is a useful technique because it removes the magnitudes and units from the variable, and it allows us to look at its fluctuations only. There are some notable properties of Gaussian distribution.

$$P(-1 < z < +1) = \int_{-1}^{+1} f(x)dx = 0.68 \qquad (1.13)$$

If a variable is normally distributed, approximately two thirds of the population remains within ± 1 standard deviation about the mean. So there is about one in three chance that a random selection can result in outside of ± 1 standard deviation about the mean.

$$P(-2 < z < +2) = \int_{-2}^{+2} f(x)dx = 0.95 \qquad (1.14)$$

If we consider a wider envelope of ± 2 standard deviation about the mean, approximately 95% of population is contained within this range. That is one in twenty chance that a randomly selected variable can be outside of the ± 2 standard deviation.

Central Limit Theorem

Consider random drawing of N samples from a population with the mean of μ and the standard deviation of σ. When you calculate the mean of the N samples, we get the sample mean, M, which is not necessarily the same as the population mean (μ).

However, you can repeat and make many sets of the N-member samples. When you have sufficiently large number of the samples, *the central limit theorem* states that the mean of the sample mean ($\overline{M}$) is equal to the population mean (μ).

Furthermore, the distribution of sample mean (M) follows the Gaussian distribution regardless of the distribution of the parent population. Its standard deviation is $\sigma/\sqrt{N}$, which is also called the standard error. It measures the mean distance between the population mean (μ) and the sample mean (M). As you increase the number of samples in each set, the sample mean better approximate the population mean.

Let's think about rolling the dice as an example of sampling random data. There is equal chance of getting the numbers from 1 to 6, with the probability of 1/6 for each number. Its PDF is a constant, $f(x) = 1/6$, and is obviously NOT a Gaussian distribution. Theoretical mean and standard deviation can then be calculated as follows.

$$\mu = \Sigma_{n=1}^{6} \left(n \times \frac{1}{6} \right) = 3.5 \tag{1.15}$$

$$\sigma^2 = \Sigma_{n=1}^{6} \left\{ (n-\mu)^2 \times \frac{1}{6} \right\} = 1.71^2 \tag{1.16}$$

Next, we simulate the making the N-member samples by generating a random integer for 1,000 times in MATLAB. As an example, we set N=20.

```
N=20;
for i=1:1000
    sample = ceil(6*rand(N,1));
    M(i) = mean(sample);
end
hist(M,[2:.25:5]);
xlabel('sample mean');
ylabel('number of occurrence');
```

This script draws a histogram of the sample mean. As predicted by the central limit theorem, the histogram is approximately centered at μ=3.5, and its standard deviation is approximately $\sigma/\sqrt{20}$=0.38.

In the previous example of the statistical distribution of Atlanta's temperature, you can imagine that the monthly or yearly mean temperature represents the sample means of randomly varying weather events. Because it averages sufficiently large number of weather events and we have approximately 140 years of data, the dataset indeed forms the Gaussian distribution consistent with the central limit theorem.

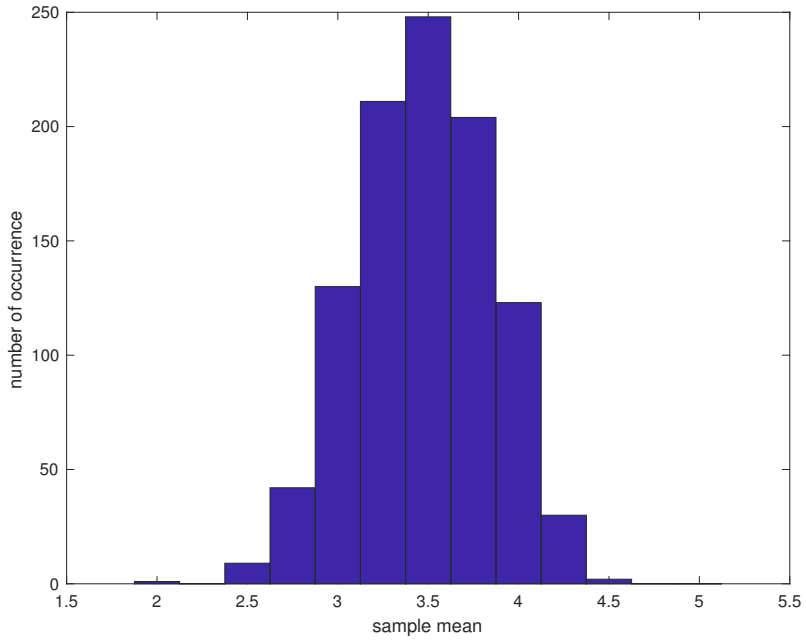

Figure 1.4: Histogram of the simulated sample mean based on rolling dice $N(= 20)$ times. It is simulated 1,000 times using the rand function in MATLAB.

Key points: Central Limit Theorem

1. For large enough sampling, the mean of sample mean is equal to the population mean, $\overline{M} = \mu$.

2. The standard error $\sigma/\sqrt{N}$ measures average distance between sample mean M and population mean μ.

3. For sufficiently large N, the distribution of M forms a Gaussian regardless of the distribution of the parent population.

Box-Whiskar plot

In addition to the histogram (Fig 1.3), box-whiskar plot is a useful way of visualizing the statistical distribution. It visualizes five statistics that characterizes the data. The box-whiskar plot of the Atlanta temperature is shown in Fig 1.5. First, it shows a box, centered at the median, and the two sides are set to the 25 and 75 percentiles. In another words, the width of the box is the IQR. There are two lines (whiskars) extending from the sides of the box, indicating the minimum and maximum values.

The box-whiskar diagram (Fig 1.5) was generated by the "boxplot" command in MATLAB.

```
>> boxplot(T);
>> ylabel('Temperature, deg F');
```

Exercises

1. Download the monthly mean temperature of Atlanta from the course website, `http://shadow.eas.gatech.edu/~Ito/webdata/EAS2655`.

2. Calculate and display the mean, median, standard deviation and IQR of the July temperature.

3. Reproduce Figure 1.1 and 1.3 for the month of July.

4. Climatology refers to the long-term mean values. Display the monthly statistics of Atlanta's temper-

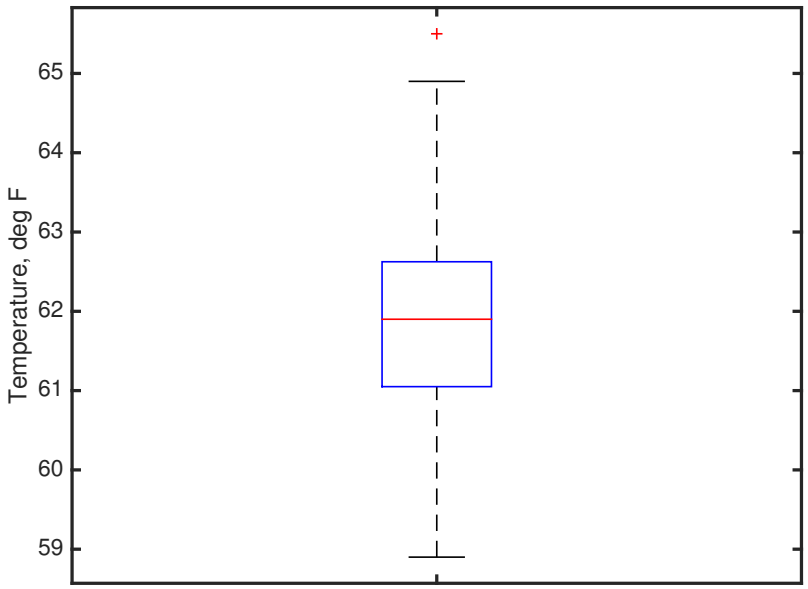

Figure 1.5: Box-Whiskar plot of the annual mean temperature of Atlanta, Georgia, USA. The red cross indicates outlier(s). The box is centered at median value, and its vertical extent is the IQR bounded by 25 and 75 percentile values. Whiskar indicates the maximum and minimum values.

ature by plotting Box-Whiskar diagram for each of the 12 months and plot them together in a single panel using month as the x-axis.

5. **HW1** Publish the MATLAB script that performs all of the activity above, and submit it as a report in the PDF format.

1.2 Data acquisition and visualization

Environmental science is transforming because of the rapidly increasing quantity of data and the computing power to analyze them. It is important to develop computer skills to efficiently process and analyze the data. In this section we briefly review how to read in a few common data formats into MATLAB.

Graphic interface: Import Data

You can download the monthly temperature of Atlanta in the CSV format from the course website, `http://shadow.eas.gatech.edu/~Ito/webdata/EAS2655/Atlanta_Monthly_Mean_Temp.csv`. It is the same data used in the previous exercise, but this time the data is in the CSV (comma separated value) format which is typically exported from a spreadsheet.

You can open this file as a spread sheet, but you can also use MATLAB to read into its workspace. To do so, use "Import Data" graphic interface to read the data into MATLAB workspace (see Figure 1.6 and read the caption). I suggest to save the machine-generated MATLAB script that imports the data. This way, you can read the script to understand how to write a program that handles external file.

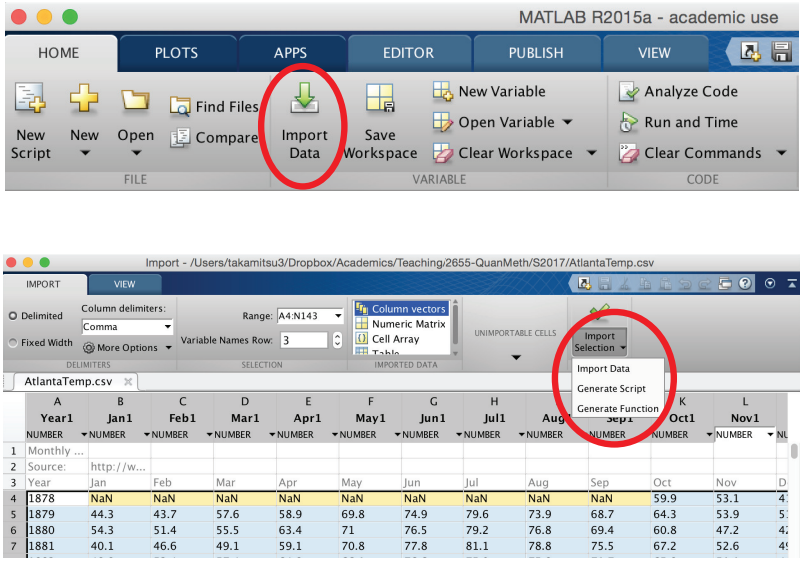

Figure 1.6: Graphic interface to read electronic data into MATLAB workspace. (top) Under the "HOME" tab, choose "Import Data" to open a file. (bottom) Once the file is open, it automatically interprets the data content and suggest data regions and number format. Users are allowed to make changes and have options to directly import data, generate script, or generate functions. I recommend to use "generate script" option so you can read the script.

Meta data

In the simplest form, data can be a bunch of numbers. However, more information is typically needed to make use of the data. Often a dataset includes so-called "meta data", which is the information explaining the data. For example, a meta data should answer; What are the definition and units of the data? How and when is the measurement taken?

Without these information it is hard to make use of the data except for the people who generated it. If you are using the dataset generated by someone else, it is important to read, and understand the meta data before you use the data itself. In the CSV format, often the first several lines of the file includes the meta data.

NetCDF

NetCDF is a self-describing, machine-independent data format, and is commonly used in environmental sciences.

In a single NetCDF data file, there are two parts. The first part is a header including the meta data that describes the names, sizes, units, etc. Then the second part contains the data itself. First we normally read the header only, and find what variable you need to read from the second part.

Let's work with an example. You can download the monthly surface air temperature

(*air.mon.mean.nc*) from the NCEP reanalysis website, `https://www.esrl.noaa.gov/psd/data/gridded/reanalysis`. Once in the website, choose "the monthly and other derived data", and then choose "surface". NCEP reanalysis blends atmospheric observations with the numerical weather model to estimate the state of the atmosphere such as temperature, humidity, and winds.

Generally a NetCDF file contains two parts; a header and contents. The header describes the data itself. What are the dimensions of the data? How many data points are there? What are the units of the data?, etc etc. This is why we call NetCDF a self-describing data. It contains all kinds of information about the data itself, and after that the actual data contents are stored.

To display the header in MATLAB, use the command:

```
>> ncdisp('air.mon.mean.nc');
```

This will produce a list of variables and its name, dimensions, units and all other attributes. You will see that *air* is an array of $144 \times 73 \times 827$ containing the monthly mean air temperature at sigma level 0.995. Furthermore, we see that there are 144 longitude points (lon), 73 latitude points (lat) and 827 time points (time). The values of coordinates and time are also stored in the same NetCDF file. To read in the information, write a script such as:

```
clear all;
X = ncread('air.mon.mean.nc','lon');
Y = ncread('air.mon.mean.nc','lat');
T = ncread('air.mon.mean.nc','time');
TMP = ncread('air.mon.mean.nc','air');
```

Then you have created four arrays for longitude (X), latitude (Y), time (T), and air temperature (TMP). Note that the units for time is hours since 1800-01-01. To convert the time to more familiar value, we have to divide T with $24(hr/day)\times 365.25(day/year)$ and add 1800, and the unit transforms to year.

```
yr = T/24/365.25 + 1800;
```

Plotting using MATLAB

Let's have a quick look at the most recent temperature for the entire grid. Here is an example:

```
pcolor(X,Y,TMP(:,:,end)');
shading flat;
colormap('jet');
colorbar;
xlabel('longitude');
ylabel('latitude');
title('surface air temperature, degree C');
```

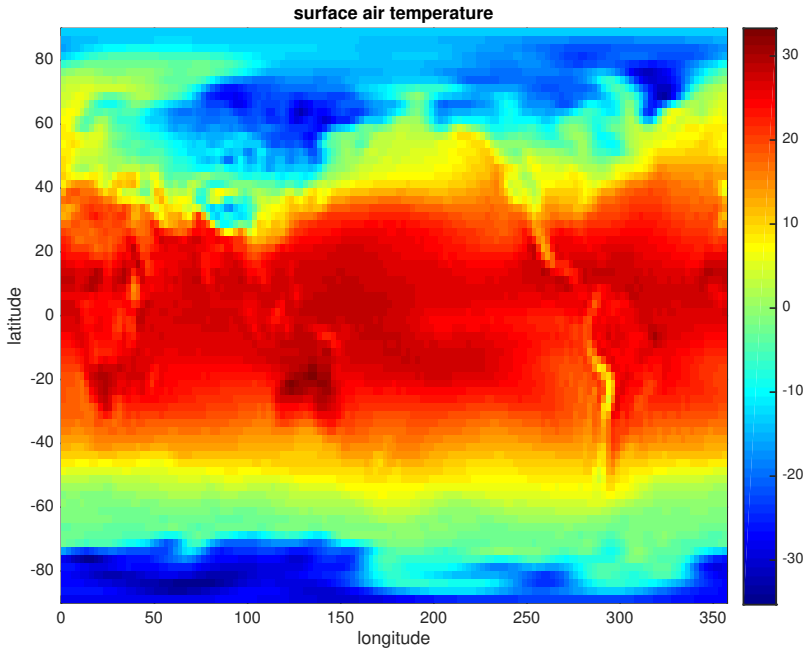

Figure 1.7: Surface air temperature ($^{\circ}C$) for November, 2016, based on NCEP-reanalysis product.

When the data is plotted, always make sure to include units and properly label all axes. *pcolor* is a quick and easy way to plot the data (Figure 1.2). It plots a matrix of colored squares and it gives an impression of the resolution well.

Making it nicer and generating an output file

We can fine tune the details of the figure before printing it out. Here, we use larger font for the axis labels so it is easier to read, and then generate a PDF file named "temp.pdf":

```
ax=get(gcf,'currentaxes');
set(ax,'fontsize',14);
set(ax,'linewidth',2);
set(ax,'layer','top);
grid on;
print -dpdf temp.pdf;
```

If you have access to graphic editing software such as Adobe Illustrator, you can further refine the figure manually. It is important to keep all the text readable when it is presented.

m_map package

Sometimes we wish to plot with different projection. *m_map* package provides a wide range of plotting option. For the first time, we have to download the package from `https://www.eoas.ubc.ca/~rich/map.html`. Read this website and you will see a wide range of examples.

Here is an example for a global projection (Figure 1.8):

```
m_proj('robinson','clongitude',-150);
X(end+1)=X(1)+360;
TMP(end+1,:,:)=TMP(1,:,:);
m_pcolor(X,Y,TMP(:,:,end)');
hold on;
m_pcolor(X-360,Y,TMP(:,:,end)');
hold off;
shading flat;
m_coast;
m_grid('xaxis','middle');
colormap('jet');
colorbar;
print -dpdf temp_v2.pdf;
```

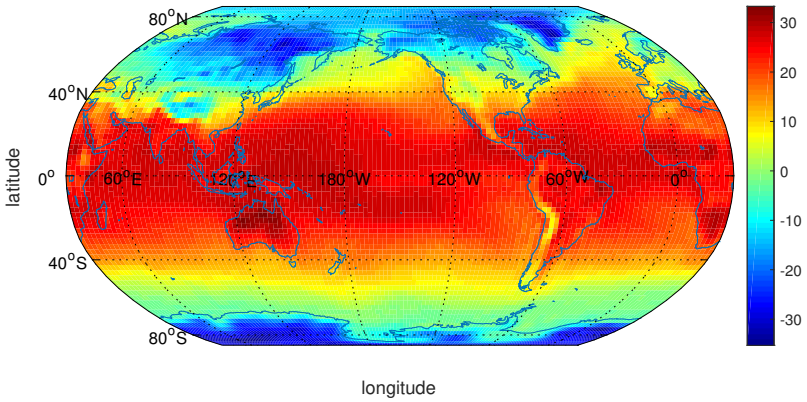

Figure 1.8: Surface air temperature ($^{\circ}C$) for November, 2016, based on NCEP-reanalysis product in the global equal-area projection.

Let's plot the same data over the Antarctica. (Figure 1.9):

```
m_proj('stereographic','longitude',0,'latitude',-90
m_pcolor(X,Y,TMP(:,:,end)');
shading flat;
m_coast;
m_grid('xaxis','middle');
colormap('jet');
colorbar;
print -dpdf temp_v3.pdf;
```

Exercises

1. Download the monthly surface air temperature and the *m_map* package from `https://www.eoas.ubc.ca/~rich/map.html`. Add the *m_map* package to your MATLAB path using *addpath* command. Then you should be able to make plots using this package.

2. Make a global map of median July temperature following the projection of Figure 1.8 using the *m_map* package

3. Same as 2 but plot the median January temperature. Make sure that you use consistent color scale so you can compare (hint: use caxis command).

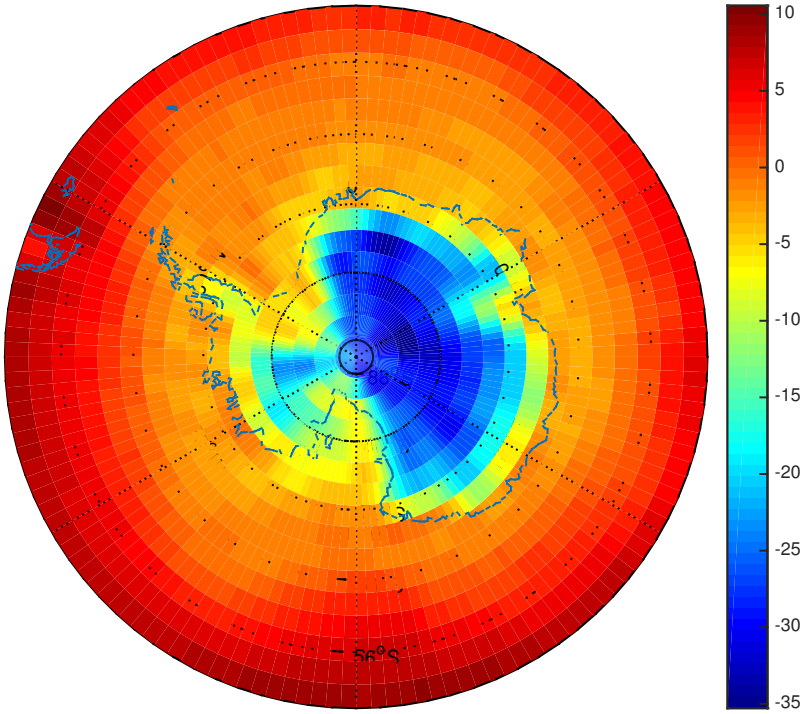

Figure 1.9: Surface air temperature ($^{\circ}C$) for November, 2016, based on NCEP-reanalysis product in the polar projection.

Make sure axis are properly labeled (hint: use xlabel and ylabel commands).

4. Repeat 2 and 3 with different projections. Briefly comment on what you see.