than the data being used by, which may be necessary for certain applications in comparison to extraction with Landsat-8 imagery used by EO4SDG.

INDICATOR 15.1.1 forest area as a proportion of total land area

Looking at figures such as 4.12 and 4.13 in comparison to Figure 4.10, it is clear that forested areas in this study area not only have a higher relative normalised occurrence of vegetation-like semi-concepts, but also have a different physical structure. The increased normalised occurrence paired with information of the distribution and variation of detected semi-concepts over time, as well as some sort of shape information (e.g. texture, size, compactness) could be used to delineate certain kinds of forests from other kinds of vegetated land cover.

This is the only indicator of the three listed here that is rated at "tier 1", meaning that established and acceptable methodology exists and data are already widely available. There ought to be enough spatially-explicit information and geospatial datasets that exist to test different object-based methods used on output from a system that can then be checked for agreement.

5.1.4 *Challenges for Interpretation*

Various challenges in interpreting the output were already mentioned, but here is a brief overview:

- variability of acquisitions, "clean" pixels and total observations meeting query results make interpretation complex
- lack of methods and metrics to take variability through time into account
- calculating differences between seasons or years based on aggregated output may be misleading (i.e. are they even comparable?)
- knowledge of the periodicity of per-pixel semi-concepts that change, but also their stability over longer time-frames would improve interpretation (i.e. lack of changing from observation to observation over time)
- finding ways to characterise not only variability, but patterns of differences through time, or ways take it them account during interpretation
- ability to give information about seasonality and dynamics (e.g. type of crop, harvest or mowing cycles, . . .)
- more complex queries are probably more difficult to interpret (e.g. automated post-classification change detection instead of just detection of occurrence)
- grouping of pixels containing shadows and water into water-like semi-concepts

- other kinds of semi-concepts may present different challenges or require different kinds of information towards building meaningful output
- lack of data for the study area that is up-to-date enough to serve testing for agreement or validity
- these kind of queries need to be interpreted in the seasonality in which they occur (e.g. a 3 year aggregation of data fir a location that experiences a monsoon season can only be interpreted knowing that the result has temporal gaps)

5.2 DISCUSSION OF DATA

The only data used in this implementation were Sentinel-2 L1C data, and information layers generated from them. For detecting objects larger than 8 pixels (e.g 800m^2) the 10m resolution is likely sufficient. It could be that output might not be able to be validated in big data domains, but that means that there is an even stronger need for proper validation of methods and source data.

5.2.1 *ESA Cloud-Masks*

Scene related quality information is offered with Sentinel-2 products, including an automatically generated cloud cover mask. This mask is used to calculate the average cloud cover for the scene. The algorithms used to calculate this mask are proprietary and not freely or openly available.

There was a consideration at some point to include a rasterised version of the cloud masks that are provided in the implementation used here, but the quality seemed not to be reliable and fairly inconsistent. There are many different kinds of clouds and not all of them are detectable by the algorithm being used, probably because ESA wants to reduce false positives (i.e. incorrectly labelling a pixel as being clouded) and only provides results that are very certain to be true. Thin clouds, for example, are often not masked as clouds, but provide plenty of problems for analysis.

Further examination of the existing cloud masks must be conducted in order to completely rule out incorporation. Even if the masks do not include all of the cloud cover, if the pixels that are masked are, in fact, quite often correct, then they could be compared with the cloud-like semi-concepts to improve confidence. They could also be incorporated in some way to establish a layer of likelihood that the pixels labelled as being cloud-like are actually clouds. Without knowing how the ESA algorithm works, this makes little sense.

5.2.2 *ESA Level-2A*

Images must to be calibrated from digital numbers to at least ToA reflectances in order to be comparable to each other. SIAM™ requires data to be calibrated to at least ToA reflectance in order to generate meaningful output. Sentinel-2 data are offered as ESA L1C, which means they are calibrated to ToA. While still considered open data, the pre-processing algorithms applied to the EO data provided by the Copernicus programme are not open, but proprietary. This has many implications for understanding the basis of scientific analysis conducted with them, because these pre-processing steps are not transparent.

Including a robust, reliable and automated method for calibrating the images to bottom-of-atmosphere (BoA) or SURF might improve pre-classification results. This sort of calibration ultimately requires data about the atmosphere at the moment of observation, which current methods can only approximate. This sometimes lead to undesirable or misleading output that is also not necessarily comparable between different images or different geographic or temporal locations.

At the time of writing, ESA intends to start offering world-wide level-2A (L2A) products, which include data calibrated to BoA and a scene classification map (SCM). These products are also generated using proprietary algorithms, which makes it difficult to decompose what processing has occurred and to ascertain an idea of quality, reliability and different kinds of variation (e.g. does it work well over snow?; does it work well in different climate zones? what is the variation of quality or reliability through space and time within the archive?). A further step to consider would be whether or not to include the L2A data and SCM in a data cube, as well as to test the ability of SIAM™ to handle the data supposedly calibrated to BoA.

5.2.3 *Challenges*

An overview of some of the challenges faced in using the currently available processed version of Sentinel-2 data processed to L1C include:

- long path names
- swaths vs. granules
- projections if a similar implementation is transferred to other areas
- data redundancy (keeping own copy) and size
- Sentinel-2 geometric alignment inconsistencies

The issue of long path names was avoided here by using Linux instead of a Windows OS. It is important to note that while the file names

of Sentinel-2 products have since been shortened, this could still pose issues to anyone wanting to use the data on Windows.

Swaths and the granule tiling-scheme that Sentinel-2 products are provided in sometimes conflict with each other. One example is granule 37SCA covering Syria and Turkey in this implementation. Any data at any point in time acquired for this granule does not cover it completely due to the swath overlap. This has different implications for calculating granule-based statistics, since two small triangles of the granule are always comprised of completely different sets of observations, whereas the middle is covered by both swaths. Perhaps this has little effect on average statistical metrics over longer time-spans, but this may misrepresent the data that exists in the archive in certain cases, especially so long as archive access is based on granule-wide metadata and not the actual content of images. When searching for scenes to use in analysis over smaller areas, pay attention to the relative orbit number to make sure that you are getting data acquired by the swath that covers your area of interest.

Luckily all data used in this implementation was already provided in a common projection (UTM zone 37N, EPSG:32637). However, this will not be the case for most other big Earth data applications, especially meant for areas larger than 30,000km^2. It definitely makes sense to start thinking of solutions to handle multiple projections. Perhaps this can be accommodated by using methods to re-project data on the fly depending on the spatial extent and location of a query, or to re-project data before loading into a data cube implementation. The pros and cons of different methods need to be assessed, especially since re-projection requires resampling, which changes the values of the data, and could impact analysis, results and the reproducibility of results if not properly documented.

The current implementation has a lot of data redundancy. This could be discussed further in discussing methods, but the idea of keeping the all of the original Sentinel-2 data used in the implementation has already proven to be important. The processing baselines for Sentinel-2 L1C data change and have changed, and sometimes includes re-processing of already archived data, replacing data processed with an older processing baseline. For example, a recent announcement was made that the entire archive would be re-processed to meet a need for improved geometric corrections (European Space Agency, 2018b), which is very relevant for a pixel-based implementation such as the one presented here. Keeping the original data used in an implementation such as this could be important if differences between processing baselines could impact whatever analysis.

5.3 DISCUSSION OF METHODS

Any number of software and technologies could have been used in this proof-of-concept implementation. A bit more information on the justification and challenges surrounding specific tools and methods are discussed.

5.3.1 *Data Acquisition*

Since the automated workflow was implemented in January 2018, the download hub has seemed to be fairly stable. The Hub's Python API has also made data access fairly uncomplicated to automate. However, downloading data is the part of the automated workflow that takes the longest amount of time, and is the least reliable due to external factors (e.g. Internet connection, Hub status).

It might make sense to avoid downloading the original Sentinel-2 data entirely by processing the data with SIAM™ where they are located, and only saving the generated information layers. However, this sort of set-up creates a bit of tension with the reproducibility of results. The Sentinel-2 data source is a huge and growing body of data, which is likely to not always be available everywhere for the entire duration of acquisition, but distributed among various mirrors. The Sentinel-2 archive is also about to be completely re-processed, which will replace the existing products that Sentinel-2-based analysis conducted up to now has relied on. These results will not be reproducible unless the researchers keep a copy of the data used for their analysis.

5.3.2 *Re-formatting*

Data redundancy trade-off for reproducibility of results.

This step in the automated workflow essentially produces redundant data that takes up more space on the server. These data are not used once the SIAM™ information layers have been generated. However, if changes occur to any of the Python packages used to generate these re-formatted layers, keeping them could be used to identify differences, and whether or not they produce comparable output for the task at hand.

Methods also exist for improved resampling of lower resolution bands, that work a bit like pan sharpening based on characteristics of the available 10m bands. Whether or not this would have an affect on resulting information layers, in what way and how much would be something to explore.

No-Data Mask Generation

It has been assumed that pixels with a measured value of 0 in any of the six re-formatted Sentinel-2 bands be excluded (i.e. masked), but this may be a fault assumption more often than I am aware. However, information on pixels not containing valid data for each band within a scene is not yet supplied with Sentinel-2 products. Even if the image footprint is supplied in the metadata, each band's measurements at the edge of an orbit swath are most often not identical. Pixels with a measured value of 0 in any of the six bands are thus excluded from semantic enrichment.

I found this assumption useful as a relatively simple way to drastically reduce faulty semi-concept assignment to pixels lacking valid data in any of the six bands within an image at a given time, as happens mostly at the edge of an orbit swath. However, this assumption may also occasionally exclude meaningful information (i.e. when a valid measurement has a value of 0), as can be seen in Figure 5.4. A few pixels, almost exclusively over what is the Mediterranean Sea, are excluded at least once, likely due to a measured value of 0 in one of the six relevant bands.

5.3.3 *Pre-classification*

A huge array of classification algorithms and methods exist in the realm of EO. SIAM™ was chosen because it produces reproducible and repeatable output in an automated way without any user-parametrisation, uses multi-dimensional decision rules, and can be transferred to multiple optical sensors, including Landsat. A future project would be to incorporate Landsat data also processed with SIAM™ as another produce in the ODC to be able to ask similar semantic queries. This would lengthen the temporal extent considerably.

The semi-concept granularites generated by SIAM™ group pixels based on what they look like, not necessarily on what they contain, and ought to be used as building blocks in further methods. Many other classifiers require samples and are not transferable to different sensors, much less produce output that are transferable to different images. Exploration of other solutions is a point of further research, but SIAM™ was chosen, despite its closed-source license, because there seems to be a lot of potential in what can be built on top of the information layers it generates.

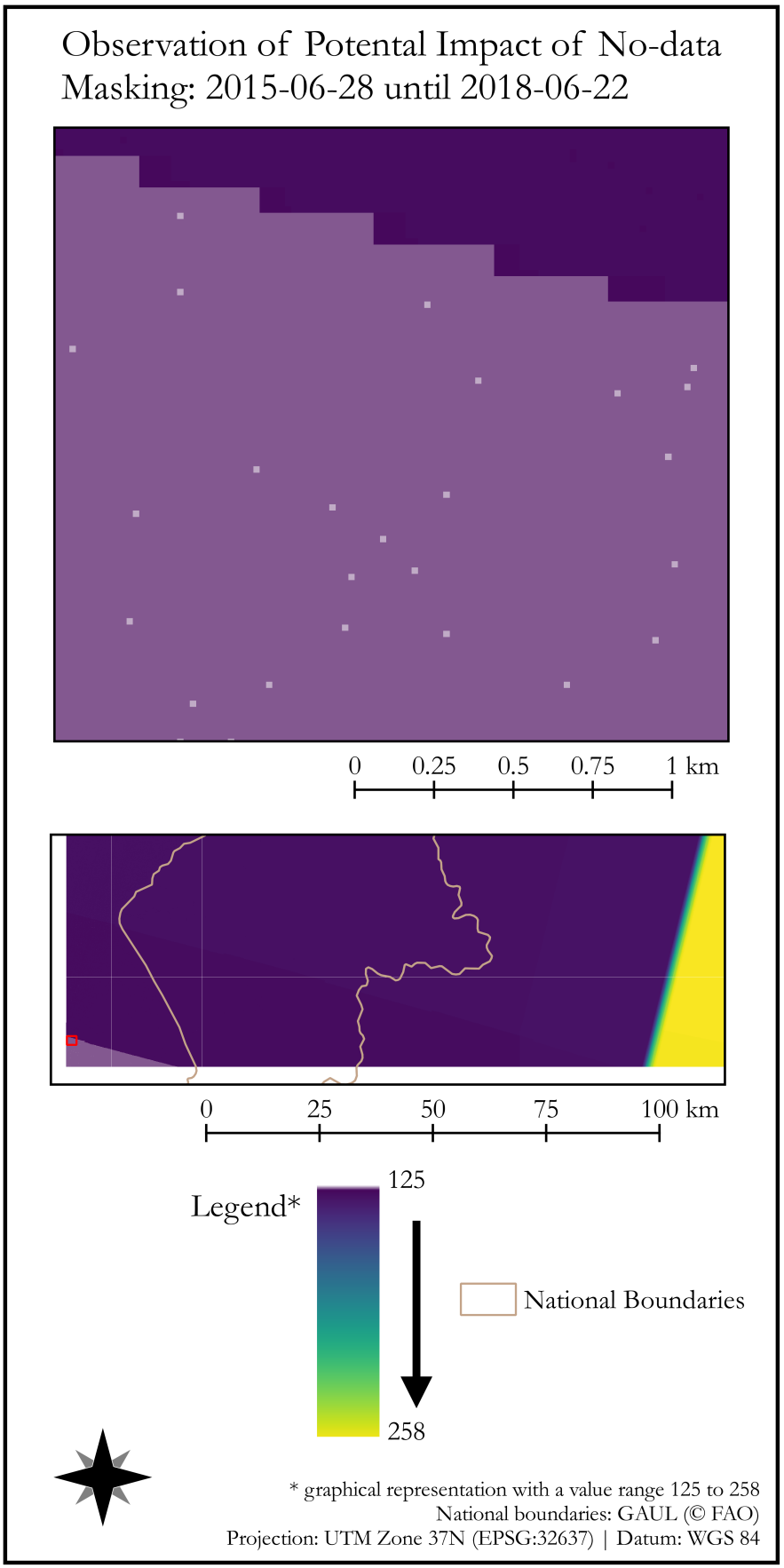

Figure 5.4: A closer look at the number of total pixels containing measured values not equal to 0 in the original Sentinel-2 data based on nearly 3 years of data. Values 0-125 are white. One can see a bit of a salt-and-pepper effect in the upper inset map, assumed to be caused by this simplified no-data masking.

Exclusion of Thermal Decision-rules

SIAM™ ultimately requires a thermal band for its decision rules. The current implementation basically creates a fake thermal band with a constant value of 110 to avoid thermal decision rules. This may, however, be part of the reason that highly reflective clouds are assigned to a bare-soil semi-concept, because there is a lack of relatively colder thermal-related values. Further exploration is required, but this likely has a low impact on aggregated dynamics analysis, especially in environments with low average cloud cover like the study area focused on here in Syria and Turkey.

5.3.4 *Data Cube*

Spread the cubes, any cubes!

Rasdaman was almost chosen for this implementation, but interest in a completely open-source movement to enable data cube implementations for everyone was interesting. Seeing as the ODC software is already being used in a few larger-scale implementations (e.g. AGDC, SDC, CDCol in Columbia), the scalability was not in question for any intended future use, even if it may pose other challenges. Two points deserve a bit more detailed discussion, and that includes the tiling scheme used in this implementation as well as the datacube API class called *GridWorkflow*, which allows distributed processing of arrays that exceed in-memory capacity.

Ingestion Tiling Scheme

At the moment, this implementation uses a tiling scheme of 10km by 10km by one moment in time. The idea behind this was to keep tiles fairly small, in the case that processing had to occur in-memory. A tiling scheme of 10km by 10km by multiple observations in time may make much more sense. Further research into the implications of different partitioning schemes on ODC performance and analysis is necessary. These sort of explorations also ought to test performance of different kinds of queries that prioritise different dimensions or different numbers of variables at once (e.g 33 semi-concepts, 96 semi-concepts, vegetation mask). However, this kind of assessment is definitely beyond the scope of this thesis.

GridWorkflow

One challenge is that processing using the Python API occurs predominantly using in-memory data. This complicates implementation on the current hardware as it requires a user to load the complete dataset

prior to analysis and is a limitation for smaller institutions. The *GridWorkflow* class offers tools to work on analysis in bite-sized chunks, but proves challenges for any analysis where the context that pixels are situated in needs to be taken into consideration (e.g. neighbours). This is a data dependency issue that needs further exploration within data cube technologies including the ODC.

An artefact from using *GridWorkflow* was noticed when loading individual chunked tiles, as well as the tiles mosaiced using GDAL after processing. This artefact is 1 pixel (i.e. 10m) wide along the edges of the tiling scheme. The *dask*[1] chunks for processing were based on the tiling scheme. After speaking with someone at Commonwealth Scientific and Industrial Research Organisation (CSIRO) who knows the code base a bit better, it could stem from a registration error between different language libraries used in implementing the *GridWorkflow* class (i.e. a difference between indexing starting at 0 or 1). A planned update to the newest release of the ODC software may solve this issue.

5.3.5 *Informal Benchmarks*

This was a proof-of-concept implementation of a semantic data cube and was never intended to optimise or even evaluate the speed or efficiency of processes or workflows. However, the volume of data that will accumulate even over one more year of operation and the limitations on in-memory processing seem to be the largest hurdles in terms of future implementation of a similar system. Just to summarise, downloading data all the way up to ingestion of the generated information layers into an implementation of the ODC takes the following tasks around so much time, given a bit of wiggle room:

- download: 10 minutes
- prep for SIAM™: 1 minute
- SIAM™ processing: 5 minutes
- indexing: 10 seconds
- ingestion: 1 minute

Conservatively estimated at 17 minutes per granule-sized scene.

This means that even with relatively slow downloads (e.g. often it only takes a few minutes), one granule can be processed and incorporated in around 17 minutes. Downloads can occur simultaneously, depending on a multitude of variables, and SIAM™ processing can easily be distributed among different cores. Just as an exercise of curiosity, an estimate of how long it would take to rebuild this implementation will be estimated. These values are based on the current existence of around 600 Sentinel-2 granules. There is also perhaps unreasonable assumptions of linear

1 *Dask* is a Python package for scaling multi-dimensional data analysis using *Numpy* (another Python package) arrays, which is a bit more of what happens under the hood of the ODC.

processing, that granule-based Sentinel-2 products are all the same size, and that no interruptions in Internet connectivity, Hub service, etc. occur. Based on these assumptions, it would take approximately so long for each step to rebuild the implementation in a linear way, including downloads, since everything has already been automated:

- download: (600 granules * 10 minutes) = 6,000 minutes
- prep for SIAM™: (600 * 1 minute) = 600 minutes
- SIAM™ processing: (600 * 5 minutes) = 3,000 minutes
- indexing: (600 * 10 seconds) = 100 minutes
- ingestion: (600 * 1 minute) = 600 minutes

Complete recreation estimated currently at roughly 3.16 days

This results in 10,300 minutes, or 7.153 days of processing, assuming nothing runs in parallel. However, SIAM™, for example, was distributed among 12 CPUs when creating this implementation, and downloads occurred using at least 2 Copernicus Open Access Hub accounts. This results in the following fairly conservative estimation of 4,550 minutes (i.e. 75.83 hours, or 3.16 days).

5.3.6 *Jupyter notebooks*

More of these!

All I can say is that these things are great, especially for sharing and documenting code in a learning environment.

5.4 REVISITING RESEARCH QUESTIONS

This thesis aimed towards answering some specific lines of inquiry. Very brief replies are shared here, or relevant sections in the rest of this thesis are referenced that already offer some thoughts towards potential answers or solutions.

5.4.1 *Context*

What are some ways that EO contributes or can be envisioned as contributing towards spatially-explicit evidence for SDGs indicators?

Free and open EO data are one of the few global data sources independent of borders. *See* Section 2.2.1 and Section 5.1.3

What are some current examples of free and open EO-based indicators or evidence for indicators?

There are many existing examples. See Section 2.3.1 for a very small selection.

5.4.2 *Implementation*

Is it possible to automatically download Sentinel-2 data and automatically enrich it semantically?

Yes. Improving the level of the automatic semantic enrichment, what it can be used for and how is an on-going field of research blending domains of EO, computer science, cognition, image understanding and more.

Can all Sentinel-2 data available for a region be automatically incorporated in a semantically enriched data cube?

Yes, and this implementation can even be reasonably considered to scale-up to a few times larger in data volume using a similar implementation as here. See Section 5.3.5 for a very rough idea of how long this could take.

Is the available hardware sufficient for such an automated workflow and queries within a reasonable execution time?

It depends on the query. For the purposes of this thesis, most exploratory queries could be conducted in 30 minutes or less. Even processing all of the available ingested spatio-temporal extent only took around 4 hours for each relatively uncomplicated query. More complex queries (i.e. more array functions) years of data over large spatial extents ought to consider a hardware upgrade.

Is the ODC software conducive to handling relatively simple semantic queries based on semi-concepts?

Yes. The question is now whether simple semantic queries based on semi-concepts serve the applications they are intended to. See Chapter 4 for example query outputs.

Are semi-concepts sufficient for ad-hoc, semi-automated monitoring of vegetation and water dynamics over time?

Considering they are ad-hoc not definitive, most certainly. However, before being used as the basis for decisions or actions that could have longer reaching consequences than curiosity, the utility, validity or

agreement with existing data, limitations and challenges need to be assessed for different applications. See Section 5.1.

Can information generated from querying vegetation-like or water-like semi-concepts utilising time be used in a meaningful way in the context of existing indicators?

This is still unclear and requires further research, but the exploratory results are a promising start. See Section 5.1.3.

How does this information characterise changes to water and vegetation cover for the temporal extent of the implemented data cube?

It can characterise longer-term spatial extents (e.g. occurrence over multiple years, seasons, etc.) or shorter term monitoring (e.g. aggregations of a few weeks to changes detected using the newest semantically enriched information layer). Interpreting such output requires existing and perhaps new statistical methods, especially when it comes to characterising the distribution, variation, variability and uncertainty of observations through time, accounting for space. Metrics and methods exist to assess those aspects while accounting for time, but also not necessarily in the context of big data. See Section 2.1.2 and Section 5.1.

What sort of information is needed to better assess the quality and confidence of aggregated indicator-like results?

It is clear that having a better understanding of the larger archive data are a part of (i.e. global information), and also the data available for the study area (i.e. local information, if you will), are necessary to inform analysis and interpretation. If a selection of images is chosen, these two collections of information are also incredibly relevant, since the data being used is chosen for a specific set of criteria that could bias or influence analysis in some way. See Section 5.1 for some qualified efforts to interpret aggregated, indicator-like output.

5.4.3 *Results*

Can differences in vegetation dynamics in agricultural areas be detected between Syria and Turkey using this implementation?

This was not a topic explicitly addressed in this thesis. However, without making a claim as to the cause, a clear difference over an aggregated 3 years of Sentinel-2 data, the border between the two countries is clearly

visible and differences in vegetation, most probably irrigated agriculture, are also visible (Figures 4.10 and 4.11).

Part III

LOOKING FORWARD

6

OUTLOOK

In the course of working on this thesis, many more questions were generated than could possibly be considered, explored or answered. If nothing else, the largest gift of having worked on this topic, beyond improving scripting skills, was waking a curiosity and interest about the distribution, variation, variability and uncertainty in multi-temporal EO analysis and archives that cover relatively large spatial extents. In the era of big Earth data, it is increasingly important to have meaningful, comprehensive and standardised methods to characterise uneven distribution, uncertainty and variability and variation in data quality and spatio-temporal coverage for different EO data sources and archives. Certain types of analysis and algorithms are not transferable to different geographic areas, sensors, moments in time, seasons, etc. This curiosity will likely shape any future work that I do, even if it is not the main focus.

6.1 DATA AND ANALYSIS

The trend towards more free and open data has been steadily getting stronger. Methods exist for intercalibrating Sentinel-2 and Landsat data (X. Li et al., 2017), which may also expand possibilities for analysis over longer temporal extents and assessing relatively slower processes and trajectories. The challenges for using two different data sources with similar but not identical sensors will produce different levels of uncertainty, and different distributions of acquisitions through time that analysis is based on. These differences are only masked by proprietary algorithms that obscure what kinds of pre-processing have been conducted, and how that may differ through space and time based on the availability of auxiliary data or changes to processing baselines over time.

It will be interesting to see if a standardised baseline definition of ARD is developed in the months or years to come. The concept of ARD (*see* Section 2.1.3) is relevant in the context of this thesis, since a data cube is provided allowing access to data with user-defined extents and generic semantic enrichment suitable for multiple kinds of analysis using semantic queries. Depending on how ARD is defined, the created ODC implementation could be considered as analysis-ready. Semi-concepts provide an automated semi-semantic layer that under current circum-

stances, could be considered as moving towards semantically enriched ARD.

We are likely to see finer spatial resolutions and more frequent observations, even if the data is not free and open (e.g. cubesats (McCabe et al., 2017)), which will open completely new possibilities. This influx of even more frequent, high- to very-high-resolution data will not solve most of the problems faced by EO researchers, though. That challenge is still largely framed in turning data into meaningful information in an automated way so that this data deluge can be utilised rather than just stored somewhere.

Object-based image analysis may be one set of methods to turn higher-resolution images into meaningful information when paired with semantics. As seen in the output generated in {ch:proof}, recognisable structures are very clearly visible, but cannot be utilised in a machine-readable way in the current implementation. Incorporating some form of iterative, reproducible segmentation to output, or coding characteristics of the neighbourhood around a pixel into an array as an additional information layer, would allow for contextual, spatial information to be utilised (e.g. neighborhood, shape, size, texture). This would could be a step further in the direction of automated detection of specific land cover classes, where shape and context are important for making the distinction between some natural and human-made features. For example, methods could be developed to automatically detect and discern a certain definition of forest-like objects (as seen in (b) and (c) in Figure 4.12) from what are clearly agricultural fields in Figure 4.10).

Incorporation of additional cloud and cloud-shadow algorithms (e.g. Fmask by Zhu & Woodcock (2012)) could be considered to include in an implementation like the one presented in this thesis. However, if Sentinel-2 is the main data source, a lack of thermal bands will continue to make this task a challenging one.

6.2 ENERGY CONSUMPTION

There has been very little consideration concerning the computing power required for storage, handling and analysis of big Earth data, and where the energy comes to support such work. Some algorithms are considerably more computationally expensive, and the energy efficiency of different data centres is something that also is not taken into account (Whitehead, Andrews, Shah, & Maidment, 2014). It is important to keep in mind that even if analysis is automated, happening in the cloud, on servers located somewhere else, that it is still consuming energy on infrastructure taking up physical space somewhere in the world. Clouds are very much of this world. If the hope is to generate information to support initiatives such as the SDGs, looking more closely into the

sustainability of various methods makes even more sense, perhaps even as a sort of metric to compare processes and algorithms that produce similar output.

Another possibility is to not get lost in an ocean of data, but rather to optimise data used for a given analysis by selecting only relevant Sentinel-2 images and not the whole stack (Kempeneers & Soille, 2017) – perhaps some information is lost, but perhaps the trade-off for reduced computation time and resources is worth it, especially if the methods for choosing data can be documented and reproduced.

6.3 REPRODUCIBLE EO-ANALYSIS

Some concepts of reproducibility were mentioned in Section 2.1.6, but how they specifically apply to EO, especially in the era of big Earth data, is an area of continued research. Hopefully we will see in increase in reproducible computing environments used to access and process data. This will hopefully include not just language-spcecific virutal environments, but reproducible, containerised environments including all dependencies, such as Docker containers.

Not everyone that works with EO data has the chops to configure reproducible computing environments, or work exclusively with open-source software, which sometimes requires users to be a bit more tech-savy by not hiding all functionality behind some sort of GUI. However, everyone can produce better documentation. Initiatives such as Zenodo and Open Science Framework (OSF) are providing free and open online frameworks and tools to better document the entire research process, including conception, collaboration, coding, documentation, communication of results and publications with long(er)-term storage with persistent links. I have never been one to keep a journal, but getting into the habit of documenting what you do thoroughly and consistently in a public forum is a good habit for anyone to develop, especially in the service of improving the reproducibility of the methods and results you produce.

6.4 DATA CUBES

I am convinced that data cubes are going to increase in use in years to come. Bringing new people to using them in data storage, access and analysis will lead to innovation in methods and more. Currently data cubes are defined as massive multi-dimensional gridded arrays. What could happen if we moved beyond the restricting confines of pixels or regular grids? What could objects look like in a data cube implementation?

6.5 PRIVACY AND MONITORING

What does privacy look like in the context of regular, constant, global EO data collection?

How has the legal and policy world been dealing with issues surrounding optical EO data with increasing spatial resolution and temporal frequency?

Is there a code of ethics or conduct that exists for the use of free and open EO data and the information derived from it? (Probably not, but would one make sense?)

6.6 LIVELIHOOD-RELATED CRISIS INDICATORS

The broad framing of the proof-of-concept examples was situated in the context of the SDGs. However, the scope of the SDGs may officially only reach until 2030, but the larger drive is longer-term. Due to the relatively high-frequency of acquisitions in comparison to what was available in the past, many new applications exist or are being envisioned for EO data cubes. One of these is related to situational awareness and providing more immediate information for livelihood-related crisis indicators, whether related to natural disasters or man-made conflicts. EO data is not only useful for monitoring longer-term international goals, but also for generating spatially-explicit evidence to better assessing the impact of events producing more rapid change, such as flooding, deforestation, wildfires, damage to irrigation infrastructure in the dry season, etc.

A struggle with this line of research is that "crisis" is a very subjective, yet sometimes very tangible concept or state of being. The concept of "crisis" is related to what is being affected, who or what it impacts and who or what the actors are. People can speak of an "immigration crisis", but what does that mean? And if it means closing borders and keep those people who have left their homes to wait at the border of another country in a potentially longer-term state of crisis, I am not sure it matters what it means.

Focusing on livelihood frames the very ambiguous concept of crisis in, perhaps, more tangible terms. The concept of "crisis" has a longer history connect to emergency management, but these are usually state-sponsored services with links to the military industrial complex. Free and open EO data have offered a potential source of information that was previously limited to state and military actors.

Free and open EO data are a reliable and objective global data source that can also serve humanitarian organisations and initiatives in moments of emergency or crisis, however defined. A handful of research with

the purpose of developing what are known as *crisis indicators* exists, utilising EO-based information. Connecting detectible changes in land cover or land use as being caused or related to a defined crisis requires a lot of inference, and additional data sources and knowledge for validation, which may not be available at the time of analysis due to limitations and restrictions to data collection, access, etc. associated with the identified crisis. In this sense, an implementation like the one presented in this thesis might be useful, but could also generate information that is misleading in moments where reliable information is absolutely necessary. Further research is necessary to assess what kinds of indicators and information are most useful to humanitarian organisations, including the level of reliability is acceptable and that can be offered.

6.7 FURTHER RESEARCH QUESTIONS

What are some ways or methods that output from such a data cube implementation produced over large areas using dense EO time-series can be validated, or better tested for agreement?

What are some existing measures of spatio-temporal confidence for EO analysis over dense time-series with big geospatial extents?

What are existing methods that take into account the spatio-temporal complexity and variability in EO analysis, but more importantly, interpretation, based on the Vs of big data?[1]

Is aggregated time-series output from different years or seasons comparable?

How could comparability be established, quantified, characterised or explored?

What spatio-temporal methods exist that could offer a way to test the statistical significance of changes or differences based on an aggregated time-series of scenes, like those generated for Afrin?

Is big data too big for traditional statistical measures to have meaning?

What semi-concepts are particularly semantically ambiguous even if they look alike?

What influence does this have on different semantic queries based on semi-concepts?

What information, rule-sets, or knowledge would be necessary to remove at least some of this semantic ambiguity?

1 For example, some methods do exist for looking at spatio-temporal accuracy (https://www.sciencedirect.com/science/article/pii/S0303243415000975#fig0015), but they still require producing a map for each time-step, which, in the case of big EO could mean hundreds, if not thousands of scenes – a task that ought not be done manually.

7

CONCLUSION

Extracting information from EO-time-series is a challenging task due to the volume, velocity and variety of EO images. The largest benefit of the semantic data cube implemented here is that it fully automates data acquisition, a certain level of semantic enrichment and access to data ready for analysis. The generic, application-independent semantic enrichment utilised enables queries and EO-based indicator extraction for many thematic tasks. Since the information layers (i.e. basis for semantic queries and analysis) continue to exist in the data cube and are stable concepts, given solid documentation on methods applied to data cube output, reproducible results and repeatable analysis ought to be possible. This could be particularly relevant for supporting global initiatives with information based on data collected independent of political borders and in a constant, unbiased way. Just because the data can be considered unbiased does not mean that information generated from it are unbiased, or that they will be used and understood in ways that are not misleading.

The contribution and innovation presented in this thesis is the automated set-up of a semantic data cube. This contrasts to most other existing implementations because it stores information and data together in an analysis-ready way, allowing ad-hoc multi-temporal, spatial and semantic queries and supporting reproducible results. However, it is not only the technical implementation, rather also an initial exploration of some of the challenges faced when working with dense time-series of EO data over larger spatial areas. These challenges are posed by qualities characteristic of big Earth data, more broadly, such as their volume, velocity, and variety.

Developing methods to extract reliable indicators from big EO sources is one way to leverage the potential these data have to offer. The ad-hoc queries that can be conducted using this implementation can hopefully offer meaningful information to support international initiatives. They can hopefully produce information about visible land cover changes through or utilising time instead of controlling for it. Exploratory outputs exemplarily showing dynamics of surface water and vegetation based on water- and vegetation-like pixels, based on semi-concepts, are presented and situated in the scope of the SDGs. Further research will include developing more reliable indicators tested for agreement with existing sources, as well as exploring methods and metrics to better

assess the distribution, variation, variability and uncertainty inherent in multi-temporal EO analysis and archives.

8

REFERENCES

Abdo, H. G. (2018). Impacts of war in Syria on vegetation dynamics and erosion risks in Safita area, Tartous, Syria. *Regional Environmental Change*, 1–13. https://doi.org/10.1007/s10113-018-1280-3

Anaconda Inc. (2017). Conda — Conda documentation. Retrieved June 30, 2018, from https://conda.io/docs/

Anderson, K., Ryan, B., Sonntag, W., Kavvada, A., & Friedl, L. (2017). Earth observation in service of the 2030 Agenda for Sustainable Development. *Geo-Spatial Information Science*, *20*(2), 77–96. https://doi.org/10.1080/10095020.2017.1333230

Ariza-Porras, C., Bravo, G., Villamizar, M., Moreno, A., Castro, H., Galindo, G., … Lozano, P. (2017). CDCol: A Geoscience Data Cube that Meets Colombian Needs. In *Advances in Computing* (pp. 87–99). Springer, Cham. https://doi.org/10.1007/978-3-319-66562-7_7

Arvor, D., Daher, F. R. G., Briand, D., Dufour, S., Rollet, A.-J., Simões, M., & Ferraz, R. P. D. (2018). Monitoring thirty years of small water reservoirs proliferation in the southern Brazilian Amazon with Landsat time series. *ISPRS Journal of Photogrammetry and Remote Sensing*. https://doi.org/10.1016/j.isprsjprs.2018.03.015

Augustin, H., Sudmanns, M., Tiede, D., & Baraldi, A. (2018). A Semantic Earth Observation Data Cube for Monitoring Environmental Changes during the Syrian Conflict. *GI_Forum 2018, Volume 1*, 214–227. https://doi.org/10.1553/giscience2018_01_s214

Baraldi, A. (2011). Satellite Image Automatic Mapper™ (SIAM™) - A Turnkey Software Executable for Automatic Near Real-Time Multi-Sensor Multi-Resolution Spectral Rule-Based Preliminary Classification of Spaceborne Multi-Spectral Images. *Recent Patents on Space Technology*, *1*(2), 81–106.

Baraldi, A. (2018, January 8). Satellite Image Automatic Mapper™ System and Products Description. Retrieved January 29, 2018, from http://siam.andreabaraldi.com/content/Documentation/SIAM_Report_BACRES_v1.18.pdf

Baraldi, A., & Boschetti, L. (2012a). Operational Automatic Remote Sensing Image Understanding Systems: Beyond Geographic Object-Based and Object-Oriented Image Analysis (GEOBIA/GEOOIA). Part 1: Introduction. *Remote Sensing*, *4*(9), 2694–2735. https://doi.org/10.3390/rs4092694

Baraldi, A., & Boschetti, L. (2012b). Operational Automatic Remote Sensing Image Understanding Systems: Beyond Geographic Object-Based and Object-Oriented Image Analysis (GEOBIA/GEOOIA). Part 2: Novel system Architecture, Information/Knowledge Representation, Algorithm Design and Implementation. *Remote Sensing*, *4*(9), 2768–2817. https://doi.org/10.3390/rs4092768

Baraldi, A., Durieux, L., Simonetti, D., Conchedda, G., Holecz, F., & Blonda, P. (2010). Automatic Spectral-Rule-Based Preliminary Classification of Radiometrically Calibrated SPOT-4/-5/IRS, AVHRR/MSG, AATSR, IKONOS/QuickBird/OrbView/GeoEye, and DMC/SPOT-1/-2 Imagery – Part I: System Design

and Implementation. *IEEE Transactions on Geoscience and Remote Sensing*, *48*(3), 1299–1325. https://doi.org/10.1109/TGRS.2009.2032457

Baumann, P. (2017). The Datacube Manifesto. Retrieved January 30, 2018, from http://www.earthserver.eu/tech/datacube-manifesto

Baumann, P., Dehmel, A., Furtado, P., Ritsch, R., & Widmann, N. (1998). The Multidimensional Database System RasDaMan. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data* (pp. 575–577). New York, NY, USA: ACM. https://doi.org/10.1145/276304.276386

Baumann, P., Mazzetti, P., Ungar, J., Barbera, R., Barboni, D., Beccati, A., … Wagner, S. (2016). Big Data Analytics for Earth Sciences: The EarthServer approach. *International Journal of Digital Earth*, *9*(1), 3–29. https://doi.org/10.1080/17538947.2014.1003106

Broich, M., Huete, A., Paget, M., Ma, X., Tulbure, M., Coupe, N. R., … Held, A. (2015). A spatially explicit land surface phenology data product for science, monitoring and natural resources management applications. *Environmental Modelling & Software*, *64*, 191–204. https://doi.org/10.1016/j.envsoft.2014.11.017

Camara, G., Queiroz, G., Vinhas, L., Ferreira, K., Cartaxo, R., Simoes, R., … Sanchez, A. (2017). The e-sensing Architecture for Big Earth Observation Data Analysis. In P. Soille & P. G. Marchetti (Eds.), *Proceedings of the 2017 conference on Big Data from Space (BiDS'17) 28th-30th November 2017.* (pp. 48–51). Toulouse (France). https://doi.org/10.2760/383579

CEOS. (2018). Data Cube Installation Guide. Retrieved July 2, 2018, from http://www.ceos-cube.org/docs/installation/index.html

CEOS-SEO. (2017). *Data_cube_notebooks: Jupyter Notebook examples for our Data Cube capable algorithms and functions.* Retrieved from https://github.com/ceos-seo/data_cube_notebooks (Original work published 2016)

Chen, M., Mao, S., & Liu, Y. (2014). Big Data: A Survey. *Mobile Networks and Applications*, *19*(2), 171–209. https://doi.org/10.1007/s11036-013-0489-0

Committee on Earth Observation Satellites. (2017). The CEOS open data cube initiative. Retrieved from https://docs.wixstatic.com/ugd/f9d4ea_1aea90c5bb7149c8a730890c0f791496.pdf

Committee on Earth Observation Satellites. (2018). *Satellite Earth Observations in Support of the Sustainable Development Goals* (Special 2018 ed.). Retrieved from http://eohandbook.com/sdg/files/CEOS_EOHB_2018_SDG.pdf

Cook, B. I., Anchukaitis, K. J., Touchan, R., Meko, D. M., & Cook, E. R. (2016). Spatiotemporal drought variability in the Mediterranean over the last 900 years. *Journal of Geophysical Research: Atmospheres*, *121*(5), 2060–2074. https://doi.org/10.1002/2015JD023929

Copernicus. (2017a). Research and User Support. Retrieved June 23, 2018, from https://rus-copernicus.eu/portal/

Copernicus. (2017b, May 23). The upcoming Copernicus Data and Information Access Services (DIAS). Retrieved June 23, 2018, from http://copernicus.eu/news/upcoming-copernicus-data-and-information-access-services-dias

Corbane, C., Kemper, T., Pesaresi, M., Freire, S., & Louvrier, C. (2016). *Monitoring the Syrian Humanitarian Crisis with the JRC's Global Human Settlement Layer and Night-Time Satellite Data.* https://doi.org/10.2788/297909

Corbane, C., Pesaresi, M., Politis, P., Syrris, V., Florczyk, A. J., Soille, P., … Kemper, T. (2017). Big earth data analytics on Sentinel-1 and Landsat imagery

in support to global human settlements mapping. *Big Earth Data*, *1*(1-2), 118–144. https://doi.org/10.1080/20964471.2017.1397899

Drusch, M., Del Bello, U., Carlier, S., Colin, O., Fernandez, V., Gascon, F., ... Bargellini, P. (2012). Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services. *Remote Sensing of Environment*, *120*, 25–36. https://doi.org/10.1016/j.rse.2011.11.026

Elvidge, C. D., Baugh, K., Zhizhin, M., Hsu, F. C., & Ghosh, T. (2017). VIIRS night-time lights. *International Journal of Remote Sensing*, *38*(21), 5860–5879. https://doi.org/10.1080/01431161.2017.1342050

European Commission Joint Research Centre. (2016). Global Surface Water Explorer. Retrieved June 29, 2018, from https://global-surface-water.appspot.com/

European Space Agency. (2015, July 24). Sentinel-2 User Handbook, issue 1, revision 2. Retrieved from https://sentinel.esa.int/documents/247904/685211/Sentinel-2_User_Handbook

European Space Agency. (2017, January 27). Sentinel High Level Operations Plan (HLOP): COPE-S1OP-EOPG-PL-15-0020. Retrieved from https://earth.esa.int/documents/247904/685154/Sentinel_High_Level_Operations_Plan

European Space Agency. (2018a, June 14). Publication of the 3rd batch of Sentinel-2A single tile repackaged products: 15 June 2018 [Copernicus Open Access Hub News]. Retrieved June 30, 2018, from https://scihub.copernicus.eu/news/News00338

European Space Agency. (2018b, May). Mission Status Report 131. Retrieved from https://sentinel.esa.int/documents/247904/3347201/Sentinel-2A-B-Mission-Status-Report-131-12-25-May-2018.pdf

Evans, B., Allen, C., Antony, J., Bastrakova, I., Gohar, K., Porter, D., ... Wyborn, L. (2015). NCI's High Performance Computing (HPC) and High Performance Data (HPD) Computing Platform for Environmental and Earth System Data Science. In (Vol. 17, p. 8273). Presented at the EGU General Assembly Conference Abstracts. Retrieved from http://adsabs.harvard.edu/abs/2015EGUGA..17.8273E

Fehling, M., Nelson, B. D., & Venkatapuram, S. (2013). Limitations of the Millennium Development Goals: A literature review. *Global Public Health*, *8*(10), 1109–1122. https://doi.org/10.1080/17441692.2013.845676

Fountain, H. (2018, January 19). Researchers Link Syrian Conflict to a Drought Made Worse by Climate Change. *The New York Times: Science*. Retrieved from https://www.nytimes.com/2015/03/03/science/earth/study-links-syria-conflict-to-drought-caused-by-climate-change.html

Frampton, W. J., Dash, J., Watmough, G., & Milton, E. J. (2013). Evaluating the capabilities of Sentinel-2 for quantitative estimation of biophysical variables in vegetation. *ISPRS Journal of Photogrammetry and Remote Sensing*, *82*, 83–92. https://doi.org/10.1016/j.isprsjprs.2013.04.007

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management*, *35*(2), 137–144. https://doi.org/10.1016/j.ijinfomgt.2014.10.007

Geoscience Australia, CSIRO, & NCI. (2017a). Data Access API — Open Data Cube 1.4.0+0.G7682003 documentation. Retrieved January 30, 2018, from https://datacube-core.readthedocs.io/en/stable/dev/api.html

Geoscience Australia, CSIRO, & NCI. (2017b). Datacube.Api.GridWorkflow — Open Data Cube 1.6rc2+2.G6aa9d16 documentation. Retrieved July 5, 2018, from

https://datacube-core.readthedocs.io/en/latest/dev/api/generate/datacube.api.GridWorkflow.html#datacube.api.GridWorkflow

Geoscience Australia, CSIRO, & NCI. (2017c). Indexing Data — Open Data Cube 1.4.0+0.G7682003 documentation. Retrieved January 29, 2018, from https://datacube-core.readthedocs.io/en/stable/ops/indexing.html

Geoscience Australia, CSIRO, & NCI. (2017d). Ingesting Data — Open Data Cube 1.4.0+0.G7682003 documentation. Retrieved January 29, 2018, from http://datacube-core.readthedocs.io/en/stable/ops/ingest.html

Geoscience Australia, CSIRO, & NCI. (2017e). Miniconda (recommended) — Open Data Cube 1.4.0+0.G7682003 documentation. Retrieved January 29, 2018, from http://datacube-core.readthedocs.io/en/stable/ops/conda.html

Giuliani, G., Chatenoux, B., Bono, A. D., Rodila, D., Richard, J.-P., Allenbach, K., … Peduzzi, P. (2017a). Building an Earth Observations Data Cube: Lessons learned from the Swiss Data Cube (SDC) on generating Analysis Ready Data (ARD). *Big Earth Data*, *0*(0), 1–18. https://doi.org/10.1080/20964471.2017.1398903

Giuliani, G., Dao, H., De Bono, A., Chatenoux, B., Allenbach, K., De Laborie, P., … Peduzzi, P. (2017b). Live Monitoring of Earth Surface (LiMES): A framework for monitoring environmental changes from Earth Observations. *Remote Sensing of Environment*. https://doi.org/10.1016/j.rse.2017.05.040

Goodman, S. N., Fanelli, D., & Ioannidis, J. P. A. (2016). What does research reproducibility mean? *Science Translational Medicine*, *8*(341), 341ps12–341ps12. https://doi.org/10.1126/scitranslmed.aaf5027

Gore, A. (1998). The digital earth: Understanding our planet in the 21st century. *Australian Surveyor*, *43*(2), 89–91. https://doi.org/10.1080/00050326.1998.10441850

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, *202*, 18–27. https://doi.org/10.1016/j.rse.2017.06.031

Group on Earth Observations. (2017, April). Earth observations and geospatial information: Supporting official statistics in monitoring and achieving the 2030 agenda. Retrieved from http://earthobservations.org/documents/publications/201704_geo_unggim_4pager.pdf

Group on Earth Observations, UN Environment, & NASA. (2017). Reporting on SDG Indicator 6.6.1 Using Satellite Earth Observations. Retrieved from http://eo4sdg.org/wp-content/uploads/2017/07/SDG6.6.1_brief_GEO_Week_2017-2.pdf

Guo, H. (2017). Big data drives the development of Earth science. *Big Earth Data*, *1*(1-2), 1–3. https://doi.org/10.1080/20964471.2017.1405925

Hagenlocher, M., Tiede, D., Wendt, L., & Lang, S. (2015). An Earth Observation-based Approach for the Assessment of the Environmental Impact of Refugee and IDP Camps. *GI_Forum*, *2015*, 420–423. https://doi.org/10.1553/giscience2015s420

Hagolle, O. (2018). *Automated download of Sentinel-2 L1C data from ESA (through wget)*. Retrieved from https://github.com/olivierhagolle/Sentinel-download (Original work published 2015)

Hansen, M. C., Potapov, P. V., Moore, R., Hancher, M., Turubanova, S. A., Tyukavina, A., … Townshend, J. R. G. (2013). High-Resolution Global Maps of 21st-

Century Forest Cover Change. *Science*, *342*(6160), 850–853. https://doi.org/10.1126/science.1244693

Hák, T., Janoušková, S., & Moldan, B. (2016). Sustainable Development Goals: A need for relevant indicators. *Ecological Indicators*, *60*, 565–573. https://doi.org/10.1016/j.ecolind.2015.08.003

Holmes, C. (2018a, April 25). Analysis Ready Data Defined. Retrieved June 23, 2018, from https://medium.com/planet-stories/analysis-ready-data-defined-5694f6f48815

Holmes, C. (2018b, April 26). Towards On-Demand Analysis Ready Data. Retrieved June 23, 2018, from https://medium.com/planet-stories/towards-on-demand-analysis-ready-data-f94d6eb226fc

Hossain, M. A., Dwivedi, Y. K., & Rana, N. P. (2016). State-of-the-art in open data research: Insights from existing literature and a research agenda. *Journal of Organizational Computing and Electronic Commerce*, *26*(1-2), 14–40. https://doi.org/10.1080/10919392.2015.1124007

INPE. (2016). E-sensing: Big earth observation data analytics for land use and land cover change information. Retrieved June 29, 2018, from http://www.esensing.org/

Janoušková, S., Hák, T., & Moldan, B. (2018). Global SDGs Assessments: Helping or Confusing Indicators? *Sustainability*, *10*(5), 1540. https://doi.org/10.3390/su10051540

Janssen, M., Charalabidis, Y., & Zuiderwijk, A. (2012). Benefits, Adoption Barriers and Myths of Open Data and Open Government. *Information Systems Management*, *29*(4), 258–268. https://doi.org/10.1080/10580530.2012.716740

Kelley, C. P., Mohtadi, S., Cane, M. A., Seager, R., & Kushnir, Y. (2015). Climate change in the Fertile Crescent and implications of the recent Syrian drought. *Proceedings of the National Academy of Sciences*, *112*(11), 3241–3246. https://doi.org/10.1073/pnas.1421533112

Kempeneers, P., & Soille, P. (2017). Optimizing Sentinel-2 image selection in a Big Data context. *Big Earth Data*, *1*(1-2), 145–158. https://doi.org/10.1080/20964471.2017.1407489

King, M. D., Platnick, S., Menzel, W. P., Ackerman, S. A., & Hubanks, P. A. (2013). Spatial and Temporal Distribution of Clouds Observed by MODIS Onboard the Terra and Aqua Satellites. *IEEE Transactions on Geoscience and Remote Sensing*, *51*(7), 3826–3852. https://doi.org/10.1109/TGRS.2012.2227333

Kohiyama, M., Hayashi, H., Maki, N., Higashida, M., Kroehl, H. W., Elvidge, C. D., & Hobson, V. R. (2004). Early damaged area estimation system using DMSP-OLS night-time imagery. *International Journal of Remote Sensing*, *25*(11), 2015–2036. https://doi.org/10.1080/01431160310001595033

Laney, D. (2001). *3-D data management: Controlling data volume, velocity and variety.* (Application delivery strategies by meta group inc.). Retrieved from https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

Langer, S., Tiede, D., & Lüthje, F. (2015). Long-term Monitoring of the Environmental Impact of a Refugee Camp Based on Landsat Time Series: The Example of Deforestation and Reforestation During the whole Lifespan of the Camp Lukole, Tanzania. *GI_Forum*, *2015*, 434–437. https://doi.org/10.1553/giscience2015s434

Lefebvre, A., Sannier, C., & Corpetti, T. (2016). Monitoring Urban Areas with Sentinel-2A Data: Application to the Update of the Copernicus High Resolution

Layer Imperviousness Degree. *Remote Sensing*, *8*(7), 606. https://doi.org/10.3390/rs8070606

Lewis, A., Oliver, S., Lymburner, L., Evans, B., Wyborn, L., Mueller, N., ... Wang, L.-W. (2017). The Australian Geoscience Data Cube — Foundations and lessons learned. *Remote Sensing of Environment*, *202*(Supplement C), 276–292. https://doi.org/10.1016/j.rse.2017.03.015

Li, X., Li, D., Xu, H., & Wu, C. (2017). Intercalibration between DMSP/OLS and VIIRS night-time light images to evaluate city light dynamics of Syria's major human settlement during Syrian Civil War. *International Journal of Remote Sensing*, *38*(21), 5934–5951. https://doi.org/10.1080/01431161.2017.1331476

Liu, H., Li, Q., Shi, T., Hu, S., Wu, G., & Zhou, Q. (2017). Application of Sentinel 2 MSI Images to Retrieve Suspended Particulate Matter Concentrations in Poyang Lake. *Remote Sensing*, *9*(7), 761. https://doi.org/10.3390/rs9070761

Maini, R., Clarke, L., Blanchard, K., & Murray, V. (2017). The Sendai Framework for Disaster Risk Reduction and Its Indicators—Where Does Health Fit in? *International Journal of Disaster Risk Science*, *8*(2), 150–155. https://doi.org/10.1007/s13753-017-0120-2

Marr, D. (1982). *Vision: A computational investigation into the human representation and processing of visual information*. San Francisco: W.H. Freeman.

McCabe, M. F., Rodell, M., Alsdorf, D. E., Miralles, D. G., Uijlenhoet, R., Wagner, W., ... Wood, E. F. (2017). The future of Earth observation in hydrology. *Hydrology and Earth System Sciences; Katlenburg-Lindau*, *21*(7), 3879–3914. https://doi.org/http://dx.doi.org/10.5194/hess-21-3879-2017

Molloy, J. C. (2011). The Open Knowledge Foundation: Open Data Means Better Science. *PLOS Biology*, *9*(12), e1001195. https://doi.org/10.1371/journal.pbio.1001195

Mubareka, S., & Ehrlich, D. (2010). Identifying and modelling environmental indicators for assessing population vulnerability to conflict using ground and satellite data. *Ecological Indicators*, *10*(2), 493–503. https://doi.org/10.1016/j.ecolind.2009.09.002

Mueller, N., Lewis, A., Roberts, D., Ring, S., Melrose, R., Sixsmith, J., ... Ip, A. (2016). Water observations from space: Mapping surface water from 25 years of Landsat imagery across Australia. *Remote Sensing of Environment*, *Complete*(174), 341–352. https://doi.org/10.1016/j.rse.2015.11.003

Nativi, S., Mazzetti, P., & Craglia, M. (2017). A view-based model of data-cube to support big earth data systems interoperability. *Big Earth Data*, *1*(1-2), 75–99. https://doi.org/10.1080/20964471.2017.1404232

Nativi, S., Mazzetti, P., Santoro, M., Papeschi, F., Craglia, M., & Ochiai, O. (2015). Big Data challenges in building the Global Earth Observation System of Systems.

Environmental Modelling & Software, *68*, 1–26. https://doi.org/10.1016/j.envsoft.2015.01.017

Nüst, D., Granell, C., Hofer, B., Konkol, M., Ostermann, F. O., Sileryte, R., & Cerutti, V. (2018). Reproducible research and GIScience: An evaluation using AGILE conference papers. https://doi.org/10.7287/peerj.preprints.26561v1

Open Data Cube. (2017). Opendatacube | CEOS. Retrieved January 30, 2018, from https://www.opendatacube.org/ceos

Open Knowledge Foundation. (2018). Open Definition 2.1 - Open Definition - Defining Open in Open Data, Open Content and Open Knowledge. Retrieved June 21, 2018, from https://opendefinition.org/od/2.1/en/

Peel, M. C., Finlayson, B. L., & McMahon, T. A. (2007). Updated world map of the Köppen-Geiger climate classification. *Hydrol. Earth Syst. Sci.*, *11*(5), 1633–1644. https://doi.org/10.5194/hess-11-1633-2007

Pekel, J.-F., Cottam, A., Gorelick, N., & Belward, A. S. (2016). High-resolution mapping of global surface water and its long-term changes. *Nature*, *540*(7633), 418. https://doi.org/10.1038/nature20584

Pesaresi, M., Ehrlich, D., Ferri, S., Florczyk, A., Freire, S., Halkia, M., … Syrris, V. (2016). Operating procedure for the production of the Global Human Settlement Layer from Landsat data of the epochs 1975, 1990, 2000, and 2014. *Luxembourg: Publications Office of the European Union*.

Planthaber, G., Stonebraker, M., & Frew, J. (2012). EarthDB: Scalable Analysis of MODIS Data Using SciDB. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data* (pp. 11–19). New York, NY, USA: ACM. https://doi.org/10.1145/2447481.2447483

Plesser, H. E. (2018). Reproducibility vs. Replicability: A Brief History of a Confused Terminology. *Frontiers in Neuroinformatics*, *11*. https://doi.org/10.3389/fninf.2017.00076

Potapov, P., Hansen, M. C., Laestadius, L., Turubanova, S., Yaroshenko, A., Thies, C., … Esipova, E. (2017). The last frontiers of wilderness: Tracking loss of intact forest landscapes from 2000 to 2013. *Science Advances*, *3*(1), e1600821. https://doi.org/10.1126/sciadv.1600821

Soille, P., Burger, A., De Marchi, D., Kempeneers, P., Rodriguez, D., Syrris, V., & Vasilev, V. (2018). A versatile data-intensive computing platform for information retrieval from big geospatial data. *Future Generation Computer Systems*, *81*, 30–40. https://doi.org/10.1016/j.future.2017.11.007

Stonebraker, M., Brown, P., Zhang, D., & Becla, J. (2013). SciDB: A Database Management System for Applications with Complex Analytics. *Computing in Science Engineering*, *15*(3), 54–62. https://doi.org/10.1109/MCSE.2013.19

Strobl, P., Baumann, P., Lewis, A., Szantoi, Z., Killough, B., Purss, M., … Dhu, T. (2017). The Six Faces of the Data Cube. In. Presented at the 2017 Conference on Big Data from Space, Toulouse.

Sudmanns, M., & Augustin, H. (2018, July 20). EO-Compass. Retrieved from osf.io/72nc9

Sudmanns, M., Augustin, H., Cavallaro, A.-M., Tiede, D., & Lang, S. (2017). Sentinel-2 Dashboard for spatio-temporal analysis of global scene coverage. In *Proceedings*

of the 2017 conference on Big Data from Space (BiDS'17) 28th-30th November 2017. (pp. 173–176). Toulouse (France). https://doi.org/10.2760/383579

Sudmanns, M., Tiede, D., Augsten, N., Baraldi, A., Belgiu, M., & Lang, S. (2016). Array-Datenbanken für semantische inhaltsbasierte Suche und Analyse in Satellitenbildarchiven. *Dreiländertagung Der DGPF, Der OVG Und Der SGPF in Bern, Schweiz – Publikationen Der DGPF*, *25*, 555–564. Retrieved from http://www.dgpf.de/src/tagung/jt2016/proceedings/papers/07_KKP_DLT2016_Sudmanns_et_al.pdf

Sudmanns, M., Tiede, D., Wendt, L., & Baraldi, A. (2017). Automatic Ex-post Flood Assessment Using Long Time Series of Optical Earth Observation Images. *GI_Forum 2017, Volume 1*, 217–227. https://doi.org/10.1553/giscience2017_01_s217

Syria's civil war explained from the beginning. (2018, April 14). *Al Jazeera News*. Retrieved from https://www.aljazeera.com/news/2016/05/syria-civil-war-explained-160505084119966.html

Tiede, D., Baraldi, A., Sudmanns, M., Belgiu, M., & Lang, S. (2016). ImageQuerying – Earth Observation Image Content Extraction & Querying across Time and Space. In P. Soille & P. G. Marchetti (Eds.), *Proceedings of the 2016 conference on Big Data from Space (BiDS'16) 15th-17th March 2016.* (pp. 192–195). Santa Cruz de Tenerife (Spain). https://doi.org/10.2788/854791

Tiede, D., Baraldi, A., Sudmanns, M., Belgiu, M., & Lang, S. (2017). Architecture and prototypical implementation of a semantic querying system for big Earth observation image bases. *European Journal of Remote Sensing*, *50*(1), 452–463. https://doi.org/10.1080/22797254.2017.1357432

Tiede, D., Lüthje, F., & Baraldi, A. (2014). Automatic post-classification land cover change detection in Landsat images: Analysis of changes in agricultural areas during the Syrian crisis. In *Seyfert, E., Gülch, E., Heipke, C., Schiewe, J., Sester, M. (Eds.), Band 23: Geoinformationen Öffnen Das Tor Zur Welt, 34. Jahrestagung in Hamburg 2014. Publikationen der Deutschen Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation (DGPF) e.V., Potsdam.*

Tulbure, M. G., Broich, M., Stehman, S. V., & Kommareddy, A. (2016). Surface water extent dynamics from three decades of seasonally continuous Landsat time series

at subcontinental scale in a semi-arid region. *Remote Sensing of Environment*, *178*, 142–157. https://doi.org/10.1016/j.rse.2016.02.034

UNHCR. (2018, April 19). Syria emergency. Retrieved May 29, 2018, from http://www.unhcr.org/syria-emergency.html

UNHCR, & Government of Turkey. (2018). Situation Syria Regional Refugee Response. Retrieved May 29, 2018, from https://data2.unhcr.org/en/situations/syria#_ga=2.83878550.126037859.1527592307-1137743097.1527592307

Union of Concerned Scientists. (2017, November 7). UCS Satellite Database. Retrieved June 22, 2018, from https://www.ucsusa.org/nuclear-weapons/space-weapons/satellite-database

United Nations. (2012). *Resolution 66/288: The future we want.* United Nations General Assembly. Retrieved from https://undocs.org/A/RES/66/288

United Nations. (2015a). *Resolution 69/283: Sendai Framework for Disaster Risk Reduction 2015–2030*. United Nations General Assembly. Retrieved from https://undocs.org/A/RES/69/283

United Nations. (2015b). *Resolution 70/1: Transforming our world: The 2030 agenda for sustainable development*. United Nations General Assembly. Retrieved from https://undocs.org/A/RES/70/1

United Nations. (2017). *Resolution 71/313: Global indicator framework for the Sustainable Development Goals and targets of the 2030 Agenda for Sustainable Development* (pp. 4–25). United Nations General Assembly. Retrieved from https://undocs.org/A/RES/71/313

United Nations. (2018). Communications materials. Retrieved June 6, 2018, from https://www.un.org/sustainabledevelopment/news/communications-material/

USGS. (2018, April 25). U.S. Landsat Analysis Ready Data (ARD) | Landsat Missions. Retrieved June 23, 2018, from https://landsat.usgs.gov/ard

Whitehead, B., Andrews, D., Shah, A., & Maidment, G. (2014). Assessing the environmental impact of data centres part 1: Background, energy use and metrics. *Building and Environment*, *82*, 151–159. https://doi.org/10.1016/j.buildenv.2014.08.021

Wulder, M. A., Masek, J. G., Cohen, W. B., Loveland, T. R., & Woodcock, C. E. (2012). Opening the archive: How free data has enabled the science and monitoring promise of Landsat. *Remote Sensing of Environment*, *122*, 2–10. https://doi.org/10.1016/j.rse.2012.01.010

Zhang, Q., Levin, N., Chalkias, C., & Letu, H. (2015). Nighttime Light Remote Sensing: Monitoring Human Societies from Outer Space. In P. S. Thenkabail (Ed.), *Remote Sensing of Water Resources, Disasters, and Urban Studies* (pp. 289–310). New York: CRC Press. Retrieved from https://ebookcentral.proquest.com/lib/unisalzburg-ebooks/reader.action?docID=4009607&ppg=323

Zhu, Z., & Woodcock, C. E. (2012). Object-based cloud and cloud shadow detection in Landsat imagery. *Remote Sensing of Environment*, *118*, 83–94. https://doi.org/10.1016/j.rse.2011.10.028

Part IV

APPENDIX

A

DATA AND CODE

A.1 DATA

A.1.1 *Data Availability Statement*

The data that this thesis is based on are publicly available for download from the Copernicus Open Access Hub (https://scihub.copernicus.eu/), and if not, hopefully from a mirror intended to archive data for a longer time period.

Due to the relatively large amount of data and the location on a virtual machine on the University of Salzburg network, the availability of the processed data cannot be guaranteed following thesis submission. However, if the processed data is still available and I still have access to where they are stored, I will kindly pass them along upon reasonable request.

A.1.2 *Sentinel-2 Products Used*

These are the names of all of the Sentinel-2 products indexed in the ODC implementation and associated granule at the time of writing. Some of the earlier products are listed more than once because they contained multiple scenes.

Table A.1: All of the 591 Sentinel-2 products containing all of the scenes that could have possibly been used in any of the example results (i.e. up to 22 June 2018).

GRANULE	PRODUCT NAME
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20151201T174419_R121_V20151128T083732_20151128T083732.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20151208T155919_R121_V20151208T083248_20151208T083248.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20151211T153444_R021_V20151211T084342_20151211T084342.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20151218T144629_R121_V20151218T083345_20151218T083345.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20151222T124756_R021_V20151221T084611_20151221T084611.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20151228T171121_R121_V20151228T083611_20151228T083611.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20151231T150040_R021_V20151231T084654_20151231T084654.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160107T183308_R121_V20160107T083407_20160107T083407.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160110T140846_R021_V20160110T083957_20160110T083957.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160119T035708_R121_V20160117T083220_20160117T083220.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160120T182045_R021_V20160120T084226_20160120T084226.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160130T185806_R021_V20160130T083204_20160130T083204.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160209T195633_R021_V20160209T083522_20160209T083522.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160216T151057_R121_V20160216T082059_20160216T082059.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160222T164411_R021_V20160219T083039_20160219T083039.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160309T011131_R121_V20160307T082908_20160307T082908.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160311T153104_R021_V20160310T083000_20160310T083000.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160317T125751_R121_V20160317T082111_20160317T082111.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160320T171753_R021_V20160320T083816_20160320T083816.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160322T230003_R021_V20150813T083848_20150813T083848.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160330T140707_R121_V20160327T082559_20160327T082559.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160401T013523_R021_V20160330T083921_20160330T083921.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160407T083809_R121_V20150830T082754_20150830T082754.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160408T003540_R121_V20160406T082552_20160406T082552.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160412T005002_R021_V20160409T083007_20160409T083007.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160416T230134_R121_V20160416T082220_20160416T082220.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160421T012620_R021_V20150823T083854_20150823T083854.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160425T154840_R121_V20150820T082942_20150820T082942.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160426T151752_R121_V20160426T082713_20160426T082713.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160429T144659_R021_V20160429T083428_20160429T083428.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160506T153311_R121_V20160506T082013_20160506T082013.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160509T152812_R021_V20160509T083548_20160509T083548.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160516T193553_R121_V20160516T082404_20160516T082404.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160516T194647_R121_V20160516T082404_20160516T082404.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160519T160445_R021_V20160519T083015_20160519T083015.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160526T163330_R121_V20160526T082013_20160526T082013.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160529T150630_R021_V20160529T083722_20160529T083722.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160608T191219_R021_V20160608T083013_20160608T083013.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160616T120638_R121_V20160615T082010_20160615T082010.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160618T173608_R021_V20160618T083357_20160618T083357.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160625T151634_R121_V20160625T082331_20160625T082331.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160628T141328_R021_V20160628T083813_20160628T083813.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160708T211337_R021_V20160708T083444 _20160708T083444.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160714T041913_R021_V20150902T083049 _20150902T083049.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160715T143738_R121_V20160715T082333 _20160715T082333.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160718T134305_R021_V20160718T083856 _20160718T083856.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160726T114226_R121_V20160725T082012 _20160725T082012.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160806T013445_R121_V20160804T082333 _20160804T082333.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160814T171938_R121_V20160814T082012 _20160814T082011.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160825T211737_R121_V20160824T081602 _20160824T082324.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160904T182052_R121_V20160903T082012 _20160903T082007.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160915T145948_R121_V20160913T081602 _20160913T082934.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160925T053337_R121_V20160923T082002 _20160923T082003.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20160929T014817_R121_V20150820T082006 _20150820T082942.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161003T143345_R121_V20161003T081752 _20161003T082412.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161007T034851_R121_V20160605T081612 _20160605T082403.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161007T104254_R121_V20150830T082006 _20150830T082754.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161014T010543_R121_V20161013T082002 _20161013T082137.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161014T214358_R121_V20150909T081736 _20150909T082324.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161023T030427_R121_V20150919T081736 _20150919T081736.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161023T184631_R121_V20161023T082012 _20161023T082012.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161028T234308_R121_V20150929T081736 _20150929T081736.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161102T153702_R121_V20161102T082112 _20161102T082112.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161112T135805_R121_V20161112T082202 _20161112T082202.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161122T180800_R121_V20161122T082242 _20161122T082242.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20161202T121547_R121_V20161202T082312 _20161202T082312.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20170121T104656_R121_V20150820T082006 _20150820T082006.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20170121T123813_R121_V20150820T082006 _20150820T082006.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20170122T074436_R121_V20150830T082006 _20150830T082006.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20170505T182613_R121_V20151128T082302 _20151128T082302.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20170510T062738_R121_V20151128T082302 _20151128T082302.SAFE
37SBA	S2A_OPER_PRD_MSIL1C_PDMC_20170530T041416_R121_V20151208T082332 _20151208T082332.SAFE
37SBA	S2A_MSIL1C_20161212T082332_N0204_R121_T37SBA_20161212T082908.SAFE
37SBA	S2A_MSIL1C_20161222T082342_N0204_R121_T37SBA_20161222T082338.SAFE
37SBA	S2A_MSIL1C_20170101T082332_N0204_R121_T37SBA_20170101T082343.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SBA	S2A_MSIL1C_20170111T082311_N0204_R121_T37SBA_20170111T082310.SAFE
37SBA	S2A_MSIL1C_20170131T082151_N0204_R121_T37SBA_20170131T082920.SAFE
37SBA	S2A_MSIL1C_20170210T082051_N0204_R121_T37SBA_20170210T082239.SAFE
37SBA	S2A_MSIL1C_20170220T082001_N0204_R121_T37SBA_20170220T082921.SAFE
37SBA	S2A_MSIL1C_20170302T081841_N0204_R121_T37SBA_20170302T082555.SAFE
37SBA	S2A_MSIL1C_20170312T082001_N0204_R121_T37SBA_20170312T081959.SAFE
37SBA	S2A_MSIL1C_20170322T081611_N0204_R121_T37SBA_20170322T082817.SAFE
37SBA	S2A_MSIL1C_20170401T082001_N0204_R121_T37SBA_20170401T082644.SAFE
37SBA	S2A_MSIL1C_20170411T081601_N0204_R121_T37SBA_20170411T082408.SAFE
37SBA	S2A_MSIL1C_20170421T082011_N0204_R121_T37SBA_20170421T082009.SAFE
37SBA	S2A_MSIL1C_20170501T081611_N0205_R121_T37SBA_20170501T082412.SAFE
37SBA	S2A_MSIL1C_20170511T082011_N0205_R121_T37SBA_20170511T082011.SAFE
37SBA	S2A_MSIL1C_20170521T081611_N0205_R121_T37SBA_20170521T082413.SAFE
37SBA	S2A_MSIL1C_20170531T082011_N0205_R121_T37SBA_20170531T082010.SAFE
37SBA	S2A_MSIL1C_20170610T081601_N0205_R121_T37SBA_20170610T082422.SAFE
37SBA	S2A_MSIL1C_20170620T082011_N0205_R121_T37SBA_20170620T082007.SAFE
37SBA	S2A_MSIL1C_20170630T081601_N0205_R121_T37SBA_20170630T082408.SAFE
37SBA	S2A_MSIL1C_20170710T082011_N0205_R121_T37SBA_20170710T082008.SAFE
37SBA	S2A_MSIL1C_20170720T081601_N0205_R121_T37SBA_20170720T082400.SAFE
37SBA	S2A_MSIL1C_20170730T082011_N0205_R121_T37SBA_20170730T082009.SAFE
37SBA	S2A_MSIL1C_20170809T081601_N0205_R121_T37SBA_20170809T082538.SAFE
37SBA	S2A_MSIL1C_20170819T082011_N0205_R121_T37SBA_20170819T082049.SAFE
37SBA	S2A_MSIL1C_20170829T081601_N0205_R121_T37SBA_20170829T082551.SAFE
37SBA	S2A_MSIL1C_20170908T082011_N0205_R121_T37SBA_20170908T082024.SAFE
37SBA	S2A_MSIL1C_20170918T081601_N0205_R121_T37SBA_20170918T082603.SAFE
37SBA	S2A_MSIL1C_20170928T082001_N0205_R121_T37SBA_20170928T082040.SAFE
37SBA	S2A_MSIL1C_20171008T081831_N0205_R121_T37SBA_20171008T082754.SAFE
37SBA	S2A_MSIL1C_20171018T082011_N0205_R121_T37SBA_20171018T082007.SAFE
37SBA	S2A_MSIL1C_20171028T082041_N0206_R121_T37SBA_20171028T120048.SAFE
37SBA	S2A_MSIL1C_20171107T082131_N0206_R121_T37SBA_20171108T171955.SAFE
37SBA	S2A_MSIL1C_20171117T082221_N0206_R121_T37SBA_20171117T120540.SAFE
37SBA	S2A_MSIL1C_20171127T082301_N0206_R121_T37SBA_20171127T104217.SAFE
37SBA	S2A_MSIL1C_20171207T082321_N0206_R121_T37SBA_20171207T105621.SAFE
37SBA	S2A_MSIL1C_20171217T082341_N0206_R121_T37SBA_20171217T105414.SAFE
37SBA	S2A_MSIL1C_20171227T082341_N0206_R121_T37SBA_20171227T105455.SAFE
37SBA	S2A_MSIL1C_20180106T082321_N0206_R121_T37SBA_20180106T104442.SAFE
37SBA	S2A_MSIL1C_20180116T082251_N0206_R121_T37SBA_20180116T105258.SAFE
37SBA	S2A_MSIL1C_20180126T082221_N0206_R121_T37SBA_20180126T120500.SAFE
37SBA	S2A_MSIL1C_20180205T082251_N0206_R121_T37SBA_20180205T134751.SAFE
37SBA	S2A_MSIL1C_20180215T082031_N0206_R121_T37SBA_20180215T120545.SAFE
37SBA	S2A_MSIL1C_20180225T081921_N0206_R121_T37SBA_20180225T120737.SAFE
37SBA	S2A_MSIL1C_20180307T081801_N0206_R121_T37SBA_20180307T093658.SAFE
37SBA	S2A_MSIL1C_20180317T081651_N0206_R121_T37SBA_20180317T103427.SAFE
37SBA	S2A_MSIL1C_20180327T081601_N0206_R121_T37SBA_20180327T122521.SAFE
37SBA	S2A_MSIL1C_20180406T081601_N0206_R121_T37SBA_20180406T094534.SAFE
37SBA	S2A_MSIL1C_20180416T081601_N0206_R121_T37SBA_20180416T103632.SAFE
37SBA	S2A_MSIL1C_20180426T081701_N0206_R121_T37SBA_20180426T093652.SAFE
37SBA	S2A_MSIL1C_20180506T081611_N0206_R121_T37SBA_20180506T094812.SAFE
37SBA	S2A_MSIL1C_20180516T081611_N0206_R121_T37SBA_20180516T102815.SAFE
37SBA	S2A_MSIL1C_20180526T081601_N0206_R121_T37SBA_20180526T120617.SAFE
37SBA	S2A_MSIL1C_20180605T081601_N0206_R121_T37SBA_20180605T094602.SAFE
37SBA	S2A_MSIL1C_20180615T081601_N0206_R121_T37SBA_20180615T102050.SAFE
37SBA	S2B_MSIL1C_20170705T082009_N0205_R121_T37SBA_20170705T082529.SAFE
37SBA	S2B_MSIL1C_20170725T082009_N0205_R121_T37SBA_20170725T082007.SAFE
37SBA	S2B_MSIL1C_20170804T081559_N0205_R121_T37SBA_20170804T082409.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SBA	S2B_MSIL1C_20170814T082009_N0205_R121_T37SBA_20170814T082005.SAFE
37SBA	S2B_MSIL1C_20170824T081559_N0205_R121_T37SBA_20170824T082406.SAFE
37SBA	S2B_MSIL1C_20170903T081959_N0205_R121_T37SBA_20170903T082037.SAFE
37SBA	S2B_MSIL1C_20170923T081959_N0205_R121_T37SBA_20170923T081954.SAFE
37SBA	S2B_MSIL1C_20171003T081739_N0205_R121_T37SBA_20171003T082726.SAFE
37SBA	S2B_MSIL1C_20171013T081959_N0205_R121_T37SBA_20171013T082228.SAFE
37SBA	S2B_MSIL1C_20171023T081959_N0206_R121_T37SBA_20171023T120122.SAFE
37SBA	S2B_MSIL1C_20171102T082059_N0206_R121_T37SBA_20171102T104010.SAFE
37SBA	S2B_MSIL1C_20171112T082149_N0206_R121_T37SBA_20171112T102302.SAFE
37SBA	S2B_MSIL1C_20171122T082229_N0206_R121_T37SBA_20171122T104032.SAFE
37SBA	S2B_MSIL1C_20171202T082309_N0206_R121_T37SBA_20171202T120455.SAFE
37SBA	S2B_MSIL1C_20171212T082329_N0206_R121_T37SBA_20171212T102303.SAFE
37SBA	S2B_MSIL1C_20171222T082329_N0206_R121_T37SBA_20171222T102532.SAFE
37SBA	S2B_MSIL1C_20180101T082329_N0206_R121_T37SBA_20180101T102359.SAFE
37SBA	S2B_MSIL1C_20180111T082309_N0206_R121_T37SBA_20180111T102539.SAFE
37SBA	S2B_MSIL1C_20180121T082239_N0206_R121_T37SBA_20180121T104245.SAFE
37SBA	S2B_MSIL1C_20180131T082149_N0206_R121_T37SBA_20180131T120612.SAFE
37SBA	S2B_MSIL1C_20180210T082049_N0206_R121_T37SBA_20180210T104042.SAFE
37SBA	S2B_MSIL1C_20180220T081949_N0206_R121_T37SBA_20180220T120900.SAFE
37SBA	S2B_MSIL1C_20180302T081839_N0206_R121_T37SBA_20180302T120321.SAFE
37SBA	S2B_MSIL1C_20180312T081729_N0206_R121_T37SBA_20180312T120809.SAFE
37SBA	S2B_MSIL1C_20180322T081619_N0206_R121_T37SBA_20180322T111859.SAFE
37SBA	S2B_MSIL1C_20180401T081559_N0206_R121_T37SBA_20180401T102741.SAFE
37SBA	S2B_MSIL1C_20180411T081559_N0206_R121_T37SBA_20180411T120732.SAFE
37SBA	S2B_MSIL1C_20180421T081559_N0206_R121_T37SBA_20180421T102449.SAFE
37SBA	S2B_MSIL1C_20180501T081559_N0206_R121_T37SBA_20180501T120625.SAFE
37SBA	S2B_MSIL1C_20180511T081559_N0206_R121_T37SBA_20180511T103319.SAFE
37SBA	S2B_MSIL1C_20180521T081559_N0206_R121_T37SBA_20180521T103556.SAFE
37SBA	S2B_MSIL1C_20180531T081559_N0206_R121_T37SBA_20180531T102527.SAFE
37SBA	S2B_MSIL1C_20180610T081559_N0206_R121_T37SBA_20180610T102259.SAFE
37SBA	S2B_MSIL1C_20180620T081859_N0206_R121_T37SBA_20180620T120921.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20151201T174419_R121_V20151128T083732 _20151128T083732.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20151205T143809_R078_V20151205T082946 _20151205T082946.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20151208T155919_R121_V20151208T083248 _20151208T083248.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20151218T144629_R121_V20151218T083345 _20151218T083345.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20151225T161241_R078_V20151225T082528 _20151225T082528.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20151228T171121_R121_V20151228T083611 _20151228T083611.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160104T163452_R078_V20160104T082046 _20160104T082046.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160107T183308_R121_V20160107T083407 _20160107T083407.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160114T191519_R078_V20160114T082234 _20160114T082234.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160119T035708_R121_V20160117T083220 _20160117T083220.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160127T203324_R078_V20160124T082009 _20160124T082914.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160216T151057_R121_V20160216T082059 _20160216T082059.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160309T011131_R121_V20160307T082908 _20160307T082908.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160315T045037_R078_V20160314T081549 _20160314T081549.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160317T125751_R121_V20160317T082111_20160317T082111.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160324T161907_R078_V20160324T081459_20160324T081459.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160330T140707_R121_V20160327T082559_20160327T082559.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160404T081632_R078_V20160403T081016_20160403T081016.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160407T083809_R121_V20150830T082754_20150830T082754.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160408T003540_R121_V20160406T082552_20160406T082552.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160413T222128_R078_V20150827T081634_20150827T081634.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160414T032437_R078_V20160413T081002_20160413T081002.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160416T230134_R121_V20160416T082220_20160416T082220.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160423T155640_R078_V20160423T081436_20160423T081436.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160425T154840_R121_V20150820T082942_20150820T082942.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160425T224958_R121_V20150820T082204_20150820T082204.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160426T151752_R121_V20160426T082713_20160426T082713.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160502T141447_R078_V20150817T081623_20150817T081623.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160506T153311_R121_V20160506T082013_20160506T082013.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160514T020204_R078_V20160513T081543_20160513T081543.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160514T020720_R078_V20160513T081543_20160513T081543.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160516T193553_R121_V20160516T082404_20160516T082404.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160516T194647_R121_V20160516T082404_20160516T082404.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160526T163330_R121_V20160526T082013_20160526T082013.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160526T192034_R078_V20160503T081407_20160503T081407.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160607T113333_R075_V20150628T081327_20150628T081327.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160612T142004_R078_V20160612T081923_20160612T081923.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160616T120638_R121_V20160615T082010_20160615T082010.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160622T134752_R078_V20160622T081612_20160622T081612.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160622T210937_R078_V20160503T081407_20160503T081407.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160625T151634_R121_V20160625T082331_20160625T082331.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160702T131143_R078_V20160702T081922_20160702T081922.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160712T141811_R078_V20160712T081013_20160712T081013.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160715T143738_R121_V20160715T082333_20160715T082333.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160722T144707_R078_V20160722T080852_20160722T080852.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160726T114226_R121_V20160725T082012_20160725T082012.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160801T164420_R078_V20160801T081155 _20160801T081155.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160806T013445_R121_V20160804T082333 _20160804T082333.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160807T153639_R078_V20150708T081152 _20150708T081152.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160812T215032_R078_V20160811T080612 _20160811T081933.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160814T171938_R121_V20160814T082012 _20160814T082011.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160822T193838_R078_V20160821T081002 _20160821T081150.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160825T012733_R078_V20160821T081002 _20160821T081150.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160825T211737_R121_V20160824T081602 _20160824T082324.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160901T164052_R078_V20160831T080612 _20160831T081920.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160904T182052_R121_V20160903T082012 _20160903T082007.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160913T003913_R078_V20160910T081002 _20160910T081308.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160915T145948_R121_V20160913T081602 _20160913T082934.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160922T024204_R078_V20160920T080612 _20160920T081928.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160923T044542_R078_V20150817T081006 _20150817T081623.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160925T053337_R121_V20160923T082002 _20160923T082003.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160929T015037_R121_V20150820T082006 _20150820T082942.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20160930T151921_R078_V20160930T081002 _20160930T081338.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161003T143438_R121_V20161003T081752 _20161003T082412.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161005T070005_R078_V20150827T081006 _20150827T081634.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161007T034851_R121_V20160605T081612 _20160605T082403.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161007T104250_R121_V20150830T082006 _20150830T082754.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161010T162800_R078_V20161010T080842 _20161010T080908.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161014T011122_R121_V20161013T082002 _20161013T082137.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161014T214358_R121_V20150909T081736 _20150909T082324.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161020T134409_R078_V20161020T081002 _20161020T081002.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161023T030427_R121_V20150919T081736 _20150919T081736.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161023T184610_R121_V20161023T082012 _20161023T082012.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161028T234308_R121_V20150929T081736 _20150929T081736.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161030T155306_R078_V20161030T081042 _20161030T081042.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161102T153803_R121_V20161102T082112 _20161102T082112.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161110T165749_R078_V20161109T081142 _20161109T081142.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161112T135839_R121_V20161112T082202 _20161112T082202.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161122T180819_R121_V20161122T082242_20161122T082242.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161129T172837_R078_V20161129T081302_20161129T081302.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161202T121659_R121_V20161202T082312_20161202T082312.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20161220T214931_R078_V20160602T081012_20160602T081319.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20170121T104656_R121_V20150820T082006_20150820T082006.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20170121T111253_R078_V20150827T081006_20150827T081006.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20170121T123813_R121_V20150820T082006_20150820T082006.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20170122T074436_R121_V20150830T082006_20150830T082006.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20170227T124612_R078_V20151125T081252_20151125T081252.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20170505T182613_R121_V20151128T082302_20151128T082302.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20170510T062738_R121_V20151128T082302_20151128T082302.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20170520T033253_R078_V20151205T081322_20151205T081322.SAFE
37SCA	S2A_OPER_PRD_MSIL1C_PDMC_20170530T041416_R121_V20151208T082332_20151208T082332.SAFE
37SCA	S2A_MSIL1C_20161209T081332_N0204_R078_T37SCA_20161209T081749.SAFE
37SCA	S2A_MSIL1C_20161212T082332_N0204_R121_T37SCA_20161212T082908.SAFE
37SCA	S2A_MSIL1C_20161219T081332_N0204_R078_T37SCA_20161219T081334.SAFE
37SCA	S2A_MSIL1C_20161222T082342_N0204_R121_T37SCA_20161222T082338.SAFE
37SCA	S2A_MSIL1C_20161229T081332_N0204_R078_T37SCA_20161229T081838.SAFE
37SCA	S2A_MSIL1C_20170101T082332_N0204_R121_T37SCA_20170101T082343.SAFE
37SCA	S2A_MSIL1C_20170108T081311_N0204_R078_T37SCA_20170108T081313.SAFE
37SCA	S2A_MSIL1C_20170111T082311_N0204_R121_T37SCA_20170111T082310.SAFE
37SCA	S2A_MSIL1C_20170118T081241_N0204_R078_T37SCA_20170118T081845.SAFE
37SCA	S2A_MSIL1C_20170128T081201_N0204_R078_T37SCA_20170128T081200.SAFE
37SCA	S2A_MSIL1C_20170131T082151_N0204_R121_T37SCA_20170131T082920.SAFE
37SCA	S2A_MSIL1C_20170207T081111_N0204_R078_T37SCA_20170207T081511.SAFE
37SCA	S2A_MSIL1C_20170210T082051_N0204_R121_T37SCA_20170210T082239.SAFE
37SCA	S2A_MSIL1C_20170217T081001_N0204_R078_T37SCA_20170217T081512.SAFE
37SCA	S2A_MSIL1C_20170220T082001_N0204_R121_T37SCA_20170220T082921.SAFE
37SCA	S2A_MSIL1C_20170227T081001_N0204_R078_T37SCA_20170227T081644.SAFE
37SCA	S2A_MSIL1C_20170302T081841_N0204_R121_T37SCA_20170302T082555.SAFE
37SCA	S2A_MSIL1C_20170309T080741_N0204_R078_T37SCA_20170309T081438.SAFE
37SCA	S2A_MSIL1C_20170312T082001_N0204_R121_T37SCA_20170312T081959.SAFE
37SCA	S2A_MSIL1C_20170319T080951_N0204_R078_T37SCA_20170319T081428.SAFE
37SCA	S2A_MSIL1C_20170322T081611_N0204_R121_T37SCA_20170322T082817.SAFE
37SCA	S2A_MSIL1C_20170329T080601_N0204_R078_T37SCA_20170329T081603.SAFE
37SCA	S2A_MSIL1C_20170401T082001_N0204_R121_T37SCA_20170401T082644.SAFE
37SCA	S2A_MSIL1C_20170408T081001_N0204_R078_T37SCA_20170408T081305.SAFE
37SCA	S2A_MSIL1C_20170411T081601_N0204_R121_T37SCA_20170411T082408.SAFE
37SCA	S2A_MSIL1C_20170418T080611_N0204_R078_T37SCA_20170418T081149.SAFE
37SCA	S2A_MSIL1C_20170421T082011_N0204_R121_T37SCA_20170421T082009.SAFE
37SCA	S2A_MSIL1C_20170428T081011_N0205_R078_T37SCA_20170428T081258.SAFE
37SCA	S2A_MSIL1C_20170501T081611_N0205_R121_T37SCA_20170501T082412.SAFE
37SCA	S2A_MSIL1C_20170508T080611_N0205_R078_T37SCA_20170508T081123.SAFE
37SCA	S2A_MSIL1C_20170511T082011_N0205_R121_T37SCA_20170511T082011.SAFE
37SCA	S2A_MSIL1C_20170518T081011_N0205_R078_T37SCA_20170518T081310.SAFE
37SCA	S2A_MSIL1C_20170521T081611_N0205_R121_T37SCA_20170521T082413.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SCA	S2A_MSIL1C_20170528T080611_N0205_R078_T37SCA_20170528T081043.SAFE
37SCA	S2A_MSIL1C_20170531T082011_N0205_R121_T37SCA_20170531T082010.SAFE
37SCA	S2A_MSIL1C_20170607T081001_N0205_R078_T37SCA_20170607T081554.SAFE
37SCA	S2A_MSIL1C_20170610T081601_N0205_R121_T37SCA_20170610T082422.SAFE
37SCA	S2A_MSIL1C_20170617T080611_N0205_R078_T37SCA_20170617T081058.SAFE
37SCA	S2A_MSIL1C_20170620T082011_N0205_R121_T37SCA_20170620T082007.SAFE
37SCA	S2A_MSIL1C_20170627T081001_N0205_R078_T37SCA_20170627T081305.SAFE
37SCA	S2A_MSIL1C_20170630T081601_N0205_R121_T37SCA_20170630T082408.SAFE
37SCA	S2A_MSIL1C_20170707T080611_N0205_R078_T37SCA_20170707T081427.SAFE
37SCA	S2A_MSIL1C_20170710T082011_N0205_R121_T37SCA_20170710T082008.SAFE
37SCA	S2A_MSIL1C_20170717T081001_N0205_R078_T37SCA_20170717T081303.SAFE
37SCA	S2A_MSIL1C_20170720T081601_N0205_R121_T37SCA_20170720T082400.SAFE
37SCA	S2A_MSIL1C_20170727T080611_N0205_R078_T37SCA_20170727T080941.SAFE
37SCA	S2A_MSIL1C_20170730T082011_N0205_R121_T37SCA_20170730T082009.SAFE
37SCA	S2A_MSIL1C_20170806T081001_N0205_R078_T37SCA_20170806T081417.SAFE
37SCA	S2A_MSIL1C_20170809T081601_N0205_R121_T37SCA_20170809T082538.SAFE
37SCA	S2A_MSIL1C_20170816T080611_N0205_R078_T37SCA_20170816T081247.SAFE
37SCA	S2A_MSIL1C_20170819T082011_N0205_R121_T37SCA_20170819T082049.SAFE
37SCA	S2A_MSIL1C_20170826T081001_N0205_R078_T37SCA_20170826T081546.SAFE
37SCA	S2A_MSIL1C_20170829T081601_N0205_R121_T37SCA_20170829T082551.SAFE
37SCA	S2A_MSIL1C_20170905T080611_N0205_R078_T37SCA_20170905T081426.SAFE
37SCA	S2A_MSIL1C_20170908T082011_N0205_R121_T37SCA_20170908T082024.SAFE
37SCA	S2A_MSIL1C_20170915T081001_N0205_R078_T37SCA_20170915T081416.SAFE
37SCA	S2A_MSIL1C_20170918T081601_N0205_R121_T37SCA_20170918T082603.SAFE
37SCA	S2A_MSIL1C_20170925T080651_N0205_R078_T37SCA_20170925T081434.SAFE
37SCA	S2A_MSIL1C_20170928T082001_N0205_R121_T37SCA_20170928T082040.SAFE
37SCA	S2A_MSIL1C_20171005T081001_N0205_R078_T37SCA_20171005T081511.SAFE
37SCA	S2A_MSIL1C_20171008T081831_N0205_R121_T37SCA_20171008T082754.SAFE
37SCA	S2A_MSIL1C_20171015T080911_N0205_R078_T37SCA_20171015T080910.SAFE
37SCA	S2A_MSIL1C_20171018T082011_N0205_R121_T37SCA_20171018T082007.SAFE
37SCA	S2A_MSIL1C_20171025T081021_N0206_R078_T37SCA_20171025T115224.SAFE
37SCA	S2A_MSIL1C_20171028T082041_N0206_R121_T37SCA_20171028T120048.SAFE
37SCA	S2A_MSIL1C_20171104T081111_N0206_R078_T37SCA_20171104T103615.SAFE
37SCA	S2A_MSIL1C_20171107T082131_N0206_R121_T37SCA_20171108T171955.SAFE
37SCA	S2A_MSIL1C_20171114T081201_N0206_R078_T37SCA_20171114T103555.SAFE
37SCA	S2A_MSIL1C_20171117T082221_N0206_R121_T37SCA_20171117T120540.SAFE
37SCA	S2A_MSIL1C_20171124T081241_N0206_R078_T37SCA_20171124T103211.SAFE
37SCA	S2A_MSIL1C_20171127T082301_N0206_R121_T37SCA_20171127T104217.SAFE
37SCA	S2A_MSIL1C_20171204T081311_N0206_R078_T37SCA_20171204T110721.SAFE
37SCA	S2A_MSIL1C_20171207T082321_N0206_R121_T37SCA_20171207T105621.SAFE
37SCA	S2A_MSIL1C_20171214T081331_N0206_R078_T37SCA_20171214T101439.SAFE
37SCA	S2A_MSIL1C_20171217T082341_N0206_R121_T37SCA_20171217T105414.SAFE
37SCA	S2A_MSIL1C_20171224T081341_N0206_R078_T37SCA_20171224T103457.SAFE
37SCA	S2A_MSIL1C_20171227T082341_N0206_R121_T37SCA_20171227T105455.SAFE
37SCA	S2A_MSIL1C_20180103T081321_N0206_R078_T37SCA_20180103T101526.SAFE
37SCA	S2A_MSIL1C_20180106T082321_N0206_R121_T37SCA_20180106T104442.SAFE
37SCA	S2A_MSIL1C_20180113T081301_N0206_R078_T37SCA_20180113T115701.SAFE
37SCA	S2A_MSIL1C_20180116T082251_N0206_R121_T37SCA_20180116T105258.SAFE
37SCA	S2A_MSIL1C_20180123T081231_N0206_R078_T37SCA_20180123T115444.SAFE
37SCA	S2A_MSIL1C_20180126T082221_N0206_R121_T37SCA_20180126T120500.SAFE
37SCA	S2A_MSIL1C_20180202T081241_N0206_R078_T37SCA_20180202T133550.SAFE
37SCA	S2A_MSIL1C_20180205T082251_N0206_R121_T37SCA_20180205T134751.SAFE
37SCA	S2A_MSIL1C_20180212T081041_N0206_R078_T37SCA_20180212T120106.SAFE
37SCA	S2A_MSIL1C_20180215T082031_N0206_R121_T37SCA_20180215T120545.SAFE
37SCA	S2A_MSIL1C_20180222T080931_N0206_R078_T37SCA_20180223T160632.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SCA	S2A_MSIL1C_20180225T081921_N0206_R121_T37SCA_20180225T120737.SAFE
37SCA	S2A_MSIL1C_20180304T080821_N0206_R078_T37SCA_20180304T101233.SAFE
37SCA	S2A_MSIL1C_20180307T081801_N0206_R121_T37SCA_20180307T093658.SAFE
37SCA	S2A_MSIL1C_20180314T080711_N0206_R078_T37SCA_20180314T092948.SAFE
37SCA	S2A_MSIL1C_20180317T081651_N0206_R121_T37SCA_20180317T103427.SAFE
37SCA	S2A_MSIL1C_20180324T080601_N0206_R078_T37SCA_20180324T092749.SAFE
37SCA	S2A_MSIL1C_20180327T081601_N0206_R121_T37SCA_20180327T122521.SAFE
37SCA	S2A_MSIL1C_20180403T080611_N0206_R078_T37SCA_20180403T102935.SAFE
37SCA	S2A_MSIL1C_20180406T081601_N0206_R121_T37SCA_20180406T094534.SAFE
37SCA	S2A_MSIL1C_20180413T080611_N0206_R078_T37SCA_20180413T093219.SAFE
37SCA	S2A_MSIL1C_20180416T081601_N0206_R121_T37SCA_20180416T103632.SAFE
37SCA	S2A_MSIL1C_20180423T080611_N0206_R078_T37SCA_20180423T102835.SAFE
37SCA	S2A_MSIL1C_20180426T081701_N0206_R121_T37SCA_20180426T093652.SAFE
37SCA	S2A_MSIL1C_20180503T080611_N0206_R078_T37SCA_20180503T102643.SAFE
37SCA	S2A_MSIL1C_20180506T081611_N0206_R121_T37SCA_20180506T094812.SAFE
37SCA	S2A_MSIL1C_20180513T080611_N0206_R078_T37SCA_20180513T101254.SAFE
37SCA	S2A_MSIL1C_20180516T081611_N0206_R121_T37SCA_20180516T102815.SAFE
37SCA	S2A_MSIL1C_20180523T080711_N0206_R078_T37SCA_20180523T115450.SAFE
37SCA	S2A_MSIL1C_20180526T081601_N0206_R121_T37SCA_20180526T120617.SAFE
37SCA	S2A_MSIL1C_20180602T081011_N0206_R078_T37SCA_20180602T102641.SAFE
37SCA	S2A_MSIL1C_20180605T081601_N0206_R121_T37SCA_20180605T094602.SAFE
37SCA	S2A_MSIL1C_20180612T080611_N0206_R078_T37SCA_20180612T101157.SAFE
37SCA	S2A_MSIL1C_20180615T081601_N0206_R121_T37SCA_20180615T102050.SAFE
37SCA	S2A_MSIL1C_20180622T080611_N0206_R078_T37SCA_20180622T092729.SAFE
37SCA	S2B_MSIL1C_20170702T081009_N0205_R078_T37SCA_20170702T081231.SAFE
37SCA	S2B_MSIL1C_20170705T082009_N0205_R121_T37SCA_20170705T082529.SAFE
37SCA	S2B_MSIL1C_20170712T081009_N0205_R078_T37SCA_20170712T081307.SAFE
37SCA	S2B_MSIL1C_20170722T080609_N0205_R078_T37SCA_20170722T080953.SAFE
37SCA	S2B_MSIL1C_20170725T082009_N0205_R121_T37SCA_20170725T082007.SAFE
37SCA	S2B_MSIL1C_20170801T080959_N0205_R078_T37SCA_20170801T081306.SAFE
37SCA	S2B_MSIL1C_20170804T081559_N0205_R121_T37SCA_20170804T082409.SAFE
37SCA	S2B_MSIL1C_20170811T080609_N0205_R078_T37SCA_20170811T080944.SAFE
37SCA	S2B_MSIL1C_20170814T082009_N0205_R121_T37SCA_20170814T082005.SAFE
37SCA	S2B_MSIL1C_20170821T080959_N0205_R078_T37SCA_20170821T081046.SAFE
37SCA	S2B_MSIL1C_20170824T081559_N0205_R121_T37SCA_20170824T082406.SAFE
37SCA	S2B_MSIL1C_20170831T080559_N0205_R078_T37SCA_20170831T081240.SAFE
37SCA	S2B_MSIL1C_20170903T081959_N0205_R121_T37SCA_20170903T082037.SAFE
37SCA	S2B_MSIL1C_20170910T080959_N0205_R078_T37SCA_20170910T081359.SAFE
37SCA	S2B_MSIL1C_20170920T080609_N0205_R078_T37SCA_20170920T081256.SAFE
37SCA	S2B_MSIL1C_20170923T081959_N0205_R121_T37SCA_20170923T081954.SAFE
37SCA	S2B_MSIL1C_20170930T080949_N0205_R078_T37SCA_20170930T081407.SAFE
37SCA	S2B_MSIL1C_20171003T081739_N0205_R121_T37SCA_20171003T082726.SAFE
37SCA	S2B_MSIL1C_20171010T080829_N0205_R078_T37SCA_20171010T081722.SAFE
37SCA	S2B_MSIL1C_20171013T081959_N0205_R121_T37SCA_20171013T082228.SAFE
37SCA	S2B_MSIL1C_20171020T080949_N0205_R078_T37SCA_20171020T081821.SAFE
37SCA	S2B_MSIL1C_20171023T081959_N0206_R121_T37SCA_20171023T120122.SAFE
37SCA	S2B_MSIL1C_20171030T081039_N0206_R078_T37SCA_20171030T101035.SAFE
37SCA	S2B_MSIL1C_20171102T082059_N0206_R121_T37SCA_20171102T104010.SAFE
37SCA	S2B_MSIL1C_20171109T081129_N0206_R078_T37SCA_20171110T151005.SAFE
37SCA	S2B_MSIL1C_20171112T082149_N0206_R121_T37SCA_20171112T102302.SAFE
37SCA	S2B_MSIL1C_20171119T081219_N0206_R078_T37SCA_20171121T030017.SAFE
37SCA	S2B_MSIL1C_20171122T082229_N0206_R121_T37SCA_20171122T104032.SAFE
37SCA	S2B_MSIL1C_20171129T081249_N0206_R078_T37SCA_20171129T101327.SAFE
37SCA	S2B_MSIL1C_20171202T082309_N0206_R121_T37SCA_20171202T120455.SAFE
37SCA	S2B_MSIL1C_20171209T081319_N0206_R078_T37SCA_20171209T101509.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SCA	S2B_MSIL1C_20171212T082329_N0206_R121_T37SCA_20171212T102303.SAFE
37SCA	S2B_MSIL1C_20171219T081329_N0206_R078_T37SCA_20171219T101906.SAFE
37SCA	S2B_MSIL1C_20171222T082329_N0206_R121_T37SCA_20171222T102532.SAFE
37SCA	S2B_MSIL1C_20171229T081329_N0206_R078_T37SCA_20171229T083604.SAFE
37SCA	S2B_MSIL1C_20180101T082329_N0206_R121_T37SCA_20180101T102359.SAFE
37SCA	S2B_MSIL1C_20180108T081319_N0206_R078_T37SCA_20180108T101452.SAFE
37SCA	S2B_MSIL1C_20180111T082309_N0206_R121_T37SCA_20180111T102539.SAFE
37SCA	S2B_MSIL1C_20180118T081249_N0206_R078_T37SCA_20180118T111939.SAFE
37SCA	S2B_MSIL1C_20180121T082239_N0206_R121_T37SCA_20180121T104245.SAFE
37SCA	S2B_MSIL1C_20180128T081159_N0206_R078_T37SCA_20180128T102945.SAFE
37SCA	S2B_MSIL1C_20180131T082149_N0206_R121_T37SCA_20180131T120612.SAFE
37SCA	S2B_MSIL1C_20180207T081109_N0206_R078_T37SCA_20180207T115926.SAFE
37SCA	S2B_MSIL1C_20180210T082049_N0206_R121_T37SCA_20180210T104042.SAFE
37SCA	S2B_MSIL1C_20180217T081009_N0206_R078_T37SCA_20180217T102448.SAFE
37SCA	S2B_MSIL1C_20180220T081949_N0206_R121_T37SCA_20180220T120900.SAFE
37SCA	S2B_MSIL1C_20180227T080849_N0206_R078_T37SCA_20180227T115501.SAFE
37SCA	S2B_MSIL1C_20180302T081839_N0206_R121_T37SCA_20180302T120321.SAFE
37SCA	S2B_MSIL1C_20180309T080739_N0206_R078_T37SCA_20180309T101603.SAFE
37SCA	S2B_MSIL1C_20180312T081729_N0206_R121_T37SCA_20180312T120809.SAFE
37SCA	S2B_MSIL1C_20180319T080629_N0206_R078_T37SCA_20180319T111540.SAFE
37SCA	S2B_MSIL1C_20180322T081619_N0206_R121_T37SCA_20180322T111859.SAFE
37SCA	S2B_MSIL1C_20180329T080559_N0206_R078_T37SCA_20180329T101348.SAFE
37SCA	S2B_MSIL1C_20180401T081559_N0206_R121_T37SCA_20180401T102741.SAFE
37SCA	S2B_MSIL1C_20180408T080609_N0206_R078_T37SCA_20180408T093650.SAFE
37SCA	S2B_MSIL1C_20180411T081559_N0206_R121_T37SCA_20180411T120732.SAFE
37SCA	S2B_MSIL1C_20180418T080609_N0206_R078_T37SCA_20180418T101459.SAFE
37SCA	S2B_MSIL1C_20180421T081559_N0206_R121_T37SCA_20180421T102449.SAFE
37SCA	S2B_MSIL1C_20180428T080609_N0206_R078_T37SCA_20180428T101116.SAFE
37SCA	S2B_MSIL1C_20180501T081559_N0206_R121_T37SCA_20180501T120625.SAFE
37SCA	S2B_MSIL1C_20180508T080609_N0206_R078_T37SCA_20180508T115609.SAFE
37SCA	S2B_MSIL1C_20180511T081559_N0206_R121_T37SCA_20180511T103319.SAFE
37SCA	S2B_MSIL1C_20180518T080609_N0206_R078_T37SCA_20180518T111754.SAFE
37SCA	S2B_MSIL1C_20180521T081559_N0206_R121_T37SCA_20180521T103556.SAFE
37SCA	S2B_MSIL1C_20180531T081559_N0206_R121_T37SCA_20180531T102527.SAFE
37SCA	S2B_MSIL1C_20180607T080609_N0206_R078_T37SCA_20180607T102248.SAFE
37SCA	S2B_MSIL1C_20180610T081559_N0206_R121_T37SCA_20180610T102259.SAFE
37SCA	S2B_MSIL1C_20180617T080609_N0206_R078_T37SCA_20180617T102342.SAFE
37SCA	S2B_MSIL1C_20180620T081859_N0206_R121_T37SCA_20180620T120921.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20151201T174419_R121_V20151128T083732 _20151128T083732.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20151205T143809_R078_V20151205T082946 _20151205T082946.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20151208T155919_R121_V20151208T083248 _20151208T083248.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20151218T144629_R121_V20151218T083345 _20151218T083345.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20151225T161241_R078_V20151225T082528 _20151225T082528.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20151228T171121_R121_V20151228T083611 _20151228T083611.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160104T163452_R078_V20160104T082046 _20160104T082046.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160107T183308_R121_V20160107T083407 _20160107T083407.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160114T191519_R078_V20160114T082234 _20160114T082234.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160119T035708_R121_V20160117T083220 _20160117T083220.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160127T203324_R078_V20160124T082009_20160124T082914.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160216T151057_R121_V20160216T082059_20160216T082059.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160309T011131_R121_V20160307T082908_20160307T082908.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160315T045037_R078_V20160314T081549_20160314T081549.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160317T125751_R121_V20160317T082111_20160317T082111.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160324T161907_R078_V20160324T081459_20160324T081459.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160330T140707_R121_V20160327T082559_20160327T082559.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160404T081632_R078_V20160403T081016_20160403T081016.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160407T083809_R121_V20150830T082754_20150830T082754.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160408T003540_R121_V20160406T082552_20160406T082552.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160413T222128_R078_V20150827T081634_20150827T081634.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160414T032437_R078_V20160413T081002_20160413T081002.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160416T230134_R121_V20160416T082220_20160416T082220.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160423T155640_R078_V20160423T081436_20160423T081436.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160425T154840_R121_V20150820T082942_20150820T082942.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160425T224958_R121_V20150820T082204_20150820T082204.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160426T151752_R121_V20160426T082713_20160426T082713.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160502T141447_R078_V20150817T081623_20150817T081623.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160506T153311_R121_V20160506T082013_20160506T082013.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160514T020204_R078_V20160513T081543_20160513T081543.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160514T020720_R078_V20160513T081543_20160513T081543.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160516T193553_R121_V20160516T082404_20160516T082404.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160516T194647_R121_V20160516T082404_20160516T082404.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160526T163330_R121_V20160526T082013_20160526T082013.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160526T192034_R078_V20160503T081407_20160503T081407.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160607T113333_R075_V20150628T081327_20150628T081327.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160612T142004_R078_V20160612T081923_20160612T081923.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160616T120638_R121_V20160615T082010_20160615T082010.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160622T134752_R078_V20160622T081612_20160622T081612.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160622T210937_R078_V20160503T081407_20160503T081407.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160625T151634_R121_V20160625T082331_20160625T082331.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160702T131143_R078_V20160702T081922_20160702T081922.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160712T141811_R078_V20160712T081013 _20160712T081013.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160715T143738_R121_V20160715T082333 _20160715T082333.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160722T144707_R078_V20160722T080852 _20160722T080852.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160801T164420_R078_V20160801T081155 _20160801T081155.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160807T153639_R078_V20150708T081152 _20150708T081152.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160812T215032_R078_V20160811T080612 _20160811T081933.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160822T193838_R078_V20160821T081002 _20160821T081150.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160825T012733_R078_V20160821T081002 _20160821T081150.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160901T164052_R078_V20160831T080612 _20160831T081920.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160913T003913_R078_V20160910T081002 _20160910T081308.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160922T024204_R078_V20160920T080612 _20160920T081928.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160923T044542_R078_V20150817T081006 _20150817T081623.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20160930T143752_R078_V20160930T081002 _20160930T081338.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20161005T065331_R078_V20150827T081006 _20150827T081634.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20161007T034851_R121_V20160605T081612 _20160605T082403.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20161010T160649_R078_V20161010T080842 _20161010T080908.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20161020T133812_R078_V20161020T081002 _20161020T081002.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20161030T154140_R078_V20161030T081042 _20161030T081042.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20161110T154535_R078_V20161109T081142 _20161109T081142.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20161129T173700_R078_V20161129T081302 _20161129T081302.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20161220T214931_R078_V20160602T081012 _20160602T081319.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20170121T111253_R078_V20150827T081006 _20150827T081006.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20170227T124612_R078_V20151125T081252 _20151125T081252.SAFE
37SDA	S2A_OPER_PRD_MSIL1C_PDMC_20170520T033253_R078_V20151205T081322 _20151205T081322.SAFE
37SDA	S2A_MSIL1C_20161209T081332_N0204_R078_T37SDA_20161209T081749.SAFE
37SDA	S2A_MSIL1C_20161219T081332_N0204_R078_T37SDA_20161219T081334.SAFE
37SDA	S2A_MSIL1C_20161229T081332_N0204_R078_T37SDA_20161229T081838.SAFE
37SDA	S2A_MSIL1C_20170108T081311_N0204_R078_T37SDA_20170108T081313.SAFE
37SDA	S2A_MSIL1C_20170118T081241_N0204_R078_T37SDA_20170118T081845.SAFE
37SDA	S2A_MSIL1C_20170128T081201_N0204_R078_T37SDA_20170128T081200.SAFE
37SDA	S2A_MSIL1C_20170207T081111_N0204_R078_T37SDA_20170207T081511.SAFE
37SDA	S2A_MSIL1C_20170217T081001_N0204_R078_T37SDA_20170217T081512.SAFE
37SDA	S2A_MSIL1C_20170227T081001_N0204_R078_T37SDA_20170227T081644.SAFE
37SDA	S2A_MSIL1C_20170309T080741_N0204_R078_T37SDA_20170309T081438.SAFE
37SDA	S2A_MSIL1C_20170319T080951_N0204_R078_T37SDA_20170319T081428.SAFE
37SDA	S2A_MSIL1C_20170329T080601_N0204_R078_T37SDA_20170329T081603.SAFE
37SDA	S2A_MSIL1C_20170408T081001_N0204_R078_T37SDA_20170408T081305.SAFE
37SDA	S2A_MSIL1C_20170418T080611_N0204_R078_T37SDA_20170418T081149.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SDA	S2A_MSIL1C_20170428T081011_N0205_R078_T37SDA_20170428T081258.SAFE
37SDA	S2A_MSIL1C_20170508T080611_N0205_R078_T37SDA_20170508T081123.SAFE
37SDA	S2A_MSIL1C_20170518T081011_N0205_R078_T37SDA_20170518T081310.SAFE
37SDA	S2A_MSIL1C_20170528T080611_N0205_R078_T37SDA_20170528T081043.SAFE
37SDA	S2A_MSIL1C_20170607T081001_N0205_R078_T37SDA_20170607T081554.SAFE
37SDA	S2A_MSIL1C_20170617T080611_N0205_R078_T37SDA_20170617T081058.SAFE
37SDA	S2A_MSIL1C_20170627T081001_N0205_R078_T37SDA_20170627T081305.SAFE
37SDA	S2A_MSIL1C_20170707T080611_N0205_R078_T37SDA_20170707T081427.SAFE
37SDA	S2A_MSIL1C_20170717T081001_N0205_R078_T37SDA_20170717T081303.SAFE
37SDA	S2A_MSIL1C_20170727T080611_N0205_R078_T37SDA_20170727T080941.SAFE
37SDA	S2A_MSIL1C_20170806T081001_N0205_R078_T37SDA_20170806T081417.SAFE
37SDA	S2A_MSIL1C_20170816T080611_N0205_R078_T37SDA_20170816T081247.SAFE
37SDA	S2A_MSIL1C_20170826T081001_N0205_R078_T37SDA_20170826T081546.SAFE
37SDA	S2A_MSIL1C_20170905T080611_N0205_R078_T37SDA_20170905T081426.SAFE
37SDA	S2A_MSIL1C_20170915T081001_N0205_R078_T37SDA_20170915T081416.SAFE
37SDA	S2A_MSIL1C_20170925T080651_N0205_R078_T37SDA_20170925T081434.SAFE
37SDA	S2A_MSIL1C_20171005T081001_N0205_R078_T37SDA_20171005T081511.SAFE
37SDA	S2A_MSIL1C_20171015T080911_N0205_R078_T37SDA_20171015T080910.SAFE
37SDA	S2A_MSIL1C_20171025T081021_N0206_R078_T37SDA_20171025T115224.SAFE
37SDA	S2A_MSIL1C_20171104T081111_N0206_R078_T37SDA_20171104T103615.SAFE
37SDA	S2A_MSIL1C_20171114T081201_N0206_R078_T37SDA_20171114T103555.SAFE
37SDA	S2A_MSIL1C_20171124T081241_N0206_R078_T37SDA_20171124T103211.SAFE
37SDA	S2A_MSIL1C_20171204T081311_N0206_R078_T37SDA_20171204T110721.SAFE
37SDA	S2A_MSIL1C_20171214T081331_N0206_R078_T37SDA_20171214T101439.SAFE
37SDA	S2A_MSIL1C_20171224T081341_N0206_R078_T37SDA_20171224T103457.SAFE
37SDA	S2A_MSIL1C_20180103T081321_N0206_R078_T37SDA_20180103T101526.SAFE
37SDA	S2A_MSIL1C_20180113T081301_N0206_R078_T37SDA_20180113T115701.SAFE
37SDA	S2A_MSIL1C_20180123T081231_N0206_R078_T37SDA_20180123T115444.SAFE
37SDA	S2A_MSIL1C_20180202T081241_N0206_R078_T37SDA_20180202T133550.SAFE
37SDA	S2A_MSIL1C_20180212T081041_N0206_R078_T37SDA_20180212T120106.SAFE
37SDA	S2A_MSIL1C_20180222T080931_N0206_R078_T37SDA_20180223T160632.SAFE
37SDA	S2A_MSIL1C_20180304T080821_N0206_R078_T37SDA_20180304T101233.SAFE
37SDA	S2A_MSIL1C_20180314T080711_N0206_R078_T37SDA_20180314T092948.SAFE
37SDA	S2A_MSIL1C_20180324T080601_N0206_R078_T37SDA_20180324T092749.SAFE
37SDA	S2A_MSIL1C_20180403T080611_N0206_R078_T37SDA_20180403T102935.SAFE
37SDA	S2A_MSIL1C_20180413T080611_N0206_R078_T37SDA_20180413T093219.SAFE
37SDA	S2A_MSIL1C_20180423T080611_N0206_R078_T37SDA_20180423T102835.SAFE
37SDA	S2A_MSIL1C_20180503T080611_N0206_R078_T37SDA_20180503T102643.SAFE
37SDA	S2A_MSIL1C_20180513T080611_N0206_R078_T37SDA_20180513T101254.SAFE
37SDA	S2A_MSIL1C_20180523T080711_N0206_R078_T37SDA_20180523T115450.SAFE
37SDA	S2A_MSIL1C_20180602T081011_N0206_R078_T37SDA_20180602T102641.SAFE
37SDA	S2A_MSIL1C_20180612T080611_N0206_R078_T37SDA_20180612T101157.SAFE
37SDA	S2A_MSIL1C_20180622T080611_N0206_R078_T37SDA_20180622T092729.SAFE
37SDA	S2B_MSIL1C_20170702T081009_N0205_R078_T37SDA_20170702T081231.SAFE
37SDA	S2B_MSIL1C_20170712T081009_N0205_R078_T37SDA_20170712T081307.SAFE
37SDA	S2B_MSIL1C_20170722T080609_N0205_R078_T37SDA_20170722T080953.SAFE
37SDA	S2B_MSIL1C_20170801T080959_N0205_R078_T37SDA_20170801T081306.SAFE
37SDA	S2B_MSIL1C_20170811T080609_N0205_R078_T37SDA_20170811T080944.SAFE
37SDA	S2B_MSIL1C_20170821T080959_N0205_R078_T37SDA_20170821T081046.SAFE
37SDA	S2B_MSIL1C_20170831T080559_N0205_R078_T37SDA_20170831T081240.SAFE
37SDA	S2B_MSIL1C_20170910T080959_N0205_R078_T37SDA_20170910T081359.SAFE
37SDA	S2B_MSIL1C_20170920T080609_N0205_R078_T37SDA_20170920T081256.SAFE
37SDA	S2B_MSIL1C_20170930T080949_N0205_R078_T37SDA_20170930T081407.SAFE
37SDA	S2B_MSIL1C_20171010T080829_N0205_R078_T37SDA_20171010T081722.SAFE
37SDA	S2B_MSIL1C_20171020T080949_N0205_R078_T37SDA_20171020T081821.SAFE

Table A.1: *continued ...*

GRANULE	PRODUCT NAME
37SDA	S2B_MSIL1C_20171030T081039_N0206_R078_T37SDA_20171030T101035.SAFE
37SDA	S2B_MSIL1C_20171109T081129_N0206_R078_T37SDA_20171110T151005.SAFE
37SDA	S2B_MSIL1C_20171119T081219_N0206_R078_T37SDA_20171121T030017.SAFE
37SDA	S2B_MSIL1C_20171129T081249_N0206_R078_T37SDA_20171129T101327.SAFE
37SDA	S2B_MSIL1C_20171209T081319_N0206_R078_T37SDA_20171209T101509.SAFE
37SDA	S2B_MSIL1C_20171219T081329_N0206_R078_T37SDA_20171219T101906.SAFE
37SDA	S2B_MSIL1C_20171229T081329_N0206_R078_T37SDA_20171229T083604.SAFE
37SDA	S2B_MSIL1C_20180108T081319_N0206_R078_T37SDA_20180108T101452.SAFE
37SDA	S2B_MSIL1C_20180118T081249_N0206_R078_T37SDA_20180118T111939.SAFE
37SDA	S2B_MSIL1C_20180128T081159_N0206_R078_T37SDA_20180128T102945.SAFE
37SDA	S2B_MSIL1C_20180207T081109_N0206_R078_T37SDA_20180207T115926.SAFE
37SDA	S2B_MSIL1C_20180217T081009_N0206_R078_T37SDA_20180217T102448.SAFE
37SDA	S2B_MSIL1C_20180227T080849_N0206_R078_T37SDA_20180227T115501.SAFE
37SDA	S2B_MSIL1C_20180309T080739_N0206_R078_T37SDA_20180309T101603.SAFE
37SDA	S2B_MSIL1C_20180319T080629_N0206_R078_T37SDA_20180319T111540.SAFE
37SDA	S2B_MSIL1C_20180329T080559_N0206_R078_T37SDA_20180329T101348.SAFE
37SDA	S2B_MSIL1C_20180408T080609_N0206_R078_T37SDA_20180408T093650.SAFE
37SDA	S2B_MSIL1C_20180418T080609_N0206_R078_T37SDA_20180418T101459.SAFE
37SDA	S2B_MSIL1C_20180428T080609_N0206_R078_T37SDA_20180428T101116.SAFE
37SDA	S2B_MSIL1C_20180508T080609_N0206_R078_T37SDA_20180508T115609.SAFE
37SDA	S2B_MSIL1C_20180518T080609_N0206_R078_T37SDA_20180518T111754.SAFE
37SDA	S2B_MSIL1C_20180607T080609_N0206_R078_T37SDA_20180607T102248.SAFE
37SDA	S2B_MSIL1C_20180617T080609_N0206_R078_T37SDA_20180617T102342.SAFE

A.2 CODE

A.2.1 *License Information for Code*

The code developed in the purview of this thesis is licensed under a GNU General Public License v3.0 (GNU GPLv3).

It 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 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, which can be accessed at: https://www.gnu.org/licenses/gpl-3.0.html

A.2.2 *Actual Code*

Code is provided in a printed version for longer-term documentation in the case that the digital repository ever ceases to exist. Some of the Python scripts have been modified slightly without testing for better display (i.e. only the few lines that exceeded the Python style guide of 79 characters and thus the margins of this appendix). Please refer to the existing digital code repository if intending to use parts of code.

At the time of writing, all code is available in some form at the following GitHub repository belonging to the author (@augustinh22): http://github.com/augustinh22/AIQ

This file defines a conda environment, which can be reproduced using this file and the following command: `conda env create -file=FILE.yaml`

Listing A.1: conda environment definition for download to processing with SIAM™ (`aiq27_environment.yaml`)

```
name: aiq27
channels:
- conda-forge
- defaults
dependencies:
- asn1crypto=0.22.0=py27_0
- blas=1.1=openblas
- boost=1.65.1=py27_0
- boost-cpp=1.65.1=1
- bzip2=1.0.6=1
- ca-certificates=2017.11.5=0
- cairo=1.14.6=5
- certifi=2017.11.5=py27_0
- cffi=1.11.2=py27_0
- chardet=3.0.4=py27_0
- cryptography=2.0.3=py27_0
- curl=7.55.1=0
- enum34=1.1.6=py27_1
- expat=2.1.0=3
- fontconfig=2.12.1=6
- freetype=2.7=2
- freexl=1.0.4=0
- gdal=2.2.3=py27_0
- geos=3.6.2=1
- geotiff=1.4.2=0
- gettext=0.19.7=1
- giflib=5.1.4=0
- glib=2.51.4=0
- hdf4=4.2.13=0
- hdf5=1.10.1=1
- icu=58.2=0
- idna=2.6=py27_1
- ipaddress=1.0.18=py27_0
- jpeg=9b=2
- json-c=0.12.1=0
- kealib=1.4.7=4
- krb5=1.14.2=0
- libdap4=3.18.3=2
- libffi=3.2.1=3
- libgdal=2.2.3=1
- libiconv=1.15=0
- libkml=1.3.0=4
- libnetcdf=4.4.1.1=10
- libpng=1.6.28=2
- libpq=9.6.3=0
- libspatialite=4.3.0a=18
- libssh2=1.8.0=2
- libtiff=4.0.7=1
- libxml2=2.9.5=2
- ncurses=5.9=10
- numpy=1.13.3=py27_blas_openblas_201
- openblas=0.2.20=4
- openjpeg=2.3.0=1
- openssl=1.0.2m=0
- pcre=8.39=0
- pip=9.0.1=py27_0
- pixman=0.34.0=1
- poppler=0.61.1=2
- poppler-data=0.4.8=0
- proj4=4.9.3=5
- pycparser=2.18=py27_0
- pyopenssl=17.2.0=py27_0
- pysocks=1.6.7=py27_0
- python=2.7.14=1
- readline=6.2=0
- requests=2.18.4=py27_1
- scipy=1.0.0=py27_blas_openblas_201
- setuptools=37.0.0=py27_0
- six=1.11.0=py27_1
- sqlite=3.13.0=1
- tk=8.5.19=2
- urllib3=1.22=py27_0
- wheel=0.30.0=py_1
- xerces-c=3.2.0=0
- xz=5.2.3=0
- zlib=1.2.11=0
- libgfortran=3.0.0=1
```

```
78 - util-linux=2.21=0
79 prefix: /home/hannah/.conda/envs/aiq27
```

Listing A.2: Download data from the Copernicus Open Access Hub (`download_linux.py`)

This script handles http downloads of a given granule from the Copernicus Open Access Hub. It also enables more complex queries of the archive and batch downloads from the CLI. While most lines fit within the 79 character style guideline, a few longer lines were broken without testing for this printed version. Unnecessary spaces between lines were also removed.

```
#-------------------------------------------------------------------------------
# Name:         Sentinel2 Downloader
# Purpose:      This script uses wget and requests to download Sentinel2
#               images from the Sentinel API or Open Access Data Hub on Linux.
#
# Author:       h.Augustin
#
# Created:      07.06.2017
#
#-------------------------------------------------------------------------------

#! /usr/bin/env python
# -- coding: iso-8859-1 --

import os
import sys
import zipfile
import argparse
import xml.etree.ElementTree as etree
from datetime import date
from datetime import datetime
import ast
import logging

import requests

################################################################################
def get_args():
    #
    # Create download tool help response.
    #
    prog = os.path.basename(sys.argv[0])

    if len(sys.argv) == 1:

        print('\n        {0} [options]'
            '\n        Help: {1} --help'
            '\n        or: {1} -h'
            '\nexample python {0} --lat 43.6 --lon 1.44\n').format(
            sys.argv[0], prog)

        sys.exit(-1)

    else:
        parser = argparse.ArgumentParser(prog=prog,
            usage='%(prog)s [options]',
            description='Sentinel data downloader.',
            argument_default=None,
            epilog='Go get \'em!')
        #
        # Authorization and directory related commands
        #
        parser.add_argument(
                '-a', '--auth', dest='auth', action='store',
                type=str, help='Sentinels Scientific Data Hub account and '
                'password file, if available')
        parser.add_argument(
                '-w', '--write_dir', dest='write_dir', action='store',
                type=str, help='Path where products should be downloaded',
                default='./tempS2')
        parser.add_argument(
                '-r', dest='MaxRecords', action='store', type=int,
                help='Maximum number of records to download (default=100)',
                default=100)
        parser.add_argument(
                '--hub', dest='hub', action='store',
                help='Try other hubs if apihub is not working', default=None)
        parser.add_argument(
                '--auto', dest='auto', action='store',
                help=('No user input necessary -- automatically downloads all '
                'matching results ingested within the last month.'),
                choices=['y','n'], default=None)
        #
        # Location related commands
        #
        parser.add_argument(
                '-t', '--tile', dest='tile', action='store',
```

```
78                  type=str, help='Sentinel-2 Tile number', default=None)
79          parser.add_argument(
80                  '--lat', dest='lat', action='store', type=float,
81                  help='Latitude in decimal degrees', default=None)
82          parser.add_argument(
83                  '--lon', dest='lon', action='store', type=float,
84                  help='Longitude in decimal degrees', default=None)
85          parser.add_argument(
86                  '--latmin', dest='latmin', action='store', type=float,
87                  help='Min latitude in decimal degrees', default=None)
88          parser.add_argument(
89                  '--latmax', dest='latmax', action='store', type=float,
90                  help='Max latitude in decimal degrees', default=None)
91          parser.add_argument(
92                  '--lonmin', dest='lonmin', action='store', type=float,
93                  help='Min longitude in decimal degrees', default=None)
94          parser.add_argument(
95                  '--lonmax', dest='lonmax', action='store', type=float,
96                  help='Max longitude in decimal degrees', default=None)
97          #
98          # Other Sentinel file related command parameters
99          #
100         parser.add_argument(
101                 '-s', '--sentinel', dest='sentinel', action='store',
102                 type=str, help='Sentinel mission considered (e.g. S1, S2A)',
103                 default='S2')
104         parser.add_argument(
105                 '-d', '--start_date', dest='start_date', action='store',
106                 type=str, help='Start date, fmt("2015-12-22")', default=None)
107         parser.add_argument(
108                 '-f', '--end_date', dest='end_date', action='store',
109                 type=str, help='End date, fmt("2015-12-23")', default=None)
110         parser.add_argument(
111                 '--id', '--start_ingest_date',
112                 dest='start_ingest_date', action='store', type=str,
113                 help='Start ingestion date, fmt("2015-12-22")', default=None)
114         parser.add_argument(
115                 '--if', '--end_ingest_date',
116                 dest='end_ingest_date', action='store', type=str,
117                 help='end ingestion date fmt("2015-12-23")', default=None)
118         parser.add_argument(
119                 '-o', '--orbit', dest='rel_orbit', action='store',
120                 type=int, help='Relative orbit path number', default=None)
121         parser.add_argument(
122                 '-od', '--orbit_dir', dest='orbit_dir', action='store',
123                 type=str, help='Orbit direction (e.g. asc or desc)',
124                 default=None)
125         parser.add_argument(
126                 '-ord', '--orderby', dest='orderby', action='store',type=str,
127                 help=('Order results by ingestion (id) or capture date (d), '
128                 'asc or desc (e.g. \'id-asc\' for ingestion date, ascending)'),
129                 default=None)
130         #
131         # Sentinel-1 specific parameters.
132         #
133         parser.add_argument(
134                 '-s1pr', '--s1product', dest='s1product', action='store',
135                 type=str, help='Sentinel-1 product (e.g. SLC, GRD, OCN)',
136                 default=None)
137         parser.add_argument(
138                 '-s1po', '--s1polar', dest='s1polar', action='store', type=str,
139                 help='Sentinel-1 polarisation mode (e.g. HH, VH, VV VH)',
140                 default=None)
141         parser.add_argument(
142                 '-s1mo', '--s1mode', dest='s1mode', action='store', type=str,
143                 help='Sentinel-1 sensor operational mode (e.g. SM, IW, EW)',
144                 default=None)
145         #
146         # Sentinel-2 specific parameters.
147         #
148         parser.add_argument(
149                 '-c', '--max_cloud', dest='max_cloud', action='store',
150                 type=float, help='Only search for products up to a certain '
151                 'cloud percentage (e.g. 50 for 50 percent)', default=None)
152         parser.add_argument(
153                 '-s2pr', '--s2product', dest='s2product', action='store',
154                 type=str, help='Sentinel-2 product (e.g. S2MSI1C, S2MSI2Ap)',
155                 default='S2MSI1C')
156
157         return parser.parse_args()
158
159
```

```
160 def kml_api(tile):
161
162     '''
163     This function returns the center point of a defined S2 tile based on an
164     API developed by M. Sudmanns, or from the kml file if request fails.
165     '''
166     #
167     # Formulate request and get it.
168     #
169     with requests.Session() as s:
170
171         api_request = ('http://cf000008.geo.sbg.ac.at/cgi-bin/s2-dashboard/'
172             'api.py?centroid={}').format(tile)
173
174         try:
175             #
176             # Read string result as a dictionary.
177             #
178             r = s.get(api_request)
179             result = {}
180             result = r.text
181             result = ast.literal_eval(result)
182         #
183         # Catch base-class exception.
184         #
185         except requests.exceptions.RequestException as e:
186
187             print '\n\n{}\n\n'.format(e)
188             result = {"status": "FAIL"}
189     #
190     # Extract lat, lon from API request, or try to get from file if failed.
191     #
192     if result["status"] == "OK" and result["data"]:
193         coords = [result["data"]["x"], result["data"]["y"]]
194     else:
195         print '\nAPI failed.\n'
196         coords = tile_coords(tile, 'point')
197
198     return coords
199
200
201 def check_kml():
202
203     '''Checks for existance of ESA kml file, and quits if not available.'''
204
205     kml_file = ('S2A_OPER_GIP_TILPAR_MPC__20151209T095117_V20150622T000000'
206         '_21000101T000000_B00.kml')
207
208     if os.path.exists(kml_file) is False:
209
210         print(
211             '\n-------------------------------------------------------------'
212             '\nPlease download the ESA Sentinel-2 kml file!'
213             '\nSee README.md for details.'
214             '\n-------------------------------------------------------------\n'
215         )
216
217         sys.exit(-1)
218
219     return kml_file
220
221
222 def tile_coords(tile, form):
223
224     '''Returns polygon or center point coordinates for a tile from the kml.'''
225     #
226     # Check for kml file.
227     #
228     kml_file = check_kml()
229
230     print '\n-------------------------------------------------------------'
231     print 'Hold on while we check the kml for the tile\'s coordinates!'
232     #
233     # Create element tree of all tiles in the kml file.
234     #
235     tree = etree.parse(kml_file)
236     #
237     # Get all placemarks (i.e. tiles).
238     #
239     placemarks = tree.findall('.//{http://www.opengis.net/kml/2.2}Placemark')
240     coords = []
241     #
```

```
242     # Iterate through the attributes within each placemark.
243     #
244     for attributes in placemarks:
245 
246         for subAttribute in attributes:
247             #
248             # Iterate through the names of each placemark.
249             #
250             name = attributes.find('.//{http://www.opengis.net/kml/2.2}name')
251             #
252             # If the name is the same as the defined tile, get coordinates,
253             # either the polygon or center point.
254             #
255             if name.text == tile and form == 'polygon':
256                 #
257                 # Find the polygon tag.
258                 #
259                 points = attributes.find('.//{http://www.opengis.net/kml/2.2}'
260                     'Polygon')
261                 #
262                 # Access and return the tile's polygon coordinates.
263                 #
264                 for unit in points:
265                     xyz = attributes.find('.//{http://www.opengis.net/kml/2.2}'
266                         'coordinates').text
267                     xyz = xyz.strip('\t\n\r')
268                     #
269                     # Original form: ['longitude,latitude,vertical', ... ]
270                     #
271                     coords = (xyz).split(' ')
272 
273                     point1 = coords[0].split(',')
274                     point2 = coords[1].split(',')
275                     point3 = coords[2].split(',')
276                     point4 = coords[3].split(',')
277                     point5 = coords[4].split(',')
278 
279                     return point1, point2, point3, point4, point5
280 
281 
282             elif name.text == tile and form == 'point':
283                 #
284                 # Find the center point tag.
285                 #
286                 points = attributes.find('.//{http://www.opengis.net/kml/2.2}'
287                     'Point')
288 
289                 for unit in points:
290                     #
291                     # Save the center point values as a list.
292                     # Form: ['longitude,latitude,vertical', ... ]
293                     #
294                     coords = (unit.text).split(',')
295 
296                     return coords
297 
298     if not coords:
299         print 'Tile not found. Try again.'
300         sys.exit(-1)
301 
302 
303 def validate_date(date_text):
304 
305     '''This function validates date argument input.'''
306 
307     try:
308         datetime.strptime(date_text, '%Y-%m-%d')
309     except ValueError:
310         raise ValueError('\nIncorrect date format, should be YYYY-MM-DD\n')
311 
312 
313 def create_query():
314 
315     '''This function creates a query string for the data hub.'''
316     #
317     # Make sure write_dir exists.
318     #
319     if options.write_dir is None:
320         #
321         # Create tempS2 folder in currend working directory.
322         #
323         cwd = os.getcwd()
```

```
324         options.write_dir = os.path.join(cwd, 'tempS2')
325
326     print '\nDirectory: {}\n'.format(options.write_dir)
327
328     prog = os.path.basename(sys.argv[0])
329     #
330     # Build in checks for valid commands related to the spatial aspect.
331     #
332     if options.tile is None or options.tile == '?':
333
334         if options.lat is None or options.lon is None:
335
336             if (options.latmin is None
337                     or options.lonmin is None
338                     or options.latmax is None
339                     or options.lonmax is None):
340
341                 geom = None
342
343             else:
344                 geom = 'rectangle'
345
346         else:
347             if (options.latmin is None
348                     and options.lonmin is None
349                     and options.latmax is None
350                     and options.lonmax is None):
351
352                 geom = 'point'
353
354             else:
355                 geom = None
356
357     else:
358         #
359         # Defines lat+lon based on tile's center point using API or kml file.
360         #
361         coords = kml_api(options.tile)
362         options.lon = coords[0]
363         options.lat = coords[1]
364
365         print '\nTile center point: {} lat, {} lon\n'.format(
366             options.lat, options.lon)
367
368         geom = 'point'
369     #
370     # Instantiate query string.
371     #
372     query = ''
373     #
374     # Create spatial parts of the query ::: point, rectangle or location name
375     # Beware of the quotation marks. For some reason double quotes alone are
376     # not registered by the data hub, so they seem to need to be done this way.
377     #
378     if geom == 'point':
379
380         query += '(footprint:\\"Intersects({} {})\\")'.format(
381             options.lon, options.lat)
382
383     elif geom == 'rectangle':
384
385         query += ('(footprint:\\"Intersects(POLYGON(({lonmin} {latmin}, '
386             '{lonmax} {latmin}, {lonmax} {latmax}, {lonmin} {latmax}, '
387             '{lonmin} {latmin})))\\")').format(latmin = options.latmin,
388             latmax = options.latmax, lonmin = options.lonmin,
389             lonmax = options.lonmax)
390
391     else:
392         pass
393     #
394     # Add orbit, if defined (default: NONE).
395     #
396     if options.rel_orbit:
397         query += ' AND (relativeorbitnumber:{})'.format(options.rel_orbit)
398     else:
399         pass
400     #
401     # Orbit direction as free text.
402     #
403     if options.orbit_dir == 'asc':
404         query += ' AND ASCENDING'
405     elif options.orbit_dir == 'desc':
```

```
406             query += ' AND DESCENDING'
407         else:
408             pass
409         #
410         # Add Sentinel mission.
411         #
412         if options.sentinel == 'S2':
413             query += ' AND (platformname:Sentinel-2)'
414         elif options.sentinel == 'S1':
415             query += ' AND (platformname:Sentinel-1)'
416         elif options.sentinel == 'S3':
417             query += ' AND (platformname:Sentinel-3)'
418         elif options.sentinel == 'S1A':
419             query += ' AND (platformname:Sentinel-1 AND filename:S1A_ )'
420         elif options.sentinel == 'S1B':
421             query += ' AND (platformname:Sentinel-1 AND filename:S1B_ )'
422         elif options.sentinel == 'S2A':
423             query += ' AND (platformname:Sentinel-2 AND filename:S2A_ )'
424         elif options.sentinel == 'S2B':
425             query += ' AND (platformname:Sentinel-2 AND filename:S2B_ )'
426         else:
427             pass
428         #
429         # Add Sentinel-1 specific query parameters, if S1 specific query.
430         #
431         if (options.sentinel == 'S1'
432                 or options.sentinel == 'S1A'
433                 or options.sentinel == 'S1B'):
434 
435             if options.s1product is not None:
436                 query += ' AND (producttype:{})'.format(options.s1product)
437             else:
438                 pass
439 
440             if options.s1polar is not None:
441                 query += ' AND (polarisationmode:{})'.format(options.s1polar)
442             else:
443                 pass
444 
445             if options.s1mode is not None:
446                 query += ' AND (sensoroperationalmode:{})'.format(options.s1mode)
447             else:
448                 pass
449         else:
450             pass
451         #
452         # Add Sentinel-2 specific query parameters, if S2 specific query.
453         #
454         if (options.sentinel == 'S2'
455                 or options.sentinel == 'S2A'
456                 or options.sentinel == 'S2B'):
457             #
458             # Add cloud cover query.
459             #
460             if options.max_cloud is not None:
461                 query += ' AND (cloudcoverpercentage:[0.0 TO {}])'.format(
462                     options.max_cloud)
463             else:
464                 pass
465 
466             if options.s2product is not None:
467                 query += ' AND (producttype:{})'.format(options.s2product)
468             else:
469                 pass
470 
471         else:
472             #
473             # Add tile query check.
474             #
475             if options.tile != None:
476                 print 'The tile option (-t) can only be used for Sentinel-2!'
477                 sys.exit(-1)
478         #
479         # Add dates of capture.
480         #
481         if options.start_date is not None:
482             #
483             # Default value is None, so must check user format.
484             #
485             validate_date(options.start_date)
486 
487         else:
```

```
488         pass
489
490     if options.end_date is not None:
491         #
492         # Default value is None, so must check user format.
493         #
494         validate_date(options.end_date)
495
496     else:
497         pass
498
499     if options.start_date is not None or options.end_date is not None:
500         #
501         # If only one is given, fill the other with today or S2 launch date.
502         #
503         if options.end_date is None:
504             options.end_date = date.today().isoformat()
505         if options.start_date is None:
506             options.start_date = '2015-06-23' # S2 launch date.
507
508         query += (' AND (beginPosition:[{0}T00:00:00.000Z TO {1}T23:59:59.999Z] '
509             'AND endPosition:[{0}T00:00:00.000Z TO {1}T23:59:59.999Z])').format(
510             options.start_date, options.end_date)
511     else:
512         pass
513     #
514     # Add dates of ingestion.
515     #
516     if options.start_ingest_date is not None:
517         #
518         # Default value is None, so must check user format.
519         #
520         validate_date(options.start_ingest_date)
521     else:
522         pass
523
524     if options.end_ingest_date is not None:
525         #
526         # Default value is None, so must check user format.
527         #
528         validate_date(options.end_ingest_date)
529     else:
530         pass
531
532
533     if options.auto is not None:
534         query += (' AND (ingestionDate:[ NOW-3DAYS TO NOW ])')
535     elif options.start_date is not None or options.end_date is not None:
536         #
537         # If only one is given, fill the other with today or S2 launch date.
538         #
539         if options.end_ingest_date is None:
540             options.end_ingest_date = date.today().isoformat()
541         if options.start_ingest_date is None:
542             options.start_ingest_date = '2015-06-23'  # S2 launch date.
543
544         query += (' AND (ingestionDate:[{}T00:00:00.000Z TO {}T23:59:59.999Z ])'
545             ).format(options.start_ingest_date, options.end_ingest_date)
546
547     else:
548         pass
549     #
550     # Sort results, if desired.
551     #
552     if options.orderby == 'id-asc':
553         options.orderby = '&orderby=ingestiondate asc'
554     elif options.orderby == 'id-desc':
555         options.orderby = '&orderby=ingestiondate desc'
556     elif options.orderby == 'd-asc':
557         options.orderby = '&orderby=beginposition asc'
558     elif options.orderby == 'd-desc':
559         options.orderby = '&orderby=beginposition desc'
560     else:
561         options.orderby = ''
562     #
563     # Correct query string if no geographic coordinates are given.
564     #
565     if query.startswith(' AND'):
566         query = query[5:]
567     return query
568
569
```

```
570 def start_session():
571     #
572     # Set data source (apihub vs dhus —— more could be added).
573     #
574     if options.hub is None:
575         huburl = 'https://scihub.copernicus.eu/apihub/'
576     elif options.hub == 'dhus':
577         huburl = 'https://scihub.copernicus.eu/dhus/'
578         #
579         # The dhus data hub has a limit of 10 records.
580         #
581         if int(options.MaxRecords) > 10:
582             print 'Max records changed to 10 due to dhus limit.'
583             options.MaxRecords = '10'
584 
585     elif options.hub == 'zamg':
586         huburl = 'https://data.sentinel.zamg.ac.at/'
587     #
588     # Use this part if you want to have your password and username saved in a
589     # textfile, with the file name as the command.
590     #
591     if options.auth is not None:
592         try:
593             f = file(options.auth)
594             (account,passwd) = f.readline().split(' ')
595 
596             if passwd.endswith('\n'):
597                 passwd=passwd[:-1]
598             f.close()
599 
600         except:
601             print 'Error with password file.'
602             url = '{}search?q='.format(huburl)
603             account = raw_input('Username: ')
604             passwd = raw_input('Password: ')
605     #
606     # Authenticate at data hub.
607     #
608     else:
609         url = '{}search?q='.format(huburl)
610         account = raw_input('Username: ')
611         passwd = raw_input('Password: ')
612     #
613     # Start session/authorization using requests module.
614     #
615     session = requests.Session()
616     session.auth = (account, passwd)
617 
618     return session, huburl, account, passwd
619 
620 
621 def set_wget_var():
622     #
623     # Set wget query variables used throughout the script.
624     #
625     url_search = '{}search?q='.format(huburl)
626     # Don't validate server's certificate and don't wait for authorization
627     # challenge from server.
628     wg = 'wget --quiet --no-check-certificate --auth-no-challenge'
629     auth = '--http-user="{}" --http-password="{}"'.format(account, passwd)
630     search_output = ' --output-document=query_results.xml'
631     wg_opt = ' --continue --tries=20 --read-timeout=60 --output-document='
632     value='\\$value'
633 
634     return url_search, wg, auth, search_output, wg_opt, value
635 
636 
637 def get_query_xml():
638     #
639     # Initialize command variable.
640     #
641     command_wget = None
642     #
643     # Check for existing xml query file and delete if present.
644     #
645     if os.path.exists('query_results.xml'):
646         os.remove('query_results.xml')
647     #
648     # Execute command to download and save query as xml-file in
649     # the same location as the python script.
650     #
651     command_wget = '{} {} {} "{}{}&rows={}"'.format(
```

```
652             wg, auth, search_output, url_search, query, options.MaxRecords,
653             options.orderby)
654
655         print command_wget
656
657         try:
658             os.system(command_wget)
659         except Exception as e:
660             logging.error(str(e) + " in getting " + query +
661                 " from " + command_wget)
662         #
663         # Parse the xml query file. The entry tag contains the results.
664         #
665         tree = etree.parse('query_results.xml')
666         entries = tree.findall('{http://www.w3.org/2005/Atom}entry')
667         #
668         # Save the number of scenes to a variable.
669         #
670         scenes = str(len(entries))
671         #
672         # Initialize variable to return total file size of results.
673         #
674         total_size = 0
675         scenes_have = 0
676
677         for entry in range(len(entries)):
678             #
679             # The UUID element is the key for creating the path to the file.
680             #
681             uuid_element = (entries[entry].find('{http://www.w3.org/2005/Atom}'
682                 'id')).text
683             sentinel_link = ("{}odata/v1/Products('{}')/{}").format(
684                 huburl, uuid_element, value)
685             #
686             # The title element contains the corresponding file name.
687             #
688             title_element = (entries[entry].find('{http://www.w3.org/2005/Atom}'
689                 'title')).text
690             summary_element = (entries[entry].find('{http://www.w3.org/2005/Atom}'
691                 'summary')).text
692             filename = (entries[entry].find('.//[@name="filename"]')).text
693             #
694             # Print each entry's info.
695             #
696             print '\n----------------------------------------------------------------'
697             print 'Scene {} of {}'.format(entry + 1, len(entries))
698             print title_element
699             print summary_element
700             #
701             # Return tile names per entry using function return_tiles if desired.
702             #
703             if options.tile == '?' and filename.startswith('S2A_OPER_'):
704
705                 found_tiles = return_tiles(uuid_element, filename)
706                 #
707                 # Print the number of tiles and their names.
708                 #
709                 print '# of Tiles: {}'.format(str(len(found_tiles[0])))
710                 print 'Tiles:{}'.format(found_tiles[1])
711
712             if options.tile == '?' and (filename.startswith('S2A_MSIL')
713                     or filename.startswith('S2B_MSIL')):
714                 #
715                 # Print the number of tiles and their names.
716                 #
717                 print '# of Tiles: 1'
718                 print 'Tile: {}'.format(filename[-26:-21])
719             #
720             # Find cloud cover percentage of Sentinel-2 images.
721             #
722             if (options.sentinel == 'S2'
723                     or options.sentinel == 'S2A'
724                     or options.sentinel == 'S2B'):
725
726                 cloud_element = (entries[entry].find(
727                     './/[@name="cloudcoverpercentage"]')).text
728
729                 print 'Cloud cover percentage: {}'.format(cloud_element)
730             #
731             # Print download link from the server.
732             #
733             print sentinel_link
```

```
734         #
735         # Return the file size of the entry.
736         #
737         size_element = (entries[entry].find('.//[@name="size"]')).text
738         #
739         # Parse size to float and add to running total of size.
740         #
741         if 'GB' in size_element:
742             size_element = size_element.replace(' GB', '')
743             size_element = float(size_element)
744             total_size += size_element
745 
746         elif 'MB' in size_element:
747             size_element = size_element.replace(' MB', '')
748             size_element = float(size_element) / 1024
749             size_element = size_element
750             total_size += size_element
751         #
752         # Check if file was already downloaded.
753         #
754         check = download_check(options.write_dir, title_element, filename)
755 
756         if check is True:
757             scenes_have += 1
758     #
759     # Turn the total size of all scenes found back into text.
760     #
761     total_size = '{0:.2f} GB'.format(total_size)
762     #
763     # Create question to continue based on number of scenes found.
764     #
765     if options.tile is None or options.tile == '?':
766         question_tile = 'Download all results you do not already have?'
767     elif options.tile is not None:
768         question_tile = ('Do you want to download only {} tiles selected '
769             'from the results you do not already have?').format(options.tile)
770 
771     question = ('\n\nTotal number of scenes found: {0}'
772         '\nYou already have {1} of {0} total scenes found.'
773         '\nTotal size of all scenes: {2}'
774         '\n\n{3}').format(scenes, scenes_have, total_size, question_tile)
775 
776     print question
777 
778     ins = None
779 
780     if options.auto is not None:
781         ins = 'y'
782     else:
783 
784         while True:
785             ins = raw_input('Answer [y/n]: ')
786 
787             if (ins == 'y' or ins == 'Y' or ins == 'yes' or ins == 'Yes'
788                     or ins == 'n' or ins == 'N' or ins == 'no' or ins == 'No'):
789 
790                 break
791 
792             else:
793                 print("Your input should indicate yes or no.")
794 
795     if ins == 'y' or ins == 'Y' or ins == 'yes' or ins == 'Yes':
796         bool_answer = True
797     else:
798         bool_answer = None
799 
800     return bool_answer, entries
801 
802 
803 def return_tiles(uuid_element, filename, tile=''):
804 
805     '''Function returns tiles incldued in the GRANULE folder of a product,
806     including the entire file name of one desired tile, if specified.'''
807     #
808     # Create link to search for tile/granule data.
809     #
810     granule_link = ("{}odata/v1/Products"
811         "('{}')/Nodes('{}')/Nodes('GRANULE')/Nodes").format(
812         huburl, uuid_element, filename)
813     #
814     # Create GET request from hub and parse it.
815     #
```

```
816     response = session.get(granule_link, stream=True)
817     granule_tree = etree.fromstring(response.content)
818     #
819     # Search for all entires (i.e. tiles)
820     #
821     granule_entries = granule_tree.findall(
822         '{http://www.w3.org/2005/Atom}entry')
823     #
824     # Empty string to fill with all tiles in the file.
825     #
826     granules = ''
827     #
828     # Go through each tile appending each name to string.
829     #
830     for granule_entry in range(len(granule_entries)):
831         #
832         # UUID element creates the path to the file.
833         #
834         granule_dir_name = (granule_entries[granule_entry].find(
835             '{http://www.w3.org/2005/Atom}title')).text
836         granule = granule_dir_name[50:55]
837         granules += ' {}'.format(granule)
838         #
839         # If one tile is given as optional arg, return entire tile file name.
840         #
841         if tile != '':
842             if tile in granule_dir_name:
843                 granule_file = granule_dir_name
844         else:
845             granule_file = ''
846     #
847     # Return the number of granules and their names, or just the individual
848     # tile file name if a specific tile was asked for.
849     #
850     if not granule_file:
851         return(granule_entries, granules)
852     else:
853         return(granule_file)
854
855
856 def download_check(write_dir, title_element, filename):
857
858     '''Function checks if files have aleady been downloaded. If yes, but
859        unzipped, it unzips them and deletes the zipped folder.'''
860
861     #
862     # Possible zipped folder name and paths.
863     #
864     zfile = '{}.zip'.format(title_element)
865     unzipped_path = os.path.join(write_dir, filename)
866     zipped_path = os.path.join(write_dir, zfile)
867
868     #
869     # Check if file was already downloaded.
870     #
871     if os.path.exists(os.path.join(write_dir, title_element)):
872         print '{} already exists in unzipped form!'.format(title_element)
873         return True
874
875     elif os.path.exists(unzipped_path):
876         print '{} already exists in unzipped form!'.format(filename)
877         return True
878
879     elif os.path.exists(zipped_path):
880         try:
881             with zipfile.ZipFile(zipped_path) as z:
882                 #
883                 # Test again to see if zipfile is corrupt, if so, remove.
884                 #
885                 ret = z.testzip()
886
887                 if ret is not None:
888                     os.remove(zipped_path)
889                     return False
890                 #
891                 # Otherwise extract and remove.
892                 #
893                 else:
894                     print ('{} has already been successfully downloaded!'
895                         ).format(zfile)
896                     z.extractall(u'{}'.format(write_dir))
897                     print '\tAnd is now unzipped.'
```

```
                    os.remove(zipped_path)

                    return True

        except zipfile.BadZipfile:
            #print 'Zipfile corrupt or hub might have a problem.'
            os.remove(zipped_path)
            return False

    else:
        return False


def return_header(uuid_element, filename):

    '''Function returns name of header xml incldued in a product.'''

    #
    # Create link to search for tile/granule data.
    #
    safe_link = ("{}odata/v1/Products"
        "('{}')/Nodes('{}')/Nodes").format(
        huburl, uuid_element, filename)
    #
    # Create GET request from hub and essentially parse it.
    #
    response = session.get(safe_link, stream=True)
    safe_tree = etree.fromstring(response.content)
    #
    # Search for all entires.
    #
    safe_entries = safe_tree.findall('{http://www.w3.org/2005/Atom}entry')
    #
    # Go through each entry in the safe folder and return header xml name.
    #
    for safe_entry in range(len(safe_entries)):
        #
        # UUID element creates the path to the file.
        #
        safe_name = (safe_entries[safe_entry].find(
            '{http://www.w3.org/2005/Atom}title')).text
        if 'SAFL1C' in safe_name:

            header_xml = safe_name

            return header_xml

    if not header_xml:
        print 'Header xml could not be located!'


def make_dir(location, filename):

    '''Creates a directory in another directory if it doesn't already exist.'''

    dir_name = '{}/{}'.format(location, filename)

    if not os.path.exists(dir_name):
        os.mkdir(dir_name)

    return dir_name


def get_tile_files(uuid_element, filename, tile_file, tile_dir):

    '''Creates structure for tile specific download (tile inside GRANULE
       folder), and fills it.'''
    #
    # Initialize command variable.
    #
    command_wget = None
    #
    # Define link to tile folder in data hub.
    #
    tile_folder_link = ("{}odata/v1/Products"
        "('{}')/Nodes('{}')/Nodes('GRANULE')/Nodes('{}')/Nodes").format(
        huburl, uuid_element, filename, tile_file)
    #
    # Connect to the server and stream the metadata as a string, parsing it.
    #
    response = session.get(tile_folder_link, stream=True)
    tile_folder_tree = etree.fromstring(response.content)
```

```
980     #
981     # Search for all entires
982     #
983     tile_folder_entries = (tile_folder_tree.findall(
984         '{http://www.w3.org/2005/Atom}entry'))
985     #
986     # Go through each entry and identify necessary information for download.
987     #
988     for tile_folder_entry in range(len(tile_folder_entries)):
989
990         tile_entry_title = (tile_folder_entries[tile_folder_entry].find(
991             '{http://www.w3.org/2005/Atom}title')).text
992
993         tile_entry_id = (tile_folder_entries[tile_folder_entry].find(
994             '{http://www.w3.org/2005/Atom}id')).text
995
996         print 'Downloading: {}'.format(tile_entry_title)
997         #
998         # Download xml file
999         #
1000        if '.xml' in tile_entry_title:
1001
1002            tile_xml_file = tile_entry_title
1003            tile_xml_link = '{}/{}'.format(tile_entry_id, value)
1004            command_wget = '{} {} {}{}/{} "{}"'.format(wg, auth, wg_opt,
1005                tile_dir, tile_xml_file, tile_xml_link)
1006
1007            try:
1008                os.system(command_wget)
1009            except Exception as e:
1010                logging.error(str(e) + " in getting " + tile_xml_file +
1011                    " from " + tile_xml_link + " in " + tile_dir)
1012                continue
1013        else:
1014            #
1015            # Create folder for files and go get them
1016            #
1017            inside_folder_dir = make_dir(tile_dir, tile_entry_title)
1018            get_inside_files(inside_folder_dir, tile_entry_id)
1019
1020
1021 def get_inside_files(inside_folder_dir, tile_entry_id):
1022
1023     '''Go deeper in the element tree and download contents to the specified
1024        folder. This is relevant for tile specific downloads in the old file
1025        structure, pre-06.12.16.'''
1026     #
1027     # Initialize command variables.
1028     #
1029     command_wget = None
1030     #
1031     # Get xml link and connect to server, parsing response as a string.
1032     #
1033     inside_folder_link = '{}/Nodes'.format(tile_entry_id)
1034     resp = session.get(inside_folder_link, stream=True)
1035     inside_folder_tree = etree.fromstring(resp.content)
1036     #
1037     # Search for all entires
1038     #
1039     inside_folder_entries = (inside_folder_tree.findall(
1040         '{http://www.w3.org/2005/Atom}entry'))
1041     #
1042     # Download each entry saving in the defined directory.
1043     #
1044     for inside_folder_entry in range(len(inside_folder_entries)):
1045
1046         inside_entry_title = (inside_folder_entries[inside_folder_entry].find(
1047             '{http://www.w3.org/2005/Atom}title')).text
1048         inside_entry_id = (inside_folder_entries[inside_folder_entry].find(
1049             '{http://www.w3.org/2005/Atom}id')).text
1050         inside_entry_file = inside_entry_title
1051         inside_entry_link = '{}/{}'.format(inside_entry_id, value)
1052
1053         command_wget = '{} {} {}{}/{} "{}"'.format(wg, auth, wg_opt,
1054             inside_folder_dir, inside_entry_file, inside_entry_link)
1055
1056         try:
1057             os.system(command_wget)
1058         except Exception as e:
1059             logging.error(str(e) + " in getting " + inside_entry_file +
1060                 " from " + inside_entry_link + " in " + inside_folder_dir)
1061             continue
```

```
1062
1063
1064  def download_results(entries):
1065      #
1066      # Initialize command variables.
1067      #
1068      command_wget = None
1069      #
1070      # Create download directory if not already existing (default = ./tempS2)
1071      #
1072      if not(os.path.exists(options.write_dir)):
1073              os.mkdir(options.write_dir)
1074      #
1075      # If you want to download all entries and did not search for a
1076      # specific tile, then downloading will begin.
1077      #
1078      if (options.tile is None or options.tile == '?'):
1079          #
1080          # Download all whole scenes matching the query.
1081          #
1082          for entry in range(len(entries)):
1083              #
1084              # Create download command for the entry.
1085              #
1086              uuid_element = (entries[entry].find(
1087                  '{http://www.w3.org/2005/Atom}id')).text
1088              sentinel_link = ("{}odata/v1/Products('{}')/{}").format(
1089                  huburl, uuid_element, value)
1090              filename = (entries[entry].find('.//[@name="filename"]')).text
1091              title_element = (entries[entry].find(
1092                  '{http://www.w3.org/2005/Atom}title')).text
1093              zfile = '{}.zip'.format(title_element)
1094              #
1095              # Skip files that have already been downloaded.
1096              #
1097              check = download_check(options.write_dir, title_element, filename)
1098
1099              if check is True:
1100                  continue
1101              else:
1102                  #
1103                  # Save to defined directory (default = ./tempS2)
1104                  #
1105                  command_wget = '{} {} {}{}/{} "{}"'.format(wg, auth, wg_opt,
1106                      options.write_dir, zfile, sentinel_link)
1107                  #
1108                  # Execute download.
1109                  #
1110                  try:
1111                      os.system(command_wget)
1112                  except Exception as e:
1113                      logging.error(str(e) + " in getting " + zfile +
1114                          " from " + sentinel_link + " in " + options.write_dir)
1115                      continue
1116
1117                  print 'Downloaded Scene #{}: {}'.format(str(entry + 1), zfile)
1118                  #
1119                  # Unzip.
1120                  #
1121                  unzipped_path = os.path.join(options.write_dir, filename)
1122                  zipped_path = os.path.join(options.write_dir, zfile)
1123
1124                  try:
1125                      with zipfile.ZipFile(zipped_path) as z:
1126
1127                          z.extractall(u'{}'.format(options.write_dir))
1128                          print 'Unzipped Scene #{}'.format(str(entry + 1))
1129
1130                  except zipfile.BadZipfile:
1131
1132                      print 'Zipfile corrupt or hub might have a problem.'
1133                      continue
1134                  #
1135                  # If the unzipped and zipped version exist,
1136                  # delete the zipped version.
1137                  #
1138                  if (os.path.exists(unzipped_path)
1139                          and os.path.exists(zipped_path)):
1140
1141                      os.remove(zipped_path)
1142
1143          print '\n----------------------------------------------------------------'
```

```
1144            print 'Downloading complete!'
1145            print '————————————————————————————————————————————————————————————\n'
1146        #
1147        # If you want to download a tile that you searched for, then it will
1148        # create the proper file structure mimicing a complete download and fill
1149        # it with data specific to the tile you want, or, post 06.12.16,
1150        # simply downloadcomplete matching tile packages.
1151        #
1152        elif options.tile is not None and options.tile is not '?':
1153
1154            #
1155            # Search through entries for matching tiles.
1156            #
1157            for entry in range(len(entries)):
1158
1159                #
1160                # Create download command for the entry.
1161                #
1162                uuid_element = (entries[entry].find(
1163                    '{http://www.w3.org/2005/Atom}id')).text
1164                filename = (entries[entry].find(
1165                    './/[@name="filename"]')).text
1166                title_element = (entries[entry].find(
1167                    '{http://www.w3.org/2005/Atom}title')).text
1168                sentinel_link = (
1169                    "{}odata/v1/Products('{}')/Nodes('{}')/Nodes").format(
1170                    huburl, uuid_element, filename)
1171
1172                if filename.startswith('S2A_OPER_'):
1173                    #
1174                    # Find tiles in entry, returning number[0] and tile names[1]
1175                    #
1176                    included_tiles = return_tiles(uuid_element, filename)
1177
1178                elif (filename.startswith('S2A_MSIL')
1179                        or filename.startswith('S2B_MSIL')):
1180                    included_tiles = [filename[-26:-21], filename[-26:-21]]
1181                #
1182                # If the tile you want is in the entry, then it will create the
1183                # necessary file structure and fill it.
1184                #
1185                if (options.tile in included_tiles[1]
1186                        and filename.startswith('S2A_OPER_')):
1187
1188                # File structure————————————————————————————————————————
1189
1190                    product_dir_name = make_dir(options.write_dir, filename)
1191                    #
1192                    # Create GRANULE directory in product directory.
1193                    #
1194                    granule_dir = make_dir(product_dir_name, 'GRANULE')
1195                    #
1196                    # Create tile directory in GRANULE based on tile file name.
1197                    #
1198                    tile_file = return_tiles(uuid_element, filename, options.tile)
1199                    #
1200                    # If tile folder already exists, then it skips downloading.
1201                    #
1202                    if os.path.exists(os.path.join(granule_dir, tile_file)):
1203                        print 'Tile Folder already downloaded.'
1204                        continue
1205
1206                    tile_dir = make_dir(granule_dir, tile_file)
1207
1208                # Downloads————————————————————————————————————————————
1209
1210                    print 'Downloading from scene #{}'.format(str(entry + 1))
1211
1212                    #
1213                    # Download the product header file after finding the name
1214                    #
1215                    header_file = return_header(uuid_element, filename)
1216                    header_link = "{}('{}')/{}".format(
1217                        sentinel_link, header_file, value)
1218
1219                    command_wget = '{} {} {}{}/{} "{}"'.format(wg, auth, wg_opt,
1220                        product_dir_name, header_file, header_link)
1221
1222                    try:
1223                        os.system(command_wget)
1224                    except Exception as e:
1225                        logging.error(str(e) + " in getting " + header_file +
```

```
1226                        " from " + header_link + " in " + product_dir_name)
1227                #
1228                # Download INSPIRE.xml
1229                #
1230                inspire_file = 'INSPIRE.xml'
1231                inspire_link = "{}('{}')/{}".format(
1232                    sentinel_link, inspire_file, value)
1233                command_wget = '{} {} {}{}/{} "{}"'.format(wg, auth, wg_opt,
1234                    product_dir_name, inspire_file, inspire_link)
1235
1236                try:
1237                    os.system(command_wget)
1238                except Exception as e:
1239                    logging.error(str(e) + " in getting " + inspire_file +
1240                        " from " + inspire_link + " in " + product_dir_name)
1241                #
1242                # Download manifest.safe
1243                #
1244                manifest_file = 'manifest.safe'
1245                manifest_link = "{}('{}')/{}".format(
1246                    sentinel_link, manifest_file, value)
1247                command_wget = '{} {} {}{}/{} "{}"'.format(wg, auth, wg_opt,
1248                    product_dir_name, manifest_file, manifest_link)
1249
1250                try:
1251                    os.system(command_wget)
1252                except Exception as e:
1253                    logging.error(str(e) + " in getting " + manifest_file +
1254                        " from " + manifest_link + " in " + product_dir_name)
1255                #
1256                # Download tile xml file and create AUX_DATA, IMG_DATA and
1257                # QI_DATA folders in tile folder and download their contents.
1258                #
1259                get_tile_files(uuid_element, filename, tile_file, tile_dir)
1260
1261                print 'Downloaded tile {} from scene #{}\n{}'.format(
1262                    options.tile, str(entry + 1), product_dir_name)
1263
1264            elif (options.tile in included_tiles
1265                    and (filename.startswith('S2A_MSIL')
1266                    or filename.startswith('S2B_MSIL'))):
1267                #
1268                # Create download command for the entry.
1269                #
1270                sentinel_link = ("{}odata/v1/Products('{}')/{}").format(
1271                    huburl, uuid_element, value)
1272                zfile = '{}.zip'.format(title_element)
1273                #
1274                # Skip files that have already been downloaded.
1275                #
1276                check = download_check(
1277                    options.write_dir, title_element, filename)
1278
1279                if check is True:
1280                    continue
1281                else:
1282                    #
1283                    # Save to defined directory (default = ./tempS2)
1284                    #
1285                    command_wget = '{} {} {}{}/{} "{}"'.format(wg, auth,
1286                         wg_opt, options.write_dir, zfile, sentinel_link)
1287                    #
1288                    # Execute download.
1289                    #
1290                    try:
1291                        os.system(command_wget)
1292                    except Exception as e:
1293                        logging.error(str(e) + " in getting " + zfile +
1294                            " from " + sentinel_link +
1295                            " in " + options.write_dir)
1296                        continue
1297
1298                    print 'Downloaded Scene #{}: {}'.format(
1299                        str(entry + 1), zfile)
1300                    #
1301                    # Unzip the downloaded file.
1302                    #
1303                    unzipped_path = os.path.join(options.write_dir, filename)
1304                    zipped_path = os.path.join(options.write_dir, zfile)
1305
1306                    try:
1307                        with zipfile.ZipFile(zipped_path) as z:
```

```
1308
1309                            if (sys.platform.startswith('linux')
1310                                    or sys.platform.startswith('darwin')):
1311                                z.extractall(u'{}'.format(options.write_dir))
1312                            else:
1313                                z.extractall(u'\\\\?\\{}'.format(
1314                                    options.write_dir))
1315                            print 'Unzipped Scene #{}'.format(str(entry + 1))
1316
1317                        except zipfile.BadZipfile:
1318                            print 'Zipfile corrupt or problem with hub.'
1319
1320                        #
1321                        # If the unzipped and zipped version exist,
1322                        # delete the zipped version.
1323                        #
1324                        if (os.path.exists(unzipped_path)
1325                                and os.path.exists(zipped_path)):
1326
1327                            os.remove(zipped_path)
1328
1329                        else:
1330                            print '\nTile {} not in scene #{}\n'.format(
1331                                options.tile, str(entry + 1))
1332
1333            print '\n------------------------------------------------------------------'
1334            print 'Downloading complete!'
1335            print '------------------------------------------------------------------\n'
1336
1337
1338    if __name__ == '__main__':
1339        #
1340        # Set-up logger.
1341        #
1342        logging.basicConfig(filename='log/collector.log',
1343                            format='%(asctime)s:%(levelname)s:%(message)s',
1344                            level=logging.DEBUG)
1345        #
1346        # Parse command line to get global arguments.
1347        #
1348        options = get_args()
1349        #
1350        # Create hub query.
1351        #
1352        query = create_query()
1353        #
1354        # Create authenticated http session.
1355        #
1356        session, huburl, account, passwd = start_session()
1357        #
1358        # Set wget query variables used throughout the script.
1359        #
1360        url_search, wg, auth, search_output, wg_opt, value = set_wget_var()
1361        #
1362        # Query hub, print results and ask whether to continue.
1363        #
1364        download_bool, entries = get_query_xml()
1365
1366        if download_bool:
1367            download_results(entries)
1368        else:
1369            #
1370            # You decided not to download this time in the message box.
1371            #
1372            print '\n------------------------------------------------------------------'
1373            print 'Nothing downloaded, but xml file saved!'
1374            print '------------------------------------------------------------------\n'
```

This script stacks the 6 Sentinel-2 bands required from SIAM™, but first resamples bands 11 and 12 to 10m, identifies pixels with a value of 0 to create a no-data mask, converts to 8-bit and saves in ENVI format. While most lines fit within the 79 character style guideline, a few longer lines were broken without testing for this printed version. Unnecessary spaces between lines were also removed.

Listing A.3: Convert data structure and format for SIAM™ (`conversion_linux.py`)

```
# ------------------------------------------------------------------------------
# Name:         Sentinel2 'Conversion' for SIAM.
# Purpose:      Use NumPy, GDAL and SciPy to convert 6 Sentinel2 bands to
#               8-bit, resample bands 11 and 12 to 10m pixels and build a 6-band
#               stack in the ENVI format (i.e. including .hdr). It also creates
#               a single band ENVI .dat/.hdr file with a constant value of 110
#               as to avoid thermal rules in SIAM.
#               This script is based on an ArcPy Python toolbox developed by
#               Dirk Tiede.
#
# Author:       h.Augustin
#
# Created:      14.12.2016
#
# ------------------------------------------------------------------------------
# #;FROM Andrea Baraldi:
# #;    OBJECTIVE: Radiometric calibration of Sentinel-2A/2B imagery into
# #;        (i)  TOP-OF-ATMOSPHERE (TOA, PLANETARY, EXOATMOSPHERIC)
# #;             reflectance (in range [0,1]), byte-coded,
# #;             i.e., scaled into range {1, 255}, output ENVI file format:
# #;             ...calrefbyt_lndstlk, band sequential (BSQ).
# #;             Equivalent to Landsat bands 1, 2, 3, 4, 5 and 7 are
# #;             the Sentinel-2A/2B bands    2, 3, 4, 8, 11 and 12
# #;             with spatial resolutions   10, 10, 10, 10, 20, 20.
# #;        (ii) faked temperature in kelvin degrees, equivalent to
# #;             10 degree Celsius,output value = 110, output
# #;             ENVI file format: ...caltembyt_lndstlk.
# #;
# #;        where:
# #;            - Sentinel-2A/2B bands are:
# #;
# #;             1, Aerosols (nm): 443?20/2,       Spatial resolution (in m): 60
# #;             2: Vis B (like TM1), 490?65/2,    Spatial resolution (in m): 10
# #;             3: Vis G (like TM2), 560?35/2,    Spatial resolution (in m): 10
# #;             4: Vis R (like TM3), 665?30/2,    Spatial resolution (in m): 10
# #;             5: NIR1 (Red Edge1), 705?15/2,    Spatial resolution (in m): 20
# #;             6: NIR2 (Red Edge2), 740?15/2,    Spatial resolution (in m): 20
# #;             7: NIR3 (Red Edge3),783?20/2,     Spatial resolution (in m): 20
# #;             8: NIR4 (like TM4), 842?115/2,    Spatial resolution (in m): 10
# #;             8a: NIR5, 865?20/2,               Spatial resolution (in m): 20
# #;             9, Water vapour: 945?20/2,        Spatial resolution (in m): 60
# #;             10, Cirrus: 1375?30/2,            Spatial resolution (in m): 60
# #;             11: MIR1 (like TM5) 1610?90/2,    Spatial resolution (in m): 20
# #;             12: MIR2 (like TM7) 2190?180/2    Spatial resolution (in m): 20
# #;
# #;             Hence, equivalent to Landsat bands 1, 2, 3, 4, 5 and 7 are
# #;             the Sentinel-2A/2B bands           2, 3, 4, 8, 11 and 12
# #;             with spatial resolutions          10, 10, 10, 10, 20, 20.

#! /usr/bin/env python
# -- coding: iso-8859-1 --

import os
import sys
import shutil
import datetime
import fnmatch
import argparse
import logging
import xml.etree.ElementTree as etree

import gdal
import numpy
import scipy.ndimage

################################################################################


def get_args():

    '''
    Gets arguments from command line.
    '''

    #
    # Create download tool help response.
    #
```

```
78      prog = os.path.basename(sys.argv[0])
79
80      if len(sys.argv) == 1:
81
82          print(
83              '\n        {0} [options]'
84              '\n        Help: {1} --help'
85              '\n        or: {1} -h'
86              '\nexample python {0} -r /path/to/data/'
87              ).format(sys.argv[0], prog)
88
89          sys.exit(-1)
90
91      else:
92
93          parser = argparse.ArgumentParser(
94              prog=prog,
95              usage='%(prog)s [options]',
96              description='Sentinel data converter.',
97              argument_default=None,
98              epilog='Go get \'em!')
99
100         #
101         # Arguments.
102         #
103         parser.add_argument(
104             '-r', '--read_dir', dest='read_dir', action='store', type=str,
105             help='Path where downloaded products are located.', default=None)
106         parser.add_argument(
107             '--auto', dest='auto', action='store',
108             help=('No user input necessary -- automatically converts all '
109                   'previously not converted images files.'),
110             default=None)
111
112         return parser.parse_args()
113
114
115 def nodata_array(tile_bands, PROC_DATA):
116
117     '''
118     This function creates a noData mask array based on all pixels that have
119     a value of 0 in any of the original Sentinel-2 bands used to create the
120     6 band .dat SIAM input file, and saves a copy as  ' nodata.dat'.
121     These correspond to S2 bands: 2, 3, 4, 8, 10 and 11.
122     '''
123
124     #
125     # Create array with same projection, etc.
126     #
127     noData = gdal.Open(tile_bands[0], gdal.GA_ReadOnly)
128     noData_array = (noData.GetRasterBand(1)).ReadAsArray()
129     noData_array = numpy.where((noData_array > 0), (1), noData_array)
130
131     #
132     # Establish size of raster from B02 for nodata output file.
133     #
134     projection = noData.GetProjection()
135     transform = noData.GetGeoTransform()
136     img_rows = noData.RasterYSize
137     img_cols = noData.RasterXSize
138
139     #
140     # Open output format driver, see gdal_translate --formats for list.
141     #
142     gdal_format = 'ENVI'
143     driver = gdal.GetDriverByName(gdal_format)
144
145     #
146     # Test nodata mask file path.
147     #
148     band_basename = os.path.basename(tile_bands[0])
149     nodata_file = '{}nodata.dat'.format(
150         os.path.basename(band_basename[:-7]))
151     filepath = os.path.join(PROC_DATA, nodata_file)
152
153     #
154     # Print driver for nodata mask (1 band, 8-bit unsigned).
155     #
156     outDs = driver.Create(filepath, img_cols, img_rows, 1,
157                           gdal.GDT_Byte)
158     if outDs is None:
159         print 'Could not create test file.'
```

```
160             sys.exit(1)
161
162         #
163         # Georeference the nodata .dat file and set the projection.
164         #
165         outDs.SetGeoTransform(transform)
166         outDs.SetProjection(projection)
167
168         for band in tile_bands:
169
170             #
171             # Open the band as read only.
172             #
173             img = gdal.Open(band, gdal.GA_ReadOnly)
174             band_id = band[-6:-4]
175             if img is None:
176                 print 'Could not open band #{}'.format(band_id)
177                 sys.exit(1)
178             print 'Processing noData for band #{}'.format(band_id)
179
180             #
181             # Cycle through bands, removing noData.
182             #
183             band_array = (img.GetRasterBand(1)).ReadAsArray()
184
185             #
186             # Resample bands 11 and 12 from 20m to 10m resolution.
187             #
188             if band.endswith(('_B11.jp2', '_B12.jp2')):
189                 band_array = scipy.ndimage.zoom(band_array, 2, order=0)
190
191             #
192             # Adjust output layer to 0 where there is nodata.
193             #
194             noData_array = numpy.where((band_array == 0), (0), noData_array)
195
196         #
197         # Invert array to mask, where 0 are values to be processed and 1 is nodata.
198         #
199         noData_array = numpy.where((noData_array == 0), (1), (0))
200
201         #
202         # Write the data to the designated band.
203         #
204         outBand = outDs.GetRasterBand(1)
205         outBand.WriteArray(noData_array, 0, 0)
206
207         #
208         # Flush data to disk.
209         #
210         outBand.FlushCache()
211
212         #
213         # Calculate statistics.
214         #
215         stats = outBand.ComputeStatistics(False)
216         outBand.SetStatistics(stats[0], stats[1], stats[2], stats[3])
217
218         del gdal_format
219         del outDs
220         del outBand
221         del noData
222         img = None
223         band_id = None
224         band_array = None
225         outBand = None
226         stats = None
227
228         return noData_array
229
230
231     def check_imgFolders(options_in):
232
233         #
234         # Create list for IMG_DATA folder and existing PROC_DATA folder paths.
235         #
236         imgFolders = []
237         procFolders = []
238
239         for dirpath, dirnames, filenames in os.walk(
240                 options_in.read_dir, topdown=True):
241
```

```
242         for dirname in dirnames:
243
244             if dirname == 'IMG_DATA':
245
246                 imgFolders.append(os.path.join(dirpath, dirname))
247
248             elif dirname == 'PROC_DATA':
249
250                 procFolder = os.path.join(dirpath, dirname)
251
252                 procFolders.append(procFolder)
253     #
254     # Determine which tile folders have no PROC_DATA folder.
255     #
256     unprocFolders = []
257
258     for imgFolder in imgFolders:
259
260         test_path = os.path.join(os.path.dirname(imgFolder), 'PROC_DATA')
261
262         if test_path in procFolders:
263             continue
264         else:
265             unprocFolders.append(imgFolder)
266
267     #
268     # Check validity of relevant PROC_DATA contents.
269     #
270     for procFolder in procFolders:
271
272         #
273         # Initialize variables for PROC_DATA folder.
274         #
275         caltembyt_path = None
276         caltembyt_size = 0
277         calrefbyt_path = None
278         calrefbyt_size = 0
279         remove_procFolder = None
280
281         for filename in os.listdir(procFolder):
282
283             if filename.endswith('caltembyt_lndstlk.dat'):
284
285                 caltembyt_path = os.path.join(procFolder, filename)
286                 caltembyt_size = os.path.getsize(caltembyt_path)
287
288                 if caltembyt_size < 5:
289                     remove_procFolder = True
290                     logger.info('caltembyt file error: ' + caltembyt_path)
291
292             elif filename.endswith('calrefbyt_lndstlk.dat'):
293
294                 calrefbyt_path = os.path.join(procFolder, filename)
295                 calrefbyt_size = os.path.getsize(calrefbyt_path)
296
297                 if calrefbyt_size < 5:
298                     remove_procFolder = True
299                     logger.info('calrefbyt file error: ' + calrefbyt_path)
300
301         #
302         # Removes PROC_DATA folders with problem files and adds to list to be
303         # processed.
304         #
305         if remove_procFolder is True:
306             shutil.rmtree(procFolder)
307             logger.info('Removed Folder: ' + procFolder)
308             unprocFolders.append(procFolder)
309
310     #
311     # Create the content of the popup window.
312     #
313     question = (
314         'Number of tiles found: {}'
315         '\n\nDo you want to process all unprocessed folders [{}]?'
316         ).format(len(imgFolders), len(unprocFolders))
317
318     print question
319
320     ins = None
321     bool_ans = None
322
323     if options_in.auto is not None:
```

```
324
325         ins = 'y'
326
327     else:
328
329         while True:
330
331             ins = raw_input('Answer [y/n]: ')
332
333             if ins == 'y' or ins == 'n':
334                 break
335             else:
336                 print 'Your input should indicate yes [y] or no [n].'
337
338     if ins == 'y' or ins == 'Y' or ins == 'yes' or ins == 'Yes':
339
340         bool_ans = True
341
342     return bool_ans, unprocFolders
343
344
345 def convert_imgs(root_folder, imgFolders):
346
347     start_time = datetime.datetime.now()
348
349     print '================================================================'
350     print 'Hold on to your hat. This may take ~45s per S2 tile folder.'
351     print 'Number of unprocessed IMG_DATA folders found: {}'.format(
352         len(imgFolders))
353     print 'Estimated time: {} minutes'.format(int(len(imgFolders))   0.75)
354     print 'Start time: {}'.format(start_time.time())
355     print '================================================================\n'
356
357     message = (
358         'Root Folder: {} \nNumber of unprocessed IMG_DATA folders '
359         'found: {}\nStart time: {}'
360         ).format(root_folder, len(imgFolders), start_time.time())
361     logger.info(message)
362     #
363     # Register all of the GDAL drivers.
364     #
365     gdal.AllRegister()
366
367     #
368     # Possible XML Schema namespaces (plus a few potential future ones.)
369     #
370     XML_namespaces = ['https://psd-14.sentinel2.eo.esa.int/',
371                       'https://psd-12.sentinel2.eo.esa.int/',
372                       'https://psd-13.sentinel2.eo.esa.int/',
373                       'https://psd-15.sentinel2.eo.esa.int/',
374                       'https://psd-16.sentinel2.eo.esa.int/']
375
376     i = 0
377
378     for imgFolder in imgFolders:
379
380         metadata_path = []
381
382         for fn in os.listdir(os.path.dirname(imgFolder)):
383             if ((fn.startswith('S2A_') or fn.startswith('MTD'))
384                     and fn.endswith('.xml')):
385                 metadata_file = fn
386                 metadata_path.append(
387                     os.path.join(os.path.dirname(imgFolder), fn))
388
389         if len(metadata_path) > 1:
390             message = (
391                 'Make sure only the original metadata exists in the tile '
392                 'folder\n{}\nAborting.'
393                 ).format(os.path.dirname(imgFolder))
394             print message
395             logger.critical(message)
396             sys.exit()
397
398         #
399         # Parse the metadata xml-file. There should only be one path.
400         #
401         try:
402             tree = etree.parse(metadata_path[0])
403         except Exception as e:
404             message = (
405                 '{} {} in {} could not be parsed.'
```

```
406                 ).format(str(e), metadata_path[0], imgFolder)
407             logger.critical(message)
408 
409         for namespace in XML_namespaces:
410 
411             try:
412 
413                 #
414                 # Get metadata values from the General_Info element.
415                 #
416                 General_Info = tree.find(
417                     '{' + namespace + 'PSD/'
418                     'S2_PDI_Level-1C_Tile_Metadata.xsd}General_Info')
419                 TILE_ID = General_Info.find('TILE_ID').text
420                 tile_id = TILE_ID[-12:-7]
421                 SENSING_TIME = General_Info.find('SENSING_TIME').text
422 
423                 #
424                 # Get metadata values from the Geometric_Info element.
425                 #
426                 Geometric_Info = tree.find(
427                     '{' + namespace + 'PSD/'
428                     'S2_PDI_Level-1C_Tile_Metadata.xsd}Geometric_Info')
429                 HORIZONTAL_CS_NAME = Geometric_Info.find(
430                     'Tile_Geocoding').find('HORIZONTAL_CS_NAME').text
431                 HORIZONTAL_CS_CODE = Geometric_Info.find(
432                     'Tile_Geocoding').find('HORIZONTAL_CS_CODE').text
433                 break
434 
435             except Exception as e:
436                 message = ('{} {} in {} could not be parsed with {}.').format(
437                     str(e), metadata_path[0], imgFolder, namespace)
438                 logger.error(message)
439         else:
440             message = ('{} in {} could not be parsed.').format(
441                 metadata_path[0], imgFolder)
442             logger.error(message)
443             continue
444 
445         tile_bands = []
446 
447         #
448         # Retrieve desired bands from old data structure.
449         #
450         if metadata_file.startswith('S2A_'):
451             for dirpath, dirnames, filenames in os.walk(
452                     imgFolder, topdown=True):
453                 for filename in filenames:
454                     if (filename.startswith('S2A')
455                             and filename.endswith('.jp2')
456                             and (fnmatch.fnmatch(filename, '*_B02.*')
457                                  or fnmatch.fnmatch(filename, '*_B03.*')
458                                  or fnmatch.fnmatch(filename, '*_B04.*')
459                                  or fnmatch.fnmatch(filename, '*_B08.*')
460                                  or fnmatch.fnmatch(filename, '*_B11.*')
461                                  or fnmatch.fnmatch(filename, '*_B12.*'))):
462 
463                         tile_bands.append(os.path.join(dirpath, filename))
464 
465         #
466         # Retrieve desired bands from data structure.
467         #
468         elif metadata_file.startswith('M'):
469             for dirpath, dirnames, filenames in os.walk(
470                     imgFolder, topdown=True):
471                 for filename in filenames:
472                     if (filename.startswith('T')
473                             and filename.endswith('.jp2')
474                             and (fnmatch.fnmatch(filename, '*_B02.*')
475                                  or fnmatch.fnmatch(filename, '*_B03.*')
476                                  or fnmatch.fnmatch(filename, '*_B04.*')
477                                  or fnmatch.fnmatch(filename, '*_B08.*')
478                                  or fnmatch.fnmatch(filename, '*_B11.*')
479                                  or fnmatch.fnmatch(filename, '*_B12.*'))):
480 
481                         tile_bands.append(os.path.join(dirpath, filename))
482 
483         #
484         # Put bands in numeric order for processing. Redundant, keep anyways.
485         #
486         tile_bands.sort()
487 
```

```
488         #
489         # Create the folder for processed data if it doesn't exist.
490         #
491         PROC_DATA = os.path.join(os.path.dirname(imgFolder), 'PROC_DATA')
492         if not os.path.exists(PROC_DATA):
493             os.mkdir(PROC_DATA)
494
495         #
496         # Create file to save stack to -- there is probably a better way to do
497         # this! Also create fake thermal band file.
498         #
499         print tile_bands
500         noData_array = None
501         noData_array = nodata_array(tile_bands, PROC_DATA)
502
503         for band in tile_bands:
504
505             if band.endswith('_B02.jp2'):
506
507                 #
508                 # Open B02 image in order to initialize .dat files. Any band
509                 # with 10m pixel size would do. Gets georeferencing info, etc.
510                 #
511                 img = gdal.Open(band, gdal.GA_ReadOnly)
512                 band_id = band[-6:-4]
513                 if img is None:
514                     message = (
515                         '{} in {} could not be opened.'
516                         ).format(band, imgFolder)
517                     print message
518                     logger.critical(message)
519                     sys.exit(1)
520
521                 print '------------------------------------------------------------'
522                 print 'Processing tile {} sensed at {}'.format(
523                     tile_id, SENSING_TIME)
524                 print 'Coordinate system: {}, {}\n\n'.format(
525                     HORIZONTAL_CS_NAME, HORIZONTAL_CS_CODE)
526
527                 #
528                 # Get raster georeference info from B02 for output .dat files.
529                 #
530                 projection = img.GetProjection()
531                 transform = img.GetGeoTransform()
532
533                 #
534                 # Establish size of raster from B02 for stacked output file.
535                 #
536                 img_rows = img.RasterYSize
537                 img_cols = img.RasterXSize
538
539                 #
540                 # Open output format driver, see gdal_translate for formats.
541                 #
542                 gdal_format = 'ENVI'
543                 driver = gdal.GetDriverByName(gdal_format)
544
545                 #
546                 # Test stacked band file path.
547                 #
548                 stacked_file = '{}calrefbyt_lndstlk.dat'.format(
549                     os.path.basename(band)[:-7])
550                 filepath = os.path.join(PROC_DATA, stacked_file)
551
552                 #
553                 # Print driver for stacked layers (6 bands, 8-bit unsigned).
554                 #
555                 outDs = driver.Create(filepath, img_cols, img_rows, 6,
556                                       gdal.GDT_Byte)
557                 if outDs is None:
558                     print 'Could not create test file.'
559                     sys.exit(1)
560
561                 #
562                 # Georeference the stacked .dat file and set the projection.
563                 #
564                 outDs.SetGeoTransform(transform)
565                 outDs.SetProjection(projection)
566
567                 print 'Creating fake thermal band for {}\n'.format(tile_id)
568
569                 #
```

```
570                 # Create thermal band file path.
571                 #
572                 thermal_file = '{}caltembyt_lndstlk.dat'.format(
573                     os.path.basename(band)[:-7])
574                 filepath = os.path.join(PROC_DATA, thermal_file)
575 
576                 #
577                 # Print driver for fake thermal band (1 band, 8-bit unsigned).
578                 #
579                 thermDs = driver.Create(filepath, img_cols, img_rows, 1,
580                                         gdal.GDT_Byte)
581                 if thermDs is None:
582                     print 'Could not create test file.'
583                     sys.exit(1)
584 
585                 #
586                 # Georeference the fake thermal band and set the projection.
587                 #
588                 thermDs.SetGeoTransform(transform)
589                 thermDs.SetProjection(projection)
590 
591                 #
592                 # Create constant array with a value of 110.
593                 #
594                 therm_array = numpy.ones((img_rows, img_cols)).astype(int)
595                 therm_array = therm_array   110
596 
597                 #
598                 # Remove pixels having no data in any of the input bands.
599                 #
600                 therm_array = numpy.where(
601                     (noData_array == 1), (0), therm_array)
602 
603                 #
604                 # Write the data to the designated band.
605                 #
606                 outBand = thermDs.GetRasterBand(1)
607                 outBand.WriteArray(therm_array, 0, 0)
608 
609                 #
610                 # Flush data to disk and set the NoData value.
611                 #
612                 outBand.FlushCache()
613 
614                 #
615                 # Calculate statistics.
616                 #
617                 stats = outBand.ComputeStatistics(False)
618                 outBand.SetStatistics(stats[0], stats[1], stats[2], stats[3])
619 
620                 print 'Fake thermal band created.\n\n'
621                 print 'Elapsed time: {}'.format(
622                     datetime.datetime.now() - start_time)
623 
624                 #
625                 # Clean up.
626                 #
627                 del driver
628                 band_id = None
629                 therm_array = None
630                 outBand = None
631                 stats = None
632                 thermDs = None
633                 img = None
634 
635         print 'Creating 6 band stack for tile {}\n'.format(tile_id)
636 
637         for band in tile_bands:
638 
639             #
640             # Keep track of which band we are writing to in the stacked file.
641             #
642             band_in_stack = None
643 
644             if band.endswith('_B02.jp2'):
645                 band_in_stack = 1
646             if band.endswith('_B03.jp2'):
647                 band_in_stack = 2
648             if band.endswith('_B04.jp2'):
649                 band_in_stack = 3
650             if band.endswith('_B08.jp2'):
651                 band_in_stack = 4
```

```
652             if band.endswith('_B11.jp2'):
653                 band_in_stack = 5
654             if band.endswith('_B12.jp2'):
655                 band_in_stack = 6
656 
657             #
658             # This if statement is redundant now, but keep for now anyways.
659             #
660             if band.endswith(('_B02.jp2', '_B03.jp2', '_B04.jp2', '_B08.jp2',
661                               '_B11.jp2', '_B12.jp2')):
662 
663                 #
664                 # Open the band as read only.
665                 #
666                 img = gdal.Open(band, gdal.GA_ReadOnly)
667                 band_id = band[-6:-4]
668                 if img is None:
669                     message = ('Could not open band #{} in {}').format(
670                         band_id, imgFolder)
671                     print message
672                     logger.critical(message)
673                     sys.exit(1)
674                 print 'Processing band #{}'.format(band_id)
675 
676                 #
677                 # Retrieve band and get dimensions.
678                 #
679                 img_band = img.GetRasterBand(1)
680                 img_rows = img.RasterYSize
681                 img_cols = img.RasterXSize
682 
683                 #
684                 # Read image as array using GDAL.
685                 #
686                 img_array = img_band.ReadAsArray(0, 0, img_cols, img_rows)
687                 print 'Original shape: {}'.format(img_array.shape)
688 
689                 #
690                 # Adjust outliers (very high reflectance and negative).
691                 #
692                 outData = img_array / 10000.0
693                 outData = numpy.where((outData > 1), (1), outData)
694                 outData = numpy.where((outData < 0), (0), outData)
695 
696                 img_array = None
697 
698                 #
699                 # Resample bands 11 and 12 from 20m to 10m resolution.
700                 #
701                 if band.endswith(('_B11.jp2', '_B12.jp2')):
702                     print 'Resample by a factor of 2 - nearest interpolation.'
703                     outData = scipy.ndimage.zoom(outData, 2, order=0)
704                     print 'Resampled size: {}'.format(outData.shape)
705 
706                 #
707                 # Convert to 8-bit.
708                 #
709                 outData = ((numpy.absolute(outData)   255.0) + 0.5).astype(int)
710 
711                 #
712                 # Remove pixels having no data in any of the input bands.
713                 #
714                 outData = numpy.where((noData_array == 1), (0), outData)
715 
716                 #
717                 # Write the data to the designated band.
718                 #
719                 outBand = outDs.GetRasterBand(band_in_stack)
720                 outBand.WriteArray(outData, 0, 0)
721 
722                 #
723                 # Flush data to disk and set the NoData value.
724                 #
725                 outBand.FlushCache()
726 
727                 #
728                 # Calculate statistics.
729                 #
730                 stats = outBand.ComputeStatistics(False)
731                 outBand.SetStatistics(stats[0], stats[1], stats[2], stats[3])
732 
733                 print 'Band #{} completed.\n'.format(band_id)
```

```
                print 'Elapsed time: {}'.format(
                    datetime.datetime.now() - start_time)

                #
                # Clean up to avoid problems processing bands to follow.
                #
                del outData
                del outBand
                img_band = None
                band_id = None
                stats = None
                img = None

        i += 1

        message = (
            'Tile {}, {} of {} processed and stacked.'
            ).format(tile_id, str(i), len(imgFolders))
        print message
        print '----------------------------------------------------------------\n'
        logger.info(message)

        #
        # Clean up to avoid problems processing tiles to follow.
        #
        del metadata_path
        del tree
        del SENSING_TIME
        del HORIZONTAL_CS_NAME
        del HORIZONTAL_CS_CODE
        del tile_bands
        del tile_id
        outDs = None
        noData_array = None

    print '\n\n================================================================'
    print 'Done processing.'
    print 'End time: {}'.format(datetime.datetime.now().time())
    print 'Total elapsed time: {}'.format(datetime.datetime.now() - start_time)
    print '================================================================\n\n'
    message = ('End time: {}').format(datetime.datetime.now().time())
    logger.info(message)
    message = ('Total elapsed time: {}').format(
        datetime.datetime.now() - start_time)
    logger.info(message)


if __name__ == '__main__':

    #
    # Set-up logger.
    #
    logging.basicConfig(filename='log/converter.log',
                        format='%(asctime)s:%(levelname)s:%(message)s',
                        level=logging.DEBUG)
    logger = logging.getLogger('converter ')

    #
    # Define S2 root folder, where all downloads are located.
    #
    options = get_args()
    root_folder = options.read_dir

    bool_answer, imgFolders_toProcess = check_imgFolders(options)

    if not bool_answer:
        print 'No folders processed.'
        sys.exit(1)

    if imgFolders_toProcess == 0:

        print 'No new folders to process.'
        logger.info('No new folders to process.')

    else:

        convert_imgs(root_folder, imgFolders_toProcess)
```

This script searches for Sentinel-2 product folders missing SIAM™ output and uses them to create a batch file to run SIAM™ so that they can be processed. While most lines fit within the 79 character style guideline, a few longer lines were broken without testing for this printed version. Unnecessary spaces between lines were also removed.

Listing A.4: Generate SIAM™ batch script (`batch_linux.py`)

```
# ----------------------------------------------------------------------------
# Name:         SIAM batch creator
# Purpose:      This script creates a SIAM batch file based
#               on SIAM compatible files located in a target
#               directory.
#
# Author:       h.Augustin
#
# Created:      21.12.2016
# Modified:     07.12.2017
#
# ----------------------------------------------------------------------------

#! /usr/bin/env python
# -- coding: iso-8859-1 --

import os
import sys
import shutil
import fnmatch
import argparse
import logging
from time import strftime as date

import gdal


def get_args():

    '''
    Gets arguments from command line.
    '''

    #
    # Create download tool help response.
    #
    prog = os.path.basename(sys.argv[0])

    if len(sys.argv) == 1:

        print(
            '\n        {0} [options]'
            '\n        Help: {1} --help'
            '\n        or: {1} -h'
            '\nexample python {0} -r /path/to/data/ --burnt-area 1'
            ).format(sys.argv[0], prog)

        sys.exit(-1)

    else:

        parser = argparse.ArgumentParser(
            prog=prog,
            usage='%(prog)s [options]',
            description='SIAM batch creator.',
            argument_default=None,
            epilog='Go get \'em!')

        #
        # General arguments.
        #
        parser.add_argument(
            '-r', '--read_dir', dest='read_dir', action='store',
            type=str,
            help='Path where downloaded products are located.',
            default=None)
        parser.add_argument(
            '-s', '--siam', dest='siam', action='store',
            type=str,
            help='Path to SIAM executable.',
            default='/opt/siam/SIAM_compilation_Ubuntu/SIAM_Ubuntu_r88v7.exe')
        parser.add_argument(
            '-w', '--write_dir', dest='write_dir', action='store',
            type=str,
            help='Path to save SIAM batch.',
            default='/home/hannah/repos/AIQ/thesis/siam/')
        parser.add_argument(
            '--auto', dest='auto', action='store',
```

```
79              help=('Automatically creates batch file without user input.'),
80              choices=['y', 'n'],
81              default=None)
82
83          #
84          # Optional parameters.
85          #
86          parser.add_argument(
87              '-03', '--bin-mask', dest='var03', action='store',
88              help=('Use a binary mask for processing. Default 0.'),
89              choices=['1', '0'],
90              default=1)
91          parser.add_argument(
92              '-07', '--crisp', dest='var07', action='store',
93              help=('Crisp[1] or fuzzy [0] classification. Default 1.'),
94              choices=['1', '0'],
95              default=1)
96          parser.add_argument(
97              '-09', '--smoke-plume', dest='var09', action='store',
98              help=('Create smoke-plume mask. Default 0.'),
99              choices=['1', '0'],
100             default=0)
101         parser.add_argument(
102             '-10', '--cloud', dest='var10', action='store',
103             help=('Create cloud mask. Default 0.'),
104             choices=['1', '0'],
105             default=0)
106         parser.add_argument(
107             '-11', '--burnt-area', dest='var11', action='store',
108             help=('Create burnt-area mask. Default 0.'),
109             choices=['1', '0'],
110             default=0)
111         parser.add_argument(
112             '-12', '--veg-bin', dest='var12', action='store',
113             help=('Create binary vegetation mask. Default 1.'),
114             choices=['1', '0'],
115             default=1)
116         parser.add_argument(
117             '-13', '--veg-tri', dest='var13', action='store',
118             help=('Create trinary vegetation mask. Default 0.'),
119             choices=['1', '0'],
120             default=0)
121         parser.add_argument(
122             '-14', '--baresoil', dest='var14', action='store',
123             help=('Create trinary baresoil builtup mask. Default 0.'),
124             choices=['1', '0'],
125             default=0)
126         parser.add_argument(
127             '-15', '--cloud-tri', dest='var15', action='store',
128             help=('Create trinary cloud mask. Default 0.'),
129             choices=['1', '0'],
130             default=0)
131         parser.add_argument(
132             '-16', '--water-tri', dest='var16', action='store',
133             help=('Create trinary water mask. Default 0.'),
134             choices=['1', '0'],
135             default=0)
136         parser.add_argument(
137             '-17', '--shadow-tri', dest='var17', action='store',
138             help=('Create trinary shadow mask. Default 0.'),
139             choices=['1', '0'],
140             default=0)
141         parser.add_argument(
142             '-18', '--urban-bin', dest='var18', action='store',
143             help=('Create binary urban area mask. Default 0.'),
144             choices=['1', '0'],
145             default=0)
146         parser.add_argument(
147             '-19', '--shape', dest='var19', action='store',
148             help=('Calculate shape indicators. Default 0.'),
149             choices=['1', '0'],
150             default=1)
151
152         return parser.parse_args()
153
154
155 def check_procFolders(options):
156
157     procFolders = []
158     siamFolders = []
159
160     for dirpath, dirnames, filenames in os.walk(
```

```
            options.read_dir, topdown=True):

        for dirname in dirnames:

            if dirname == 'PROC_DATA':
                procFolders.append(os.path.join(dirpath, dirname))

            elif dirname == 'siamoutput':
                siamFolders.append(os.path.join(dirpath, dirname))
    #
    # Determine which tile folders have no siamoutput folder.
    #
    unprocFolders = []

    for procFolder in procFolders:

        test_path = os.path.join(procFolder, 'siamoutput')

        if test_path in siamFolders:
            if len(os.listdir(test_path)) > 0:
                continue
            else:
                unprocFolders.append(procFolder)
        else:
            unprocFolders.append(procFolder)

    #
    # Check validity of relevant siamoutput contents.
    #
    for siamFolder in siamFolders:

        #
        # Initialize variables for PROC_DATA folder.
        #
        siam_Parts = ['18SpCt',
                      '33SharedSpCt',
                      '48SpCt',
                      '96SpCt',
                      'VegBinaryMask',
                      'fRatioGreennessIndex']

        remove_siamFolder = None

        for filename in os.listdir(siamFolder):

            #
            # Check only .dat files.
            #
            if filename.endswith('.dat'):

                for part in siam_Parts:

                    siam_wildcard = ' {} '.format(part)

                    if fnmatch.fnmatch(filename, siam_wildcard):

                        file_path = os.path.join(siamFolder, filename)
                        file_size = os.path.getsize(file_path)

                        if file_size < 5:
                            remove_siamFolder = True
                            logger.info(
                                'File smaller than 5 bytes: ' + file_path)

        #
        # Removes siamoutput folders with problem files and adds to list to be
        # processed.
        #
        if remove_siamFolder is True:
            shutil.rmtree(siamFolder)
            logger.info('Removed Folder: ' + siamFolder)
            unprocFolders.append(os.path.dirname(siamFolder))

    #
    # If no unprocessed folders, skip input from user.
    #
    if len(unprocFolders) == 0:
        question = ('{} unprocessed tiles from {} found.').format(
            str(len(unprocFolders)), str(len(procFolders)))
        print question
        bool_ans = None
        return bool_ans, unprocFolders
```

```
243
244     question = (
245         '{} unprocessed tiles from {} found. '
246         'Create batch for SIAM?'
247         ).format(str(len(unprocFolders)), str(len(procFolders)))
248
249     print question
250
251     ins = None
252     bool_ans = None
253
254     if options.auto is not None:
255
256         ins = 'y'
257
258     else:
259
260         while True:
261
262             ins = raw_input('Answer [y/n]: ')
263
264             if ins == 'y' or ins == 'n':
265                 break
266             else:
267                 print 'Your input should indicate yes [y] or no [n].'
268
269     if ins == 'y' or ins == 'Y' or ins == 'yes' or ins == 'Yes':
270
271         bool_ans = True
272
273     return bool_ans, unprocFolders
274
275
276 def create_batch(options, unprocFolders):
277
278     #
279     # Register all of the GDAL drivers.
280     #
281     gdal.AllRegister()
282
283     #
284     # Create empty batch file for SIAM.
285     #
286
287     timestamp = date('%Y%m%dT%H%M%S')
288
289     if len(unprocFolders) > 1:
290
291         batFilename = 'SIAM_multiple_batch_LANDSAT_{}.sh'.format(timestamp)
292
293     else:
294
295         batFilename = 'SIAM_batch_LANDSAT_{}.sh'.format(timestamp)
296
297     batch_path = os.path.join(options.write_dir, batFilename)
298
299     with open(batch_path, 'a') as f:
300
301         f.write('#!/bin/bash\n')
302
303     #
304     # Convert binary variables and image type identifier from GUI to strings.
305     #
306     var00 = options.siam
307     var06 = str(1)
308     var07 = str(options.var07)
309     var09 = str(options.var09)
310     var10 = str(options.var10)
311     var11 = str(options.var11)
312     var12 = str(options.var12)
313     var13 = str(options.var13)
314     var14 = str(options.var14)
315     var15 = str(options.var15)
316     var16 = str(options.var16)
317     var17 = str(options.var17)
318     var18 = str(options.var18)
319     var19 = str(options.var19)
320
321     for unprocFolder in unprocFolders:
322
323         var01 = unprocFolder
324
```

```
325         #
326         # Create the folder for SIAM output data if it doesn't exist.
327         #
328         siam_output = os.path.join(unprocFolder, 'siamoutput')
329 
330         if not os.path.exists(siam_output):
331 
332             os.mkdir(siam_output)
333 
334         var08 = siam_output
335 
336         #
337         # Fill var02 variable with the calrefbyt filename.
338         #
339         for filename in os.listdir(unprocFolder):
340 
341             if (filename.endswith('.dat')
342                     and fnmatch.fnmatch(filename, ' _calrefbyt_ ')):
343 
344                 var02 = filename
345 
346             elif (filename.endswith('.dat')
347                   and fnmatch.fnmatch(filename, ' nodata.dat')):
348 
349                 noData_Mask = filename
350         #
351         # Go to next folder if calibrated, stacked raster not yet craeted.
352         #
353         if not var02:
354 
355             message = ' _calrefbyt_  not found in {}'.format(unprocFolder)
356             print message
357             logger.error(message)
358             print 'Tile not processed.'
359 
360             continue
361 
362         if str(options.var03) == '1' and noData_Mask:
363 
364             var03 = noData_Mask
365 
366         elif str(options.var03) == '1' and not noData_Mask:
367 
368             message = ' nodata.dat not found in {}'.format(unprocFolder)
369             print message
370             logger.error(message)
371             print 'Tile not processed.'
372 
373             continue
374 
375         elif str(options.var03) == '0':
376 
377             var03 = str(options.var03)
378 
379         #
380         # Path to calrefbyt.
381         #
382         calrefbyt = os.path.join(unprocFolder, var02)
383 
384         #
385         # Open image to calculate rows and columns.
386         #
387         img = gdal.Open(calrefbyt, gdal.GA_ReadOnly)
388 
389         if img is None:
390 
391             message = (
392                 'Could not open  calrefbyt .dat. '
393                 'Folder not processed: {}\n').format(unprocFolder)
394             print message
395             logger.error(message)
396 
397             continue
398 
399         var04 = str(img.RasterYSize)  # Rows.
400         var05 = str(img.RasterXSize)  # Columns.
401 
402         #
403         # Create string to write to batch file.
404         #
405         batch_entry = ' '.join((
406             var00, var01, var02, var03, var04, var05, var06, var07, var08,
```

```
407             var09, var10, var11, var12, var13, var14, var15, var16, var17,
408             var18, var19))
409
410         with open(batch_path, 'a') as f:
411
412             f.write(batch_entry + '\n')
413
414         #
415         # Clean up.
416         #
417         img = None
418
419     #
420     # Location.
421     #
422     message = (
423         '\n\n{} created.\nSaved to: {}\n\n').format(batFilename, batch_path)
424     print message
425     logger.info(message)
426
427
428 if __name__ == '__main__':
429     #
430     # Set-up logger.
431     #
432     logging.basicConfig(
433         filename='log/batch.log',
434         format='%(asctime)s:%(levelname)s:%(message)s',
435         level=logging.DEBUG)
436     logger = logging.getLogger('batch')
437
438     #
439     # Parse command line to get global arguments.
440     #
441     options = get_args()
442     bool_answer, unprocFolders = check_procFolders(options)
443
444     if bool_answer:
445
446         create_batch(options, unprocFolders)
447
448     else:
449
450         print '\nNo SIAM batch file created.\n'
451         sys.exit()
452
453
454 #-------------------------------------------------------------------------------#
455 #                Parameters from Example batch file explained.                  #
456 #-------------------------------------------------------------------------------#
457
458 # (0) C:\SIAM\SIAM_r88v7\SIAM_License_Executables\SIAM_r88v7_Windows.exe
459 #                                                                  [SIAM executable]
460 # (1) E:\S2A_L1C_TL_MTI__20150813T201603_A000734_T33TUN_N01.03
461 #                                                [location of calibrated files]
462 # (2) S2A_L1C_TL_MTI__20150813T201603_A000734_T33TUN_calrefbyt_lndstlk.dat
463 #                                                                  [calibrated file]
464 # (3) 0  [Use a binary mask for processing]
465 # (4) 10980  [rows]
466 # (5) 10980  [columns]
467 # (6) 1  [image type:
468 #                     ("LANDSAT_LIKE","1")
469 #                     ("SPOT_LIKE","2")
470 #                     ("AVHRR_LIKE","3")
471 #                     ("VHR_LIKE","4")]
472 # (7) 1  [Select crisp or fuzzy classification]
473 # (8) E:\S2A_L1C_TL_MTI__20150813T201603_A000734_T33TUN_N01.03\siamoutput
474 #                                                                    [output folder]
475 # (9) 0   [Smoke-Plume mask]
476 # (10) 0  [Cloud Mask]
477 # (11) 0  [Burnt area mask]
478 # (12) 1  [Vegetation Binary mask]
479 # (13) 0  [Vegetation Trinary mask]
480 # (14) 0  [Baresoil Builtup Trinary mask]
481 # (15) 0  [Cloud Trinary mask]
482 # (16) 0  [Water Trinary mask]
483 # (17) 0  [Shadow Trinary mask]
484 # (18) 1  [Urban area binary mask]
485 # (19) 0  [SHAPE???]
```

This shell script runs the Python scripts shown in Listing A.2, A.3 and A.4 for each desired granule (37SBA, 37SCA, 37ADA) and is tasked to run once a day on the server, ultimately automating data acquisition to pre-classification.

Listing A.5: Automate acquisition to pre-classification (`automate_downloads.sh`)

```
#!/bin/bash

source activate aiq27
cd /home/hannah/repos/AIQ/thesis/
python download_linux.py -t 37SBA --auto y -a ./key.txt -w /data/s2/37SBA
python download_linux.py -t 37SCA --auto y -a ./key.txt -w /data/s2/37SCA
python download_linux.py -t 37SDA --auto y -a ./key.txt -w /data/s2/37SDA

python conversion_linux.py --auto y -r /data/s2/37SBA
python conversion_linux.py --auto y -r /data/s2/37SCA
python conversion_linux.py --auto y -r /data/s2/37SDA

python batch_linux.py --auto y -r /data/s2/
cd /home/hannah/repos/AIQ/thesis/siam/
LATEST=$(find . -mmin -60 -type f)
if [ -z "$LATEST" ]
then
  echo "\$LATEST is empty."
else
  bash "$LATEST"
fi
```

Listing A.6: *virtualenv* environment definition for indexing, ingesting and analysing data (`datacube_environment.txt`)

This file defines a virtualenv environment for Python version 3.4.5, which should be able to be reproduced using this file and the following command: `pip install -r FILE.txt`

```
affine==2.1.0
attrs==17.3.0
backports-abc==0.5
basemap==1.0.7
bleach==2.1.2
branca==0.2.0
cachetools==2.0.1
certifi==2017.11.5
chardet==3.0.4
click==6.7
click-plugins==1.0.3
cligj==0.4.0
cloudpickle==0.5.2
cycler==0.10.0
Cython==0.27.3
dask==0.16.0
-e git+https://github.com/ceos-seo/agdc-v2.git@d348e71962e5b267152c80138f189d728db74d5b#egg=datacube
decorator==4.2.1
entrypoints==0.2.3
folium==0.5.0
GDAL==1.11.2
html5lib==1.0.1
idna==2.6
ipykernel==4.8.0
ipython==6.2.1
ipython-genutils==0.2.0
ipywidgets==7.1.0
jedi==0.11.1
Jinja2==2.10
jsonschema==2.6.0
jupyter==1.0.0
jupyter-client==5.2.1
jupyter-console==5.2.0
jupyter-core==4.4.0
lcmap-pyccd==2017.10.27
MarkupSafe==1.0
matplotlib==2.1.1
mistune==0.8.3
nbconvert==5.3.1
nbformat==4.4.0
netCDF4==1.3.1
notebook==5.3.1
numpy==1.13.3
pandas==0.21.0
pandocfilters==1.4.2
parso==0.1.1
pathlib==1.0.1
pexpect==4.3.1
pickleshare==0.7.4
prompt-toolkit==1.0.15
psycopg2==2.7.3.2
ptyprocess==0.5.2
Pygments==2.2.0
pyparsing==2.2.0
pyPEG2==2.15.2
python-dateutil==2.6.1
pytz==2017.3
PyYAML==3.12
pyzmq==16.0.3
qtconsole==4.3.1
rasterio==1.0a12
requests==2.18.4
scikit-learn==0.19.1
scipy==1.0.0
Send2Trash==1.4.2
Shapely==1.6.3
simplegeneric==0.8.1
singledispatch==3.4.0.3
six==1.11.0
sklearn==0.0
snuggs==1.4.1
SQLAlchemy==1.2.0b3
terminado==0.8.1
testpath==0.3.1
toolz==0.8.2
tornado==4.5.3
traitlets==4.3.2
```

```
typing==3.6.2
urllib3==1.22
wcwidth==0.1.7
webencodings==0.5.1
widgetsnbextension==3.1.0
xarray==0.10.0
```

This is the SIAM™ information layer product definition for the ODC. A similar definition exists for Sentinel-2 data, but is not displayed here.

Listing A.7: Prepare SIAM™ product within the ODC. This only needs to be defined once (`siam_il.yaml`)

```
name: siam_il
description: Collection of SIAM–based information layers (IL)
metadata_type: eo

metadata:
    format: {name: ENVI}
    platform: {code: SIAM_IL}
    instrument: {name: SIAM}
    product_type: INFORMATIONLAYERS

measurements:
- name: '18SpCt'
  aliases: [18spct, siam18]
  dtype: uint8
  nodata: 255
  units: 'category'
  flags_definition:
    siam:
        bits: [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,255]
        description: SIAM 18 Spectral Categories
        values:
            0: Unclassified
            1: V
            2: VSH_VWA
            3: R
            4: WR
            5: PB
            6: GH
            7: BB
            8: BBSH
            9: WASH
            10: CL
            11: SMKPLM
            12: TNCL_SHRBR_HRBCR_BB
            13: SN_WAICE
            14: SHSN
            15: SH
            16: UN
            17: BA
            18: FLAME
            255: NO_DATA_MASK
- name: '33SharedSpCt'
  aliases: [33sharedspct, siam33]
  dtype: uint8
  nodata: 255
  units: 'category'
  flags_definition:
    siam:
        bits:
     [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,255]

        description: SIAM 33 Shared Spectral Categories
        values:
            0: Unclassified
            1: SVHNIR
            2: SVLNIR
            3: AVHNIR
            4: AVLNIR
            5: WV
            6: VSH_VWA_TWASH
            7: SHRBRHNIR
            8: SHRBRLNIR
            9: HRBCR
            10: WR
            11: PB
            12: GH
            13: VBBB
            14: BBB
            15: SBB
            16: ABB
            17: DBB
            18: WBBorBBSH
            19: NIRPBB
            20: BA
            21: DPWASH
            22: SLWASH
            23: TWASH
```

```
            24: SASLWA
            25: CL
            26: SMKPLM
            27: TNCLV
            28: TNCLWA_BB
            29: SN_WAICE
            30: SHSN
            31: SH
            32: FLAME
            33: UN
            255: NO_DATA_MASK
- name: '48SpCt'
  aliases: [48spct, siam48]
  dtype: uint8
  nodata: 255
  units: 'category'
  flags_definition:
    siam:
        bits:
     [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35

        description: SIAM 48 Spectral Categories
        values:
            0: Unclassified
            1: SV
            2: AV
            3: WV
            4: RWESH
            5: SHV
            6: SSHRBR
            7: ASHRBR
            8: SHR
            9: AHRBCR
            10: WR
            11: WEDR
            12: VSN
            13: PB
            14: GH
            15: VWA
            16: BBB_VBBB
            17: SBB
            18: ABB
            19: DBB
            20: WBB
            21: NIRPBB
            22: NIRPSABA
            23: BBSH
            24: RBA
            25: BSFAFS
            26: OBA
            27: BSFS
            28: DPWASH
            29: SLWASH
            30: TWA
            31: SASLWA
            32: CRCL
            33: TKCL
            34: SMKPLMWA
            35: SMKPLMV
            36: SMKPLMBB
            37: TNCLV_SHRBR_HRBCR
            38: TNCLWA_BB
            39: SN_WAICE
            40: SHSN
            41: WASHLTIR
            42: VDTWASH
            43: DTWASH
            44: VTNCLWA
            45: FLAME
            46: WAICE_SHSN
            47: UN2
            48: UN3
            255: NO_DATA_MASK
- name: '96SpCt'
  aliases: [96spct, siam96]
  dtype: uint8
  nodata: 255
  units: 'category'
  flags_definition:
    siam:
```

```
154         bits:
        [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
155         description: SIAM 96 Spectral Categories
156         values:
157             0: Unclassified
158             1: SVVH2NIR
159             2: SVVH1NIR
160             3: SVVHNIR
161             4: SVHNIR
162             5: SVMNIR
163             6: SVLNIR
164             7: SVVLNIR
165             8: AVVH1NIR
166             9: AVVHNIR
167             10: AVHNIR
168             11: AVMNIR
169             12: AVLNIR
170             13: AVVLNIR
171             14: WVHNIR
172             15: WVLNIR
173             16: RWESH
174             17: SHV
175             18: SSHRBR
176             19: ASHRBRVH1NIR
177             20: ASHRBRVHNIR
178             21: ASHRBRHNIR
179             22: ASHRBRMNIR
180             23: ASHRBRLNIR
181             24: ASHRBRVLNIR
182             25: SHRBCR
183             26: AHRBCR
184             27: WR
185             28: WEDR
186             29: VSN
187             30: PB
188             31: GH
189             32: VWA
190             33: VBBBFVHTIR
191             34: VBBBNFVHTIR
192             35: VBBBFLTIR
193             36: VBBBNFLTIR
194             37: BBBFVHTIR
195             38: BBBNFVHTIR
196             39: BBBFLTIR
197             40: BBBFLTIR
198             41: SBBVFVHTIR
199             42: SBBFVHTIR
200             43: SBBNFVHTIR
201             44: SBBVFLTIR
202             45: SBBFLTIR
203             46: SBBNFLTIR
204             47: ABBVFVHTIR
205             48: ABBFVHTIR
206             49: ABBNFVHTIR
207             50: ABBVFLTIR
208             51: ABBFLTIR
209             52: ABBNFLTIR
210             53: DBBVFVHTIR
211             54: DBBFVHTIR
212             55: DBBNFVHTIR
213             56: DBBVFLTIR
214             57: DBBFLTIR
215             58: DBBNFLTIR
216             59: WBB
217             60: NIRPBB
218             61: NIRPSABA
219             62: BBSH
220             63: RBA
221             64: BSFAFS
222             65: OBA
223             66: BSFS
224             67: DPWASHLTIR
225             68: DPWASHMTIR
226             69: DPWASHHTIR
227             70: DPWASHVHTIR
228             71: SLWASHLTIR
229             72: SLWASHMTIR
230             73: SLWASHHTIR
231             74: SLWASHVHTIR
232             75: TWA
233             76: SASLWA
```

```
            77: CRCL
            78: TKCL
            79: TKSMKPLMWA
            80: TNSMKPLMWA
            81: SMKPLMV
            82: SMKPLMBB
            83: TNCLV_SHRBR
            84: TNCLV_HRBCR
            85: TNCLWA
            86: TNCLWA_BB
            87: SN_WAICE
            88: SHSN
            89: WASHLTIR
            90: VDTWASH
            91: DTWASH
            92: VTNCLWA
            93: FLAME
            94: WAICE_SHSN
            95: UN2
            96: UN3
            255: NO_DATA_MASK
- name: 'HazePentarnaryMask'
  aliases: [haze, Haze, haze_mask]
  dtype: uint8
  units: 'category'
  nodata: 0
  flags_definition:
    siam:
        bits: [50,100,150,200,250]
        description: Haze Pentarnary Mask
        values:
            50: Very low
            100: Low
            150: Medium
            200: High
            250: Very high
- name: 'VegBinaryMask'
  aliases: [vegbinarymask, VegetationBinaryMask, vegetation_mask]
  dtype: uint8
  units: 'category'
  nodata: 2
  flags_definition:
    siam:
        bits: [0,1,2]
        description: Vegetation Binary Mask
        values:
            0: Likely not vegetation
            1: Looks like vegetation
- name: 'fRatioGreennessIndex'
  aliases: [greenness, greenness_index]
  dtype: float32
  nodata: 0
  units: 'greenness'
- name: 'cBrightness'
  aliases: [brightness]
  dtype: uint8
  nodata: 0
  units: 'brightness'
- name: 'nodata'
  aliases: [noData, NoData, no_data]
  dtype: uint8
  units: 'category'
  nodata: 2
  flags_definition:
    siam:
        bits: [0,1,2]
        description: NoData Binary Mask
        values:
            0: Data
            1: No data
```

Listing A.8: Prepare SIAM™ metadata for indexing. (`prep_siam.py`)

This script prepares metadata for SIAM™ required for indexing. A similar script exists for Sentinel-2 data, but is not displayed here. While most lines fit within the 79 character style guideline, a few longer lines were broken without testing for this printed version. Unnecessary spaces between lines were also removed.

```
# coding=utf-8
"""
Ingest data from the command-line.
"""
from __future__ import absolute_import

import uuid
import logging
from xml.etree import ElementTree
from pathlib import Path
import yaml
import click
from osgeo import osr
import os
import sys
import datetime
# image boundary imports
import rasterio
from rasterio.errors import RasterioIOError
import rasterio.features
import shapely.affinity
import shapely.geometry
import shapely.ops


# IMAGE BOUNDARY CODE


def safe_valid_region(images, mask_value=None):
    try:
        return valid_region(images, mask_value)
    except (OSError, RasterioIOError):
        return None


def valid_region(images, mask_value=None):
    mask = None
    for fname in images:
        # ensure formats match
        with rasterio.open(str(fname), 'r') as ds:
            transform = ds.transform
            img = ds.read(1)

            if mask_value is not None:
                new_mask = img & mask_value == mask_value
            else:
                # new_mask = img != ds.nodata
                new_mask = img != 0
            if mask is None:
                mask = new_mask
            else:
                mask |= new_mask

    shapes = rasterio.features.shapes(mask.astype('uint8'), mask=mask)
    shape = shapely.ops.unary_union(
        [shapely.geometry.shape(shape) for shape, val in shapes if val == 1])
    type(shapes)
    # convex hull
    geom = shape.convex_hull

    # buffer by 1 pixel
    geom = geom.buffer(1, join_style=3, cap_style=3)

    # simplify with 1 pixel radius
    geom = geom.simplify(1)

    # intersect with image bounding box
    geom = geom.intersection(shapely.geometry.box(
        0, 0, mask.shape[1], mask.shape[0]))

    # transform from pixel space into CRS space
    geom = shapely.affinity.affine_transform(geom, (
        transform.a, transform.b,
        transform.d, transform.e,
        transform.xoff, transform.yoff))

    output = shapely.geometry.mapping(geom)

```

```
    return geom
    # output['coordinates'] = _to_lists(output['coordinates'])
    # return output


def _to_lists(x):
    """
    Returns lists of lists when given tuples of tuples
    """
    if isinstance(x, tuple):
        return [_to_lists(el) for el in x]

    return x


def get_geo_ref_points(root):
    nrows = int(root.findall(
        './/Tile_Geocoding/Size[@resolution="10"]/NROWS')[0].text)
    ncols = int(root.findall(
        './/Tile_Geocoding/Size[@resolution="10"]/NCOLS')[0].text)

    ulx = int(root.findall(
        './/Tile_Geocoding/Geoposition[@resolution="10"]/ULX')[0].text)
    uly = int(root.findall(
        './/Tile_Geocoding/Geoposition[@resolution="10"]/ULY')[0].text)

    xdim = int(root.findall(
        './/Tile_Geocoding/Geoposition[@resolution="10"]/XDIM')[0].text)
    ydim = int(root.findall(
        './/Tile_Geocoding/Geoposition[@resolution="10"]/YDIM')[0].text)

    return {
        'ul': {'x': ulx, 'y': uly},
        'ur': {'x': ulx + ncols   abs(xdim), 'y': uly},
        'll': {'x': ulx, 'y': uly — nrows   abs(ydim)},
        'lr': {'x': ulx + ncols   abs(xdim), 'y': uly — nrows   abs(ydim)},
    }


def get_coords(geo_ref_points, spatial_ref):
    t = osr.CoordinateTransformation(spatial_ref, spatial_ref.CloneGeogCS())

    def transform(p):
        lon, lat, z = t.TransformPoint(p['x'], p['y'])
        return {'lon': lon, 'lat': lat}

    return {key: transform(p) for key, p in geo_ref_points.items()}


def get_relevantpaths(path):
    for dirpath, dirnames, filenames in os.walk(str(path.parent)):
        for dirname in dirnames:
            if dirname == 'siamoutput':
                siamoutput = os.path.join(dirpath, dirname)
        for filename in filenames:
            if filename == 'datacube—metadata.yaml':
                datacube_YAML = os.path.join(dirpath, filename)
    return siamoutput, datacube_YAML

def get_PROC_DATA(path):

    for dirpath, dirnames, filenames in os.walk(path):
        for dirname in dirnames:
            if dirname == 'PROC_DATA':
                PROC_DATA =  os.path.join(dirpath, dirname)

    return PROC_DATA

def get_siamlayers(path):

    layers = ['18SpCt', '33SharedSpCt', '48SpCt', '96SpCt',
              'HazePentarnaryMask', 'VegBinaryMask',
              'fRatioGreennessIndex', 'cBrightness']
    siam_dict = {}
    print(path)
    for item in os.listdir(path):
        for layer in layers:
            if layer in item and item.endswith('.dat'):
                siam_dict[layer] = os.path.join('siamoutput', item)
    for item in os.listdir(os.path.dirname(path)):
        if item.endswith('nodata.dat'):
            siam_dict['nodata'] = item
```

```
161
162     return siam_dict
163
164 def prepare_dataset(path):
165
166     root = ElementTree.parse(str(path)).getroot()
167
168     siamoutput_path, datacubeYAML_path = get_relevantpaths(path)
169     siam_dict = get_siamlayers(siamoutput_path)
170
171     siamoutput_file = os.path.join(siamoutput_path, (
172         os.listdir(siamoutput_path))[0])
173     ct_time = (datetime.datetime.fromtimestamp(
174         os.path.getmtime(siamoutput_file)).strftime('%Y-%m-%dT%H:%M:%SZ'))
175     del siamoutput_file
176
177     with open(datacubeYAML_path, 'r') as f:
178         datacube_YAML = yaml.load(f.read())
179
180     granules = {granule.get('granuleIdentifier'): [
181                 imid.text for imid in granule.findall('IMAGE_ID')]
182                 for granule in root.findall(
183                     './/Product_Info/Product_Organisation/'
184                     'Granule_List/Granules')}
185
186     if not granules:
187         granules = {granule.get('granuleIdentifier'): [
188             imid.text for imid in granule.findall('IMAGE_ID')]
189             for granule in root.findall(
190                 './/Product_Info/Product_Organisation/Granule_List/Granule')}
191
192     for key, value in granules.items():
193         found_IMAGE_IDs = value
194
195     if len(granules) > 10 or len(found_IMAGE_IDs) > 10:
196
197         documents = []
198         for granule_id, images in granules.items():
199             gran_path = str(path.parent.joinpath(
200                 'GRANULE', granule_id, granule_id[:-7].replace(
201                     'MSI', 'MTD') + '.xml'))
202             if os.path.exists(gran_path):
203                 root = ElementTree.parse(gran_path).getroot()
204
205                 sensing_time = root.findall('.//SENSING_TIME')[0].text
206                 cs_code = root.findall(
207                     './/Tile_Geocoding/HORIZONTAL_CS_CODE')[0].text
208                 spatial_ref = osr.SpatialReference()
209                 spatial_ref.SetFromUserInput(cs_code)
210                 geo_ref_points = get_geo_ref_points(root)
211
212                 documents.append({
213                 'id': str(uuid.uuid4()),
214                 'product_type': 'INFORMATIONLAYERS',
215                 'creation_dt': ct_time,
216                 'platform': {'code': 'SIAM_IL'},
217                 'instrument': {'name': 'SIAM'},
218                 'extent': {
219                     'from_dt': sensing_time,
220                     'to_dt': sensing_time,
221                     'center_dt': sensing_time,
222                     'coord': get_coords(geo_ref_points, spatial_ref),
223                 },
224                 'format': {'name': 'ENVI'},
225                 'grid_spatial': {
226                     'projection': {
227                         'geo_ref_points': geo_ref_points,
228                         'spatial_reference': spatial_ref.ExportToWkt(),
229                     }
230                 },
231                 'image': {
232                     'bands': {
233                         key: {
234                             'path': value,
235                             'layer': 1,
236                         } for key, value in siam_dict.items()
237                         }
238                 },
239
240                 'lineage': {
241                     'source_datasets': {
242                         'level1': datacube_YAML }},
```

```
            })

    else:
        granules = {granule.get('granuleIdentifier'): [
                    imid.text for imid in granule.findall('IMAGE_FILE')]
                    for granule in root.findall(
                        './/Product_Info/Product_Organisation/'
                        'Granule_List/Granule')}

        documents = []
        for granule_id, images in granules.items():

            gran_path = str(path.parent.joinpath((
                os.path.dirname(os.path.dirname(images[0]))), 'MTD_TL.xml'))
            root = ElementTree.parse(gran_path).getroot()

            sensing_time = root.findall('.//SENSING_TIME')[0].text
            cs_code = root.findall(
                './/Tile_Geocoding/HORIZONTAL_CS_CODE')[0].text
            spatial_ref = osr.SpatialReference()
            spatial_ref.SetFromUserInput(cs_code)
            geo_ref_points = get_geo_ref_points(root)

            documents.append({
                'id': str(uuid.uuid4()),
                'product_type': 'INFORMATIONLAYERS',
                'creation_dt': ct_time,
                'platform': {'code': 'SIAM_IL'},
                'instrument': {'name': 'SIAM'},
                'extent': {
                    'from_dt': sensing_time,
                    'to_dt': sensing_time,
                    'center_dt': sensing_time,
                    'coord': get_coords(geo_ref_points, spatial_ref),
                },
                'format': {'name': 'ENVI'},
                'grid_spatial': {
                    'projection': {
                        'geo_ref_points': geo_ref_points,
                        'spatial_reference': spatial_ref.ExportToWkt(),
                    }
                },
                'image': {
                    'bands': {
                       key : {
                            'path': value,
                            'layer': 1,
                        } for key, value in siam_dict.items()
                        }
                },

                'lineage': {
                    'source_datasets': {
                        'level1': datacube_YAML }},
            })
    return documents


@click.command(help="Prepare Sentinel 2 dataset for ingestion into Data Cube.")
@click.argument('datasets',
                type=click.Path(exists=True, readable=True, writable=True),
                nargs=-1)
def main(datasets):
    logging.basicConfig(format='%(asctime)s %(levelname)s %(message)s',
        level=logging.INFO)

    for dataset in datasets:
        path = Path(dataset)

        if path.is_dir():
            print(path)
            xml_path = Path(path.joinpath(path.stem.replace(
                'PRD_MSIL1C', 'MTD_SAFL1C') + '.xml'))
            if not xml_path.exists():
                xml_path = Path(path.joinpath('MTD_MSIL1C' + '.xml'))
                print (xml_path)
            if xml_path.exists():
                path = xml_path
                del xml_path
        if path.suffix != '.xml':
            raise RuntimeError('want xml')

```

```
325         logging.info("Processing %s", path)
326         documents = prepare_dataset(path)
327         print(documents)
328         #
329         # BE sure to save in PROC_DATA folder.
330         #
331         PROC_DATA = get_PROC_DATA(dataset)
332         if documents:
333             yaml_path = os.path.join(PROC_DATA, 'siam-metadata.yaml')
334             logging.info("Writing %s dataset(s) into %s",
335                 len(documents), yaml_path)
336             with open(yaml_path, 'w') as stream:
337                 yaml.dump_all(documents, stream)
338         else:
339             logging.info("No datasets discovered. Bye!")
340
341
342 if __name__ == "__main__":
343     root_folder = '/data/s2/'
344     granule_folders = ['37SBA', '37SCA', '37SDA']
345     datasets = []
346     for tile in granule_folders:
347         folder_path = os.path.join(root_folder, tile)
348         for ds in os.listdir(folder_path):
349             dataset = os.path.join(folder_path, ds)
350             test_path = get_PROC_DATA(dataset)
351             yaml_test = Path(os.path.join(test_path, 'siam-metadata.yaml'))
352             if not yaml_test.exists():
353                 datasets.append(dataset)
354     main(datasets)
```

This script indexes the SIAM™ information layers based on the product defined in Listing A.7. A similar script exists for Sentinel-2 data, but is not displayed here. While most lines fit within the 79 character style guideline, a few longer lines were broken without testing for this printed version. Unnecessary spaces between lines were also removed.

Listing A.9: Index SIAM™ information layers in the ODC (`index_siam.py`)

```
import os
import sys
import logging
import datetime
import subprocess

def main(datasets):

    for yaml_path in datasets:
        cmd = ['datacube', '-v', 'dataset', 'add',  yaml_path]

        logger.info('Indexing dataset: {}'.format(yaml_path))

        ps = subprocess.Popen(cmd, stdout=subprocess.PIPE)

        output = ps.communicate()[0]
        for line in output.splitlines():
            logger.debug("[ ] {0}".format(line))

if __name__ == "__main__":
    #
    # Set-up logger.
    #
    log_dir = os.path.join(
        '/home/odci/Datacube/agdc-v2/ingest/prepare_scripts/siam/', 'log')
    if not os.path.exists(log_dir):
        os.mkdir(log_dir)

    logger_filepath = os.path.join(log_dir, 'index_s2.log')

    logging.basicConfig(filename=logger_filepath,
                        format='%(asctime)s:%(levelname)s:%(message)s',
                        level=logging.DEBUG)
    logger = logging.getLogger('index_siam')

    datasets = []
    for dirpath, dirnames, filenames in os.walk('/data/s2/', topdown=True):
        for filename in filenames:
            if filename.endswith('siam-metadata.yaml'):
                #
                # Only adds files older than 5 days.
                #
                test_path = os.path.join(dirpath, filename)
                creation_time = (datetime.datetime.fromtimestamp(
                    os.path.getmtime(test_path)))
                now = datetime.datetime.now()
                diff = (now-creation_time).days
                if diff <= 6:
                    datasets.append(os.path.join(dirpath, filename))
    main(datasets)
```

This is the SIAM™ information layer product definition for ingesting into the ODC.

Listing A.10: Defines SIAM™ product with chunks of 10km by 10km by 1 time-step for ingestion (`s2_siam_epsg32637_syria_10km.yaml`)

```
source_type: siam_il
output_type: siam_utm37n_10km

description: SIAM results processed using all Sentinel-2 data for granules 37
     SBA, 37SCA and 37SDA (10meter, 10km tile, EPSG:32637)

location: '/data/s2/ingested_data'
file_path_template: 'siam_syria/v2/S2_MSI_LC1_SIAM_32637_10km_{start_time}_{
     tile_index[0]}_{tile_index[1]}.nc'

global_attributes:
  title: Syria Data Cube Sentinel-2 SIAM Data
  summary: Sentinel-2 Multispectral Imager Precision data processed using SIAM
     from Dr. Andrea Baraldi  by h.Augustin at Z_GIS (University of Salzburg).
  source: S2 MSI Level-1c granules processed using SIAM
  history: This data represents a tile of Sentinel-2 MSI granule data processed
      with SIAM from Dr. Andrea Baraldi.
  institution: Z_GIS
  instrument: SIAM
  cdm_data_type: Grid
  keywords: EARTH SCIENCE
  keywords_vocabulary: GCMD
  platform: SIAM_IL
  product_version: '1.0.0'
  product_suite: SIAM
  project: Syria Master Thesis
  publisher_email: hannahlucille.augustin@sbg.ac.at
  publisher_name: Hannah Augustin
  publisher_url: obia.zgis.at
  coverage_content_type: preclassification
  license: https://creativecommons.org/licenses/by/4.0/
  naming_authority: obia.zgis.at
  acknowledgment: Sentinel-2 data is provided by the European Space Agency on
     behalf of the European Commission via download.

storage:
  driver: NetCDF CF

  crs: EPSG:32637
  tile_size:
          x: 10000.0
          y: 10000.0
  resolution:
          x: 10
          y: -10
  chunking:
      x: 200
      y: 200
      time: 1
  dimension_order: ['time', 'y', 'x']

fuse_data: copy

measurements:
    - name: siam18
      dtype: uint8
      nodata: 255
      src_varname: '18SpCt'
      resampling_method: nearest
      zlib: True
      attrs:
          long_name: "18 SIAM spectral categories"
          alias: "18SpCt"
    - name: siam33
      dtype: uint8
      nodata: 255
      resampling_method: nearest
      src_varname: '33SharedSpCt'
      zlib: True
      attrs:
          long_name: "33 SIAM spectral categories"
          alias: "33SpCt"
    - name: siam48
      dtype: uint8
      nodata: 255
      resampling_method: nearest
      src_varname: '48SpCt'
```

```
      zlib: True
      attrs:
          long_name: "48 SIAM spectral categories"
          alias: "48SpCt"
    - name: siam96
      dtype: uint8
      nodata: 255
      resampling_method: nearest
      src_varname: '96SpCt'
      zlib: True
      attrs:
          long_name: "96 SIAM spectral categories"
          alias: "96SpCt"
    - name: haze_mask
      dtype: uint8
      nodata: 0
      resampling_method: nearest
      src_varname: 'HazePentarnaryMask'
      zlib: True
      attrs:
          long_name: "SIAM Haze Mask"
          alias: "haze"
    - name: veg_mask
      dtype: uint8
      nodata: 2
      resampling_method: nearest
      src_varname: 'VegBinaryMask'
      zlib: True
      attrs:
          long_name: "SIAM Binary Vegetation Mask"
          alias: "veg"
    - name: greenness
      dtype: float32
      nodata: 0
      resampling_method: nearest
      src_varname: 'fRatioGreennessIndex'
      zlib: True
      attrs:
          long_name: "SIAM greenness index"
          alias: "green"
    - name: brightness
      dtype: uint8
      nodata: 0
      resampling_method: nearest
      src_varname: 'cBrightness'
      zlib: True
      attrs:
          long_name: "SIAM brightness"
          alias: "bright"
    - name: nodata_mask
      dtype: uint8
      nodata: 2
      resampling_method: nearest
      src_varname: 'nodata'
      zlib: True
      attrs:
          long_name: "SIAM nodata mask"
          alias: "nodata"
```

Listing A.11: Automate creation of necessary metadata, as well as indexation and ingestion in the ODC (`automate_ODC.sh`)

```
#!/bin/bash

source ~/Datacube/datacube_env/bin/activate

cd /home/odci/Datacube/agdc-v2/ingest/prepare_scripts/sentinel_2/

python prep_s2.py
python index_s2.py

cd /home/odci/Datacube/agdc-v2/ingest/prepare_scripts/siam/

python prep_siam.py
python index_siam.py

datacube -v ingest -c /home/odci/Datacube/agdc-v2/ingest/ingestion_configs/siam
    /s2_siam_epsg32637_syria_10km.yaml --executor multiproc 8
```

This shell script runs the scripts shown in Listing A.8 and A.9 and ingests data into the product defined by Listing A.10 and is tasked to run once a day on the server, ultimately automating indexing the original Sentinel-2 data and SIAM™-generated information layers into the ODC implementation, and then ingesting the SIAM™ layers as a product for analysis.

This file defines a conda environment for Python version 3.5.4, which should be able to be reproduced using this file and the following command: `conda env create -file=FILE.yaml`

Listing A.12: *conda* environment definition for analysis (`cubeenv_2_environment.yaml`)

```
name: cubeenv_2
channels:
  - conda-forge
  - defaults
dependencies:
  - affine=2.1.0=py_1
  - amqp=2.2.2=py35_0
  - asn1crypto=0.22.0=py35_0
  - attrs=17.4.0=py_0
  - backports=1.0=py35_1
  - backports.functools_lru_cache=1.4=py35_1
  - basemap=1.1.0=py35_3
  - basemap-data-hires=1.1.0=0
  - billiard=3.5.0.2=py35_0
  - blas=1.1=openblas
  - bleach=2.0.0=py35_0
  - bokeh=0.12.13=py35_0
  - boost=1.66.0=py35_1
  - boost-cpp=1.66.0=1
  - boto3=1.5.14=py35_0
  - botocore=1.5.92=py35_0
  - bzip2=1.0.6=1
  - ca-certificates=2017.11.5=0
  - cachetools=2.0.0=py35_0
  - cairo=1.14.10=0
  - celery=4.1.0=py35_0
  - certifi=2017.11.5=py35_0
  - cffi=1.11.2=py35_0
  - click=6.7=py_1
  - click-plugins=1.0.3=py35_0
  - cligj=0.4.0=py35_0
  - cloudpickle=0.5.2=py_0
  - cryptography=2.1.4=py35_0
  - curl=7.55.1=0
  - cycler=0.10.0=py35_0
  - dask=0.16.1=py_0
  - dask-core=0.16.1=py_0
  - datacube=1.5.4=py35_0
  - dbus=1.10.22=0
  - decorator=4.1.2=py35_0
  - distributed=1.20.2=py35_0
  - docutils=0.14=py35_0
  - entrypoints=0.2.3=py35_1
  - expat=2.2.5=0
  - fontconfig=2.12.6=0
  - freetype=2.8.1=0
  - freexl=1.0.4=0
  - gdal=2.2.3=py35_0
  - geos=3.6.2=1
  - geotiff=1.4.2=1
  - gettext=0.19.7=1
  - giflib=5.1.4=0
  - glib=2.55.0=0
  - gmp=6.1.2=0
  - graphite2=1.3.10=0
  - graphviz=2.38.0=7
  - gst-plugins-base=1.8.0=0
  - gstreamer=1.8.0=1
  - harfbuzz=1.7.1=0
  - hdf4=4.2.13=0
  - hdf5=1.10.1=1
  - heapdict=1.0.0=py35_0
  - html5lib=1.0.1=py_0
  - icu=58.2=0
  - idna=2.6=py35_1
  - ipykernel=4.7.0=py35_0
  - ipython=6.2.1=py35_1
  - ipython_genutils=0.2.0=py35_0
  - ipywidgets=7.1.0=py35_0
  - jedi=0.10.2=py35_0
  - jinja2=2.10=py35_0
  - jmespath=0.9.3=py35_0
  - jpeg=9b=2
  - json-c=0.12.1=0
  - jsonschema=2.6.0=py35_0
  - jupyter=1.0.0=py35_0
  - jupyter_client=5.2.1=py35_0
```

```
78   - jupyter_console=5.2.0=py35_0
79   - jupyter_core=4.4.0=py_0
80   - kealib=1.4.7=4
81   - kombu=4.1.0=py35_0
82   - krb5=1.14.2=0
83   - libdap4=3.18.3=2
84   - libffi=3.2.1=3
85   - libgdal=2.2.3=2
86   - libiconv=1.15=0
87   - libkml=1.3.0=6
88   - libnetcdf=4.4.1.1=10
89   - libpng=1.6.34=0
90   - libpq=9.6.3=0
91   - libsodium=1.0.15=1
92   - libspatialite=4.3.0a=19
93   - libssh2=1.8.0=2
94   - libtiff=4.0.9=0
95   - libtool=2.4.6=0
96   - libxcb=1.12=1
97   - libxml2=2.9.7=0
98   - libxslt=1.1.32=0
99   - locket=0.2.0=py35_1
100  - lxml=4.1.1=py35_0
101  - markupsafe=1.0=py35_0
102  - matplotlib=2.1.2=py35_0
103  - mistune=0.8.3=py_0
104  - msgpack-python=0.4.8=py35_0
105  - nbconvert=5.3.1=py_1
106  - nbformat=4.4.0=py35_0
107  - ncurses=5.9=10
108  - netcdf4=1.3.1=py35_1
109  - notebook=5.2.2=py35_1
110  - numpy=1.14.0=py35_blas_openblas_200
111  - openblas=0.2.20=7
112  - openjpeg=2.3.0=2
113  - openssl=1.0.2n=0
114  - pandas=0.22.0=py35_0
115  - pandoc=2.1=0
116  - pandocfilters=1.4.1=py35_0
117  - pango=1.40.14=0
118  - paramiko=2.3.1=py_0
119  - partd=0.3.8=py35_0
120  - pathlib=1.0.1=py35_0
121  - pcre=8.39=0
122  - pexpect=4.3.1=py35_0
123  - pickleshare=0.7.4=py35_0
124  - pip=9.0.1=py35_1
125  - pixman=0.34.0=1
126  - poppler=0.61.1=3
127  - poppler-data=0.4.8=0
128  - proj4=4.9.3=5
129  - prompt_toolkit=1.0.15=py35_0
130  - psutil=5.4.0=py35_0
131  - psycopg2=2.7.3.2=py35_0
132  - ptyprocess=0.5.2=py35_0
133  - pyasn1=0.4.2=py_0
134  - pycparser=2.18=py35_0
135  - pygments=2.2.0=py35_0
136  - pynacl=1.1.2=py35_0
137  - pyparsing=2.2.0=py35_0
138  - pypeg2=2.15.2=py35_0
139  - pyproj=1.9.5.1=py35_0
140  - pyqt=5.6.0=py35_4
141  - pyshp=1.2.12=py_0
142  - python=3.5.4=3
143  - python-dateutil=2.6.1=py35_0
144  - python-graphviz=0.8=py35_0
145  - pytz=2017.3=py_2
146  - pyyaml=3.12=py35_1
147  - pyzmq=16.0.2=py35_3
148  - qt=5.6.2=7
149  - qtconsole=4.3.1=py35_0
150  - readline=7.0=0
151  - redis-py=2.10.6=py_0
152  - s3transfer=0.1.11=py35_0
153  - scipy=1.0.0=py35_blas_openblas_201
154  - setuptools=38.4.0=py35_0
155  - simplegeneric=0.8.1=py35_0
156  - simplejson=3.11.1=py35_0
157  - singledispatch=3.4.0.3=py35_0
158  - sip=4.18=py35_1
159  - six=1.11.0=py35_1
```

```
  - snuggs=1.4.1=py35_0
  - sortedcontainers=1.5.7=py35_0
  - sqlalchemy=1.1.13=py35_0
  - sqlite=3.20.1=2
  - sshtunnel=0.1.1=py35_0
  - tblib=1.3.2=py35_0
  - terminado=0.8.1=py35_0
  - testpath=0.3.1=py35_0
  - tk=8.6.7=0
  - toolz=0.8.2=py_2
  - tornado=4.5.3=py35_0
  - tqdm=4.19.5=py_0
  - traitlets=4.3.2=py35_0
  - vine=1.1.4=py35_0
  - wcwidth=0.1.7=py35_0
  - webencodings=0.5=py35_0
  - wheel=0.30.0=py35_2
  - widgetsnbextension=3.1.0=py35_0
  - xerces-c=3.2.0=0
  - xorg-libxau=1.0.8=3
  - xorg-libxdmcp=1.1.2=3
  - xz=5.2.3=0
  - yaml=0.1.6=0
  - zeromq=4.2.1=1
  - zict=0.1.3=py_0
  - zlib=1.2.11=0
  - bcrypt=3.1.4=py35ha35c455_0
  - intel-openmp=2018.0.0=hc7b2577_8
  - libgcc-ng=7.2.0=h7cc24e2_2
  - libgfortran=3.0.0=1
  - libgfortran-ng=7.2.0=h9f7466a_2
  - libstdcxx-ng=7.2.0=h7a57d05_2
  - mkl=2018.0.1=h19d6760_4
  - rasterio=0.36.0=py35h94a1d1c_1
  - util-linux=2.21=0
  - xarray=0.10.0=py35h17cbfd9_0
  - pip:
    - boltons==17.2.0
    - datacube-stats==0.9a7
    - fiona==1.7.11
    - munch==2.2.0
    - pydash==4.3.1
    - redis==2.10.6
    - voluptuous==0.10.5
prefix: /home/odci/miniconda3/envs/cubeenv_2
```

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede and Ivo Pletikosić with minor modifications by Hannah Augustin, and a set-up to write the body of the text in Markdown, leaving compilation and reference management to pandoc. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of July 12, 2018 (version 2.1).