Mathematics and big data technology development to visualize, deliver and, analyze IMS and ARGO data

---

A Thesis

Presented to the

Faculty of

San Diego State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Applied Mathematics

---

by

Tyler Tucker

Spring 2018

SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the

Thesis of Tyler Tucker:

Mathematical applications

to big data visualization and analysis

with applications in climate and oceanographic data

Sam Shen, Chair
Department of Mathematics and Statistics

Barbara Bailey
Department of Mathematics and Statistics

Fernando De Sales
Department of Geography

Approval Date

DEDICATION

Dedicated to Juliana Johnson, as the person who supported me through this whole adventure.

But that man who sets himself the task of singling out the thread of order from the tapestry will by the decision alone have taken charge of the world and it is only by such taking charge that he will effect a way to dictate the terms of his own fate.

Cormac McCarthy, Blood Meridian, or the Evening Redness in the West

ABSTRACT OF THE THESIS

Mathematical applications
to big data visualization and analysis
with applications in climate and oceanographic data
by
Tyler Tucker
Master of Science in Applied Mathematics
San Diego State University, 2018

This research addresses current trends in big data, and how to apply them to mathematical, statistical, and climate science applications.

The first chapter describes a toolkit for snow-cover area calculation and display (SACD) based on the Interactive Multisensor Snow and Ice Mapping System (IMS). The paper uses the Tibetan Plateau region as an example to describe the toolkit's method, results, and usage. The National Snow and Ice Data Center (NSIDC) provides to the public IMS, a well-used system for monitoring the snow and ice cover. The newly developed toolkit is based on a simple shoe-lace formula for a grid box area on a sphere and can be conveniently used to calculate the total area of snow cover given the IMS data. The toolkit has been made available as an open source Python software on GitHub. The toolkit generates the time-series of the daily snow-covered area for any region over the Northern Hemisphere from 4 February 1997. The toolkit also creates maps showing snow and ice coverage with an elevation background. The Tibetan Plateau (TP) region $(25^\circ - 45^\circ N) \times (65^\circ - 105^\circ E)$ is used as an example to demonstrate our work on SACD. The IMS products at 24, 4, and 1 km resolutions include each grid's latitude and longitude coordinates that are used to calculate the grid box's area using the shoe-lace formula. The total TP area calculated by the sum of the areas of all the grid boxes approximates the true spherical TP area bounded by $(25^\circ - 45^\circ N) \times (65^\circ - 105^\circ E)$ with a difference 0.046 % for the 24 km grid and 0.033 % for the 4 km grid. The differences in the snow-cover area reported by the 24 km and 4 km grids vary between -2.34 and 6.24 %. The temporal variations of the daily TP snow cover are displayed in time series from 4 February 1997 to present with 4 km and 24 km resolutions.

The third chapter extends toolkit into a fully fledged web app for the Argo dataset. The Argo program has generated close to two million temperature, salinity, and pressure (T/S/P) profiles in the upper 2000 meters of the ocean. A new web app named Argovis at `www.argovis.com` provides easy access to Argo profile data and gridded products for both scientists and the general public. The RESTful application allows users or even other apps to interact with a database through the URL. In this case, users query a MongoDB database filled with Argo profiles. Users input a set of latitude-longitude coordinates, date range, and pressure range. The Argovis app sends a response: either raw JSON or HTML page with profile measurements and metadata that fall within these queries. Jupyter notebooks written in Python outline the API interface. Time series and gridded products are generated by API, offering a way for researchers to tailor the web app to their specific needs. The topics covered

in this thesis entail big data sets and how to visualize and retrieve data quickly. The issues posed in this thesis extend past 3D/4D climate data sets. Open source toolkits and web apps for data visualization and accessibility pertain big data sets in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PAGE

GLOSSARY

Glossary

.asc File that stores ascii characters.

AIC Akaike information criterion.

AMSU Advanced Microwave Sounding Unit.

API application programming interface.

ARGO Array for Real-time Geostrophic Oceanography is a global array of 3,800 free-drifting profiling floats whos primary measurement includes temperature and salinity of the upper 2000 m of the ocean..

AVHRR Advanced Very High-Resolution Radiometer.

AWS Amazon Web Services (AWS) provides on-demand cloud computing platforms to individuals, companies and governments, on a paid subscription basis..

CMC Common Mapping Client.

cron A time-based job scheduler in Unix-like computer operating systems. Cron enables scripts to run at periodically at a fixed time..

DAC Data Assembly Centres.

DAT NASA Sea Level Change Data Analysis Tool.

DataFrame 2D Table like object with columns of potentially different types. Similar to a SQL table or MS Excel spreadsheet..

dataseries A one-dimensional labeled array capable of holding any data type (integers, strings, floating point numbers, Python objects, etc.)..

DMSP Defense Meteorological Satellite Program.

document-oriented database Is a type of NoSQL database, used to store semi-structured data, often in key-value schemas, such as XML or JSON. Unlike traditional relational databases, the data model in a document database is not structured in a table format of rows and columns. This allows flexibility in data structures and modeling..

express.js The standard server web application framework for Node.js. Used to make web-apps..

FileZilla Free software, cross-platform FTP application, consisting of FileZilla Client and FileZilla Server. Client binaries are available for Windows, Linux, and macOS..

FOSS free open source software.

FTP File Transfer Protocol.

GDAC Global Data Assembly Centres.

GMS Geostationary Meteorological Satellite.

Hadoop Apache Hadoop is an open-source software framework used for distributed storage and processing of datasets of big data using the MapReduce programming model. It consists of computer clusters built from commodity hardware..

HDF5 Hierarchical Data Format is a set of file formats designed to store and organize large amounts of data..

IMS Interactive Multisensor Snow and Ice Mapping System.

Ipython IPython is an interactive command shell for the Python language..

Iridium Iridium Communications Inc. is a company that supplies Argo floats an antenna for data transmission..

JSON JavaScript object notation.

Jupyter A browser based notebook. Jupyter is a FOSS web application that allows, text, equations, and figure integration in a number of languages, includeing Python..

LAEAP Lambert azimuthal equal area projection.

leaflet Open source JavaScript library used to build web mapping applications. Includes plugins for drawing objects, using different map projections, and using different map tiles..

LOESS local regression.

METEOSAT European Weather Satellite.

MODIS Moderate Resolution Imaging Spectrometer.

MongoDB MongoDB (from humongous) is document-oriented database program, classified as a NoSQL. MongoDB uses JSON-like documents with schemas. MongoDB is developed by MongoDB Inc., and is published under a combination of the GNU Affero General Public License and the Apache License..

MVC Modelviewcontroller (MVC) is a software architectural pattern commonly used for developing user interfaces that divides an application into three interconnected parts..

NASA National Aeronautics and Space Administration.

NetCDF Network Common Data Form is a file storage system, denoted by .nc. It allows the creation, access, and sharing of array-oriented scientific data. The ARGO program releases data NetCDF on two FTP sites..

NH Northern Hemisphere.

NOAA National Oceanic and Atmospheric Administration.

node.js Node.js JavaScript run-time environment for executing JavaScript code server-side. Historically, javaScript was run on the client side only, with Node.js, javascript is run like any high-level interepreted language..

NSIDC National Snow and Ice Data Center.

psu practical salinity unit.

QC quality control.

RESTfull REpresentational State Transfer, or RESTful, web services provide interoperability between computer systems on the Internet..

rsync rsync is a utility for efficiently transferring and synchronizing files across computer systems, by checking the timestamp and size of files..

SACD snow cover area calculation and display.

SSMI Special Sensor Microwave Imager.

T/S/P temperature salinity pressure.

TP Tibetan Plateau.

TSM Tibetan Snow Man.

URL Uniform Resource Locator.

WGS-84 World Geodetic System established in 1984, is an Earth-centered and Earth-fixed terrestrial reference system and geodetic datum describing the Earth's shape, size, gravity and geomagnetic fields..

WMO World Meteorological Organization.

XML extensible markup language.

ACKNOWLEDGMENTS

Dr. Sam Shen - for his unrelenting encouragement and support. Dr. Barbara Baily - for her expertise in time series analyis. Dr. Fernando De Sales - for his keen eye for detail. This study is supported and monitored by The National Oceanic and Atmospheric Administration Cooperative Science Center for Earth System Sciences and Remote Sensing Technologies (NOAA-CREST) under the Cooperative Agreement Grant #: NA16SEC4810008. The authors would like to thank The City College of New York, NOAA-CREST program and NOAA Office of Education, Educational Partnership Program for full fellowship support for Tyler Tucker. The statements contained within the manuscript/research article are not the opinions of the funding agency or the U.S. government, but reflect the authors opinions.

CHAPTER 1
REVIEW OF DATA VISUALIZATION AND DELIVERY TECHNIQUES

The advent of big data has put pressure on the Mathematics and Statistics community to rethink how traditional mathematics/ statistics can be applied data availability on the scale of petabytes. Both climate and oceanographic science disciplines are feeling this data explosion and are struggling to scale. Unprecedented amounts of data are made available by governmental agencies like National Oceanic and Atmospheric Administration (NOAA) and National Aeronautics and Space Administration (NASA), and heavily used by the scientific community to draw inferences. The traditional visualization and delivery tools can be improved to handle a surge of new data delivery. The techniques used to visualize and retrieve these data needs to be reevaluated. Global predictions need modern web app technologies for professional and amateur to visualize and receive data quickly, accurately, and as painlessly as possible.

Traditional methods of parsing through a remote/local archive of files, selecting relevant files, opening the files and performing analysis on a single pc do not scale with big data. Scientists are spending more time navigating through data rather than finding discoveries. Worse still, amateurs in the form of young people, hobbyists, students and potential scientists get frustrated and give up on using these rich data sets. Toolkits and web apps are suggested in this thesis to relieve this pressure. Modern RESTfull app design gives the community a walking stick to help them wade through the big data mire.

It's worth mentioning here another big data paradigm. Distributed systems/cloud computing offers the scientific community a means to handle big data at the expense of complicated overhead and expensive cloud-based services, such as AWS. Most scientists do not want, or in many cases do not even need these services. They need a subset of big data, or to be able to break down a data set into smaller pieces for their studies. Cloud computing services do not address the issue of accessibility and visualization. Comparing toolkits/web apps to cloud platforms are not opposing viewpoints. Instead, they can be used (or not) to varying degrees.

An unsettling reality is that as new technologies unfurl, mathematicians, statisticians, and scientists are pressured to adopt them without question or complaint. Expecting the community to embrace new paradigms is impracticable, and branding scientists who are reluctant to embrace as 'old school' manifest the fallacy that new is better, young outpaces the

old, and what you learned yesterday has no use today. This fallacy also cloaks the issue with these new paradigms: they are not needed for most scientific applications. In the context of the problem posed in this thesis, we are interested in big data visualization and delivery, and tailor this solution to what the community wants/needs. In this scope, big data is divided piecemeal into relevant sections that can be visualized and delivered with minimal domain expertise.

Time series analysis and gridded interpolation of climate systems are usually done on local spatiotemporal realms. The mathematical models that describe complex systems simplify climate regimes to seasonal, spatial, and temporal scales small enough to be computed on a single machine. A butterfly's flapping its wings in the Amazon is not modeled for Gulf rainfall after all! In this sense, the methods used to simplify the data is under investigation. Models are evaluated by how well they fit different data sets, and it is of interest to the community to try as many models as possible and to compare results with each other. In this sense, data visualization and accessibility aid the scientific process, as it allows ease of data for modeling, and is means of comparing. In the next sections, we hope to introduce current visualization platforms available for two different datasets. A 3D gridded satellite product, and a 4D in situ ocean sensor array product.

1.1 INTERACTIVE MULTISENSOR SNOW AND ICE MAPPING SYSTEM DATA VISUALIZATION

The daily snow and ice coverage dataset over the Northern Hemisphere (NH) based on the Interactive Multisensor Snow and Ice Mapping System (IMS) maintained by the United States' NSIDC [16] is a useful tool for numerous applications, such as monitoring day-to-day weather variations for water resources and agriculture, and estimating the snowstorm severity for insurance industries and governmental agencies. The IMS data derives from remote sensing of multiple satellites, including NOAA POLES satellites, MODIS Aqua and Terra, as well as *in situ* observations. [10, 24] The dataset began on 4 February 1997 and has numerous users, ranging from professionals including climate scientists, satellite remote sensing experts, hydrological engineers to high school students and climate science enthusiasts. A few visualization sites cater to this broad audience, offering a quick visualization platform.

Rutgers University Global Snow Lab provides a daily snow-cover `https://climate.rutgers.edu/snowcover/` [25]. Here users can scroll through preprocessed images of the daily snow cover. Visualization is limited to scrolling through images. There are no zooming features, or time series offered.

The United States National Snow and Ice data center provide another image archive at `http://www.natice.noaa.gov/ims/` [17]. Here users also can download images of IMSs daily northern hemisphere snow and ice coverage. Additionally, they can view and download

United States, Alaska, Europe/Asia, and Afghanistan regions. As with Rutgers site, the users are unable to zoom or download the actual data. Furthermore, the actual square coverage for this site is limited to pixels on an image. Estimating square km areas are limited to a 3-day mean ice coverage chart, showing maximum and minimum bars along with a dotted line average shown in Figure 1.1

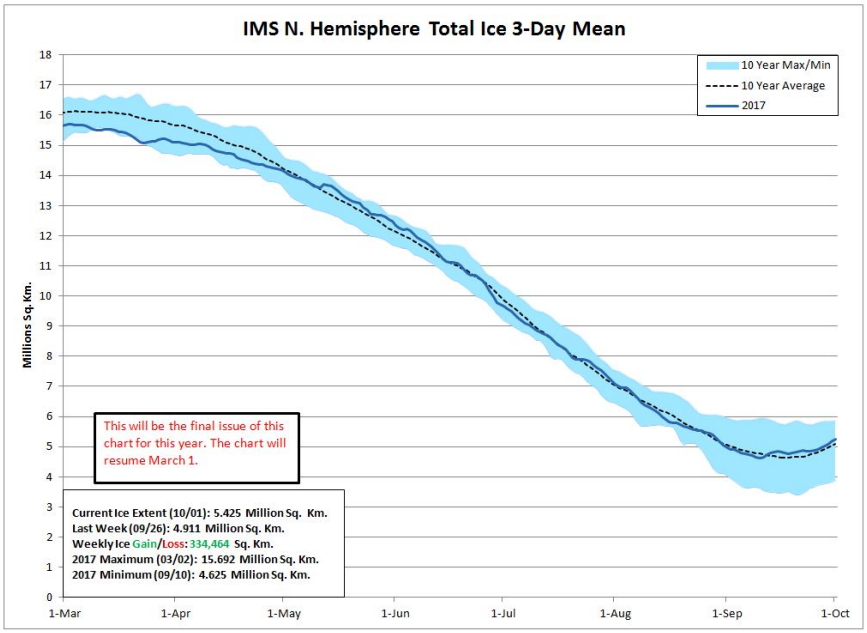

Figure 1.1. Sea and Lake Ice coverage of IMS data using 4 KM resolution. Ice coverages are calculated using a three day running mean from March until September each year. Blue border displays maximum and minimum values for the season. Areas are calculated using the Lambert Azimuthal Equal Area Projection with a WGS84 Datum. [17]

Users are left to examine two figures with square kilometers provided but are not able to see how the area is calculated. The IMS' calculation of the actual snow-covered area based on each grid box needs clarification. Because IMS provides images on a polar stereographic projection, the pixel area is a distortion of the true square kilometer coverage. Projecting a three-dimensional sphere/ellipsoid on a 2d plane using a stereographic projection stretches out low latitude areas and contracts high latitudes [27]. The IMS community would benefit to know how to make this calculation, or better still, have software calculates the true area for them.

Chapter 2 of this thesis describes a software toolkit written in Python that calculates areas of the coverage provided by the Northern hemisphere. This toolkit also parses through the IMS data set for a given latitude-longitude. This thesis shall provide the theoretical basis

for the toolkit, along with a time series analysis of the Tibetan Plateau Area. This toolkit is used by the Institute of Tibetan Plateau Research Chinese Academy of Sciences to produce daily snow coverage images located at `http://www.tpedatabase.cn/tibetSnow.jsp` [20].

Information and improvements regarding the toolkit shall be discussed, chiefly the opportunity for the creation of a web application that provides users with visualization and data retrieval built with modern website technology. Providing the source code limits the users to those who know Python; on the other hand, a web app connected to an intuitively designed user interface is a pragmatic way to serve the current IMS community to expand its user base.

1.2 ARGO VISUALIZATION AND DATA RETRIEVAL

For over a decade, the Argo program has provided temperature, salinity and pressure data (T/S/P) on a global scale for depths as far as 2000 decibar (dbar), with unprecedented spatial and temporal resolution and no seasonal bias [4]. Close to two million profiles have been collected and made publicly available on Argos Global Data Assembly Centres (GDACs) on File Transfer Protocol (FTP) servers. Data is recorded by various types of floats that drift at a resting depth and raise every few days to transmit information to GDACS. From there data underwent quality control markers and adjustments and released as a NetCDF product on FTP servers. Users can download either profile (one cycle of a floating measurement) or the entire profile history of a float. Though there are visualization tools available, almost all of them do not allow querying based on a spatial-temporal selection. For example, retrieving Argo profile data for North Sea regions at a certain depth and date. Chapter 3 will describe a web app that is used to allow such a data retrieval service.

1.2.1 Profile Cycles for Floats

Argo floats are equipped with an inflatable bladder, allowing them to float and sink. Once deployed the floats are programmed to drift at a 1000 meter depth for nine days. After nine days before descending to 2000 meters, then rising at a rate of 10 cm/s. While ascending the float sensors begin sampling parameters described in Table 1.1. Ascension takes 6 hours. Once at the surface the float transmits data via satellite to the GDAC, taking 6-12 hours for floats using the Argos1 and Argos2 satellite systems. [3].

As early as 2005, floats with GPS navigation have been introduced [31]. Iridium and Argos-3 satellite systems data transmitted in a matter of minutes. The ratio of ARGOS to GPS floats is displayed over time by Figure 1.2, collected from the Argovis database. According to this plot, Argos makes up $\approx 36\%$ of all profiles released, and GPS makes up $\approx 64\%$. Not only is GPS faster, but also allows higher frequency sampling. Floats that use GPS transmit

$\approx$ 1000 samples for each parameter. In contrast, Argos1 and Argos2 transmits only $\approx$ 100. The Argo dataset is growing as more floats are deployed *and* with increased sampling rates.

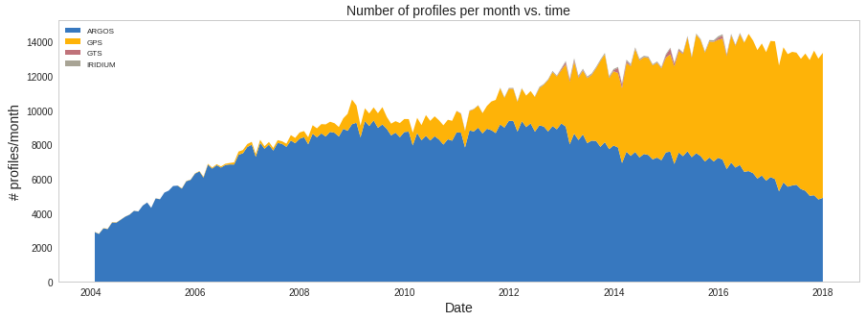

Figure 1.2. Satalite Communication Type by month. Iridium floats also use the GPS system.

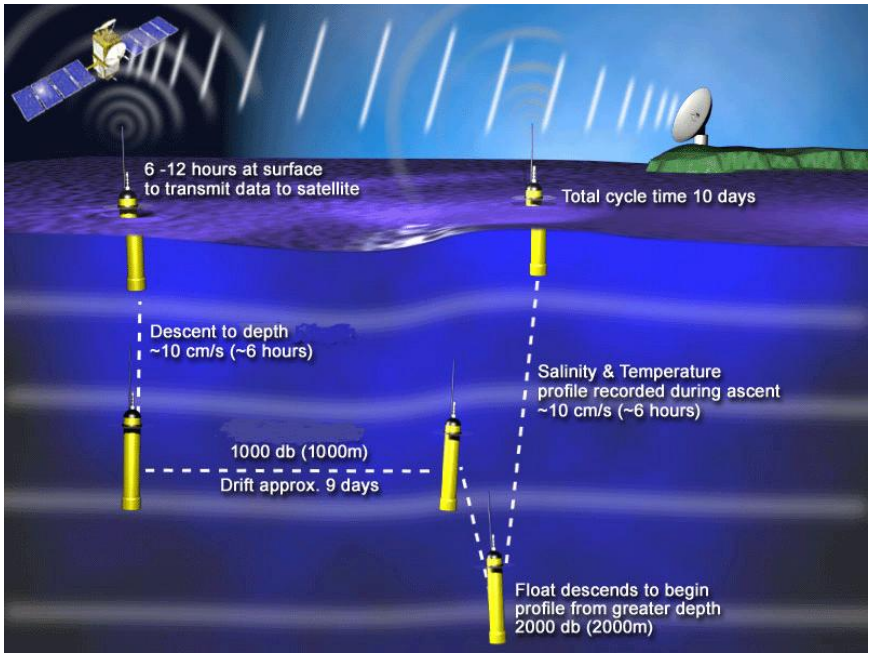

Figure 1.3. Detail of one profile cycle[4]

1.2.2 Float Hardware and Data Transmission

An international program such as Argo consists of many float types for different functions. In addition to T/S/P, some floats optic/biologic measurements, as shown in table 1.1

Table 1.1. Abridged list of Argo mission parameters.

Parameter name	long name	units
CNDC	conductivity	Siemens/m
PRES	sea water pressure	decibar
PSAL	practical salinity	practical salinity unit (psu)
TEMP	sea temperature	Celsius
DOXY	dissolved oxygen	$\mu Mol/kg$
BBPx	particle backscattering at x nanometers	m^-1
TURBIDITY	sea water turbidity	nephelometric turbidity unit (NTU)
CPx	Particle beam attenuation at x nanometers	m^-1
CHLA	chlorophyll-A	mg/m^3
CDOM	concentration of coloured dissolved organic matter in sea water	ppb
NITRATE	nitrates	$\mu Mol/kg$
BISULFIDE	bisulfides	$\mu Mol/kg$
PH_IN_SITU_TOTAL	pH	dimensionless
DOWN_IRRADIANCEx	downwelling irradiance at x nanometers	$W/m^2/nm$
UP_IRRADIANCEx	upwelling irradiance at x nanometers	$W/m^2/nm/sr$
DOWNWELLING_PAR	downwelling photosynthetic available radiation	$\mu Mol\ Quanta\ sec/m^2$

1.2.3 Data Delivery

All Argo data collected since its inception is available at a US-based GDAC FTP server at `ftp://usgodae.org/pub/outgoing/argo/` or a Europian Server at `ftp://ftp.ifremer.fr/ifremer/argo/`. Users navigate through directory structure with sparse documentation. Each DAC folder contains float and profile netCDF files. From there users can either download netCDF, either individually or bundled together by its float. The server is designed for those familiar with the Argo manual [31]. Each float file is named by its WMO number. Each profile file is designated by its WMO number appended by the cycle number and prefixed with a real-time/delayed-time marker. The amount of data available to the user is on the scale of hundreds of Gigabytes, yet there is no way to select data based on location, date, measured parameters, etc. A few opportunities to allow better navigation arise

if the Argo data is transferred to a database. By design, databases can be queried quickly by indexing frequently queried fields such as latitude, longitude, date, WMO, et cetera.

Databases are designed to accomplish storing and accessing large datasets. As Argo grows and expands, methods for reporting large and more complex data is essential to allow access to this information to as many people as possible. This database required to have an intuitive interface for selection, such as that made by the Coriolis group shown in Figure 1.4. The next section introduces a proposed database and web-application interface.

1.2.4 Argo Visualization Tools

The limitations of selecting by profile and platform number through an FTP server has caused visualization tools to sprout up among the Argo community. The majority of these applications are based on visualization, and their utility is somewhat limited to displaying points on a map. A few prominent ones are mentioned below. Their merits and limitations shall be covered in detail before revealing the Argovis web app covered in Section 1.2.5 and Chapter 3.

1.2.4.1 Coriolis Visualization Application

Coriolis, Argo's French group, includes an app showing profiles on a tile map interface.[2]. Users can pan, zoom in and out, and select floats on the map, represented by dots. See Figure 1.4 for a screenshot of the webpage. Moreover, the site includes velocity fields derived from drifting floats provided via the ANDRO product [21]. Data displayed on the map is limited to profiles reported within the past week. Upon clicking a point, a sidebar opens, revealing interactive charts of Pressure vs. Temperature and Pressure vs. Salinity. The side panel also includes meta-data: float identification number (WMO), date, DAC, project name, primary investigator and float model. Data can be viewed and downloaded in greater detail by clicking on links to another page.

Argo-active-network-map's primary function is visualization but restrictive even at that. Users can only view one profile at a time for profiles reported in the last seven days. More importantly, they restricted from making spatial-temporal selections. Tilemap interfaces have draw plugins that allow users to draw polygon shapes. These points can then be used to select and view profiles contained within that polygon. It can be useful for users interested in climate change to view data on decadal timescales. Nonetheless, the app provided by Coriolis is a significant advancement over the main source of accessing Argo data on FTP server.

1.2.4.2 NASA Sea Level Change Data Analysis Tool

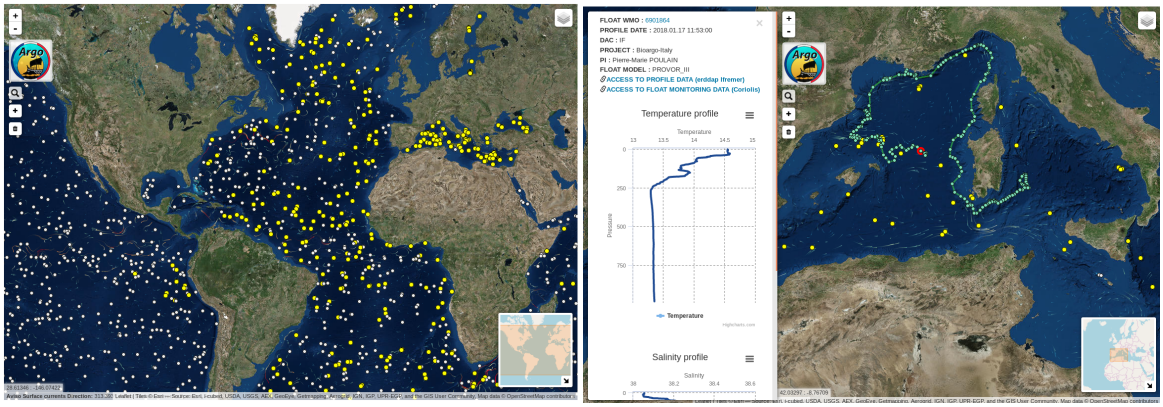

Figure 1.4. Argo Active network map (left) `http://www.argo-france.fr/en/argo-active-network-map/`**[2] Showing argo profiles owned by the Coriolis (French) DAC. Yellow dots represent floats operated by the Coriolis project. Selecting a dot reveals a side panel with additional information (right). Velocity field animations overlay the profile dots showing currents generated via ANDRO.**

The NASA Sea Level Change Data Analysis Tool (DAT) has been designed to allow for quick-look comparisons and analysis of NASA datasets of sea level change. The project is in its beta phase and can be viewed at `https://sealevel.nasa.gov/data-analysis-tool/`[15]. Gridded Argo products, among many other in situ and remote sensing products make up this app. The project states that its goal is to track sea level rise, it can visualize Argo temperature and salinity data.

DAT comprises of interactive map and side control bar shown on ee Figure 1.5. The mapping component uses an open source library Common Mapping Client (CMC) library, located at `https://github.com/nasa/common-mapping-client`.

CMC allows drawing points, lines and rectangles on the Mercator projection only. Lines vertically slice the ocean, whose temperature and salinity is viewed in a color chart. Shapes are editable, and plots are redrawn by clicking a new button. Unfortunately, this feature is not available for Argo data at this time.

Images of temperature and salinity overlay the CNC map. Users select which depth time, and an image of either temperature or salinity anomalies are displayed. See Figure 1.6

Views and data are accessed through the URL. This RESTfull design extends the app beyond the browser. It allows API extensions data access without the need of the user manually retrieving data by a series of clicks on the browser. Scientists can automate these tasks for Repeatability and scalability.

DAT is an unprecedented tool in making selections from large climate data sets, yet some features limit its potential to be widely used by the Argo community.

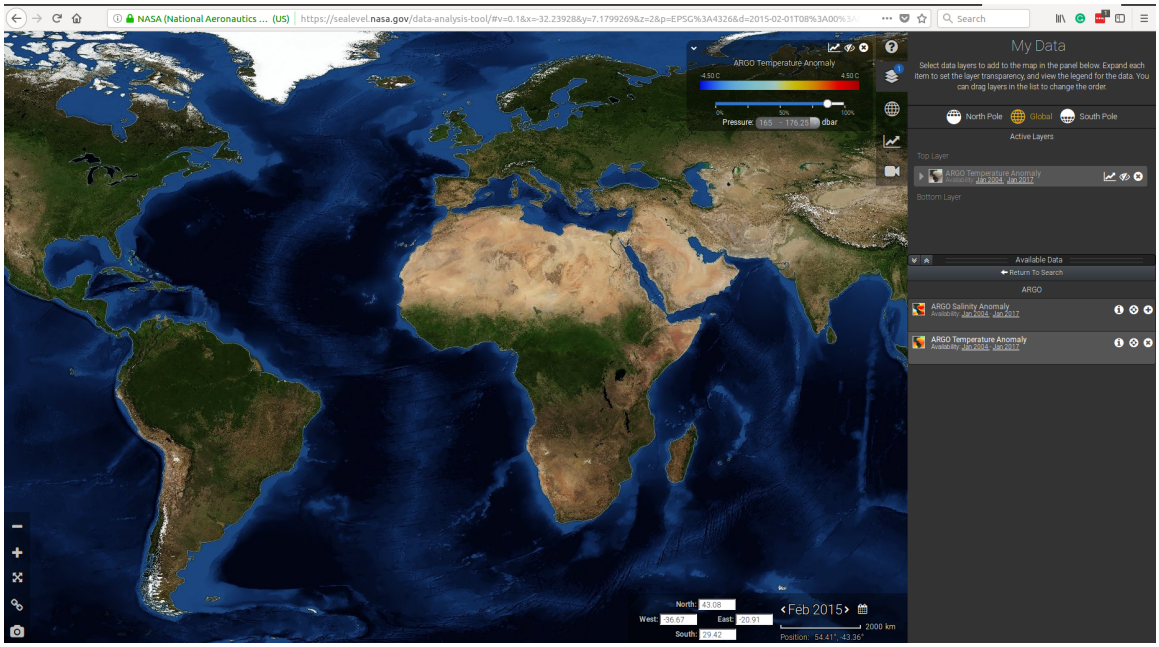

Figure 1.5. DAT web app format with Data tab displayed on the side bar. Argo derived gridded salinity and temperature products from January 2004 to January 2017 are available to plot.

1. The map cuts off at 180 degrees latitude. Unfortunately, the map cuts off in the middle of the Pacific basin.
2. It is unclear how the gridded product interpolates Argo profiles. With no reference to the source, scientists are reluctant to use this data other than visualization.
3. The CMC library is capable of plotting points, yet Argo products are limited to gridded products. DAT would benefit from having a platform plotting feature, similar to the app covered in Section 1.2.4.1.
4. Does not allow images to show using different projections. South Sea and North Sea ocean views are limited. Stereographic projections would better serve these regions of interest. Storing data and plotting it using a heat map plugin displays the data in any map projection.

The app generates plots on the server-side. The client side doesn't get strained, making the app more responsive. The downside here is that the server requires more resources to work efficiently, making the app expensive to run and maintain.

This project sets a new precedence for data delivery and visualization. However, the cons may discourage its use as a scientific tool. In the later sections, these pros and cons are addressed. A compromise attempted that balances visualization features with data transparency in Section 1.2.5.

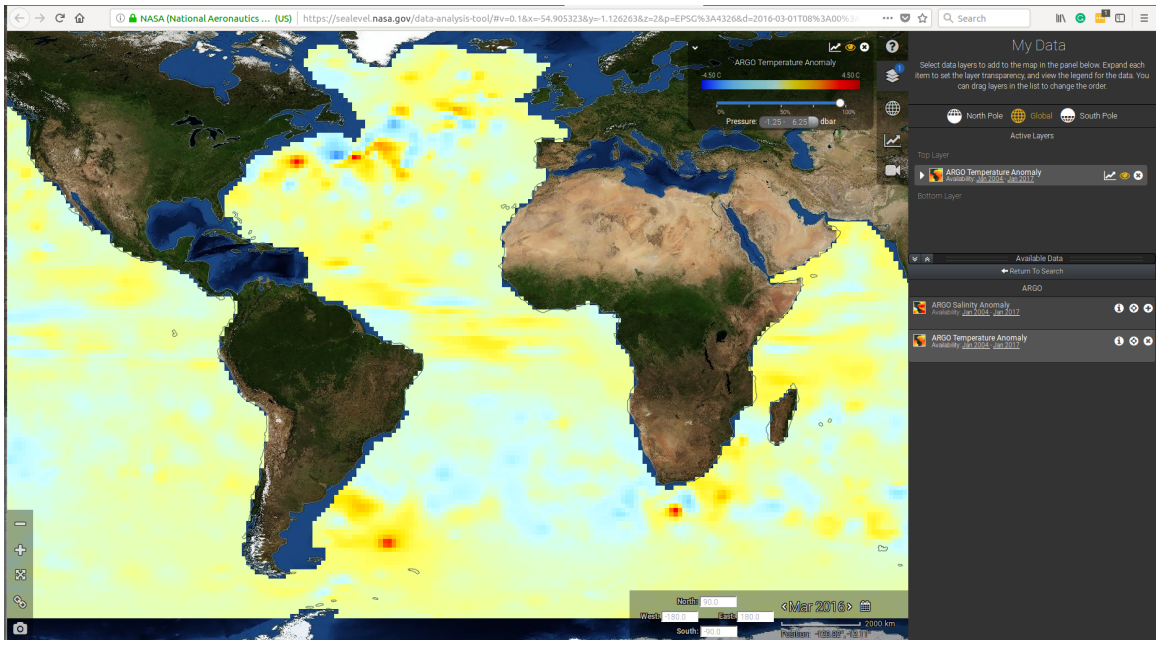

Figure 1.6. DAT web app format with temperature anomalies layer displayed. Anomolies are for March 2016, indicated by the date selection on the bottom right. Pressure ranges are from -1.25 to 6.25 dbar, as indicated by the drop-down menu shown on the top right portion of the map.

1.2.4.3 4D Visual Delivery of Big Climate Data

Pierret et alli released another RESTfull app that displays big data sets on either a Web Mercator projection or a 3d globe [23]. A 3d orthographic projection greets users upon visiting `http://climate2.mrsharky.com/`. Different views are made by spinning and zooming the globe, shown in Figure 1.7.

Data is added after selecting the sidebar tab and choosing different data sets in a drop-down menu. Argo data sets are not currently available; gridded data can be added similar to DAT.

MrSharky will also generate time series by clicking on the globe (Figure 1.7). A popup window indicates the data value and the location in latitude-longitude coordinates. A screenshot showing the data is generated and downloaded with a click of a button.

This app is intuitive enough to use, yet allows a great deal of customization. The sidebar control lets users select different projections, datasets, color bars, adjust sphere resolution, et cetera.

The bulk of the server side involves the transfer of HTML scripts and data[23]. Pushing the plotting, rastering, and graphics manipulation tasks to the client side unburden the

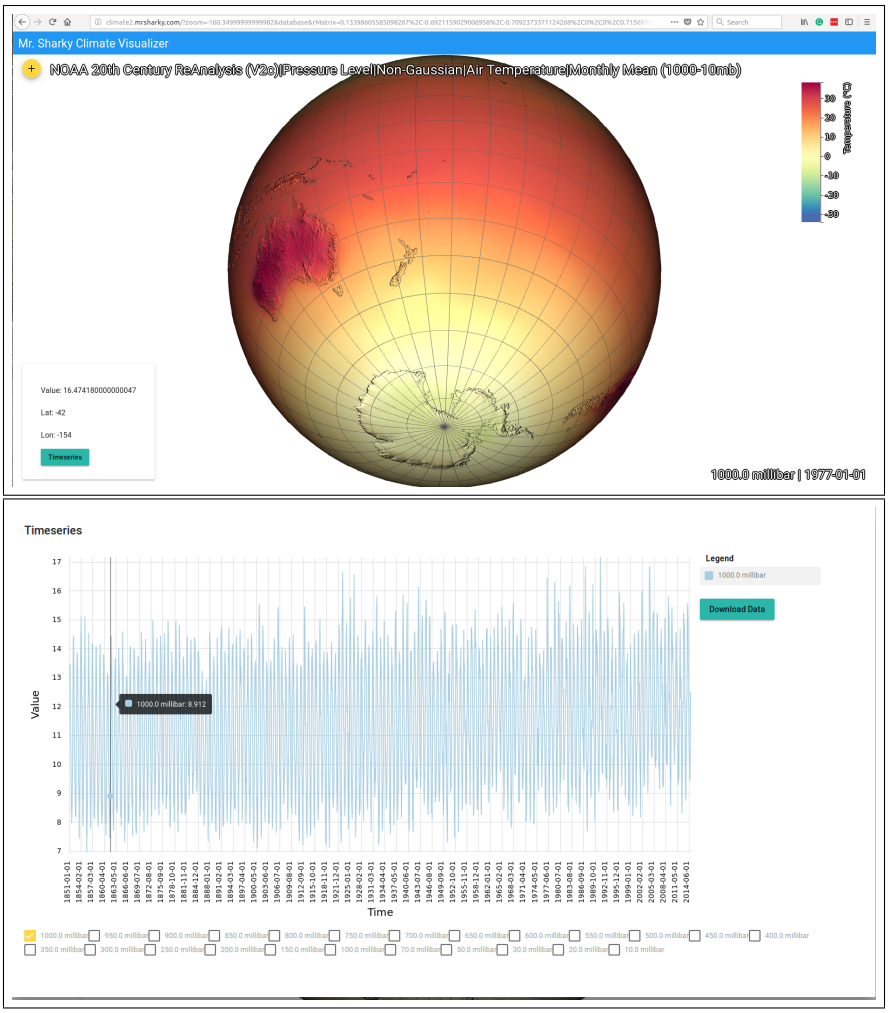

Figure 1.7. Clicking a point on the globe will generate a box element showing the point's value and latitude-longitude coordinates, shown in the lower left corner of the (top) subfigure. Clicking the teal button will generate a time series (bottom) that can be downloaded as a .csv file.

server, reducing the hardware and networking resource requirements and lowering the cost to run the app.

MrSharky's architecture and data delivery method has influenced the web app described in Section 1.2.5 and in Chapter 3 of this thesis.

1.2.5 Visualization and data retrieval tool Argovis

This thesis introduces a new website at `www.argovis.com` that allows easy navigation of profiles from the Argo GDACs. This web app offers a simple user interface and allows both scientists and the general public to draw polygons on an interactive map and view the profiles within given date and depth ranges. After selection, the user may browse T/S/P profiles on interactive charts, either from a single float or those found within the drawn polygons. Some

metadata about the floats are displayed as well, along with a list of links to locate the data of interest on the GDACs.

Overall, Argovis offers seamless navigation of the Argo dataset written with a representational state transfer (REST) architecture. RESTful design allows the opportunity to feature API and cloud computing applications, e.g., map comparison from existing gridded Argo products, as well as derived variables. Argo now has a maintainable, scalable, and portable tool to support this ever-expanding big data set. Chapter 3 shall provide further detail on the web app's front end and back end architecture and database information, along with some time series and grid-interpolation applications.

CHAPTER 2
DAILY SNOW COVER MONTITORING FOR THE TIBETAN PLATEAU

2.1 INTRODUCTION

Handling Interactive Multisensor Snow and Ice Mapping System (IMS) data at several resolutions requires significant data management efforts. Universal tasks include: Parsing through dates, filtering latitude and longitude coordinates, calculating snow/ice areas, and other functions that characterize snow cover area calculation and display (SACD). Individual groups who use IMS data dedicate time and resources developing and maintaining code built to handle these tasks. Currently, there is no free open source software (FOSS) available for users. Consequentially groups and users are developing personal libraries to handle data, where they probably should be analyzing and drawing inferences from that data. In response to this challenge, a convenient toolkit for the snow-SACD has been created, with the Tibetan Plateau (TP) used as an example of the toolkit's capabilities. The TP region, bounded by (25,45)°N and (65,105)°E, also known as the Third Pole of the Earth and includes the Himalaya Mountains, is used as an example to describe the SACD's method and usage. TP region's average altitude is over 4000 meters. TP's snow cover area has a large annual cycle and a considerable temporal variation due to TP's strong interaction with the atmospheric flows during the Indian and the East Asian monsoons in the summer, as well as the westerlies from the Eurasian continent in the winter. These interactions significantly affect the Earth surface's heat exchange with the atmosphere and can then influence the climate of southern and eastern Asia, even the global climate [33, 32]. Thus, TPs snow cover has a notable influence on its regional and global climate. Its accurate monitoring is essential for understanding the climate variations of both TP and the world. Furthermore, accurate snow-cover information that the IMS data provides is critical for weather prediction and hydro-logical resource management efforts. In one example, Peings and Douville [2010] correlated the Indian monsoon and Eurasian Snowfall. They used 0.5 resolution satellite data over a long timescale. A finer resolution product such as the IMS can be used to re-evaluate their findings. Another study by Wu and Qian [2003] provided evidence for positive correlations between Tibetan snow and the East Asian Monsoon. They considered not only snow-covered area but also the snow depth from the field measurements of 60 Tibetan stations. The high-resolution IMS data can help more accurately quantify their results of snow

cover. Although Shen et al. [2015] described the spatiotemporal variation of the TP snow cover using the IMS data, their results displayed in map projection percentage coverage. They did not provide a quantitative snow-cover area calculation in square kilometers and did not explore the actual TP surface area of each grid box in the polar stereographic projection. The nominal 24 km resolution on the projection plane needs to find its corresponding Earth surface grid boxes and their associated areas. The SACD toolkit described in this thesis is designed to solve this problem by providing each grid boxs surface area as shown in Figures 2.4 and 2.6. It can display IMS data with maps, and with a geographic background, and hence enriches the IMSs software suites. Thus, the toolkit is useful for further study of snow cover, snow depth-snow water equivalence, climate forecast, and hydrological engineering planning and management. The SACD toolkit can map any region on the northern hemisphere In addition to this thesis, the website (www.itsonlyamodel.us) was created to explain how to use the SACD toolkit, named Tibetan Snow Man (TSM) for its application to the TP region, and is written in python. MIT open-source license ensures TSM will always remain FOSS. TSM is available on GitHub along with an Ipython notebook that provides a walk-through on how to use it [Tucker (2017a,b)]. Users can expand on TSM to develop similar products for other regions over the northern hemisphere. Two datasets on the Ipython notebook in TSM can be used to create time series of snow cover. Additionally, `http://www.itsonlyamodel.us` has plotting features that facilitate snow-cover visualization, compute various kinds of statistics, and interpret the climate conditions from the snow-cover data for the study region.

Section 2.2 describe data and methods, Section 2.3 contains results, and the conclusions are in Section 2.5.

2.2 Data and Methods

2.2.1 Interactive Multi-sensor Snow and Ice Mapping System (IMS)

IMS [16] uses both satellites remote sensing data and in situ observations, including Advanced Very High-Resolution Radiometer (AVHRR), the Moderate Resolution Imaging Spectrometer (MODIS), the Geostationary Operational Environmental Satellites (GOES), the Geostationary Meteorological Satellite (GMS), the European Weather Satellite (METEOSAT), the Special Sensor Microwave Imager (SSMI), the Defense Meteorological Satellite Program (DMSP), the Advanced Microwave Sounding Unit (AMSU), and field measurements. The current daily snow cover data are blended results of these remote sensing and ground observations, with three different spatial resolutions: 24 km, 4 km, and 1 km.

The 24 km in this resolution refers to the unscaled distance of a stereographic projection. Mapping a 3D surface to a 2D plane via stereographic projection introduces scale

distortion [27]. The point at which the scale error is zero is what the IMS refers to as the resolution. Each point on the projection has a latitude-longitude (lat-lon) coordinate, determined by a set of assumptions.

2.2.2 Mathematics of the stereographic mapping

For the coarse 24x24 resolution, models the Earth as a sphere of a fixed radius of 6371.2 km [16]. The stereographical projection transformation is obtained by one-to-one mapping between a point on the projected plane with Cartesian coordinates (x, y) and a point on the sphere with lat-lon coordinates (ϕ, λ). Following [27], the mapping formulas from a sphere point (ϕ, λ) to a Cartesian point (x, y) are below:

$$x = 2R\, k_0 \tan(\frac{\pi}{4} - \frac{\phi}{2}) \sin(\lambda - \lambda_0), \tag{2.1}$$

$$y = -2R\, k_0 \tan(\frac{\pi}{4} - \frac{\phi}{2}) \cos(\lambda - \lambda_0). \tag{2.2}$$

The length scale factor k for the projection is given by

$$k = \frac{2k_0}{1 + \sin(\phi)}, \tag{2.3}$$

and (ϕ_0, λ_0) is a designated center point. IMS uses $(\phi_0 = 90°, \lambda_0 = 80°)$. The stereographic projection is conformal on the same latitude as k is fixed in all directions. The $k_0 = 1/2 + \sqrt{3}/4$ is the k value for the designated point and can be determined by the following method. We set the constraint that the scale factor $k = 1$ at $60°$

$$k = 1 = \frac{2k_0}{1 + \sin(\phi = \frac{\pi}{3})} \tag{2.4}$$

Solving this equation for $k_0$yields

$$k_0 = \frac{1 + \frac{\sqrt{3}}{2}}{2} = 1/2 + \sqrt{3}/4 \approx 0.933013, \tag{2.5}$$

since $\sin(\pi/3) = \sqrt{3}/2$. Thus, k is simplified to

$$k = \frac{1 + \frac{\sqrt{3}}{2}}{1 + \sin(\phi)}. \tag{2.6}$$

The inverse mapping from Cartesian coordiantes (x, y) over the projected plane to the sphere's latitude and longitude coorinates (ϕ, λ) is below:

$$\lambda = \lambda_0 + \arctan(x \,/\, (-y)), \tag{2.7}$$

$$\phi = \arcsin(\cos(c)), \tag{2.8}$$

where c is the angle between the (x, y) point and the origin (South Pole).

$$c = 2\arctan(\frac{\rho}{2Rk_0}) \tag{2.9}$$

$$\rho = \sqrt{x^2 + y^2} \tag{2.10}$$

The finer resolutions of 4 km and 1 km require better accuracy of the mapping and require us to treat the Earth as an ellipsoid, because a round ball Earth assumption can lead to errors at finer resolutions. IMS uses the WGS-84 ellipsoid to model the Earth for the 4x4 and 1x1 km resolutions. The mapping formulas are shown below.

$$x = \rho\ \sin(\lambda - \lambda_0), \tag{2.11}$$

$$y = -\rho\ \sin(\lambda - \lambda_0). \tag{2.12}$$

Here, ρ is scale constant occurring at latitude $\phi_c = 60^\circ$ defined as

$$\rho = a\ m_c \frac{t}{t_c}, \tag{2.13}$$

where t and m_c are expressed by

$$t = \left[\left(\frac{1 - \sin(\phi)}{1 + \sin(\phi)}\right)\left(\frac{1 + e\sin(\phi)}{1 - e\sin(\phi)}\right)^e\right]^{\frac{1}{2}}, \tag{2.14}$$

$$m_c = \frac{\cos(\phi_c)}{(1 - e^2\sin^2(\phi_c))^{\frac{1}{2}}}, \tag{2.15}$$

e is the ellipsoid's eccentricity

$$e = \left(1 - \frac{b^2}{a^2}\right)^{\frac{1}{2}}, \tag{2.16}$$

a and b are the ellipsoid major and minor axes respectively, and t_c is t at $\phi = \phi_c = \pi/3$. Using $\cos(\phi_c) = \frac{1}{2}$ and $\sin(\phi_c) = \frac{\sqrt{3}}{2}$, we can evaluate m_c and t_c

$$m_c = \frac{1/2}{(1 - 3e^2/4)^{\frac{1}{2}}} \tag{2.17}$$

$$t_c = \left[\left(\frac{1 - \frac{\sqrt{3}}{2}}{1 + \frac{\sqrt{3}}{2}}\right)\left(\frac{1 + e\,\frac{\sqrt{3}}{2}}{1 - e\,\frac{\sqrt{3}}{2}}\right)^e\right]^{\frac{1}{2}} \tag{2.18}$$

The scale factor for this ellipsoid's projection is

$$k = \frac{\rho}{a\, m} \tag{2.19}$$

The inverse formulas for the north polar ellipsoidal stereographic mapping are as follows:

$$\phi = \frac{\pi}{2} - 2\arctan\left[t\left(\frac{1 - e\sin(\phi)}{1 + e\sin(\phi)}\right)^{\frac{e}{2}}\right] \tag{2.20}$$

$$\lambda = \lambda_0 + \arctan(x \,/\, (-y)) \tag{2.21}$$

Equation (2.20) defines an implicit function for latitude ϕ and can be solved iteratively.

Mapping both the sphere and ellipsoidal surface to a plane using stereographic projection results in the points far from the origin to bunch close together, and the points near the origin to stretch apart. In the case of IMS, there are more points further at the equator, which means that the lat-lon grid is non-uniform. The distances between the points decreases with latitude as shown by Figure 2.1

This non-uniformity can be a source of confusion, leading to misinterpretations of the square kilometer estimation of snow coverage. This paper attempts to eliminate this confusion by explicitly calculating the surface area around each grid box of the 24×24 and 4×4 km resolution datasets.

2.2.3 IMS data description

IMS stores its data in zipped .asc files for each day. Each file includes header and a body. The body is in the form of a $n \times n$ matrix. The 24×24 km resolution has $n = 1024$ and 4×4 km resolution has $n = 6144$. Each point on the body is digits 0, 1, 2, 3, and 4 representing categorical information listed in Table 1.

Table 2.1. Meaning of the IMS categorical data.

Data Marker	Meaning
0	Out of the Earth surface range
1	Water
2	Land
3	Ice
4	Snow

The .asc files provided have two different formats; one described earlier and another deprecated format using 164 and 165 for ice and snow respectively. The latter format, occurring for years 1997 and 1998, does not include land and water values and uses irregular

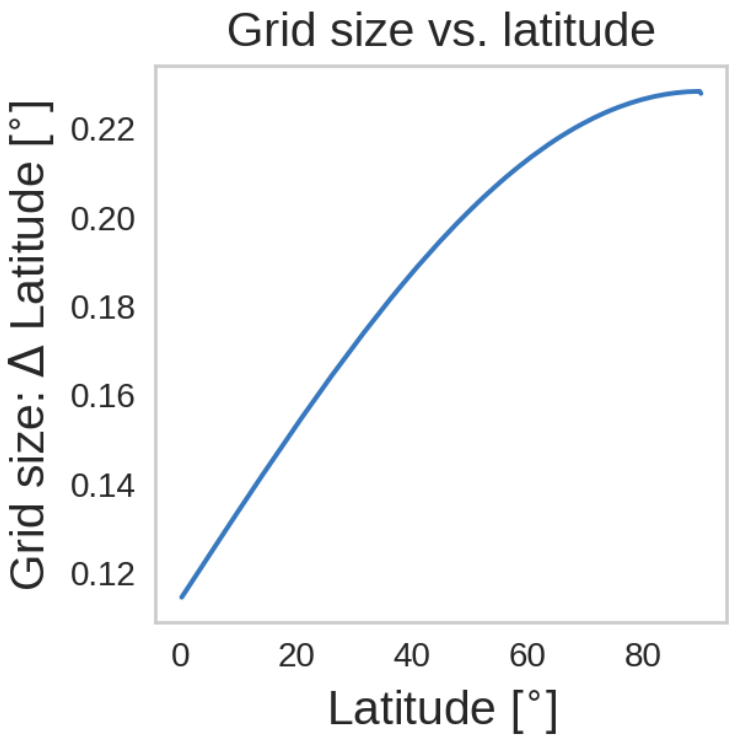

Figure 2.1. Latitude was taken from IMS latitude file: rows 0 to 512, column 512, points plotted lie approximately at $80°E$. Grid points are distributed densely near the equator, with latitude grid spacing of $.11°$ at the equator. The latitude grid spacing increases to $.23°$ at the north pole.

line breaks. TSM reformats the deprecated .asc files to include land and water digits and reshapes the data to match the standard $n \times n$ matrix format.

For example, one .asc file displayed below shows only ice and snow (164 and 165), with irregular line breaks

```
0   164   164   164   0   164   0   0   165   0   0   0
0   0     0     0     0   164   164 164 165   0   0   0
```

TSM later adds in land and sea values and converts the ice and snow values to 3 and 4, respectively. It also removes the irregular line breaks. The final data format is shown below.

```
13331312422222222334000
```

Under the new format, data for each day is in the an $n \times n$ matrix format, indexed by (row, column). The data matrix starts at (0,0) and ends at the point (n-1,n-1) See Fig. 2.2 for the $n \times n$ matrix coordinates.

Figure 2.2 is an illustration of the grid coordinate ticks on a stereographic projection square from NH without snow and ice information. The four black corner areas are marked as 0, which are out of the Earth surface range in the stereographic projection. The prime meridian in Fig. 2.2 is rotated clockwise by about 10.22°. Additionally, the data is provided as a mirror image to what is shown in Figure 2.2.

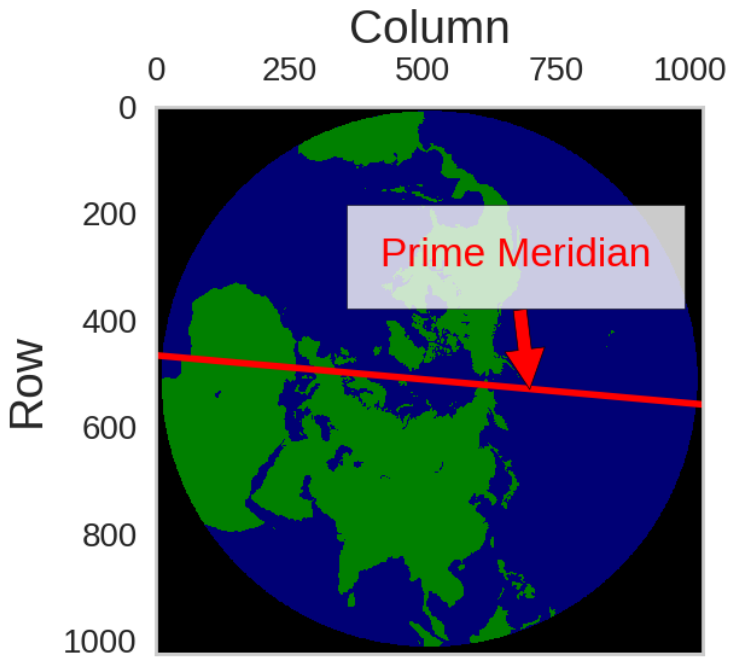

Figure 2.2. Grid coordinate ticks of NH without snow and ice cover. IMS provides daily files in an $n \times n$ format. The data is given in .asc files but mirrored to the figure shown. The file represents a stereographic projection, with the 80° meridian line pointing up. IMS rotates the prime meridian by about 10.22° clockwise to the horizontal. This figure shows the 24km resolution data, which has n = 1024.

2.2.4 IMS data delivery

Data is placed on a FTP server at `ftp://sidads.colorado.edu/pub/DATASETS/NOAA/G02156/`. FTP directory includes separate folders for lat-lon grids (metadata) and for each resolution. Data can be downloaded by selecting files in a directory while working in the browser. This process can be automated an FTP client/server software such as FileZilla. The dataset can be downloaded in parts or in

its entirety and stored locally for manipulation. FileZilla can also syncronize local and remote directories. FTP server updates daily for each resolution folder.

2.2.5 IMS data manipulation and analysis

Each IMS data file corresponds to one day. TSM main routine parses through each file and calculates the given area of snow and ice coverage. Python's Numpy, Pandas, and H5py libraries are used to handle daily datasets, grid area estimation, and database archives. TSM filters each file by the appropriate latitude and longitude and flattens it into a dataseries object. TSM groups each series by year and stores them as a Hierarchal Data Format 5 (HDF5) file for each year. Basemap library is used for the projection transformations and plotting contours on maps.

TSM's main task calculates snow coverage for a given region. First, the .asc files are loaded into TSM, filtered to a region and then flattened into a vector, with each element is given a categorical index, notated by $T_k \in \{0, 1, 2, 3, 4\}$. T_k is transformed into $B_k \in \{0, 1\}$ where ice and snow values are transformed to 1 and everything else is transformed to 0. Each index has a corresponding area, computed by the shoe-lace formula and denoted by A_k. The total area of snow and ice cover for a given day t is simply the dot product between vectors B_k and A_k:

$$Area_t = B_k^T A_k \tag{2.22}$$

Superscript T is matrix transpose operator. The resulted data vector $Area_t$ is added to an HDF5 database for each year.

A_k is calculated by interpolating the latitude (lat) and longitude (lon) points, provided by NSIDC [16]. The details are further explained later in this section. Latitude and longitude stored as binary .bin files and are loaded into TSM and formatted into $n \times n$ latitude and longitude matrices $\phi_{ij}, \lambda_{ij} \in \mathbb{R}$, with i being the row of the .asc file and j the column.

Map projections showing snow and ice coverage from data files are generated using the Basemap library. TSM code has been placed in the public domain GitHub [28]. A blog about the TSM usage is posted on www.itsonlyamodel.us [29].

TP region $25° - 45°$N and $65° - 105°$E is shown in Figure 2.3. The grid box areas are shown by the contour plot in Figure 2.3 illustrate how the stereographic projection distorts the latitude spacing as latitude approaches the north pole, becoming coarser as latitude increases. Hence the area of a grid box varies according to latitude, as shown by the color contour in Figure 2.4. One of TSM's main features is that it can calculate these grid areas.

Before calculating the area of a grid box, TSM estimates the box's corner points. Four points define a grid box, heron called centroids, because they are found by interpolating between four given lat-lon coordinates, (i,j), (i-1,j-1), (i,j-1), and (i-1,j). A zoomed in section

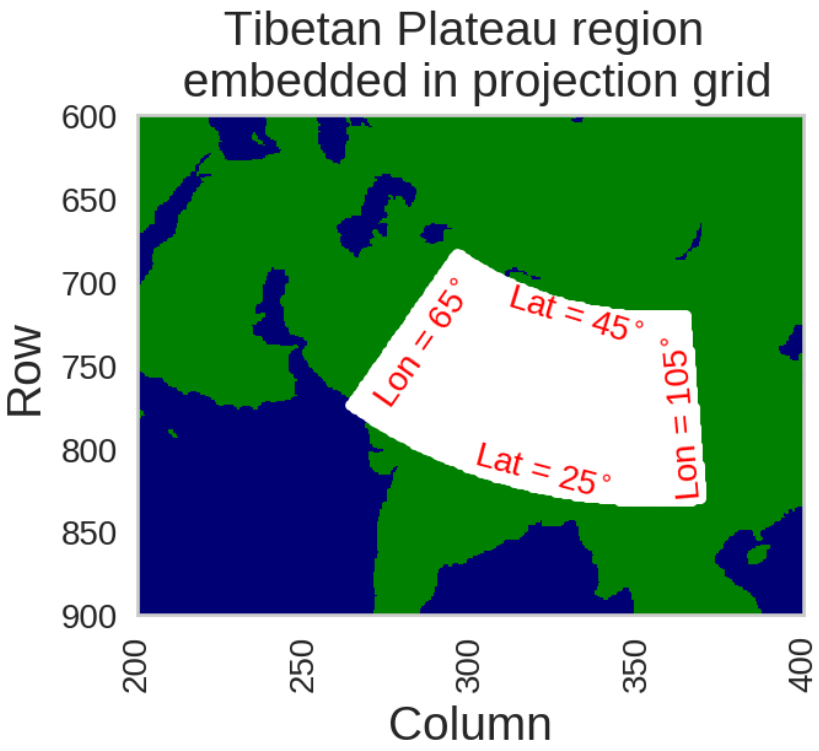

Figure 2.3. The TP region ($25° - 45°$N and $65° - 105°$E) shown bounded in white is shown on a stereographic projection.

showing this detail is provided for further clarification (see Figure 2.7 and 2.5). The surrounding four centroids $C_{i,j,k}$, indexed by $k = 1, 2, 3, 4$ for a given point i, j are calculated as shown below.

$$C_{ij1} = \frac{1}{4}(p_{i-1,j-1} + p_{i,j-1} + p_{i,j} + p_{i-1,j}) \tag{2.23}$$

$$C_{ij2} = \frac{1}{4}(p_{i-1,j} + p_{i-1,j+1} + p_{i,j+1} + p_{i,j}) \tag{2.24}$$

$$C_{ij3} = \frac{1}{4}(p_{i,j} + p_{i,j+1} + p_{i+1,j+1} + p_{i+1,j}) \tag{2.25}$$

$$C_{ij4} = \frac{1}{4}(p_{i-1,j} + p_{i,j} + p_{i+1,j} + p_{i-1,j+1}) \tag{2.26}$$

The area bounded four corners are calculated by first mapping them onto a 2d surface, and then by using the shoe-lace formula. See Figure 2.5 for reference.

The points are then projected onto a flat surface via the Lambert Azimuthal Equal Area Projection (LAEAP)[27]. LAEAP converts the lat-lon coordinates into Cartesian point x,y in meters about an arbitrary reference point ϕ_1, λ_0 using the following equations.

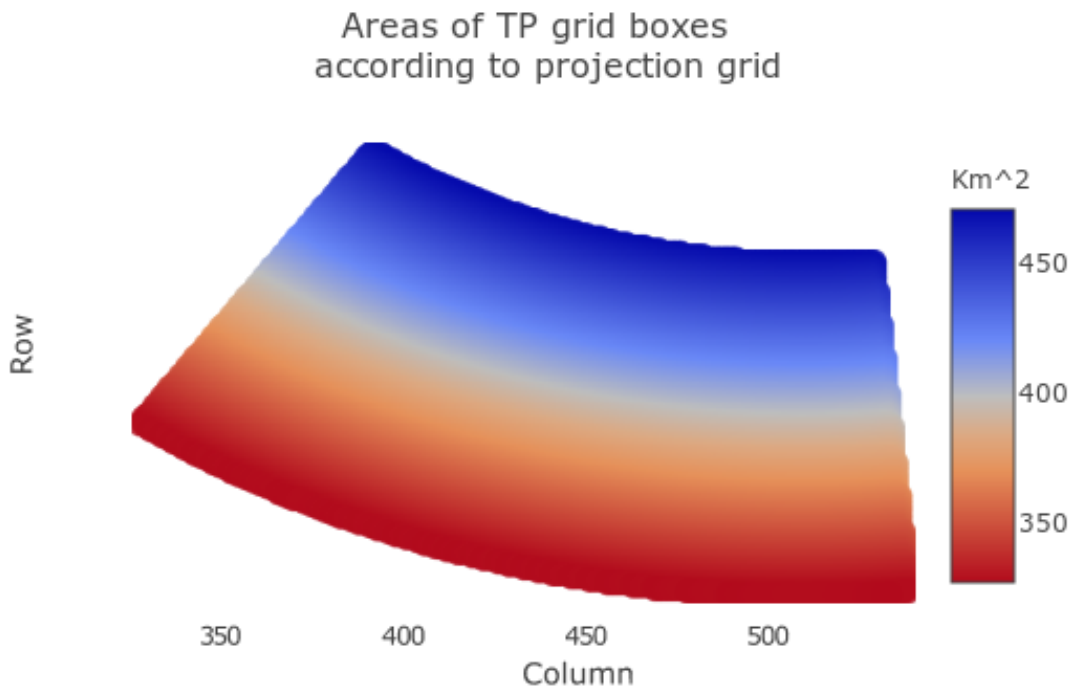

Figure 2.4. Color chart bottom shows grid box areas calculated using shoe-lace formula

$$x = Rk' \cos\phi_1 \sin(\lambda - \lambda_0) \tag{2.27}$$

$$y = Rk'[\cos\phi_1 \sin\phi - \sin\phi_1 \cos\phi \cos(\lambda - \lambda_0)] \tag{2.28}$$

$$k' = \frac{\sqrt{2}}{\sqrt{1 + \sin\phi_1 \sin\phi + \cos\phi_1 \cos\phi \cos(\lambda - \lambda_0)}} \tag{2.29}$$

Where $R = 6,371km$ is the Earth's radius. For the TP, $\phi_1 = 35°$ $\lambda_0 = 85°$ Treating these points as corners of a polygon, the areas are found using the discrete version of Green's Theorem, also known as the shoe-lace [8] formula shown below.

$$A_{ij} = \frac{1}{2}|\sum_{k=1}^{3} x_k y_{i+1} + x_n y_1 - \sum_{k=1}^{3} x_{k+1} y_k - x_1 y_n| \tag{2.30}$$

where $A_{i,j}$ is area of point i, j, $k = 1, 2, 3$ represent the centroids that make up the corners of the polygon, x_k and y_k are vectors containing respective horizontal and vertical centroid coordinates. Once all the points were calculated, the ratio between largest and smallest grid area in Figure 2.6 was measured as $A_{ratio} = 1.44$. To compare the areas gathered by the TSM

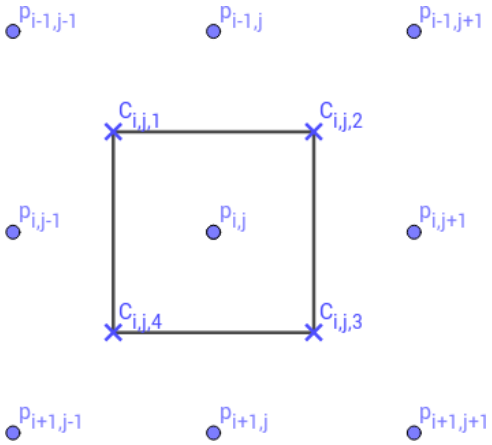

Figure 2.5. Grid box showing grid box corners $C_{i,j,k}$ for lat-lon point $p_{i,j}$ calculated using shoe-lace formula (2.30).

project, an exact solution of the surface area of the TP region calculated by l'Huilier's formula for spherical triangles gives an area of 8,059,061.81 km^2. The 24×24 km area estimation made by Lambert equal area projection using the shoe-lace formula is 8,062,815.10 km (0.046 % difference). The 4x4km area estimation is 8,061,716.81 km^2 (0.033 % difference).

Zoomed in a small section, Figure 2.7 shows the four points surrounding each region.

IMS collects Snow data daily. The algorithm indicates snow and ice with a 1 and replaces land and sea with 0. That is, a given vector containing given values $T_k \in \{0, 1, 2, 3, 4\}$ is transformed to $B_k \in \{0, 1\}$.

$$T_k \in \{0, 1, 2, 3, 4\} \mapsto B_k \in \{0, 1\} \tag{2.31}$$

The vector B_k and the area vector A_k is multiplied together by a dot product.

$$Area_t = B_k^T A_k \tag{2.32}$$

As an example, the top five points in Figure 2.8 are covered in snow and are have B_k values of 1. Shown in Figure 2.7, the corners surrounding point i,k are calculated by taking the centroid of its surrounding four points.

2.2.6 Comparison with exact solution

The Earth is modeled as a sphere of radius $R = 6,371\ km$ The Haversine function calculates the great circle distance between two lat-lon points in degrees.

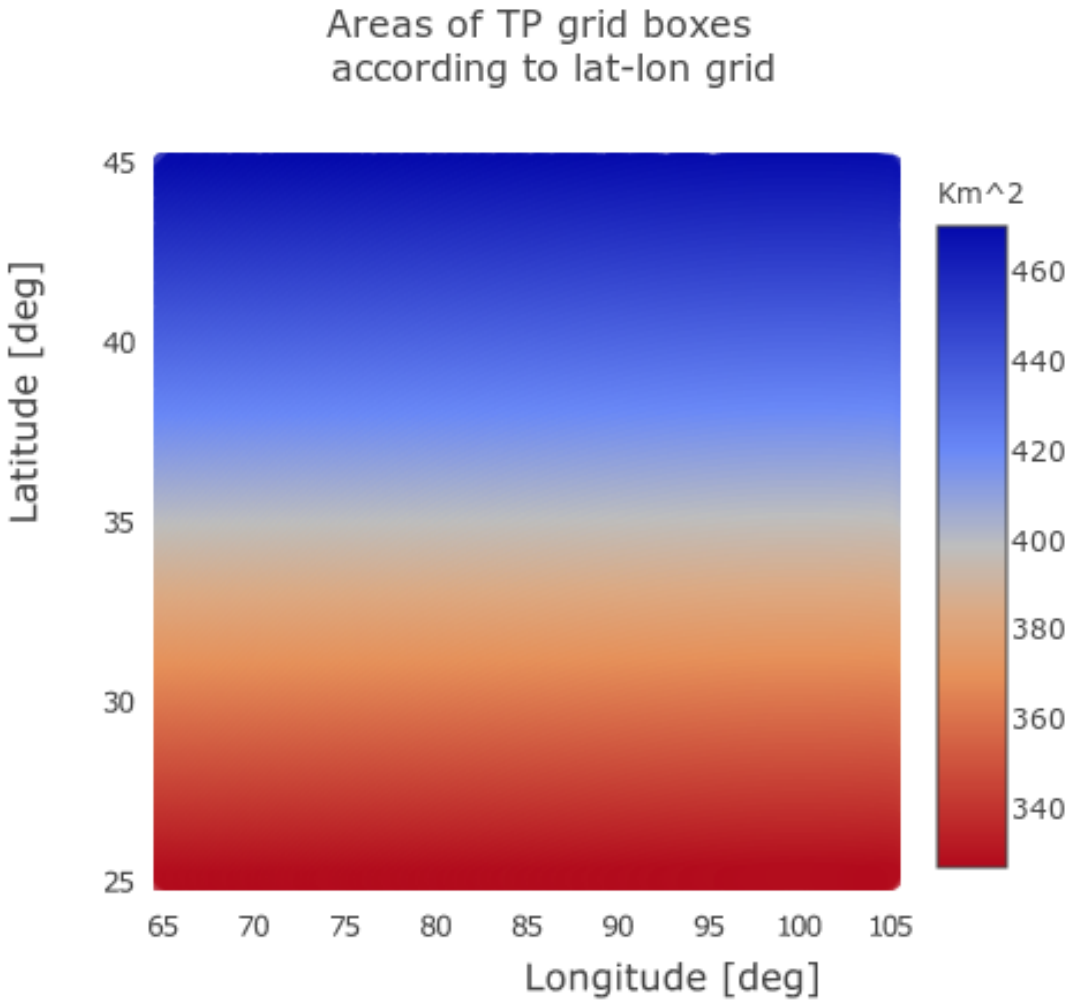

Figure 2.6. The area of a grid box varies according to latitude.

$$\Delta\phi = \phi_2 - \phi_1 \tag{2.33}$$

$$\Delta\lambda = \lambda_2 - \lambda_1 \tag{2.34}$$

$$a = \sin(\frac{\Delta\phi}{2})^2 + \cos(\phi_1)\cos(\phi_2)\sin(\frac{\Delta\lambda}{2})^2 \tag{2.35}$$

$$c = 2\arcsin(\sqrt{(a)}) \tag{2.36}$$

where ϕ represents latitude and λ represents longitude and c is the great circle length in degrees. Next, a semi perimeter area is calculated

$$s = c + \pi + \frac{\phi_1 + \phi_2}{2} \tag{2.37}$$

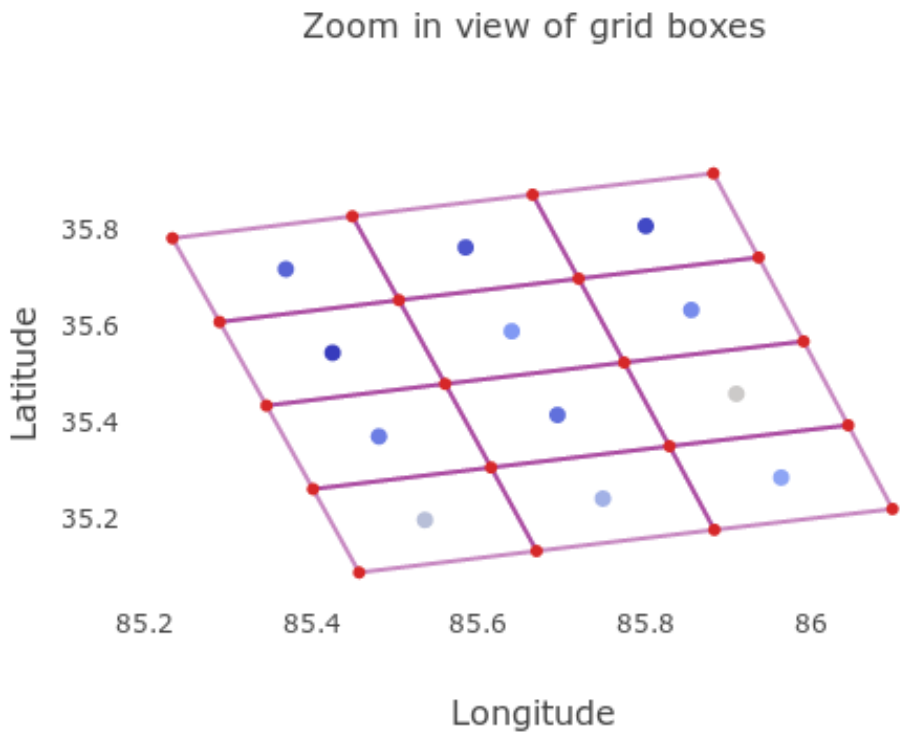

Figure 2.7. A zoom-in display of grid boxes and their centroid points provided a detailed view on how areas are estimated. Each central circle represents a lat-lon point provided by IMS. TSM calculates the red corner points followed by the bounding boxes' area.

Finally, The area of a spherical triangle is given by the l'Huilier formula

$$b = \left(\tan(\frac{s}{2})\tan(\frac{s-d}{2})\tan(\frac{s-\pi/2+\phi_1}{2})\tan(\frac{s-\pi/2+\phi_2}{2})\right)^{\frac{1}{2}} \quad (2.38)$$

$$\Delta = 4.0 * \arctan(b) \quad (2.39)$$

where Δ is the radial area of a spherical triangle.

Treating the longitude points $\lambda_1 = 65^\circ$ $\lambda_2 = 105^\circ$ and setting $\phi_1 = \phi_2$. Two different spherical triangle areas are obtained for $\phi_{top} = 45^\circ$ $\phi_{bottom} = 25^\circ$, Δ_{top} and Δ_{bottom} are obtained. The area of Tibet in km^2 is given by

$$A_{tibet} = R^2(\Delta_{top} - \Delta_{bottom}), \quad (2.40)$$

where A_{tibet} is treated as the exact solution and compared to the sum of areas in our data frame

2.3 RESULTS

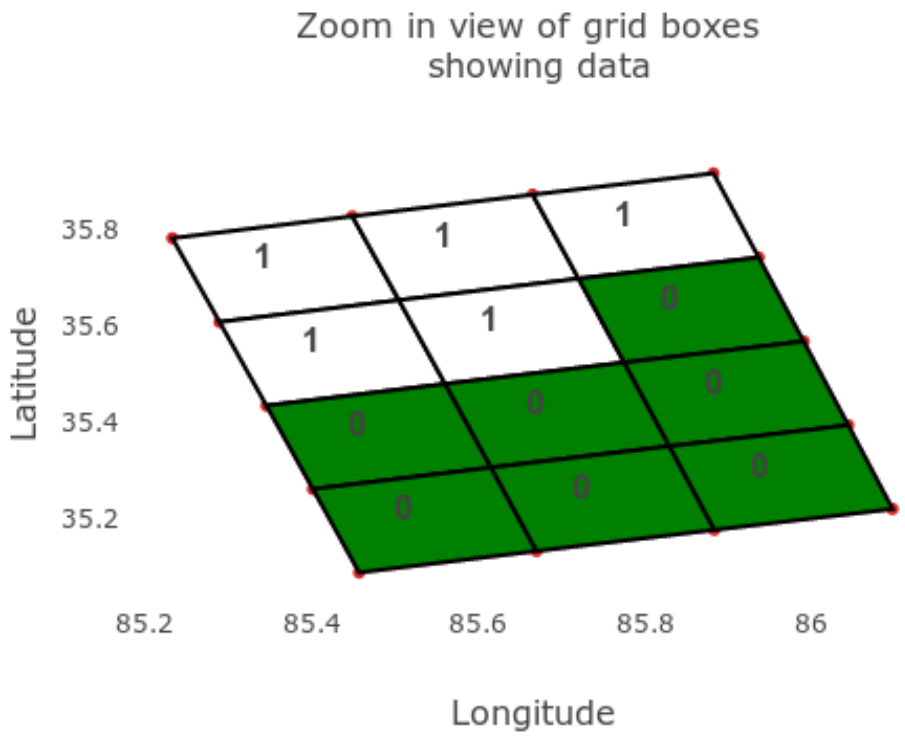

Figure 2.8. A zoom-in view of the IMS data with snow/no snow illustrations. Cells covered in snow are in white and cells with no snow are shown in green.

Snow data are visualized by plotting an interpolated contour grid onto a Mercator projection of the TP, region shown in Figure 2.9. The two grids are compared in a time series of snow coverage collected from 2004-02-24 until 2017-06-26, shown in Fig. 2.10. A percent coverage difference subplot is included to better distinguish between the two series,defined as:

$$100 * \left(\frac{ts_{24\ km} - ts_{4\ km}}{ts_{4\ km}} \right), \tag{2.41}$$

where $A_{tibet} = 8,059,061.82 km^2$ Calculated analytically via the l'Huilier's formula for spherical triangles.

Figure 2.10 indicates a percent coverage difference range of $[-10.12\%, 48.42\%]$. Note The finer resolution grid reports more snow coverage in the warmer months, suggesting the finer grid records glaciers and smaller snowfields. During winter and spring months, this trend reverses, with the finer resolution grid reporting less snow and ice coverage, indicating a positive bias for the 24x24 km resolution product.

Figure 2.11 shows a time series showing percent coverage collected until from 1997-02-04 until 2017-06-26. An area estimation conducted by Shen et al. [26] assumed a

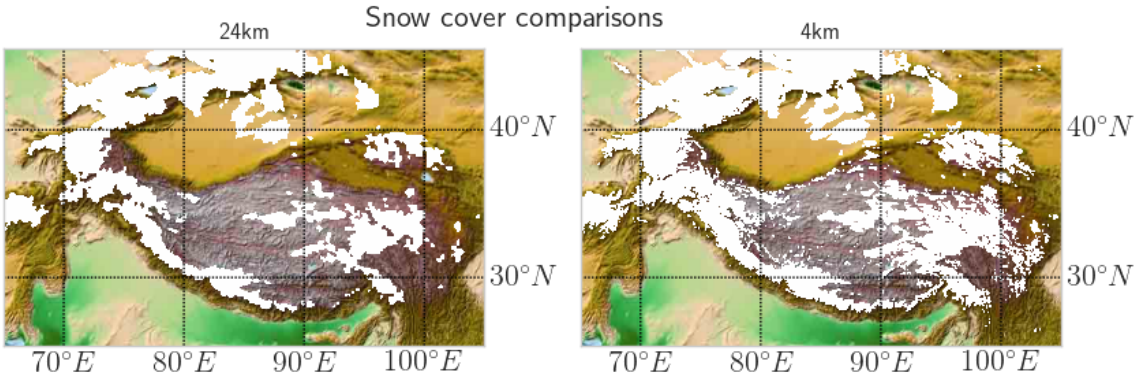

Figure 2.9. Tibetan plateau region's snow cover for 2 January 2015: The white region indicates snow cover which is over layed on the land surface topography.

uniform $24x24km^2$ grid area. This assumption causes an overestimation of snow and ice coverage by as little as 32.2 % and as much as 54.79 %.

Long-term trends in snow cover are obtained by binning the annual cycles into 5 day periods and averaging them, denoted by $clim(t)_5$. The annual snow cycle is shown in Figure 2.13 This annual snowfall is subtracted from the time series, $Snow(t)$.

$$Anomalies(t) = Snow(t) - clim(t)_5 \tag{2.42}$$

From there a linear regression is obtained that is not influenced by annual cycle, shown in Fig. 2.12

A trend line was included to investigate long-term trends on the plateau. Linear regression results indicate no significant trends in snow coverage from the start of the IMS product to the last date of capture. There is little indication of a considerable change in snow coverage with the data provided; though, there is a high certainty of a negative change. Still, the yearly averaged anomaly bar chart show greater frequencies of yearly averaged negative anomalies in recent years, suggesting there is some change taking place in the TP snowpack.

Climate averaging $clim(t)_5$ shown in Figure 2.13 provides an expected annual snow coverage profile for the TP. Maximum snow coverage at $2.786\ 10^6 km^2$ occurs in January 21th till January 25th. Minimum snow coverage at $0.151\ 10^6 km^2$ occurs from August 24th till August 28th. Annual mean coverage = 1.199x$10^6\ km^2$

Figure 2.14 is a histogram showing the probabilistic distribution. Kurtosis is given using Pearson's definition. [13]

Maximum snow coverage was reported on 2008-02-06, during one of most volitle weather patterns in China's recent history [6]. The surge in 2008 winter storms are thought to be caused by a combination of La Nina and cold temperatures [22].

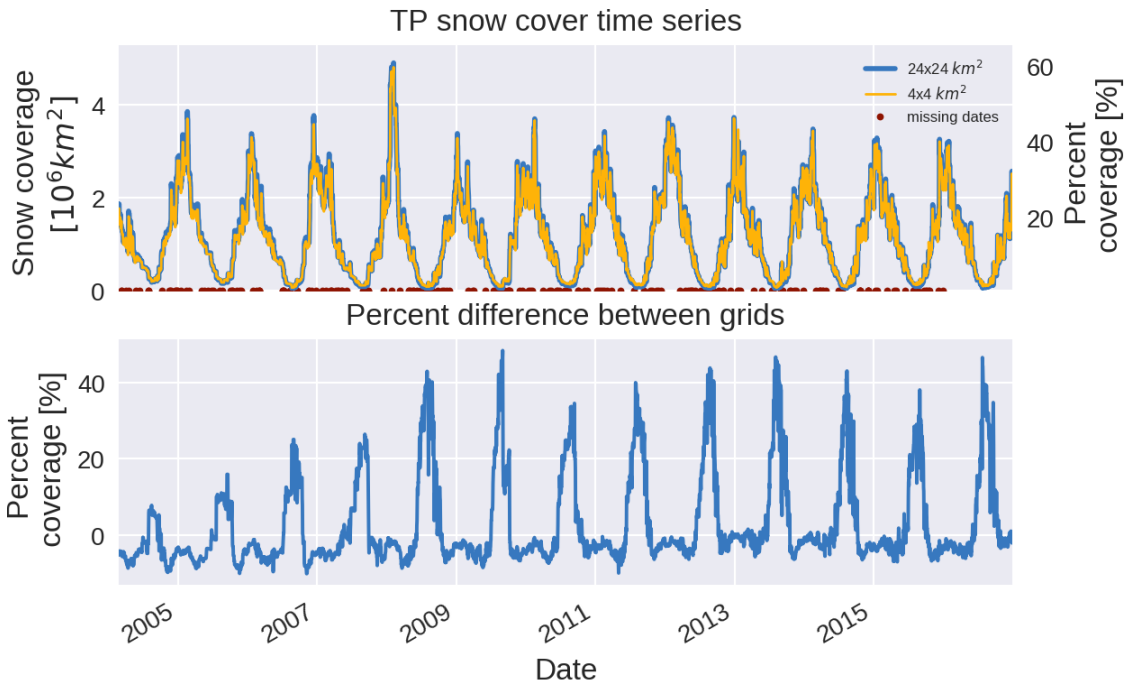

Figure 2.10. The snow cover area time series comparison between two resolutions: 4 km and 24 km. Percent difference calculated using (2.41).

Minimum snow coverage was reported on 2013-08-12, which was known for its unusually warmer weather in Southern China[18].

The standard deviation is approximately a quarter of the annual mean. Such a large standard deviation implies a high risk of snowstorms and the general volatility of water sources coming from the TP.

Both Skew and Kurtosis tests gave p-values of approximately zero.

Skew greater than 1 indicates more outliers to the right of the mean, suggesting that the TP experiences severe snow storms, compared to a steady flux of predictable snow patterns.

Table 2.2. Anomaly Distribution Properties

Statistic	Value
Mean	$-309.9 km^2$
Median	$-30992 km^2$
Min	$-1.112\ 10^6 km^2$
Max	$2.446\ 10^6 km^2$
Standard Deviation	$.34259971\ 10^6 km^2$
Skew	1.0006
Kurtosis (Pearson's)	4.351

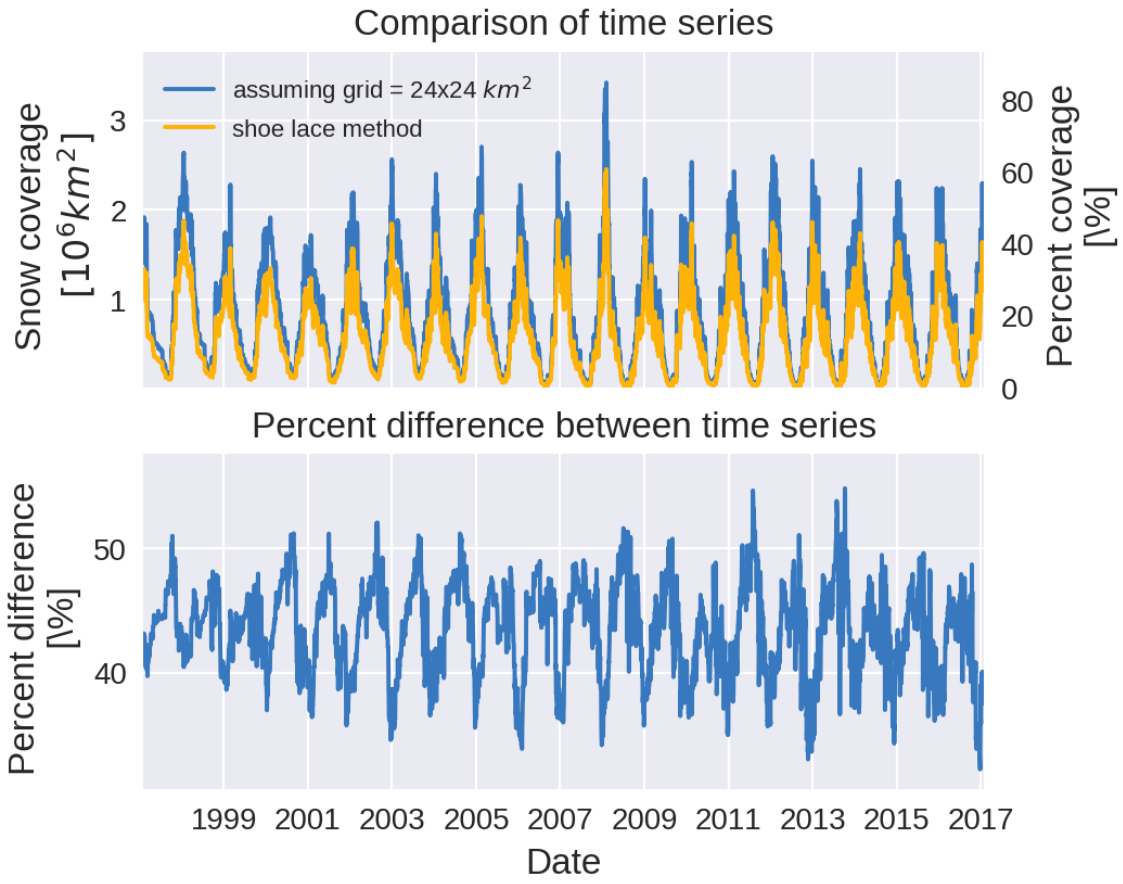

Figure 2.11. Time series comparison between $24 \times 24km^2$ assumption and area calculation via shoe-lace formula.

Kurtosis is greater than three shows that anomalies gather closer to the mean when compared to a Gaussian distribution. The expected snow cover occurs often, but infrequent storms can create large anomalies.

Climate averaged anomalies allow basic statistics. However there are some drawbacks, chiefly, the long-term trends and noise remain coupled. An alternative approach to anomalies, seasonal decomposition by LOESS can attempt breaking a time series into three components: season, trend, and remainder. This non-parametric method is described in the AppendixB. The results shown in Figure 2.15 reveals a remainder that appears to be almost identical to the five days averaged climate anomalies from Figure 2.12. The trend portion does not appear to have a clear increase or decrease. Large peaks occur on winter months 1998 and 2017 which is also seen in the yearly averaged anomalies.

Another advantage to using seasonal decomposition by LOESS is that it will fill in missing values with approximations, creating a daily time series with no missing values. By removing the seasonal portion, the remainder and trend can be modeled as a stationary linear

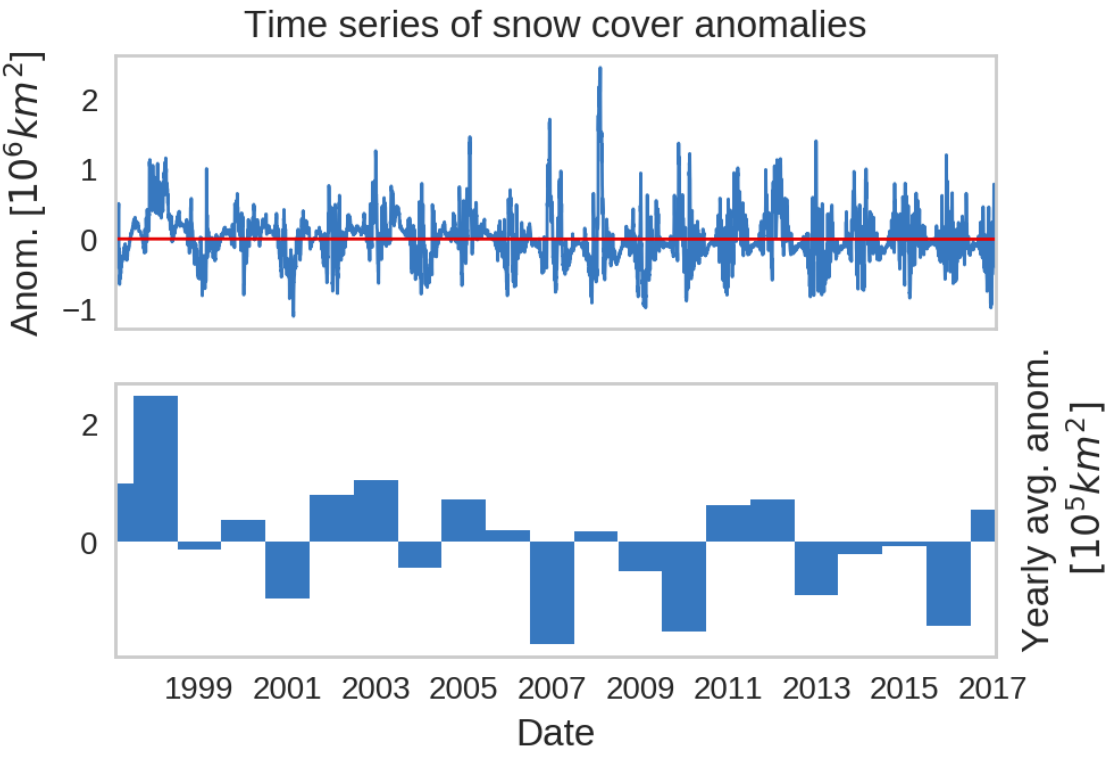

Figure 2.12. Time series of Snow coverage anomalies with included trend line, in addition to yearly averaged anomalies, shown by the bar chart.

time series. Details on the theory are explained in Appendix A. The series' variance is seasonal. Therefore the time series log transform is taken. An AR(6) model is selected to have the more favorable AIC while still passing the Box-Ljung test.

$$log(Z_t) = a_t + \theta_1 a_{t-1} + \theta_2 a_{t-2} + \theta_3 a_{t-3} + \theta_4 a_{t-4} + \theta_5 a_{t-5} + \theta_6 a_{t-6} \tag{2.43}$$

The AR(6) model with estimated parameters shown in Table 2.3 have a relatively low AIC = -12046.12 and a Box-Ljung test that fails to reject the null hypothesis, shown in Figure 2.16. All of the calculated, $p - values > .005$ indicating that the residuals after fitting the AR(6) model to the deseasonalized snow data are not autocorrelated.

2.4 TSM Applications

The TSM is currently used to generate daily snow mapping at `http://www.itsonlyamodel.us/daily-snow.html`. See Figure 2.17.

The Chinese National Earth System Science Data Sharing Infrastructure website uses TSM to archive current and past snow coverage of TP at `http://www.tpedatabase.cn/tibetSnow.jsp`[20] shown by Figure 2.18. As of now, the

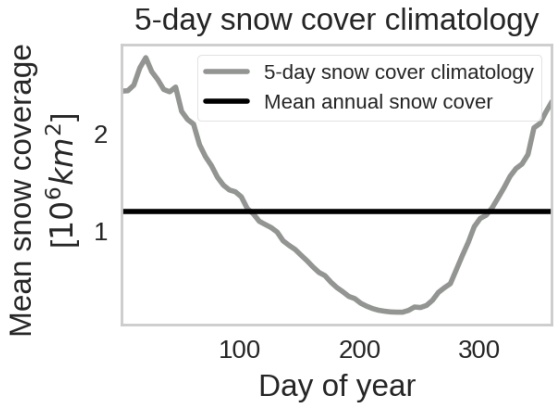

Figure 2.13. 5 day climate averages of the 24×24 km annual snow cycle.

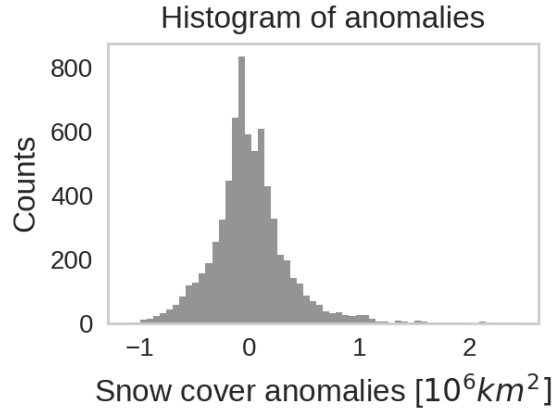

Figure 2.14. Histogram of the 24×24 anomalies.

tool is primarily used for generating images; however, we have shown that the TSM can be used for analysis.

2.5 Conclusions

The TSM project provided on GitHub [28] is a convenient tool for parsing through large IMS data sets. Statistical inferencing using this set has never been easier or more accessible to both professional and amateur alike. Anyone who can run a python environment can generate time series, snow and ice maps, histograms, spectral series over TP or any region over the Northern Hemisphere. TSM provides users with data sets that can be used for statistical inferencing as shown by the simple statistical inferencing in the results section.

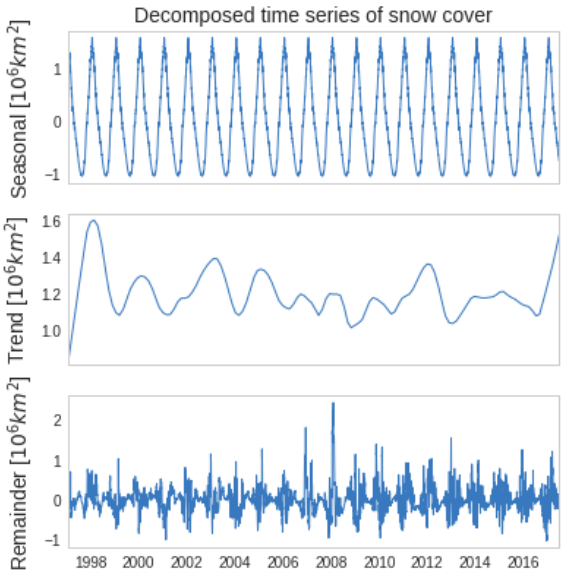

Figure 2.15. Deseasonalized using STL algorithm described in Appendix.

Table 2.3. AR(6) model parameters with AIC = -12046.12

Param	Value	Standard Error
θ_1	1.0099	0.0116
θ_2	-0.0814	0.0165
θ_3	-0.0395	0.0165
θ_4	0.0460	0.0165
θ_5	-0.0234	0.0165
θ_6	0.0214	0.0116
intercept	13.9548	0.0186
σ^2	0.01159	N/A

Regression trends point out less snow over the TP today than at the onset of the IMS project. Histogram trends outline that it is more likely for there to be negative anomalies than positive ones.

Analysis can be made using either the fine or coarse data set. The project created offers researchers a toolkit that quickly parses through massive data sets over any latitude and longitude range in the northern hemisphere.

TSM can adjust the map projection reference location, in the case that IMS begins recording snow and ice coverage over the Southern Hemisphere.

Moreover, calculated areas improve the accuracy of these gridded measurements, and aid in monitoring snow and ice data sets.

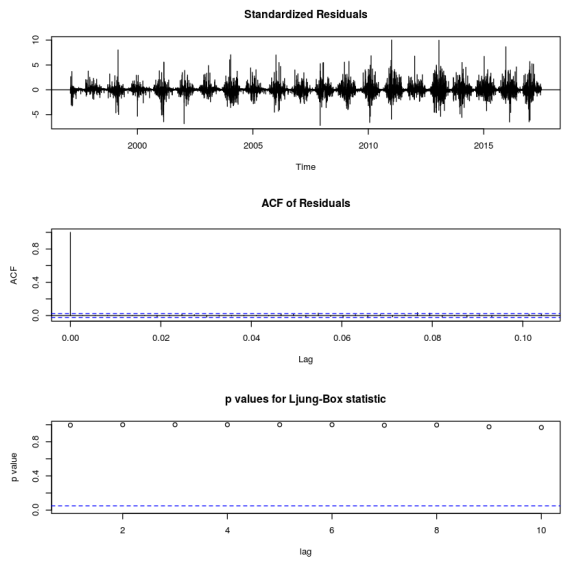

Figure 2.16. Box-Ljung diagnostic test for $m = 10$ of an AR(6) model. For all lags, $p - values > .005$ indicating that the residuals after fitting the AR(6) model are not autocorrelated.

Future developments to TSM can include better means of querying subsets of data. Current methods do not use a database but depend on local files. A database with a website interface would give more users access to more data. The scalability of the dataset also makes it possible to implement the 1x1 km data set. The higher resolution set could show the change in glacier and ice field size over time and even pinpoint regions of rapid climate change. A database design also can be continually updated with the NSIDC's daily measurements. Lastly, a website interface would give users with no python experience access TSM. The next chapter describes this web-app framework used by the ARGO project. A similar web app can be created for IMS, providing a much-needed service for SACD for this and related products.

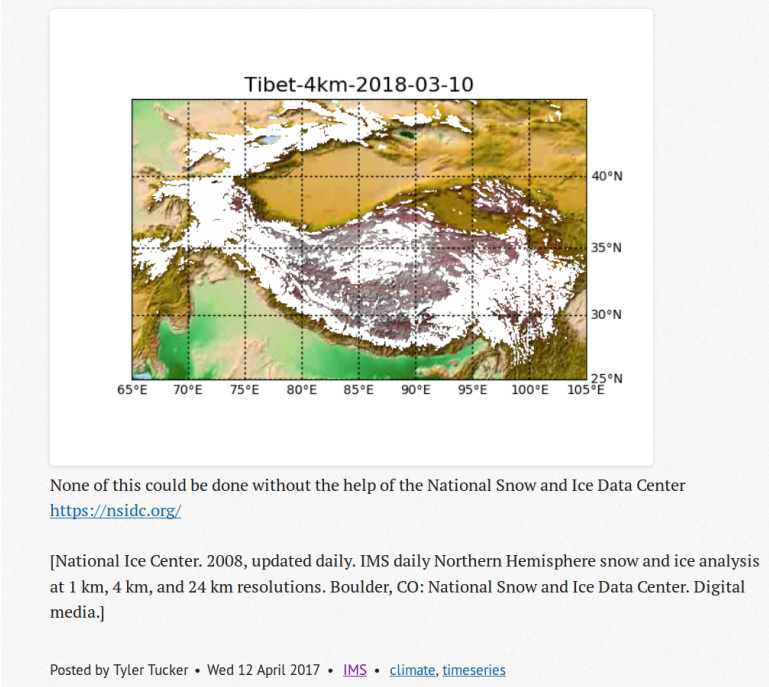

Figure 2.17. Screenshot of daily snow plot at `http://www.itsonlyamodel.us/daily-snow.html`

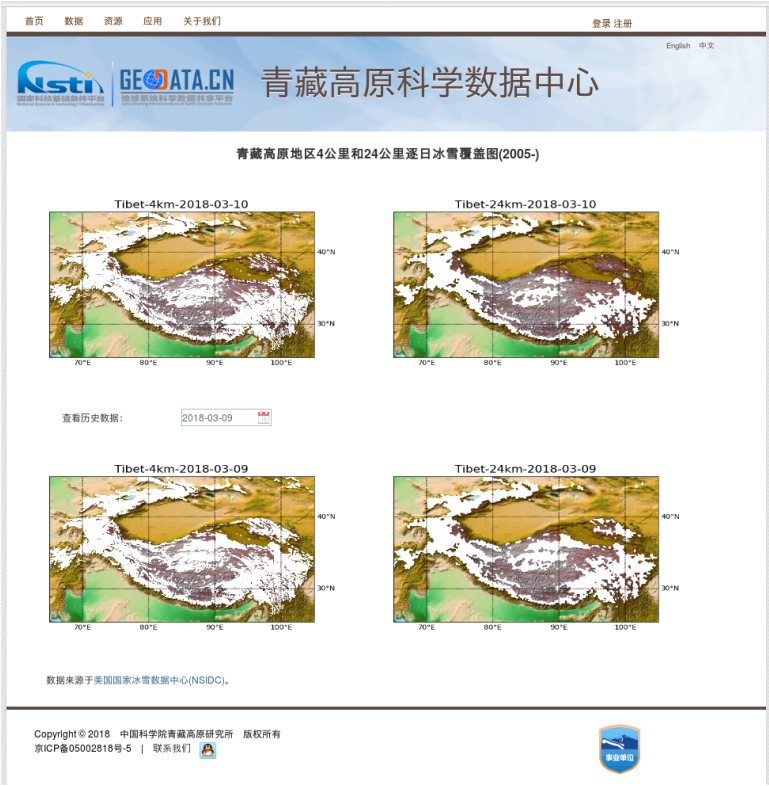

Figure 2.18. Screenshot of `http://www.tpedatabase.cn/tibetSnow.jsp` **showing daily snow plot at archives at 24 and 4 km resolution.**

CHAPTER 3
A WEB APPLICATION FOR QUICK EXTRACTION AND ANALYSIS OF ARGO DATA

3.1 INTRODUCTION

Advances in communication and sensor technologies cause the amount of data generated by the Argo program to explode. Sampling the oceans at high resolution creates a data storage problem. Figure 3.1 shows the increase as the program continues. The additional data included in each profile due to the Iridium floats compounds each profile size, begging the question as to how can this large data set is accessed. Creating gridded products, inferencing, displaying results becomes an increasingly complex issue as Argo grows in both profile payload and quantity. The web app Argovis is an attempt to address big data in climate science.

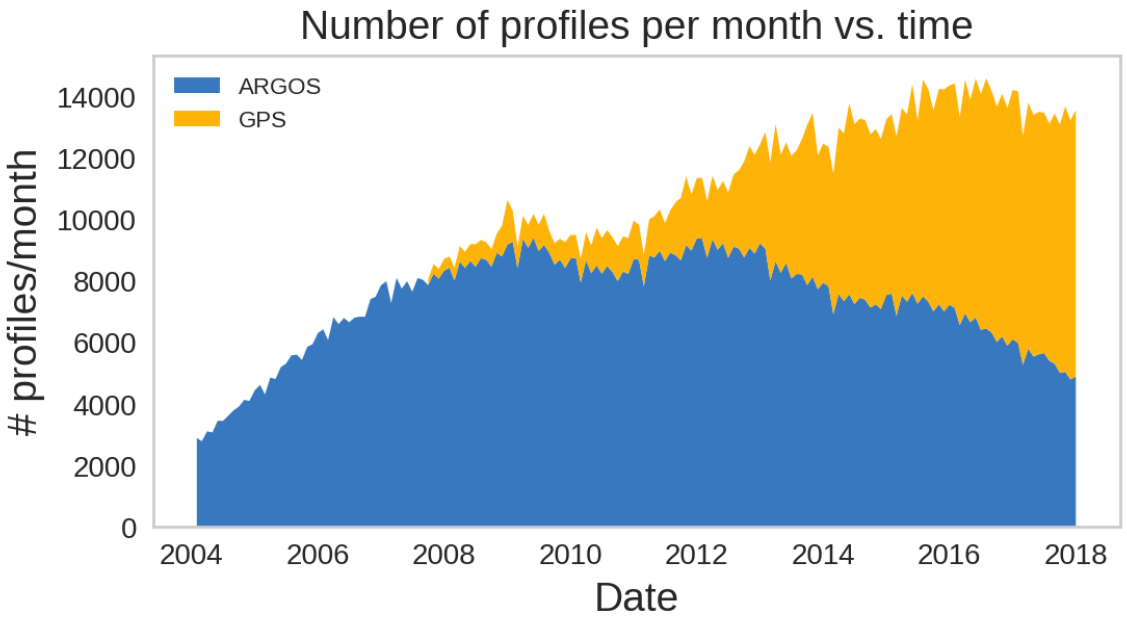

Figure 3.1. History of Argo profiles reported each month since the programs inception. Trends towards GPS measurements indicate a need for a data retrival service. Argo data generation is increasing in the number of profiles reported and by the amount of data reported by each profile.

Data is normally stored in a Global Data Assembly Centres (GDAC) FTP server, either at a US-based server at `ftp://usgodae.org/pub/outgoing/argo/` or a Europian server at `ftp://ftp.ifremer.fr/ifremer/argo/`. The data is assumed to be sufficiently quality controlled and in a stable format. Platform and profile name is the only identification. This data structure does not allow seasonal selection or spatial selection: a key feature to any climate data set. The FTP server is meant to act as an archive. Scientists are encouraged to download the data locally for their uses, a process that relies on significant domain knowledge of the FTP site. Few people will go through the trouble of understanding the FTP structure, and fewer still will take time to make the process easier for others. The current unwieldy system, Stunts the Argo community from expanding.

The Argo program depends on government and by extension, public support. Without the understanding of these parties, the program is in danger of budget cuts, sequestration, and general apathy to scientific endeavor. As is, the public is reliant on the Argo community to extract meaning from the dataset. There is an opportunity to design a system intuitive enough for Laypeople to view data.

The remainder of this chapter will describe a proposed data storage and visualization system that uses a document-oriented database to supplant the FTP server, and a RESTfull web app to view database content both spatially and temporally.

3.2 Web Application Overview

The main page, shown in Figure 3.2, consists of an interactive map and sidebar with controls, allowing users an intuitive way to navigate through profiles, either a region and date range or by entering a float's WMO number. The map is created using the leaflet library.

Typing in the known platform name on the side panel reveals the platform's history on the map. Once complete the platform will show up, see Figure 3.3. Each dot will have a popup window appear when clicked. The window gives lat-lon coordinates, profile's measurement date, a button that will show the float's history. Also included are links to either the platform page, shown in Figure 3.7 or the entire float history, shown in Figure 3.4.

Different map projections are available by clicking the link on the top of the sidebar. Currently, Web Mercator, Northern stereographic, or Southern stereographic are the only web projections available. Only Web Mercator uses map tiles; the latter two use a JSON shapefile to show land outlines. In any projection, the map includes a drawing plugin that allows users to create, edit, and delete squares or polygons. In the event a polygon is created, the app queries the database for profiles falling within the region bounded by the polygon, along with the date and pressure ranges selected on the side panel. If the query isn't too large, the app then clears the map of profiles and returns profiles returned by the database. See Figure 3.5 (top).

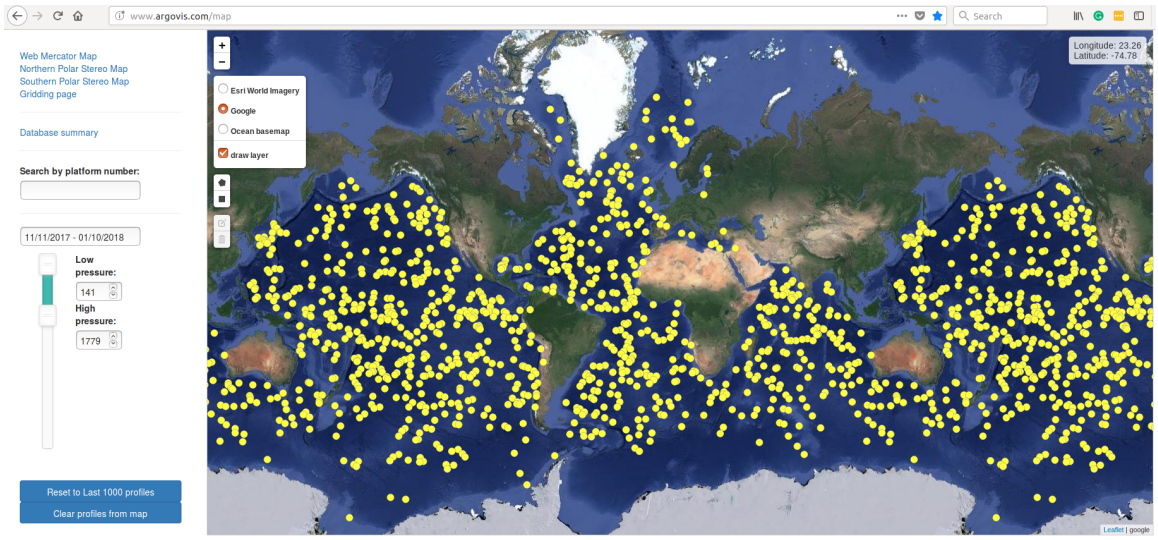

Figure 3.2. Users are greeted by a map with the latest 1000 profiles upon visiting `http://www.argovis.com` (top). Clicking on a profile marker springs open a popup window showing additional information and links to its data visualization page. Rectangle and Polygon icons on the map lets the user draw squares or polygons. Upon completion, the map clears all profiles except for the ones that fall within the drawn polygon(s). A pop up over the shape links to another page with T/S/P charts.

The completed shape will also have a popup window show with two buttons that open a page of the selection's T/P/S charts. The same goes for a platform's T/S/P charts can be generated and viewed with a click of a button as shown in Figure 3.5 (bottom).

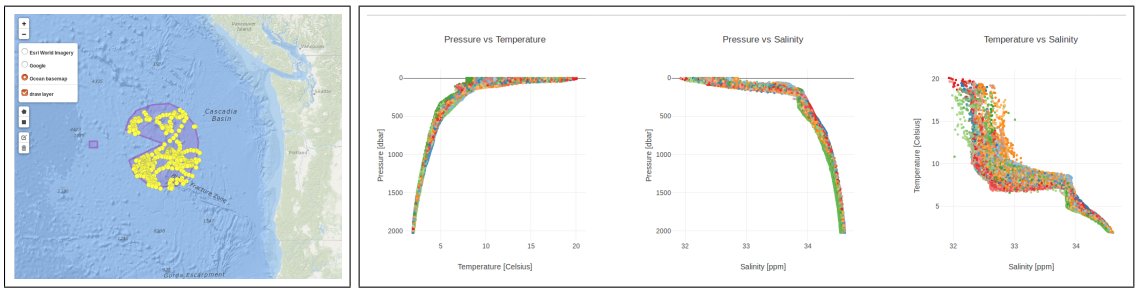

Figure 3.5. Upon completion of a shape provided by the drawing tool (top left), the map clears all profiles except for the ones that fall within the drawn polygon(s). A pop up over the shape links to another page with T/S/P charts (mid left). The sidebar includes Date and pressure (depth) range options too.

Bottom of page, shown by Figure 3.6 includes a download button, a table, and a table export feature. The page is accessed with the url: `http://www.argovis.com/selection/profiles/page?startDate=2010-11-29&endDate=2018-01-28&shape=[[[-132.601318,46.286224],[-130.557861,45.767523],[-132.403564,45.135555],[-132.`

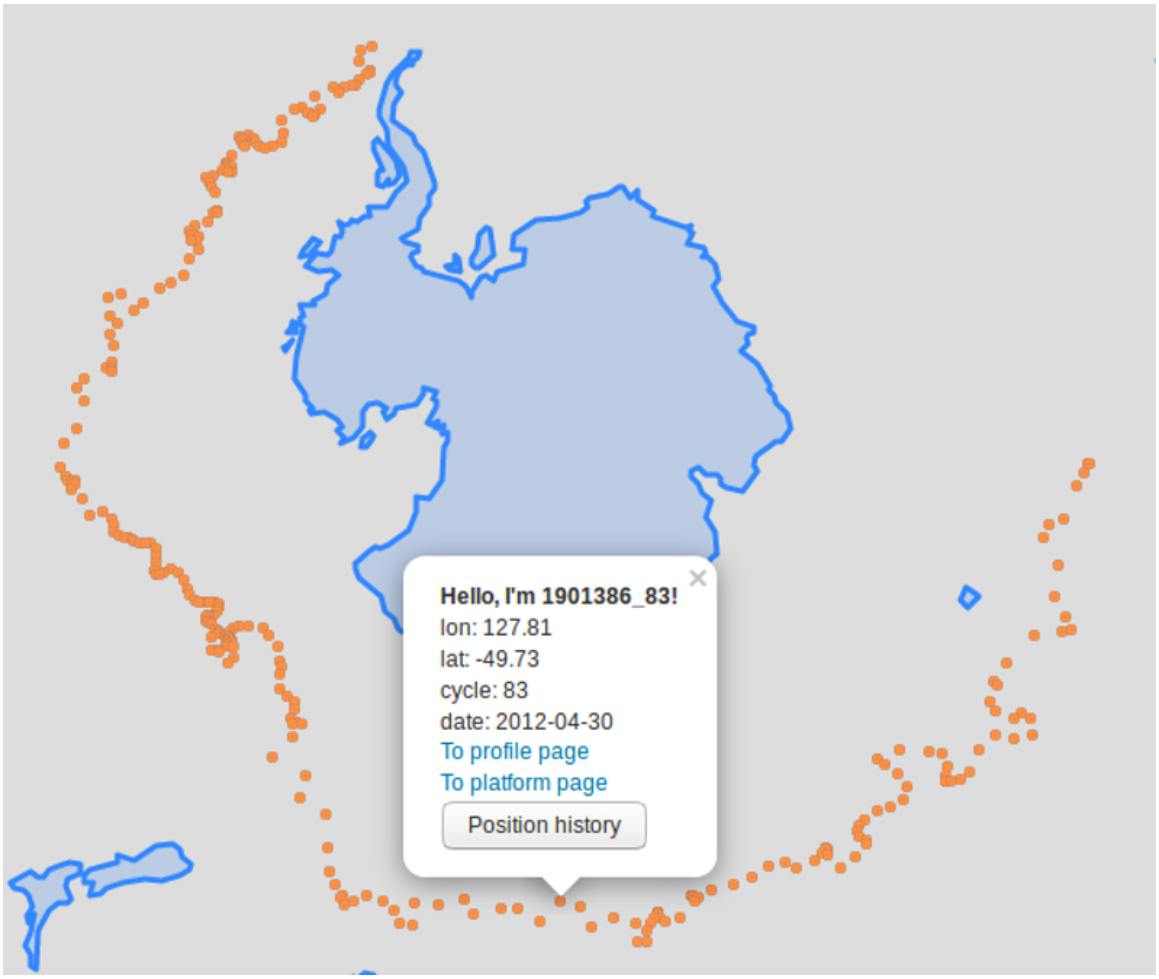

Figure 3.3. History of platform 1901386 are shown by orange dots. Each point reprents the float surfacing to transmit its profile information. Clicking on any of the orange profile markes springs open a popup showing additonal information and links to its data visualization page.

161865,44.653024],[-131.480712,44.260937],[-130.711669,44.103365],[-129.810791,44.150681],[-129.129638,44.559163],[-128.73413,45.182037],[-128.624267,45.813486],[-128.822021,46.452997],[-129.525146,47.040182],[-130.206298,47.26432],[-131.12915,47.219568],[-131.986084,47.010226],[-132.403564,46.679594],[-132.601318,46.286224]]]

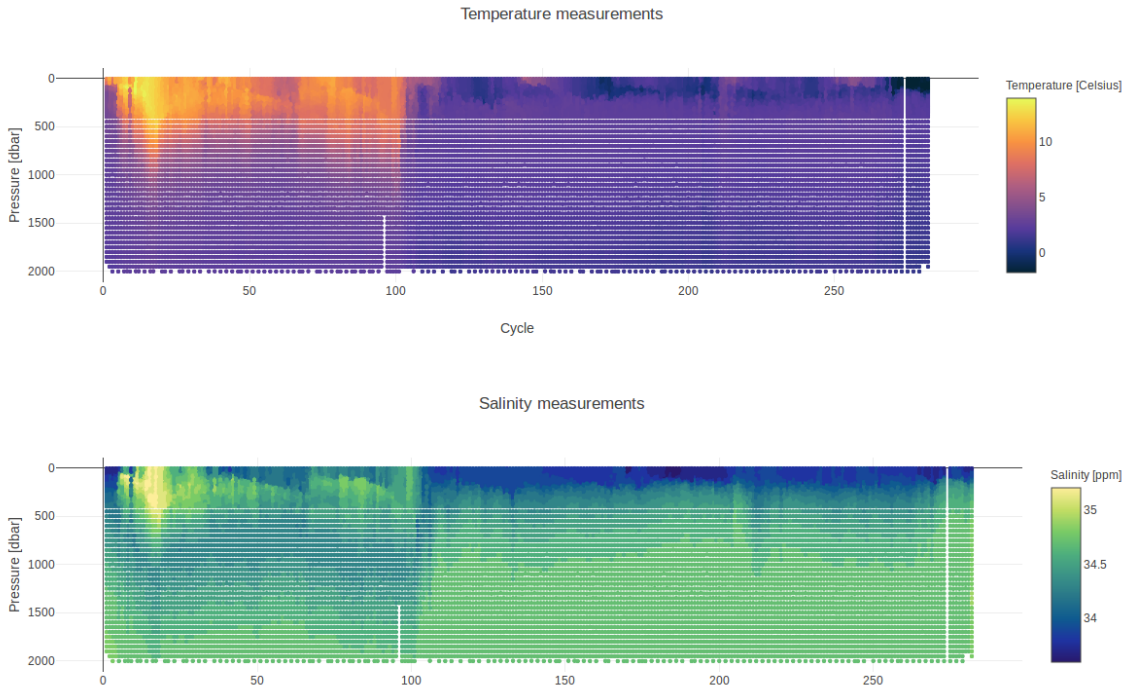

Figure 3.4. Temperature and salinity heat plot for profile 1901386.

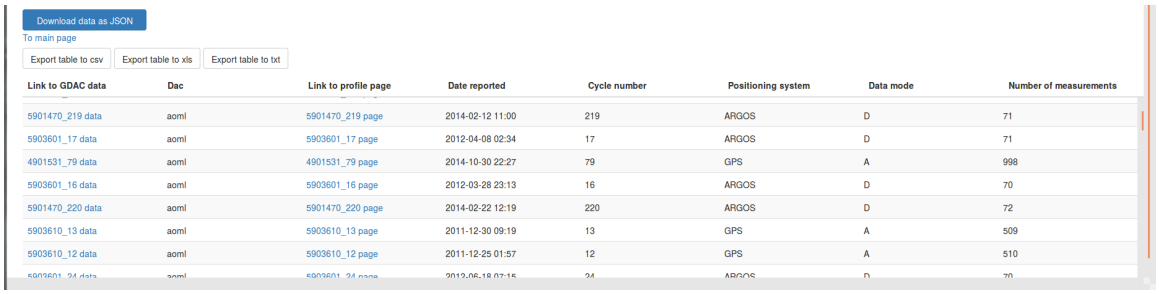

Figure 3.6. Temperature and salinity heat plot for profile 1901386.

RESTful applications allow clients (either users or other apps) to interact with a database through the URL. In the case of Argovis, users query a MongoDB database filled with ARGO profiles. Users input a set of latitude-longitude coordinates, date range, and pressure range. The Argovis app sends a response: either raw JSON or HTML page with the profile measurements and metadata that fall within these queries. See Figure 3.7. Clients can either download the data directly on their files by clicking the 'Download Data as JSON' button, or by removing from the URL, '/page.' Both profile or selection data is be accessed the same way.

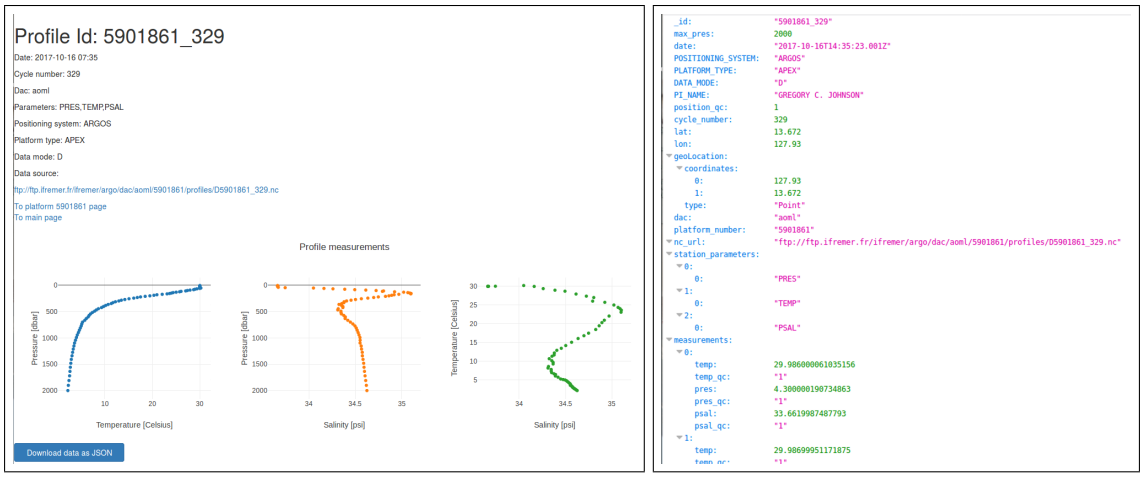

Figure 3.7. Profile page (left), and JSON (right) can be accessed by appending '/page' to the URL (`http://www.argovis.com/catalog/profiles/5901861_329/page`) or omitting it.

3.3 ARGOVIS DATA VISUALIZATION AND EXTRACTION TUTORIALS

Tutorials describe several website use cases step by step and offers the user a break from reading. The first tutorial narrates a typical use case on how a user can make a custom selection around the Labrador sea at 500-1000 meters during Summer 2017, and download the data and its metadata table locally. The second tutorial has a user track a particular profile over time, view its profiles, examine unusually cold sea surface temperatures, and download a profile directly from the IFREMER GDAC. More data selection and delivery methods are available using the API, covered in Section 3.5. `www.itsonlyamodel.us` has more tutorials.

3.3.1 Labrador Sea Summer Selection

This tutorial describes how to make selections on the main page. First, a shape is drawn, followed by a date pressure range. The profiles are viewed and downloaded on a separate page.

1. Visit `www.argovis.com`. You will be greeted by a Web Mercator projection showing profiles reported by the DACS from the past seven days. See Figure 3.2. Sidebar includes a map reset button (Figure 3.8, which will return the map to its original state. The clear map button will swipes all profile icons off the map.

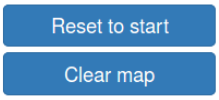

Figure 3.8. Reset buttons will return the map to how it was when first rendered by the browser. Clear will remove all profile dots and shapes from the page.

2. Zoom into the Labrador sea, east of Greenland. You can zoom in by using the mouse wheel or by clicking the zoom button on the top left corner of the map (Figure 3.9

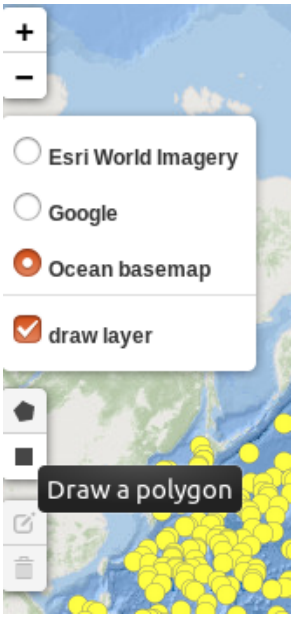

Figure 3.9. Zoom button (top element) allows zooming in and out. Tile map overlay (2nd down from top) allows the user to choose which map overlay is shown. Rectangle and polygon buttons (3rd down from top) and edit/delete buttons (bottom) lets users draw shaps on the map.

3. Click the polygon button, located below the zoom buttons and map tile layers on the top left corner of the map.
4. Create a polygon by clicking on the region around the general outline of the Labrador Sea. Circle the polygon nodes back to the point of origin to complete the shape. Upon completion, profiles will only show within the polygon, and a popup will appear (Figure 3.10. With the shape selected, we next need to specify a date and pressure range.

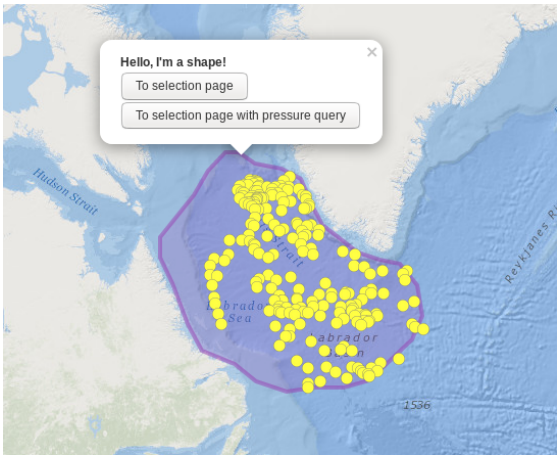

Figure 3.10. Shape is drawn by pointing and clicking. Complete the shape to query the database and plot profiles. Editing the shape will replot the points with the updated information.

5. Click on the date input element located on the sidebar. A date range window (Figure 3.11) will appear comprised of three columns. The left column is an array of buttons with relative times that users can select. The center and right columns display an interactive scrollable calendar that allows for date selections.

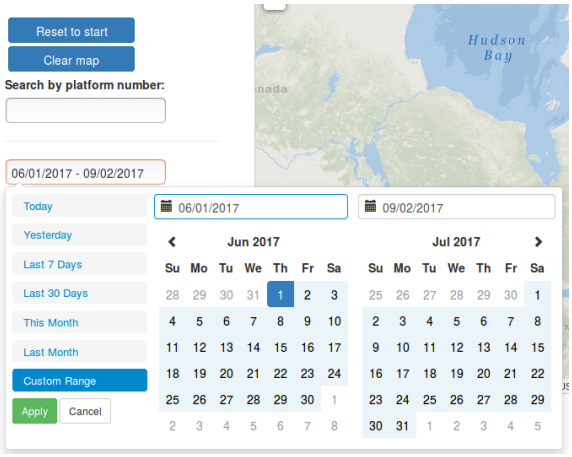

Figure 3.11. Calender feature allows users to select common date ranges (left), click interactively in ranges by clicking days on calender, or by manually typing in date ranges (top textboxes). Upon completion, map is updated.

6. Adjust the pressure range, located on the sidebar (Figure 3.12, so that the upper pressure is at 1000 dbar and the lower pressure is at 500 dbar. You can either adjust the double scroll bar using the mouse, click the arrows located in the input element. You may also type in the number of input element directly followed by pressing enter. Each time pressure is adjusted updates the profiles in the polygon area.

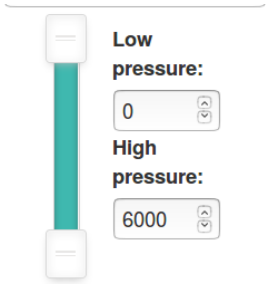

Figure 3.12. Pressure slider bar contains two sliders that set upper and lower pressure ranges. Upon a change, the map is updated. Text boxes are syncronized to slider bar such that any changes are updated on all three elements.

7. Open the selection page by clicking on the shape popup button labeled ”To selection page with pressure query. This page (Figure 3.13 can be viewed through the `http://www.argovis.com/selection/profiles/page?presRange=[500,1000]&startDate=2017-06-01&endDate=2017-09-02&shape=[[[-54.8291,53.696706],[-56.674803,54.572062],[-59.487303,55.229023],[-60.717772,56.511018],[-62.036131,58.401712],[-63.090819,59.534318],[-63.090819,60.413852],[-61.508787,61.312452],[-60.190428,62.144976],[-58.959959,63.114638],[-57.465819,63.821288],[-56.32324,63.821288],[-54.91699,63.391522],[-52.631834,63.194018],[-50.961912,62.794935],[-49.995115,61.731526],[-49.116209,60.930432],[-47.79785,60.457218],[-46.040037,59.756395],[-44.194334,59.445075],[-42.436522,59.265881],[-40.942381,58.309489],[-40.502928,56.897004],[-41.381834,55.776573],[-42.788084,55.028022],[-45.07324,53.748711],[-47.270506,53.383328],[-49.731444,53.383328],[-52.192381,53.173119],[-54.8291,53.696706]]]`.

Link to GDAC data	Dac	Link to profile page	Date reported	Cycle number	Positioning system	Data mode	Number of measurements
6901023_189 data	coriolis	6901023_189 page	2017-09-01 15:38	189	ARGOS	R	20
6901646_249 data	coriolis	6901646_249 page	2017-09-01 08:43	249	GPS	R	50
4901817_48 data	meds	4901817_48 page	2017-08-31 04:49	48	GPS	A	100
6901147_158 data	bodc	6901147_158 page	2017-08-31 03:42	158	ARGOS	A	10
4902383_48 data	meds	4902383_48 page	2017-08-31 00:54	48	GPS	D	100
4902412_2 data	meds	4902412_2 page	2017-08-30 20:47	2	GPS	A	99
6902661_73 data	coriolis	6902661_73 page	2017-08-28 13:17	73	ARGOS	R	20
5904989_246 data	coriolis	5904989_246 page	2017-08-27 23:02	246	GPS	A	250

Figure 3.13. Selection page is comprised of T/S/P plots (top). Each color represents a profile. Hovering the cursor over will reveal a popup that displays position coordinates and profile id. Clicking on a point will open the profile page in another link. ”Download data as JSON” button opens JSON file containing selection data in a separate window. ”To main page” link sends the user back to home page. Export buttons allow users to download the table of profile metadata shown below. A table includes links to download profiles directly from the GDAC.

The T/S/P plots showing selection data is instantly available for view. Each color represents a single profile. Hovering the cursor shows the detail. Clicking on while the cursor is hovering over a profile will open the page for that particular profile.

8. Download the raw data by clicking on the ”Download Data” button. A new tab on the browser will open, showing the data in a JSON format.
9. Download the table of profile Metadata as a .csv file by clicking on the ”Download as csv” button. Excel spreadsheet or text format is also available.

You should now have as much data as the database for this section locally.

3.3.2 Track a single platform in the Southern Ocean

This tutorial shows Argovis’s ability for a user to track a particular profile over time, view its profiles, examine unusually cold sea surface temperatures, and download a profile directly from the IFREMER GDAC.

1. Visit `www.argovis.com`. You are greeted by a Web Mercator projection showing profiles reported by the DACS from the past seven days.

2. Click the ”Southern Polar Stereo Projection” button that redirects you to a new page with a map that uses a Southern polar stereographic projection (Figure 3.14.

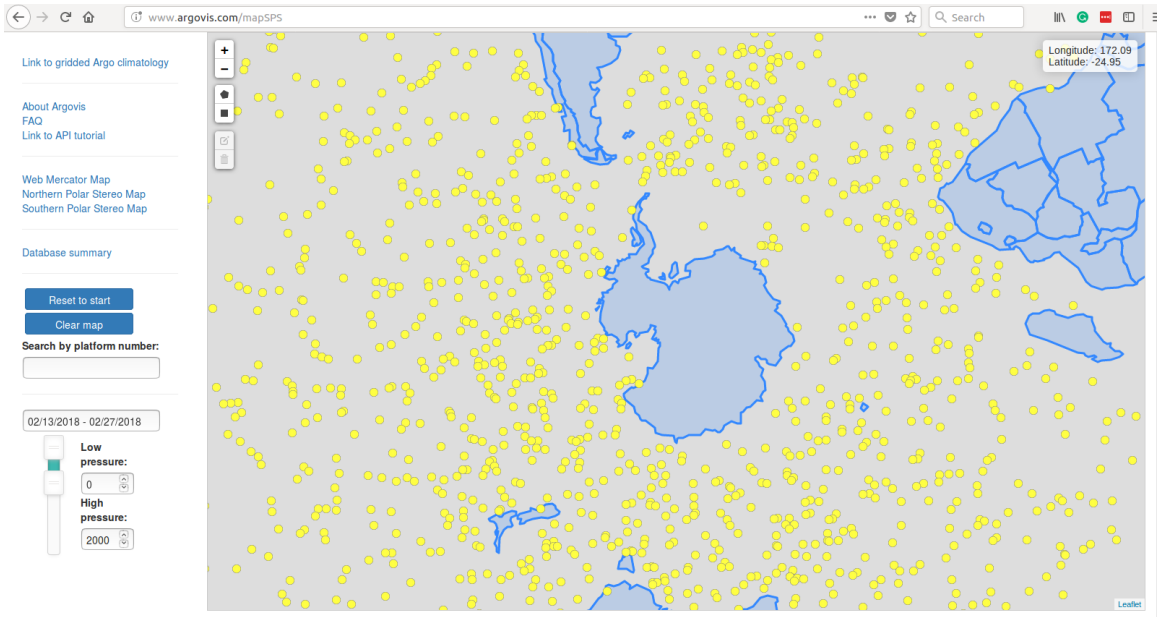

Figure 3.14. Southern Stereographic projection map view. Map tile layers are not available under this projection. Land is displayed using geoJSON shape files.

3. Clear the map by clicking the ”Clear Map” button located the sidebar.
4. Enter the WMO/platform number 1901386 in the ”Search by platform number” textbox. The profiles generated by this platform will be displayed circling Antartica (Figure 3.16).

Figure 3.15. ”Search by platform number” textbox will plot a platform’s profiles on the map. Profiles are displayed in orange. Profiles are displayed when the user enters in a profile number. Four characters or more is enough to query the database.

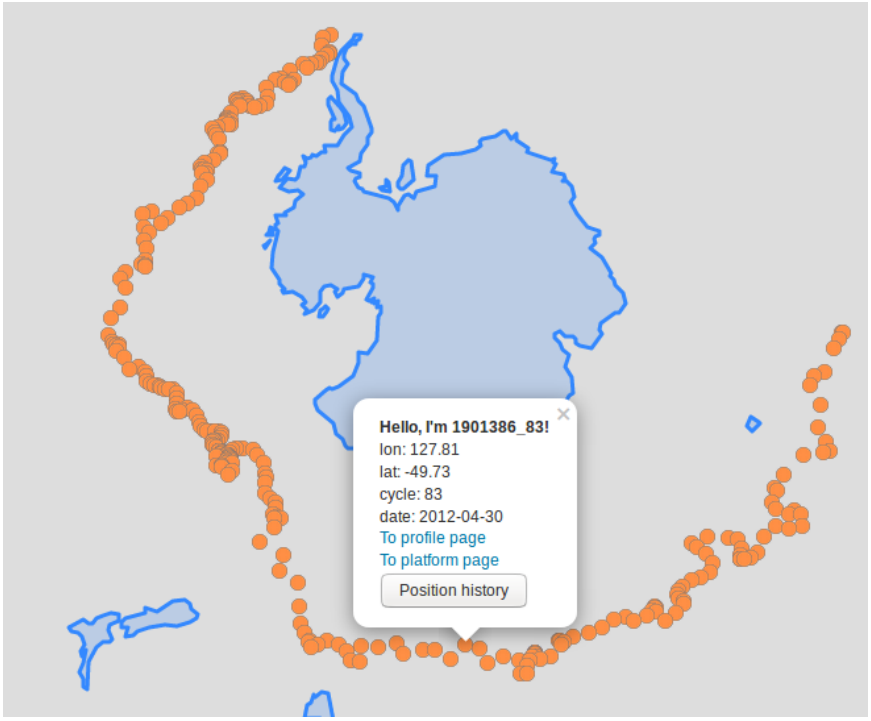

Figure 3.16. Profiles that are plotted using platform selection button appear in Orange. Clicking on any point will bring up a popup icon with the profiles id, position, cycle number, and date. Also included are links to the profile's page and its platform page. Position history button will display the other profiles of the same platform.

5. Click on any orange point to bring up a profile popup window. Click the "To platform page" link to open up the page in another browser tab (Figure 3.17). The page resides at `http://www.argovis.com/catalog/platforms/1901386/page`.

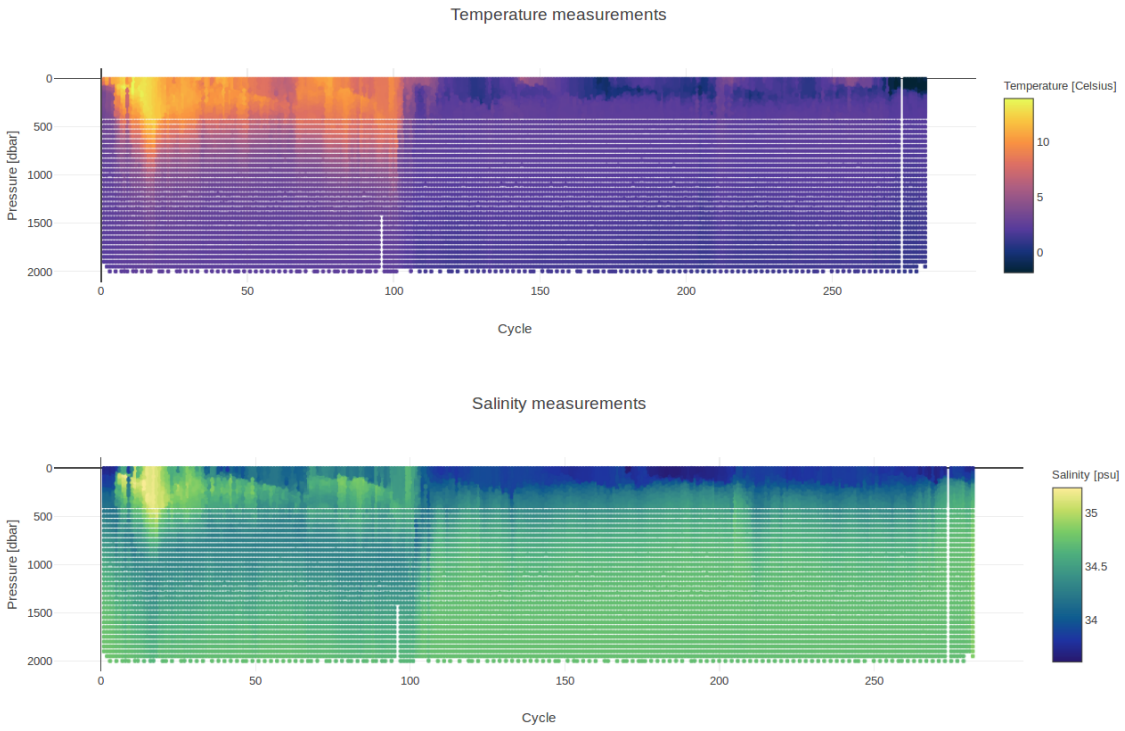

Figure 3.17. Profiles of the same platform are plotted with color charts. The platform page shows temperature (top) and Salinity (bottom). The vertical axis represents pressure (depth), and the horizontal is cycle number. Like the selection page, hovering the cursor over any point will trigger a popup with the values of the point. Clicking on the point will open the profile's page.

Color plot of shows temperature going unusually low at the surface. Negative temperatures may be an error and will be investigated as an exercise.

6. Click on the dark purple region in the upper right corner of temperature color plot. In this case. A new browser tab will open, showing the profile page you just clicked. In this example, profile 1901386_281 was selected and is shown in Figure 3.18. This page resides at `http://www.argovis.com/catalog/profiles/1901386_281/page`.

Figure 3.18. Platform page of id 1901386_281**. Meta information is displayed first. Data source links to the original netCDF file located at the GDAC. There are also links that will direct the user back to the platform page or main page. T/S/P charts are on the bottom. Data used to generate the page can be downloaded by clicking the "Download data as JSON" button.**

The date this profile surfaced was on November 3rd, 2017. Ice sheets melt, making the surface temperatures cold and brackish. Salinity is around 34 practical salinity unit (psu). The profile location is close to the Antartic peninsula, indicating that the ice sheets connected to the land are relatively close. Most likely, this data is accurate. As it turns out, sea temperatures freeze at about -2 Celsius [19]. To make sure that the reported data matches the original netCDF file reported in the GDAC the profile page includes a link that will download the netCDF file when clicked.

7. Click on the link `ftp://ftp.ifremer.fr/ifremer/argo/dac/aoml/1901386/profiles/R1901386_281.nc` to download this profile data directly from the ifremer GDAC. You now have the original data opened with software compatible with netCDF files to compare to the unusually low sea surface temperatures.

3.4 Back-End Overview

Argovis uses the express.js framework to host dynamic web pages. The framework handles page views, routing, and database interface. In addition to the base T/S/P parameters, floats may contain O_2, N_2, Conductivity, et cetera. A MongoDB database stores the profiles metadata and their measurements as Binary JSON objects. Argo data pairs well with MongoDB's flexible schema. For example, not all profiles have to have a N_2 field.

3.4.1 Data Format and Delivery

Profiles are stored as document objects. Each document has a schema as seen in Listing 3.1. The "measurements" field must contain temperature, salinity and pressure, but may also contain fields found on Table 1.1

Listing 3.1. JSON schema of mongoDB profile.

```
{
  "_id": <String>,
  "max_pres": <Int>,
  "measurements": <List[{temp: <Number>,
                         psal: <Number>,
                         pres: <Number>,
                         ...}]>,
  "date": <ISODate>,
  "POSITIONING_SYSTEM": <String>,
  "PLATFORM_TYPE": <String>,
  "DATA_MODE": <String>,
  "PI_NAME": <String>,
  "cycle_number": <Int>,
  "lat", <Number>,
  "lon", <Number>,
  "geoLocation", {"type": "Point",
                  "coordinates": [<Number[2]>],
  "dac": <String>,
  "platform_number": <String>,
  "station_parameters": <List[String]>,
  "nc_url": <String>
}
```

MongoDB outputs JavaScript object notation (JSON) data. JSON format is used to create the map, profile, platform, and selection aspects of the website. API access also uses

JSON and will be mentioned in Section 3.5. JSON 'curly braces' encapsulate the Javascript data types.

3.4.2 Database Architecture

Indexing shrinks the query time significantly. Currently, date, platform_number, cycle_number, DAC, and geoLocation are indexed.

Each day new profiles are added to the GDAC, and old profiles are updated. Argovis is synchronized with the GDAC daily to reflect new changes, such as introducing new profiles or correcting errors by the GDAC. Each day, a cron script runs a shell script that updates a local mirror of the IFREMER GDAC using the rsync command. A script of changes is created in the process and fed to a python algorithm that adds or replaces the files that are in the database. The script is found in Appendix D.

Transforming netCDF files from the GDAC FTP server involves reformatting the .nc files to the document schema shown in Listing 3.1. The script parses netCDF files and adds portions of its 'variables' field to the database. Additionally, some erroneous data and low quality are removed, as the GDACs keep all data. Each measurement in the Argo program has a quality control (QC) string value, whose meanings are shown in Table 3.1. The current version of Argovis includes measurements that have a QC flag of 1.

Table 3.1. Argo Quality control flag meanings, taked directly from Argo Manual [31]

N	Meaning
0	No QC was performed
1	Good data
2	Probably good data
3	Bad data that is potentially correctible
4	Bad data
5	Value changed
8	Interpolated value
9	Missing value

The following pseudocode 1 outlines the .nc to MongoDB process as a rough outline.

MongoDB's database accepts one document or multiple documents at a time, the latter speeds up the algorithm, hence Algorithm 1 collects documents in RAM before adding it to MongoDB. Python excels at handling complex data objects. For this reason, it was chosen to implement Algorithm 1 and 1.

Algorithm 1 profile *.nc files to MongoDB document

```
1:  procedure CONVERT FILE TO DOCUMENT
2:      Generate lists of profile names for each dac from local GDAC
3:      loop each dacList in listOfDacLists
4:          document = []
5:          loop each fileName in dacList
6:              get pathName from fileName
7:              get dacName from dacList
8:              open file, get variables field
9:              doc = MAKE_PROF_DOC(variables, pathName, dacName, remotePath)
10:             append doc to documents
11:             if documents length greater than threshold then
12:                 add documents to MongoDB
13:                 document = []
14:         add remaining documents to MongoDB
15: procedure MAKE_PROF_DOC
16:     INPUT: variables, pathName, dacName, remotePath
17:     retrieve platformNumber, stationParameters, referenceDate from variables
18:     idx = 0
19:     qcThreshold =' 1'
20:     p2D = netCDFToDoc(...)
21:     doc = p2D.get_profile_doc()
22:     OUTPUT: doc
```

The class netCDFToDoc2 takes some of netCDF's variables, performs QC, and bundles parameters into a JSON object that MongoDB can accept. In a pythonic context, dictionary types are equivalent to JSON. See Algorithm 2.

Measurements may contain values adjusted by the DACS; however, Argovis will replace the original with the adjusted if it exists(lines 9-11). The netCDFToDoc algorithm filters the data so that only QC = 1 make it to the database (line 13). Measurements are all bundled in one DataFrame object (line 14). An exception to the rule, rows of this concatenated dataframe are dropped if there are unknown pressure values (line 16); pressure values are needed to generate plots and filter on the front-end side. '-999' replaces missing values.' Finally, the QC columns, no longer needed are dropped. The remaining lines 19-34 add meta information with minimal formatting.

This process is repeated daily for new files using a cron script. New/updated profiles are detected and downloaded from the GDAC to a local directory. A list of the profiles is fed to the python script, which runs Algorithms 2 and 1.

3.5 PYTHON API

Argovis is designed to be used by both professional and amateur alike. The front end is intuitive but can be confining for automated tasks. Additional visualization features can be added, at the expense of additional development cost and maintainability. Feature creep in web apps complicates an otherwise intuitive design. On the other hand, a sparsely featured application discourages experts/scientists from using the app. application programming interface (API) services are the balance that reconciles the design conflict between ease of use and visualization features.

The definition of API is somewhat nebulous, but this context an API is access to data without the need of a browser. It is often referred to as "Machines talking to machines" In this case, it is custom code talking to Argovis. The custom code can be a process running a plotting script, a data archive service, or even another website. Scientists can make selections using a script in their preferred language by including HTTP APIs.

One such API has been implemented in Python. A suite of functions listed in the Appendix C retrieve data from Argovis. All other features for visualization and computing depend on these functions. RESTfully designed apps such as Argovis allow non-browser apps to retrieve data. Accessing Argovis is as simple as writing a few lines using the request library. The different ways to visualize data surpasses the current capability of Argovis. To address the communities need for the myriad visualization and computation techniques, a tutorial that guides the user through the Python API at was written in `http://www.itsonlyamodel.us/argovis-python-api.html`. The API described in the

Algorithm 2 netCDFToDoc class formats netCDF data into JSON object

```
1:  procedure CREATE A JSON/DICT OBJECT
2:      INPUT: variables, pathName, dacName, remotePath
3:      INPUT: platformNumber, stationParameters, referenceDate
4:      init measDataFrame
5:      measurement = ['temp', 'pres', 'psal', 'cndc', 'doxy', ...  ]
6:      loop each meas in measurements
7:          from variables get measArray, measQCArray, adjMeasArray, adjMeasQCArray
8:          loop each (value, idx) in measArray
9:              if adjMeasArray[idx] exists then
10:                 measArray[idx] = adjMeasArray[idx]
11:                 measQCArray[idx] = adjMeasQCArray[idx]
12:             combine arrays into singDataFrame
13:         drop row in singDataFrame if QC column ≠ 1
14:         concat singDataFrame to measDataFrame (preserving index)
15:     replace NaN with -999 in measDataFrame
16:     drop row in measDataFrame if measDataFrame['pres'] == -999
17:     drop all qc columns in measDataFrame
18:     init doc = dict()
19:     doc['max_pres'] = measDataFrame['pres'].max()
20:     doc['measurements'] = measDataFrame.toList(orient='records')
21:     doc['date'] = variables['JULD'] - referenceDate
22:     doc['lat'] = variables['LATITUDE']
23:     doc['lon'] = variables['LONGITUDE']
24:     doc['POSITIONING_SYSTEM'] = variables['POSITIONING_SYSTEM']
25:     doc['PLATFORM_TYPE'] = variables['PLATFORM_TYPE']
26:     doc['DATA_MODE'] = variables['DATA_MODE']
27:     doc['PI_NAME'] = variables['PI_NAME']
28:     doc['POSITION_QC'] = variables['POSITION_QC']
29:     doc['cycle_number'] = variables['CYCLE']
30:     doc['geoLocation'] = dict('type':'Point','coordinates':[lon, lat])
31:     doc['dac'] = dacName
32:     doc['platform_number'] = platformNumber
33:     doc['nc_url'] = 'ftp://ftp.ifremer.fr/ifremer/argo/dac/'+remotePath
34:     doc['_id'] = platformNumber + variables['CYCLE']
35:     OUTPUT: doc
```

next section is the baseline for additional tools and scripts written by the community to germinate.

3.6 API Applications

Example applications are provided below, involving custom plots and statistical modeling. There are myriad of analysis and modeling techniques to try. The API makes it easier for scientists to model their ideas on Argo's data.

One application is a monthly histogram of profiles, shown in Fig. 2.14. The code used to create this found at `www.itsonlyamodel.us`

API function of interest queries a section by a date range, aggregates the data, and generates time series. See Figure 3.19 for details.

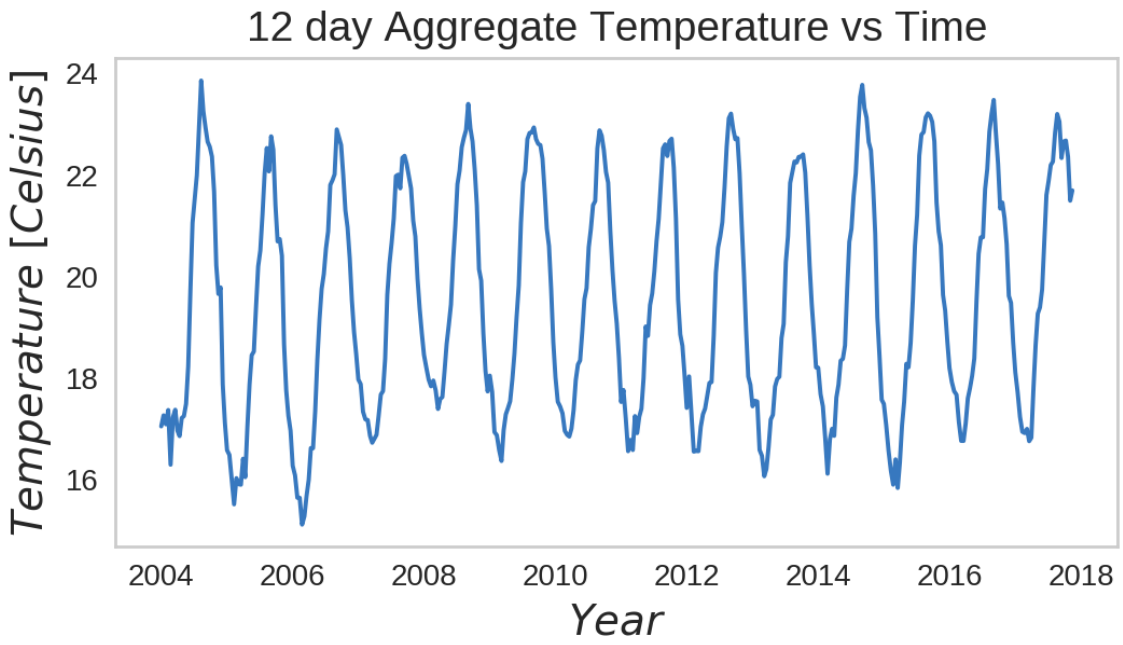

Figure 3.19. API includes a special time series generator for a selection.

Temperature data displayed in Figure 3.19 is modeled as a linear time series process using the theory described in Appendix A after applying deseasonalizing the time series the methods described in Appendix B.

Time series plotting capability is expanded two three dimensions as shown in Figure 3.20. The first 100 meters water column is aggregated into ten equal partitions. Each partition is ten dbar/meters thick. Each aggregate has its time series, whose amplitude is described by color.

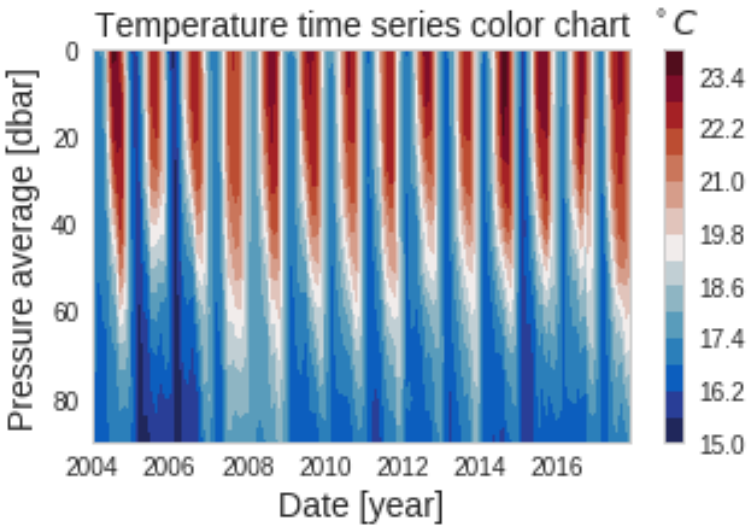

Figure 3.20. Time series at ten equally spaced partitions along the first 100 meters of the water column (10 meters/partition). Temperature is displayed as color and the y-axis is plotted as pressure (depth).

Without modeling the time series, the plotting feature shows upwelling patterns in the years 2005-2007. Colder water is brought closer to the surface as indicated by the dark blue color. Years 2007 and 2008 show upwelling patterns; the color is much brighter than the surrounding years. Another item worth mentioning is the large thermal mass forming in winter 2017/2018. Arvovis's API can monitor this warm mass evolve daily being as the app updates its database daily.

Another method generates interpolated maps, again by aggregating the data, but only for one time range, shown by Figure 3.22. The histogram summarizes the temperature distribution over the whole time series in Figure 3.21.

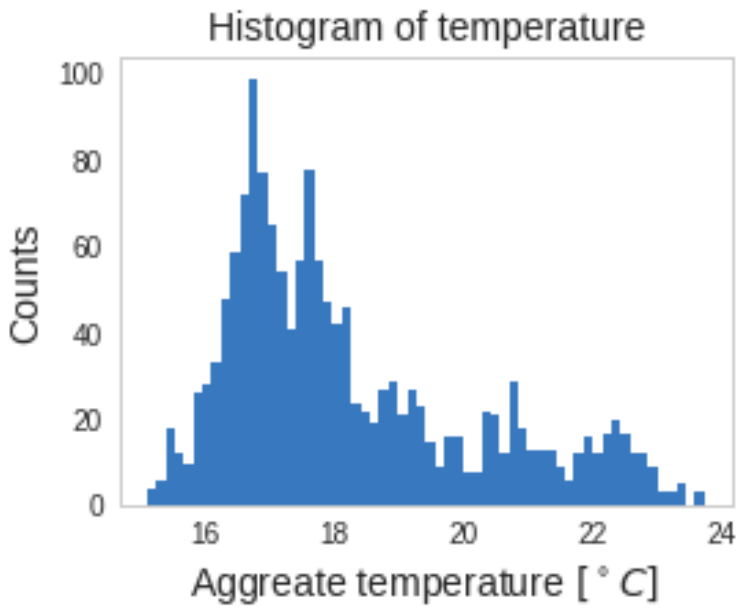

Figure 3.21. Histogram of temperatures of the first 100 dbar/meters, aggreagated into 10 dbar sections.

The histogram shows a bivariate data set (most likely seasonal) with a right-hand skew (surface temperatures). Summary statistics are provided in Table 3.2.

Table 3.2. Anomaly Distribution Properties

Statistic	**Value** $[^{\circ}C]$
Mean	18.2456
Median	17.6627
Min	15.0946
Max	23.7158
Standard Deviation	1.9496
Skew	0.8933
Kurtosis (Pearson's)	-0.1797

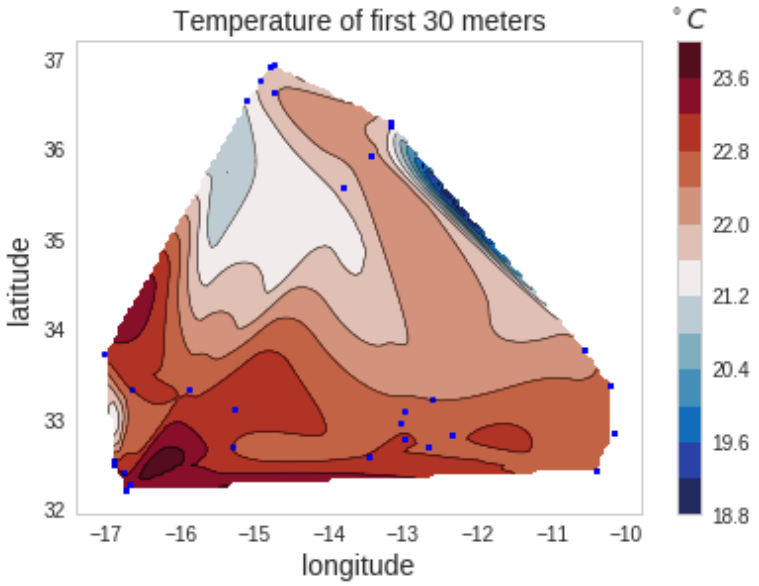

Figure 3.22. Cubic interpolation of temperature at the first 30 meters of arbitrary selection.

The API also has a built-in histogram on a map, showing profile frequency on a 1×1 degree grid, shown by Figure 3.23.

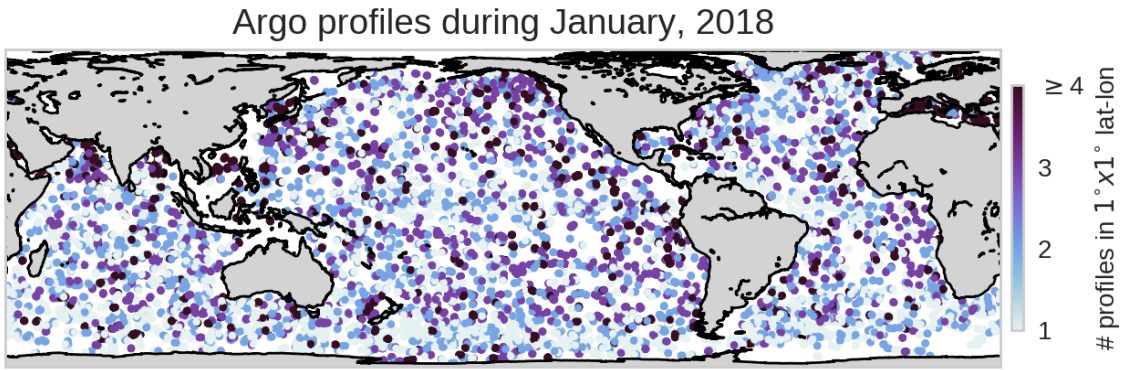

Figure 3.23. Frequency of profiles in a 1×1 degree grid during January 2018.

3.7 Conclusion

Argovis has proved that it is feasible to implement modern web-app tools for scientific application. Web-apps with similar architecture can be used as interfaces for other large climate/oceanographic datasets. A map-based interface also allows gridded products generated from the Argo datasets can be displayed side-by-side for comparison. As Argovis grows in complexity, front-end frameworks such as Angular or React.js are used to separate

client-side logic in a MVC design. Rewriting the front end this way improves the maintainability of the app and expands the capability to handle more complex features all the while sustaining the cost of running and maintaining this app.

It is in the interest of international incentives such as Argo to have a broad user community. The Argo program depends on the public support to thrive. Apps that allow ordinary people visualize their tax dollars at work bolster their support. Argovis is intuitive enough for non-programmers to use. K-12 schools may use it in their in ocean science education. Argovis can be featured at ocean science centers, displayed on public kiosks. Learning to use the app requires little to no training other than basic 'point-and-click' computer proficiency, as shown by the tutorials shown earlier in this chapter. The openness of the app lets students, in turn, to show it to their friends and family. Data visualization tools already exist for Argo but are unsuitable for researchers to use. Argovis addresses this by allowing automation tools for professionals.

The RESTfull API portion allows scientists to interface with the app's database, otherwise, they will be forced to grab data using a browser. Automation through the API ensures scientists focus more time on the non-trivial inferencing they will be doing with the data. Temperature and salinity measurements are also needed by biologists, climate scientists who undoubtedly have their preferred tools and programming languages. The API is customizable, tailoring to the needs of Argo's diverse community. The base functions displayed in Appendix C shows just how simple 'hooking' up to Argovis can be. Nothing is stopping the community from writing these four functions in R, Matlab, Julia, et cetera. Data is extracted in JSON format and adjusted into the tabular format in a few lines of code.

In sum, through its utility and ease of use, Argovis becomes the face of the Argo project.

CHAPTER 4
CONCLUSIONS AND DISCUSSION

Big data does not need to be viewed all at once. Humans do not need process petabytes at a time. A more useful application to data visualization is to have the data be accessible through an easy to use interface. This technology exists in the form of videos, where users view one frame at a time, or a browser, able to navigate complex website networks. In the case of scientific research, data should be easily divided into regions of interest. Scientists interested in the Tibetian Plateau do not need to download the Entire IMS dataset.

The tools discussed in this thesis is by its nature supplementary to their respective communities. IMS users can now make both local and global snow and ice area assertions with the calculations provided by TSM. It provides the gridded satellite community a customizable toolkit that can be changed to suit individual requirements. The source code made available at Github [28] and is explained in `www.itsonlyamodel.us` [29] aids a researcher to use TSM for their own needs. They may wish to project using another geo-coordinate system or area projection for example. Users can extend the TSM to another project that uses another gridded data set whose cell areas are unknown. Area inferencing such as TSM's snow/ice cover is extensible to vegetation, cloud cover, ocean color, et cetera. As remote sensing technology and algorithms used by the fleet of satellites products improve, more of this sort of inferencing will become more common. Consequentially, climate inferencing based on satellite methods puts pressure on the IMS community to include uncertainty in their measurements. The IMS is a blend of satellite sensing products [17], with their resolution and uncertainty. The addition of atmospheric scattering, cloud cover, vegetation, and elevation effects further complicate the physics [5]. Reducing the accuracy of these products is nontrivial, but with the increase of area-based inferencing, is predicted to be more in demand. TSM's visualization section places additional pressure, as it becomes accessible to more users by the ease of selection and time series visualization. Future work planned on TSM involves creating a 3D (latitude-longitude-time) visualization app that professional and public can use, in a similar app to `www.argovis.com` as discussed in Chapter 3.

Visualization Apps like Argovis have served two purposes. The first purpose is to serve the community; the second is to expand the community. Global data sets in the oceanographic and earth science disciplines could potentially benefit from such apps, with minimal cost to their endeavors. The architecture of Argovis is designed for high traffic

demands at low computation by placing visualization loads on the client side. The server side acts as a database and web server only; this allows a relatively high traffic web app be hosted and maintained at a low cost. Moreover, the software used is open source.

Governmental agencies NOAA and NASA are required to release their data. The amount of data gathered and released is now on the petabyte scale. Without accessibility, this amounts to little more than an archive, accessible only to domain experts. This thesis proposes that agencies and groups design their data interfaces with a user interface in mind. Argovis is designed around the following data acquisition procedure:

1. browse datasets.
2. assess its relevance to their needs.
3. visualize/select their region of interest.
4. download data in a format they can use.

An additional consideration would be to include a discussion forum to the web app, similar to `www.kaggle.com` [12] competitions, where each dataset has its discussion forum, where users post their code, figures and results in Jupyter notebooks. Forums allow outside users to contribute to the project. As the institutions themselves may be limited in manpower and funding, they may rely on forums for additional code, tutorials or discussion. For example, Argovis has an API in Python, but users proficient in other scientific languages can write APIs in other languages, such as R, Matlab, and Julia. A forum section of the site works as a source of feedback for the development team. Questions, requests, complaints can be posted and be resolved/answered by either other users or the developers themselves.

The scientific community can become bogged down in data storage problem. Open, community-driven web applications to big data visualization and analysis can help users float on this sea of data.

BIBLIOGRAPHY

[1] H. AKAIKE, *A new look at the statistical model identification*, IEEE transactions on automatic control, 19 (1974), pp. 716–723.

[2] ARGO, 2014. French Argo website `http://www.argo-france.fr/en/welcome/`. Accessed January, 2018.

[3] ARGO, 2014. UK Argo site `http://www.ukargo.net/about/technology/transmission_system/`. Accessed January, 2018.

[4] ARGO, 2018. These data were collected and made freely available by the International Argo Program and the national programs that contribute to it. (`http://www.argo.ucsd.edu`, `http://argo.jcommops.org`). The Argo Program is part of the Global Ocean Observing System. Accessed January 2017.

[5] A. BASIST, D. GARRETT, R. FERRARO, N. GRODY, AND K. MITCHELL, *A comparison between snow cover products derived from visible and microwave satellite observation*, Journal of Applied Meteorology, 35 (1996), pp. 163–177.

[6] BBC, *"snow-hit china welcomes new year"*, 2008. `http://news.bbc.co.uk/2/hi/asia-pacific/7231622.stm`. Accessed March 2018.

[7] G. E. BOX AND D. R. COX, *An analysis of transformations*, Journal of the Royal Statistical Society. Series B (Methodological), (1964), pp. 211–252.

[8] B. BRADEN, *The surveyors area formula*, The College Mathematics Journal, 17 (1986), pp. 326–337.

[9] R. B. CLEVELAND, W. S. CLEVELAND, AND I. TERPENNING, *Stl: A seasonal-trend decomposition procedure based on loess*, Journal of Official Statistics, 6 (1990), p. 3.

[10] F. FETTERER AND K. WEBSTER, *Ims daily northern hemisphere snow and ice analysis at 4 km and 24 km resolution*, National Snow and Ice Data Center, Boulder, CO, (2011).

[11] S. GUPTA, 2015. R source code for stl() function `https://github.com/SurajGupta/r-source/blob/master/src/library/stats/R/stl.R`. Accessed January, 2018.

[12] KAGGLE, 2018. Kaggle: a data science and machine learning website (`https://www.kaggle.com`).

[13] S. KOKOSKA AND D. ZWILLINGER, *Standard probability and statistics tables: Student edition*, Boca Raton, FL: CRC, (2000).

[14] G. M. LJUNG AND G. E. BOX, *On a measure of lack of fit in time series models*, Biometrika, 65 (1978), pp. 297–303.

[15] NASA, 2018. NASA JPL Sea Level Change Data Analysis Tool (DAT) `https://sealevel.nasa.gov/data/data-analysis-tool`. Accessed March, 2018.

[16] NIC, *National ice center. 2008, updated daily. ims daily northern hemisphere snow and ice analysis at 1 km, 4 km, and 24 km resolutions. boulder, co: National snow and ice data center. digital media.*, 2008. accessed November 2016.

[17] NISDC, 2017. US National Ice Center (`http://www.natice.noaa.gov/ims/`). Accessed December, 2017.

[18] NOAA, 2018. NOAA National Centers for Environmental Information, State of the Climate: Global Climate Report for Annual 2013, published online January 2014, retrieved on March 2, 2018 from https://www.ncdc.noaa.gov/sotc/global/201313. `https://www.ncdc.noaa.gov/sotc/global/201313`. Accessed March 2018.

[19] NOAA, 2018. NOAA ocean service `https://oceanservice.noaa.gov/facts/oceanfreeze.html`). Accessed February, 2018.

[20] C. A. OF SCIENCES, 2017. Institute of Tibetan Plateau Research, Chinese Academy of Sciences Third Pole Database. Accessed January, 2018.

[21] R. P. OLLITRAULT MICHEL, *Andro: An argo-based deep displacement dataset. seanoe.*, 2013. `http://doi.org/10.17882/47077`.

[22] M. PERRY, *"china's snow storms not climate change-scientists" reuters article. https://uk.reuters.com/article/idUKSYD222109._CH_.242020080131. accessed march 2018*, 2008.

[23] J. PIERRET AND S. S. SHEN, *4d visual delivery of big climate data: A fast web database application system*, Advances in Data Science and Adaptive Analysis, 9 (2017), p. 1750006.

[24] B. H. RAMSAY, *The interactive multisensor snow and ice mapping system*, Hydrological Processes, 12 (1998), pp. 1537–1546.

[25] RUTGERS, 2017. Rutgers Global Snow Lab (http://climate.rutgers.edu/snowcover), Accessed January, 2017.

[26] S. S. P. SHEN, R. YAO, J. NGO, A. M. BASIST, N. THOMAS, AND T. YAO, *Characteristics of the tibetan plateau snow cover variations based on daily data during 1997–2011*, Theoretical and Applied Climatology, 120 (2015), pp. 445–453.

[27] J. P. SNYDER, *Map projections–A working manual*, vol. 1395, US Government Printing Office, 1987.

[28] T. TUCKER, *Tibet git repository at (http://www.github.com/tylertucker202/tibet_snow_man)*, 2017.

[29] T. TUCKER, *Tibet snow man and argo blog (http://www.itsonlyamodel.us)*, 2017.

[30] W. W. WEI ET AL., *Time series analysis: univariate and multivariate methods*, Pearson Addison Wesley, 2006.

[31] C. WONG, KEELEY, *Argo users manual V3.2. http://doi.org/10.13155/29825*.

[32] T. YAO, V. MASSON-DELMOTTE, J. GAO, W. YU, X. YANG, C. RISI, C. STURM, M. WERNER, H. ZHAO, Y. HE, ET AL., *A review of climatic controls on δ18o in precipitation over the tibetan plateau: Observations and simulations*, Reviews of Geophysics, 51 (2013), pp. 525–548.

[33] T. YAO, L. THOMPSON, W. YANG, W. YU, Y. GAO, X. GUO, X. YANG, K. DUAN, H. ZHAO, B. XU, ET AL., *Different glacier status with atmospheric circulations in tibetan plateau and surroundings*, Nature Climate Change, 2 (2012), pp. 663–667.

APPENDIX

NONLINEAR LEAST SQUARES FITTING OF TIME SERIES

NONLINEAR LEAST SQUARES FITTING OF TIME SERIES

This chapter explains how one can create time series models from ARGO and IMS data sets. The programming language R uses built in time series model fitting and diagnostic tools, and are used extensively to produce model parameters. A theoretical background, borrowed heavily on Wei's textbook [30], on how these models are created is appropriate. First the theory behind linear time series shall be explained. Next model selection criterion will be discussed, followed by how model parameters are estimated.

A.1 LINEAR TIME SERIES MODELS

Time series modeling of climate data involves taking into account noise shocks. The linear time time series used in this thesis are linear combinations of past values and stochastic shocks as an input. For example sea surface temperature measurement T indexed by time t has the form

$$T_t = f(T_{t-1}, T_{t-2}, ..., a_t, a_{t-1}) \tag{A.1}$$

With a_t being a white noise process. White noise processes are time-invariant are centered about a mean μ with a standard deviation σ_a^2. A single realization at time index t produces a shock a_t on A.1.

A.1.1 STATIONARITY

It is worth noteing that a white noise process is an example of a stationary time series, that is, a finite process a_t for time index $t = 1, 2, ...n$ has a probability density function

$$P(a_t) = WN(\mu, \sigma_a^2) \tag{A.2}$$

Moreover, each shock a_t has the same probability for any time t. The first order is in reference to the marginal distribution $P(Z_t)$.

$$P(a_1) = P(a_2)... = P(a_n) = WN(\mu, \sigma_a^2) \tag{A.3}$$

Any process with this characteristic is first order stationary. This concept can be extended to any order with the inclusion of additional input parameters to $P(Z_t)$.

Definition A.1. *A finite set of random variables in a sample space ω has the form $Z_t = \{Z_1, Z_2, ..., Z_n\}$ from a stochastic process $Z(t) : t = \pm 1, \pm 2, ...$ has an n dimensional probability distribution $F_{Z_t}(x_1, x_2, ..., x_n) = P\{\omega : Z_1 \leq x_1, Z_2 \leq x_2, ..., Z_n \leq x_n\}$ for any set of real numbers $x_i = x_1, x_2, ...x_n$.*

A probability distribtion definded by A.1 with arbitrary real numbers x_i can be further refined by setting some values of x_i to be equal. If all values x_i are equal, than the distribtion is first order, and so on. In terms of a definition

Definition A.2. *Given a set of j unique real numbers from the set x_i, a process Z_t is j order stationary if its probility distribution is time invarient.*

Z_t can be 1,2,...n order stationary. Higher order stationarity m inplies lower order stationarity n for $n < m$. In the event that Z_t is n order, it is said to be strictly stationary. Linear time series can be used to model *strictly stationary* processes such as A.1. For a given real-valued process Z_t, an expected value

$$\mu_t = E(Z_t) \tag{A.4}$$

and variance

$$\sigma^2 = E(Z_t - \mu_t)^2 \tag{A.5}$$

Z_t also has an autocovariance between two times t_1 and t_2 denoted by

$$\gamma(t_1, t_2) = E(Z_{t_1} - \mu_{t_1})(Z_{t_2} - \mu_{t_2}) \tag{A.6}$$

and an autocorrelation function

$$\rho(t_1, t_2) = \frac{\gamma(t_1, t_2)}{\sqrt{\sigma^2_{t_1}}\sqrt{\sigma^2_{t_2}}} \tag{A.7}$$

Z_t is strictly stationarity when all equations A.4 to A.7 do not depend on time index t.

Linear time series can be used to model *stationary* time series. The white noise process $Z_t = a_t$ can be modeled as a linear time series because its probability density function A.2 does not depend on t and so neither do A.4 to A.7, hence it is stationary. In practice, time series data are usually shown to be covariance stationary. Only A.6 is satisfied and the rest is assumed. Hereon, the term covariance stationarity shall be refered to only as stationarity. The remaining subsections will describe the models used to make up linear time series.

A.1.2 AUTOREGRESSIVE (AR) MODELS

Past values of a time series such as A.1 often hold a critical role in climate data. To model this process, past values are weighted by a set of real numbers. This model is called an autogregressive (AR) model and is defined by

$$\dot{T}_t = \phi_1\dot{T}_{t-1} + \phi_2\dot{T}_{t-2} + \phi_p\dot{T}_{t-p} + ... + a_t = \Sigma_{k=1}^{p}\phi_k\dot{T}_{t-k} + a_t \tag{A.8}$$

Integer p is finite and ϕ_k are constants, whose values are chosen so that the model is stationary. For simplicity of notation without loss of generality, $\dot{T}_t = T_t - \mu$. Time series are

often written in a compact form using the backshift operator B. The operator acts on a time index by shifting it back once. For example $Ba_t = a_{t-1}$, $B^2a_t = a_{t-2}$ and so forth. The AR model using this notationA.9 is written below.

$$\dot{T}_t = \phi_1\dot{T}_{t-1} + \phi_2\dot{T}_{t-2} + ... + a_t = \phi_1\dot{T}_tB + \phi_2\dot{T}_tB^2 + \phi_pB^p\dot{T}_t...a_t = \Sigma_{k=1}^{p}\phi_kB^k\dot{T}_t + a_t \quad \text{(A.9)}$$

The backshift operator rewrites AR model. Temperature anomaly $\dot{T}_t$ shifts to right-hand side.

$$\dot{T}_t - \Sigma_{k=1}^{p}\phi_kB^k\dot{T}_t = a_t \quad \text{(A.10)}$$

The left-hand side compacted further by the notation

$$\dot{T}_t - \Sigma_{k=1}^{p}\phi_kB^k\dot{T}_t = \phi_p(B)\dot{T}_t \quad \text{(A.11)}$$

Where $\phi_p(B) = \Sigma_{k=0}^{p}\phi_kB^k$ with ϕ_0 always 1. So that the AR model is expressed by

$$\phi_p(B)\dot{T}_t = a_t \quad \text{(A.12)}$$

A.1.3 MOVING AVERAGE (MA) MODELS

Moving average models multiply the white noise variable a_{t-k} by a constant ϕ_k for k = 1, 2, ..., q.

$$\dot{T}_t = a_t + \theta a_{t-1} + \theta a_{t-2}...\theta a_{t-q} \quad \text{(A.13)}$$

Using the backshift notation, $a_t + \theta_1a_{t-1} + \theta_2a_{t-2}...\theta_qa_{t-q} = \theta_q(B)a_t$ so that the AR model is expressed as

$$\dot{T}_t = \theta_q(B)a_t \quad \text{(A.14)}$$

A.1.4 AUTOREGRESSIVE MOVEING AVERAGE (ARMA) MODEL

Combining AR A.12 and MA A.14 creates an ARMA model expressed by

$$\phi_p(B)\dot{T}_t = \theta_q(B)a_t \quad \text{(A.15)}$$

A.1.5 AUTOGREGRESSIVE INTEGREATED MOVING AVERAGE (ARIMA) MODEL

Up until now each of these time series are stationary. If random variable temperature for step t = a, $\dot{T}_a$ has the same probability for $\dot{T}_b$ then it is stationary.

Imagine a increasing time series; The first point is likely smaller than the last point. Then this time series is not stationary. In this case, taking the difference and applying time series satisfies stationarity. The change in temperature is constant, so its time series is stationary. In the finite realm, we take an initial differencing step corresponding to the

"integrated" part of the model. For generalization, this can be applied d times until the resulting series is stationary.

$$\dot{T}_t(1-B)^d = S_t \tag{A.16}$$

Where, S_t is a stationary time series. Until now, the time series models had to be stationary, with this mean stabilizing transformation, it is possible to model some non stationary processes using an AR, MA, or the generalized ARMA model. From a calculus framework, the model is integrated d times to return it to its original design. called an ARIMA model written as

$$\phi_p(B)(1-B)^d\dot{T}_t = \theta_q(B)a_t \tag{A.17}$$

A.1.6 VARIANCE STABILIZING TRANSFORMATION

As it was shown in the previous sections, it is possible to transform some non-stationary time series into stationary. Difference transformations are suitable for trending time series, but do not sucessfully account for variance stationarity. A common example would be the volitility of snowfall during the winter/spring months compared to small variation during the summer months, a clear indication of a non-stationary process. To stabilize the variance, assume variance is a function dependent on a moving average.

$$Var(Z_t) = cf(\mu_t) \tag{A.18}$$

Where c is a arbitrary positive constant, and $f(\mu_t)$ is a positive function dependent on an . average indexed by t. Next consider a transformation $T(Z_t$ such that

$$Var(T(Z_t)) = \sigma_T^2 \tag{A.19}$$

Taking the Taylor series about μ_t

$$T(Z_t) \approx T(\mu_t) + T'(\mu_t)(Z_t - \mu_t) \tag{A.20}$$

The variance of A.20 is

$$Var(T(Z_t)) \approx \cancelto{0}{T(\mu_t)} + T'(\mu_t)^2(Var(Z_t) - \cancelto{0}{Var(\mu_t)}) \tag{A.21}$$

Substituting A.18 into A.21 yields

$$Var(T(Z_t)) \approx T'(\mu_t)^2 cf(\mu_t) \tag{A.22}$$

Knowing that a stationary time series has constant variance, let $Var(T(Z_t)) = c$ and rearrange A.22 to solve for $T'(\mu_t)$.

$$T'(\mu_t) \approx \frac{1}{f(\mu_t)} \tag{A.23}$$

Taking the indefinite integral of A.23 yields the following criteria that the transformation function must statisfy the following.

$$T(\mu_t) = \int \frac{1}{f(\mu_t)} + C \tag{A.24}$$

In general Box and Cox provided a generalized expression for variance stabilizing functions[7], that satisfy A.24

$$T(Z_t) = \begin{cases} Z_t^{\lambda}, & \lambda \neq 0 \\ ln(Z_t), & \lambda = 0 \end{cases} \tag{A.25}$$

Note that here the natural log transformation will fail to transform $Z_t \leq 0$. Oftentimes climate data is is described as anomalies, which are centered about 0 with negative values. Another example could be temperature taken in Celcuis. Without loss of generality, it is possible to perform a linear transformation on Z_t followed by a log transformation. Adhering to temperature, this would be equivalent to taking the log transform of degrees Kelvin instead of Celcius.

$$T(Z_t) = ln(Z_t + C) \tag{A.26}$$

Where $Z_t = C \geq 0$.

Variance transformations when needed make better fitting models. In the next selection, two main criteria are used to determine which linear time series is best.

A.1.7 MODEL DIAGNOSTICS

There is no ideal model selection method. Singling out best fitting models is a subjective decision, however there do exist metrics that can be used to compare different models with one another. One well known metric is the Akaike information criterion (AIC)[1].

$$AIC = 2k - 2log(\hat{L}) \tag{A.27}$$

where k is the number of parameters and $\hat{L}$ is the maximum value for the Likilihood function for the model. The AIC provides a balanced metric that rewards goodness of fit and penalizes overfitting models.

Box and Ljung [14] also include a diagnostic that tests the lack of fit. The test is applied to the residuals of a model after fitting the model to the data by examining the first m

autocorrelations of the residuals. If the model is a good fit, there should be no autocorrelations. There are m p-values calculated. The null hypothesis states The model does not show a lack of fit. Thus large p-values indicate that the test fails to reject the null Hypothesis.

Given a time series Z_t with length n, Consider the summation

$$Q = n(n+2)\Sigma_{k=1}^{m}\frac{\hat{r}_k^2}{n-k} \tag{A.28}$$

where $\hat{r}_k^2$ is the estimated autocorrelation of the series at lag k. The Box-Ljung test rejects the null hypothesis if $Q > \chi^2_{1-\alpha,h}$. $\chi^2_{1-\alpha,h}$ is the chi-square distribution with $h = m - p - q$ degrees of freedom of a ARMA(p,q) model.

APPENDIX

SEASONAL DECOMPOSITION BASED ON LOESS

SEASONAL DECOMPOSITION BASED ON LOESS

Seasonal Decomposition Based on LOESS by Cleveland et al. [9] allows a non stationary time series with a seasonal trend to be decomposed into three parts: The trend T_t, season S_t, and the residual R_t. Where the original signal Z_t is combination of all three.

$$Z_t = T_t + S_t + R_t \tag{B.1}$$

In comparison, seasonal averaging as seen in Chapter 2 only decomposes the Seasonal element, leaving the trend and residuals intact. Modeling stationarity time series favors the LOESS approach, and modeling the residuals as a stationary time series. R's STL() function, is the implentation to this non-parametric approach by Clevelant et al. [9].

B.0.0.1 LOESS FIT

STL uses the LOESS curve $\hat{g}(x)$ is a smoothing parameter for $y(x)$.
$y(x) = \hat{g}(x) + \epsilon(x)$. The independent variable is written in a discreet form so that it is indexed by t.

$$x = x_t, t = 1, 2, ...n \tag{B.2}$$

LOESS fits linear or quadractic curves to regions of the time series. The size of the region is fixed by a focal window q number of points of some time series $z(x)$. Let this subset of points be denoted $y(x) \subseteq z(x)$. Each point in value of $y(x)$ is weighted with a tricube function.

$$W(u) = \begin{cases} (1-u^3)^3, & 0 \leq u < 1 \\ 0, & \text{otherwise} \end{cases} \tag{B.3}$$

$$v_t(x) = W\left(\frac{|x_i - x|}{\lambda_q(x)}\right) \tag{B.4}$$

Where $\lambda_q(x)$is that qth farthest distance from x_t. Note that the focal window q can be larger than n, by adjusting $\lambda_q(x)$ so that $\lambda_q(x) = \lambda_n(x)\frac{q}{n}$. A 1st or 2nd degree order polynomial $p_{x_t}(x)^{(q)}$ with weight $v_t(x)$ at point $(x_t, y(x_t)$ is fitted to $y(x)$. Additionally, a reliablity factor ρ_i is weighted to $v_t(x)$ for handling outliers in $y(x)$. For example suppose that it is known that the variance for y_t is σk_t for a known k_t. Then the weight is choosen to stabilize the variance for all $y(x)$, in this case $\rho_t = 1/k_i$, there by each point is weighted by $\rho_t v_t(x)$, thus a smoothed point $z_{smoothed}(x) = p_{x_t i}(x)^{(q)}$. This adds robustness to the LOESS smoothing. Details on how stl() handles ρ_t is discussed in the B.0.0.2. The process is continued for each point in $z(x)$.

Densely sampled seasonal time series with few missing points such as those shown in Chapters 2 and 3 work well with LOESS methods. In the next section, the STL functions underlying algorithm will show how LOESS methods and low pass filters is used to decompose Z_t.

B.0.0.2 IMPLEMENTATION OF STL ALGORITHM

The basic structure of the STL algorithm are two forloops: The outer calculates ρ_i of a completed LOESS curve $n_{(o)}$ number of times, and the inner loop decomposes a series $n_{(i)}$ number of times. Seasonal decomposition of sereries described in Chapters 2 and 3 set the outer loop length $n_{(o)} = 0$ and set $\rho_i = 1,\ \forall i$. The time series object Z_t with specified period is input into stl().

Pseudocode 3 used to describe how the time series used in Chapters 2 and 3 implement R's stl() function as desribed in Clevelant et al. [9] and R source code [11].

Algorithm 3 STL Implementation

procedure DECOMPOSE SERIES
  INPUT: Z_t *(time series object)*
  $n \leftarrow$ length of Z_t
  $period \leftarrow$ period of time series
  $s.window \leftarrow 10 * n + 1$
  $s.degree \leftarrow 0$
  $l.degree,\ t.degree \leftarrow 1$
  $t.degree \leftarrow$ **nextodd** $\left(\lceil \frac{1.5 * period}{1 - 1.5/s.window} \rceil\right)$
  $n_{(i)} \leftarrow 2$
  $k \leftarrow 0$
  $T_t^{(k)} \leftarrow 0$
  $n_{(i)} \leftarrow 2$
  $T_t^{(0)} \leftarrow$ array of zeros with length n
  for $k = 0,\ k{+}{+},$ `while` $k \leq n_{(i)}$ **do**
    $DT_t^{k+1} = Z_t - T_t^{(k)}$
    $C_t^{(k+1)} \leftarrow$ **loess**$(DT_t^{(k)}, q = s.window, degree = 1)$
    $A_t \leftarrow$ **lowpass**$(C_t^{(k+1)}, period)$
    $B_t \leftarrow$ **lowpass**$(A_t, 3)$
    $L_t^{(k+1)} \leftarrow$ **loess**$(B_t, q =$ **nextodd**$(period), degree = 1)$
    $S_t^{(k+1)} \leftarrow C_t^{(k+1)} - L_t^{(k+1)}$
    $DS_t^{(k+1)} \leftarrow Y_t - S_t^{(k+1)}$
    $T_t(k+1) \leftarrow$ **loess**$(DS_t^{(k+1)}, q = t.window, degree = 1)$
  $R_t = Y_t - S_t^{(k+1)} - T_t^{(k+1)}$
  OUTPUT : $S_t^{(k+1)},\ T_t^{(k+1)},\ R_t$

APPENDIX

PYTHON API

PYTHON API

The Python basic functionality is provided below.

```
import pandas as pd
import numpy as np
import requests
def get_profile(profile_number):
    resp =
        requests.get('http://www.argovis.com/catalog/profiles/'+profile_number)
    # Consider any status other than 2xx an error
    if not resp.status_code // 100 == 2:
        return "Error: Unexpected response {}".format(resp)
    profile = resp.json()
    return profile

def get_platform_profiles(platform_number):
    resp =
        requests.get('http://www.argovis.com/catalog/platforms/'+platform_number)
    # Consider any status other than 2xx an error
    if not resp.status_code // 100 == 2:
        return "Error: Unexpected response {}".format(resp)
    if resp.status_code == 500:
        return "Error: 500 status {}".format(resp)
    platformProfiles = resp.json()
    return platformProfiles

def get_platform_measurements(profiles):
    '''
    Retrieves all measurements included in a list of platforms.
    '''
    stationParam = []
    for profile in profiles:
        stationParam.append(profile['station_parameters'])
    flatList = [item for sublist in stationParam for item in sublist]
    if isinstance(flatList[0], list):
        flatList = [item for sublist in flatList for item in sublist]
```

```
    uniqueList = list(set(flatList))
    uniqueList = [s for s in uniqueList if s != ''] # ignore blank station
        params.

    measurement_keys = [x.lower() for x in uniqueList]
    return measurement_keys

def parse_into_df(profiles):
    #initialize dict
    meas_keys = profiles[0]['measurements'][0].keys()

    #get measurement numbers
    dfKeys = get_platform_measurements(profiles)

    df = pd.DataFrame(columns=meas_keys)
    for profile in profiles:
        profileDf = pd.DataFrame(profile['measurements']) # may inlude qc
            values
        profileDf['cycle_number'] = profile['cycle_number']
        profileDf['profile_id'] = profile['_id']
        profileDf['lat'] = profile['lat']
        profileDf['lon'] = profile['lon']
        profileDf['date'] = profile['date']
        df = pd.concat([df, profileDf])
    return df

def get_selection_profiles(startDate, endDate, shape, presRange=None):

    baseURL = 'http://www.argovis.com/selection/profiles'

    startDateQuery = '?startDate=' + startDate
    endDateQuery = '&endDate=' + endDate
    shapeQuery = '&shape='+shape.replace(' ','')
    if not presRange == None:
        pressRangeQuery = '&presRange=' + presRange
        url = baseURL + startDateQuery + endDateQuery + pressRangeQuery +
            shapeQuery
    else:
```

```
    url = baseURL + startDateQuery + endDateQuery + shapeQuery
resp = requests.get(url)
# Consider any status other than 2xx an error
if not resp.status_code // 100 == 2:
    return "Error: Unexpected response {}".format(resp)
if resp.status_code == 500:
    pdb.set_trace()
    return "Error: 500 status {}".format(resp)
selectionProfiles = resp.json()
return selectionProfiles
```

APPENDIX

SHELL SCRIPT USED TO UPDATE MONGODB DATABASE

SHELL SCRIPT USED TO UPDATE MONGODB DATABASE

Listing D.1. bash script used to syncronize local mirror of vdmzrs.ifremer.fr::argo. Python code updates MongoDB with changes.

```
#!/bin/bash
echo 'Start of rsync and List'
DATE=`date +%y-%m-%d-%H:%M`
echo $DATE
FTPDIR='/storage/ifremer/'
ARGODIR='/home/argo-database/'
QUEUEDIR=$ARGODIR'queuedFiles/'
OUTPUTNAME=$QUEUEDIR'ALL-DACS-list-of-files-synced-'$DATE'.txt'
echo 'Starting rsync: writing to '$FTPDIR
#Sync only /profiles/[RDM]*
rsync -arvzhim --delete --include='**/' --include='**/profiles/[RDM]*.nc'
    --exclude='*' --exclude='**/profiles/B*' vdmzrs.ifremer.fr::argo $FTPDIR >
    $OUTPUTNAME
ENDDATE=`date +%y-%m-%d-%H:%M`
echo 'End of rsync and List'
echo $ENDDATE

echo 'Starting to add DB'
cd $ARGODIR
python3.6 processQueue.py kadavu
PYENDDATE=`date +%y-%m-%d-%H:%M`
echo 'Added new files to DB'
echo $PYENDDATE
```