Sea Ice Parameters in Antarctica: Seasonal variations in the Sea-ice Thickness

Shreya Trivedi
mailto:shreyatrivedi26@ucla.edu

March 8, 2020

Abstract

Antarctic sea ice plays an important role in regulating the heat, moisture and moisture exchanges between the atmosphere and the ocean, and salinity of the ocean. Current studies of Antarctica confirm the area-integrated total sea ice extent grew to record maximum values in four of the last six years, whilst the 2016-17 summer has been marked by record low ice cover. The reasons for the variations in the sea ice extent are not fully understood. Given the important role of Antarctic sea ice; in this era of climate change it is vital to understand the factors contributing to its variability. Current work focuses on the sea ice concentration, which allows an understanding of the areal variation of the ice. The sea ice thickness (SIT) which allows an understanding of the variability of the volume of sea ice has not been explored to the same extent because there was a lack of data. This study uses a newly-released circum-Antarctic SIT dataset to examine the variability in Antarctic SIT and to investigate the relationships between the SIT and climatic variables such as surface and ocean temperatures as well as convergence and divergence in ocean flows over a period of 2002-2011. This study gives a first look at the spatial and temporal variability of the SIT showing how it varies between the continent and sea-ice edge, with SIT favoring increase towards the landmass. Temporal variation shows maximum SIT values concentrated at the start of spring when the sea-ice accumulation is at its peak.

Keywords: *; Sea-ice Thickness ; Climate Change; Seasonal Variability*

Contents

1 Introduction **3**

2 Methodology and Datasets **3**

2.1 Dataset . . . 3

2.2 Python Scripting . . . 4

3 Results **8**

3.1 Temporal trends: Relationship between Air Temperature and Sea-ice Thickness (SIT) . . . 8

3.2 Spatial analysis: Studying spatial extents and variations of SIT . 9

3.3 Visualization using R . . . 10

4 Discussion and Conclusions **14**

List of Figures

1 NetCDF format: Reading a climate variable . . . 3

2 Visualizing statistics for a Gridded data (Air Temperature) . . . 7

3 Trend Plot comparing SIT and air temperatures at 850hpa) . . . 9

4 Spatial Variations in SIT for October, 2002 . . . 10

5 Studying variability in SIT using the limits of Standard Deviation: The dark blue band indicates the limit formed by 1SD value while the light blue band is the limit by 2SD . . . 12

6 Seasonal(Monthly) variability in the SIT values using density curves estimates . . . 14

1 Introduction

The Antarctic sea-ice plays is of global significance owing to its role in the global ocean's uptake of heat, and potentially in the penetration of warm, ocean water to the base of Antarctic ice shelves. This makes the study of Antarctic sea ice and its spatial and temporal variations important, primarily due to two reasons, firstly, there are very limited studies which have been conducted in the research field focusing mainly upon the reasons and the climatological causalities behind the changing sea-ice dynamics in Antarctica. Secondly, to study the tele-connections which might impact the regional and global climatic interactions as a result of variations taking place in Southern Oceans. In todays's scenario where Climate Change is considered as the 'prime suspect' for changing climatological dynamics, it becomes all the more important to asses the reasons for the sea-ice variability in order to ascertain whether the observed changes are due to natural variability alone, or represent a forced anthropogenic response.

2 Methodology and Datasets

2.1 Dataset

The dataset format used in the study is called the NetCDF format (with the extension .nc). The Network Common Data Form, or netCDF, is an interface to a library of data access functions for storing and retrieving data in the form of arrays. An array is an n-dimensional (where n is 0, 1, 2, ...) rectangular structure containing items which all have the same data type (e.g., 8-bit character, 32-bit integer). A scalar (simple single value) is a 0-dimensional array.

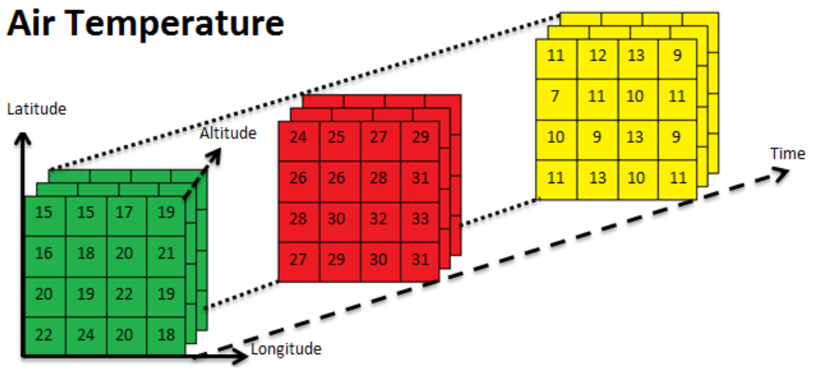

Figure 1: NetCDF format: Reading a climate variable

The study uses gridded datasets for making seasonal comparisons of the Sea-ice Thickness(SIT) values around Antarctica. A surface air temperature

dataset is also used to derive relationships between sea-ice parameters and global temperatures.
The dataset is in the form of the spatially gridded values over the time period of 2002-2011. The dataset only incorporates two seasons namely, Spring and Winters (the months of May, June, July, August, September and October)

2.2 Python Scripting

Python can read and analyse NetCDF files via two inbuilt packages namely *netCDF4 and xarray*. These packages make it very handy to iterate through this 3-D gridded files and and read the values located at every coordinate. (intersection of the latitude and longitude for sliced for every time-step.)

```python

import netCDF4 as nf # This package helps to read the 3-D gridded
    (.nc) format.
import datetime as dt #This helps to convert the time-steps in the
    netcdf files into readble date format
import numpy as np
import matplotlib.pyplot as plt #For plotting spatial variables
from mpl_toolkits.basemap import Basemap #Provides basemaps to be
    used as reference backgrounds for the
#plotting the globally referenced climate variables over space and
    time
```

Listing 1: Importing required packages

- **Reading and Iterating through the variables:**

  Both *xarray and netcdf* packages help in reading the gridded files by converting them into arrays or dictionaries respectively. The functions in the package help in reading the variables and extracting them to be used individually.

```python

data_temp = nf.Dataset("air.mon.mean_1979-2015.nc") #importing
    datasets using netcdf

#defining the variables using the ncdf4 library
time_temp = data_temp.variables["time"]
lon = data_temp.variables["lon"]
lat = data_temp.variables["lat"]
level = data_temp.variables["level"]
air = data_temp.variables["air"]


#Creating the time stamps in the form of real time date format:

time_start_temp = nf.date2index(dt.datetime(1979,1,1),time_temp,
    select="nearest")
time_end_temp = nf.date2index(dt.datetime(2015,12,31),time_temp,
    select="nearest")
```

```
dates= nf.num2date(time_temp[:],time_temp.units)
time= ([date.strftime('%Y-%m-%d') for date in dates[:]])
```

Listing 2: Reading variables in NetCDF

- **Data Slicing: Selection of desired time or pressure levels:**

  Selecting only the months of May, June, July, August, September and October and the years from 2002-2011. The filtering of the dates is done so that comparison can be made with the another NetCDF file which incorporates the sea-ice values for the above timescale.

  Since the NetCDF files contain more than one pressure levels, desired pressure level was extracted from the matrix which was relevant for the study. The pressure level at **index 2** is 850 mb which is the near surface pressure level. The variables in a NetCDF files are read in the format *current shape = 444(time), 17(levels), 73(lat), 144(long).*

```
#selecting the months:
new_mon_temp=[]
for ii in range(len(time)):
    if time[ii][5:7]=='05'or time[ii][5:7]=='06' or time[ii][5:7]==
    '07' or time[ii][5:7]=='08' or time[ii][5:7]=='09' or time[ii
    ][5:7]=='10':
        new_mon_temp.append(time[ii])


#selecting the years:
new_time_temp=[]
for ii in range(len(new_mon_temp)):
    if new_mon_temp[ii][0:4]=='2002'or new_mon_temp[ii][0:4]=='2003
    ' or new_mon_temp[ii][0:4]=='2004' or new_mon_temp[ii][0:4]=='
    2005' or new_mon_temp[ii][0:4]=='2006' or new_mon_temp[ii
    ][0:4]=='2007' or new_mon_temp[ii][0:4]=='2008' or new_mon_temp
    [ii][0:4]=='2009' or new_mon_temp[ii][0:4]=='2010' or
    new_mon_temp[ii][0:4]=='2011':
        new_time_temp.append(new_mon_temp[ii])


#selecting the 850mb pressure level:
air_temp_850hpa=air_temp[:,2,:,:]
```

Listing 3: Data Slicing (using temperature dataset)

- **Statistical operations on Gridded Variables:**

  The study required the use of temporal trends of the data for comparing variations in a variable over a specified period of time. For this purpose, functions were defined that iterated through the lat-lon matrices corresponding to every time-step and calculated an area-averaged value against each time-step in the data. These are called as *Area-Averaged Values.*

```python
#Mean:

def matrix_mean(timestep_matrix):
    temp_mean=[]
    for ii in range(0,len(timestep_matrix)):
        mean = np.mean(timestep_matrix[ii])
        temp_mean.append(mean)
    return(temp_mean)

#Standard deviation:

def matrix_std(timestep_matrix):
    temp_std=[]
    for ii in range(0,len(timestep_matrix)):
        std = np.std(timestep_matrix[ii])
        temp_std.append(std)
    return(temp_std)

#Maximum values:

def matrix_max(timestep_matrix):
    temp_max=[]
    for ii in range(0,len(timestep_matrix)):
        maxm = np.max(timestep_matrix[ii])
        temp_max.append(maxm)
    return(temp_max)

#Minimum values:

def matrix_min(timestep_matrix):
    temp_min=[]
    for ii in range(0,len(timestep_matrix)):
        minm = np.min(timestep_matrix[ii])
        temp_min.append(minm)
    return(temp_min)
```

Listing 4: Defining functions for operating on gridded variables

- **Creating Visualizations using NetCDF:** In order to create trend lines for the gridded variables, outputs from the above statistical functions were used and then plotted using the library *matplotlib*.

```python

def visualize(timestep_matrix):

    #1) Finding mean, max, min and standard deviation
    max_global = matrix_max(timestep_matrix)
    min_global = matrix_min(timestep_matrix)
    mean_global = matrix_mean(timestep_matrix)
    std_global = matrix_std(timestep_matrix)

    #2) create our three subplot holders
    fig = matplotlib.pyplot.figure(figsize=(20.0,10.0))
    axes1= fig.add_subplot(2,2,1)
    axes2= fig.add_subplot(2,2,2)
    axes3= fig.add_subplot(2,2,3)
    axes4= fig.add_subplot(2,2,4)

    #3) plot each of them
    fig.suptitle('Global Temperature Trends and Statistics:
    1979-2015', fontsize=25)
    axes1.set_ylabel('Average')
    axes1.plot(mean_global)

    axes2.set_ylabel('Maximum')
    axes2.plot(max_global)

    axes3.set_ylabel('Minimum')
    axes3.plot(min_global)
    axes3.set_xlabel('No. of Timesteps')

    axes4.set_ylabel('Standard Deviation')
    axes4.plot(std_global)
    axes4.set_xlabel('No. of Timesteps')

    #4) we want to show the plot
    fig.tight_layout
    matplotlib.pyplot.show()
```

Listing 5: Defining functions for creating Visualizations

The above code created following output:

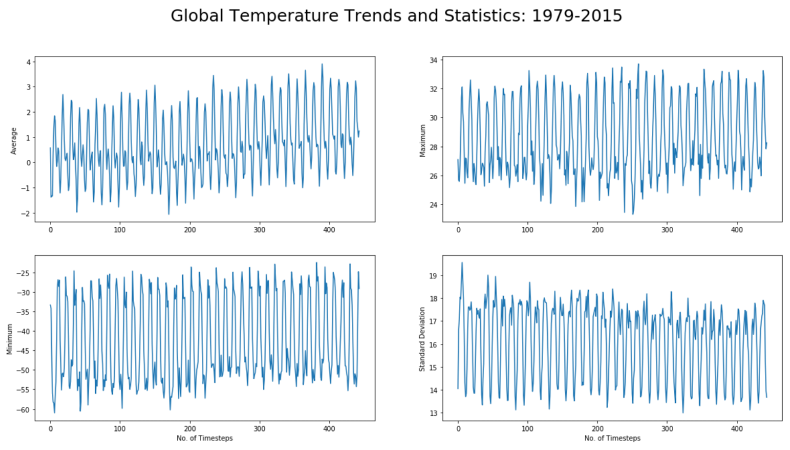

Figure 2: Visualizing statistics for a Gridded data (Air Temperature)

3 Results

3.1 Temporal trends: Relationship between Air Temperature and Sea-ice Thickness (SIT)

Studying Antarctic sea-ice thickness variability is the primary objective of the study. Besides looking at the variability trends in the dataset, the research would also try to explore the inter-relationship between the sea-ice parameters and other climatic parameters. This would help in building greater understanding of the factors controlling the changes in the sea-ice patterns across seasons and years.

A temporal trend analysis was conducted to look at the variations of the sea-ice thickness with the surface temperatures at 850 hpa.

```python
#One mean value obtained for each matrix (mesh of latitude and
    longitude) for every timestep in the gridded data.

#mean Sea Ice Thickness (SIT) array

SIT_mean=[]
for ii in range(len(SIT)):
    mean_SIT = np.nanmean(SIT[ii])#This is done to deal with the
    very high NaN values in the SIT dataset.
    SIT_mean.append(mean_SIT)
print(SIT_mean)

#mean temperature array

temp_mean=[]
for ii in range(len(air_temp_850hpa)):
    mean = np.mean(air_temp_850hpa[ii])
    temp_mean.append(mean)
print(temp_mean)


# Plotting mean values on the same plot for comparisons:

fig = plt.figure(figsize=(16.0,10.0))
plt.plot(time,SIT_mean, marker='o', markerfacecolor='blue',
    markersize=10, color='skyblue', linewidth=4,label='SIT')
plt.plot(time,temp_mean, marker='', color='red', linewidth=2,
    linestyle='dashed', label="Temperatures")
plt.xticks(np.arange(0, 55, 4.0)) #axis intervals defined
plt.xlabel('Dates') #these are dates with a well defined date
    format
plt.ylabel('Mean values')
plt.title('Global Temperatures and SIT trends:Spring and Winter
    Seasons (2002-11)')
plt.legend(loc='best') #It chooses the best positioning for legend
plt.show()
```

Listing 6: Comparing two gridded variables

The above code created following output:

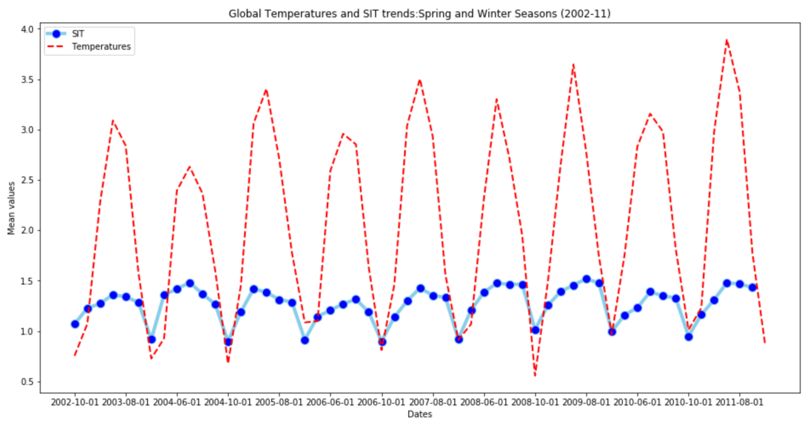

Figure 3: Trend Plot comparing SIT and air temperatures at 850hpa)

3.2 Spatial analysis: Studying spatial extents and variations of SIT

Along with temporal trends or change in the SIT values with time, it is very important to note the spatial variations of SIT for a particular time period. The spatial variations also help in understanding the changes in the extent and thickness of the sea-ice parameters during a particular season. Hence, Spatial Variability studies are very important for studying seasonal patterns.
While dealing with spatial plots, another crucial point which plays an important role in the visualization is the use of relevant projections for the maps. In our case, **Polar Projections** are the best for representing the South Pole. For this purpose, the *Basemap feature* is used for all spatial plots.

```
#Defining a function for creating Spatial Plots

def plt_map(data):
    m = Basemap(projection='spstere',boundinglat=-55,lon_0=90,
    resolution='l') #Creates a basemap with south pole oriented
    projection with the domain extending from 55S to 90S.
    x, y = m(lon, lat) #Creates a mesh out of the lat-lon matrix
    fig = plt.figure(figsize=(15,7))
    m.fillcontinents(color='white',lake_color='gray') #specifics
    for the basemap aesthetic
    m.drawcoastlines()
    m.drawparallels(np.arange(-80.,81.,20.)) #spacing and extent of
     latitudes
    m.drawmeridians(np.arange(-180.,181.,20.)) #spacing and extent
    of longitudes
    m.drawmapboundary(fill_color='skyblue')
    m.contourf(x,y,data,40,cmap=plt.cm.get_cmap('inferno'));#Using
    the fill countor method of plotting
    plt.title("Seasonal Variations in SIT for October,2002")
    plt.colorbar()
```

```
#Plotting the spatial map of SIT for the first timestep at index 0:
plt_map(SIT[0])
```

Listing 7: Creating Spatial Plots

The above code created following output:

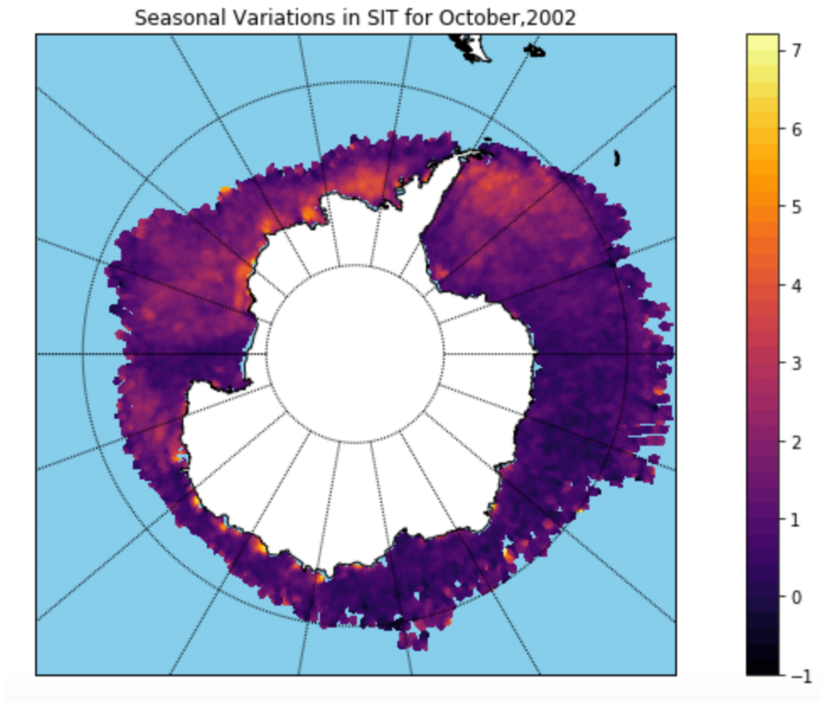

Figure 4: Spatial Variations in SIT for October, 2002

3.3 Visualization using R

R Programming lets one learn the art of visualization by offering a set of inbuilt functions and libraries to build visualizations and present data. R has several systems for making graphs, but *ggplot2* is one of the most elegant and most versatile. ggplot2 implements the grammar of graphics, a coherent system for describing and building graphs. The library ggplot2 allows one to do more faster and easy visualization by learning one system and applying it in many places.

- **Assessing variability in sea-ice data using Standard Deviation ranges:** The main objective of the study is to assess variations in the SIT values over the time-period of 11 years (2002-2011). **Standard Deviation** is the statistical measure of variability which quantifies how much a dataset varies from its mean.

Standard Deviation $= \frac{\sqrt{X - MeanofXs}}{No.ofObservations}$

Another measure of variability is called the **Anomalies**. The anomalies are calculated as follows:

Anomaly = (Observation - Mean of observations)

The anomalies help in getting rid of the extremes and plot a variability graph within a positive and negative range of values signifying the deviation from mean.

Explanation of the plot:
In the following plot, both of the above variability measures are used to quantify temporal variations in SIT. The trend line obtained by using *geom_smooth()* helps in providing a clear and tidy line plot of how the values of SIT have changed over the course of 11 years using the in-built linear model. Since, the time-period is relatively small for any concrete climatic conclusions, there is a slight increase in SIT within that period as understood by the very gentle positive slope of the trend line. The limits under *geom_ribbon()* are defined by the standard deviations and twice the standard deviations. This plot shows the degree of deviation based on whether the values lie within the limits of 1SD or 2SD. In this plot, we can conclude that the SITs are contained within the 2SD and hence vary no more than 2SD fro the mean.

```

library(ggplot2) #This package helps in creating visualizations in
    R
setwd("/home/eebc177student/Developer/Repos/eeb-c177-project/python
    -scripting") #setting the directory from where files would be
    read
sit <- read.csv("SIT_average.csv",sep = ",",header = T) #Importing
    the .csv file from the directory: those separated by commas
sit$anomaly <- sit$col1-mean(sit$col1) #Finding the anomalies and
    storing it in a new column within the dataset "sit"
St_Dev <- sd(sit$anomaly) #finding the standard deviation of the
    column anomaly St_Dev2 <- 2*sd(sit$anomaly)#Twice the deviation

tm=seq(as.Date("2002-10-15"), as.Date("2011-09-15"), by="months")#
    The as.date function creates the list of dates and spaces them
    according to defined 'by=' function which in this case is
    months. Hence, we get a list of monthly values. seq() function
    creates a continuous sequence of the dates defined.

dates <- as.data.frame(tm)#converting the dates into a data frame

```

```r
dates$month <- months(dates$tm, abbreviate = T) #creating a new
    column in the dataframe 'dates' called months which contain the
     "months" of the dates. month() helps in generating the months
    corresponding to the dates.

winter_spring <- subset(dates, month %in% c("May","Jun","Jul","Aug"
    ,"Sep","Oct")) #subset() is used for filtering the required
    values from a given dataframe. Here, it is the months from May
    to Oct.

ggplot(sit, aes(x = winter_spring$tm, y = sit$anomaly)) +
  geom_line() +
  geom_ribbon( aes(ymin = St_Dev*-1, ymax = St_Dev,fill = St_Dev),
    alpha = .15) +
  geom_ribbon( aes(ymin = St_Dev2*-1, ymax = St_Dev2,fill = St_Dev2
    ), alpha = .15)+
  geom_smooth(method = "lm", size=1,alpha = .2,level=0.99) + ylim(c
    (-0.5,0.5)) +xlab("Years")+
  ylab("Anomalies")+ggtitle("SIT Anomalies with mean deviations
    (2002-2011)")+labs(colur="Standard Deviation")

#geom_line() under ggplot package creates a line plot for anomalies
    . The line plot is overlaid by smoothed trend line using a
    method ="lm" which means linear model.
#geom_ribbon() displays a y interval defined by ymin and ymax which
     here are defined by the positive and negative values of
    Standard deviation(SD) and 2*SD.
```

Listing 8: Plotting variability in SIT

The above code created following output:

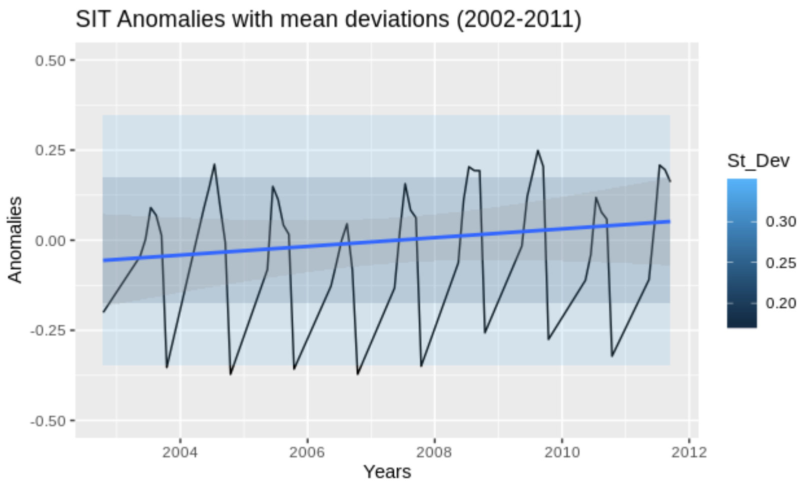

Figure 5: Studying variability in SIT using the limits of Standard Deviation: The dark blue band indicates the limit formed by 1SD value while the light blue band is the limit by 2SD

- **Seasonal variability in sea-ice data using Density curves:** A density curve is a curve that is always on or above the horizontal axis, and has area exactly 1 underneath it. When considering a specific data point, there is area to the left and area to the right. A *Normal* curve is special case of density curve that mimics a symmetric histogram and the mean and median are *equal*. Other curves may be skewed as their corresponding histogram with the mean skewed in the direction of the tail.

  Graphically, the MEDIAN of a density curve is the equal-area point, the point with half the area under the curve to its left and the other half to its right. The quartiles divide the area under the curve into quarters. 1/4 of the area is to the left of Q1 and 3/4 of the are is to the left of Q3. The *median* is the point at which the curve would *balance* if made of solid material.

  Explanation of the plot:

  The density curve mainly plots the density estimates of the dataset based on the defined categories. The categories for the plot below are the 'months' hence, the density curves are indicators of how the SIT anomalies are changing within one month for the entire time period of 11 years. These curves are an indicator of *"climatological anomalies"* which are defined as an average or baseline to evaluate climate events and provide context for year-to-year variability. Variability from these averages is typical and looks at the magnitude of extremes.

  The density plot below uses the density curves plotting the SIT for every month where the extent of the curve defines the ranges of SIT values in a particular month. An additional *fill=* function provides colour gradient for changing SIT anomalies within the months. Hence the x-axis and the fill both depict the SIT variations where the former mainly reflects the extent of SIT within a month while the colour gradient shows the SIT ranges/variations within a month.

  The plot highlights that the month of October (which happens to be the beginning of spring time in the Antarctic) is the one which shows the minimum values of SIT anomalies. This was expected because spring season has is when the sea-ice starts to melt. While it is evident that the maximum SIT values are found during June and July which are the peak winter months in the Antarctica. The SIT anomalies tend to reduce while moving away from the peak winters either towards May or towards September.

```r

dens_sit <- ggplot(data=sit,aes(x=sit$anomaly, y=winter_spring$
    month,fill=stat(x))) #ggplot() loads in the dataset and assigns
     values for x and y axes from the dataset entered. fill= is the
     command used to fill in the values with colours which here are
     the anomalies.

```

```
#Note that we need to map the calculated x value (stat(x)) onto the
     fill aesthetic, not the original temperature variable. This is
     the case because geom_density_ridges_gradient calls stat_
    density_ridges (described in the next section) which calculates
     new x values as part of its density calculation.

dd_plot=dens_sit+ geom_density_ridges_gradient(rel_min_height=0.01)
    + #This function uses density estimates and creates ridge-lines
     with colour gradients filled with the SIT anomalies in a month
    . The height represents the the max. peak heights and gaps
    between the other curves.
  scale_fill_viridis_c(name="SIT (in m)",option="C")+ # The viridis
     scales provide colour maps that are perceptually uniform in
    both colour and black-and-white.The option='C' denotes colour.
  xlab("SIT") +  ylab("Months") + ggtitle("Density Estimates: SIT
    for Winter-Spring(2002-2011)")
dd_plot
```

Listing 9: Plotting seasonal/monthly variability in SIT using Density functions

The above code created following output:

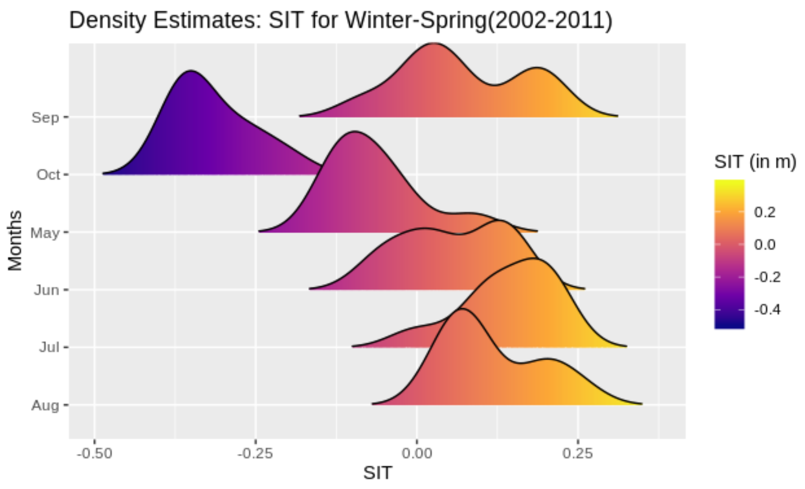

Figure 6: Seasonal(Monthly) variability in the SIT values using density curves estimates

4 Discussion and Conclusions

Several analyses are yet to be done in order to obtain the desired results leading to a well-defined conclusion.