R PROGRAMMING FOR CLIMATE DATA ANALYSIS AND VISUALIZATION

R PROGRAMMING FOR CLIMATE DATA ANALYSIS AND VISUALIZATION

Computing and plotting for NOAA data applications

Samuel S.P. Shen
San Diego State University and Scripps Institution of Oceanography

A JOHN WILEY & SONS, INC., PUBLICATION

CONTENTS

FOREWORD

This book is the instruction manual used for a short-course on R programming for Climate Data Analysis and Visualization first taught at the U.S. National Center for Environmental Information (NCEI), Asheville, North Carolina, 30 May- 2 June 2017. The purpose of the course is to train NCEI scientists and the personnel from the the Cooperative Institute for Climate Science (CICS) -North Carolina to write simple R programs for the climate data managed by the U.S. National Oceanic and Atmospheric Administration (NOAA), so that the NOAA data can be easily accessed, understood, and utilized by the general public, such as school students and teachers. `NOAAGlobalTemp` is the primary dataset used for examples of this book.

R is an open source programming language and software environment, originally designed for statistical computing and graphics first appeared in 1993. In the first 10 years, R was more or less used only in the statistics community, but now, R has become a top 20 most popular computer programming languages in 2017, ranked by Cleveroad, Techworm and others, R and its interface R Studio are free and have become a very popular tool handling big data: to make calculations and to plot. R programs are often shorter due to its sophistication of design and mathematical optimization. R calculation and plotting codes can be incorporated with the readme file of a NOAA dataset. A data user can easily use the R code in the readme file to read the data, change the data format, make some quick calculations, and plot critical figures for his applications. R maps and numerous visualization functions make R programming a convenient tool for not only NOAA data professionals, climate research scientists, and business applicants, but also to teachers and students. Thus, R programming is a convenient tool for climate data's delivery, transparency, accuracy check, and documentation.

This course is divided into six chapters. Chapter 1 describes R basics, such as arithmetic, simple curve plotting, functions, loops, matrix operations, doing statistics, if-else syntax, and logic variables. Chapter 2 is to use R for observed data which are often space-time incomplete due to missing data. NOAAGlobalTemp dataset is used as an example and is analyzed extensively. We show area-weighted spatial average, polynomial fitting, trend calculation with missing data, efficient extraction a subset of the data, data formating, and data writing. Chapter 3 discusses more advanced R graphics, including maps, multiple curves on the same figure, and margin setup and font change for publication. Chapter 4 shows how to handle large datasets in different formats, such as .nc, .bin, .csv, .asc, and .dat. It uses NCEP/NCAR reanalysis' monthly mean temperature data in .nc format as an example to show data reading, data conversion into a standard space-time data matrix, writing the matrix into a .csv file, plotting the temperature maps, calculating empirical orthogonal functions (EOFs) and principal components (PCs) efficiently by the singular mode decomposition (SVD) method, and EOF and PC plotting. Chapters 5 and 6 are on basics of linear algebra and statistics using R. They can be omitted in formal teaching and used as reference materials for the previous four chapters.

The book is intended for a wide range of audience. A high school student with some knowledge of matrices can understand most of its materials. An undergraduate students with two semesters of calculus and one semester of linear algebra can understand the entire book. Some sophisticated R programming tricks and examples are useful to climate scientists, engineers, professors, and graduate students.

Finally, a layman user can simply copy and paste the R codes in this book to produce some desired graphics, as long as he can spend 10 minutes to install R and R Studio following a Youtube instruction.

The book is designed for a one-week course total 20 hours. Half the time is used teaching and demonstration, and another half is for students practice guided by an instructor. Each student is recommended to produce an R code for her/his own work or interest with the instructor's help.

SSPS at San Diego
May 2017

CHAPTER 1

BASICS OF R PROGRAMMING

This chapter shows installation of R and R Studio, and some very basic mathematical operations and plotting by R.

1.1 Download and install R and R-Studio

For Windows users, visit the website
`https://cran.r-project.org/bin/windows/base/`
to find the instructions of R program download and installations.

For Mac users, visit
`https://cran.r-project.org/bin/macosx/`

If you experience difficulties, please refer to online resources, Google or Youtube. A recent 3-minute Youtube instruction for R installation for Windows can be found from the following link:
`https://www.youtube.com/watch?v=Ohnk9hcxf9M`

The same author also has a youtube instruction about R installation for Mac (2 minutes):
`https://www.youtube.com/watch?v=uxuuWXU-7UQ`

When R is installed, one can open R. The `R Console` window will appear. See Fig. 1.1. One can use `R Console` to perform calculations, such as typing 2+3 and hitting

return. However, most people today prefer using RStudio as the interface. To install RStudio, visit
`https://www.rstudio.com/products/rstudio/download/`
This site allows to choose Windows, or Mac OS, or Unix.

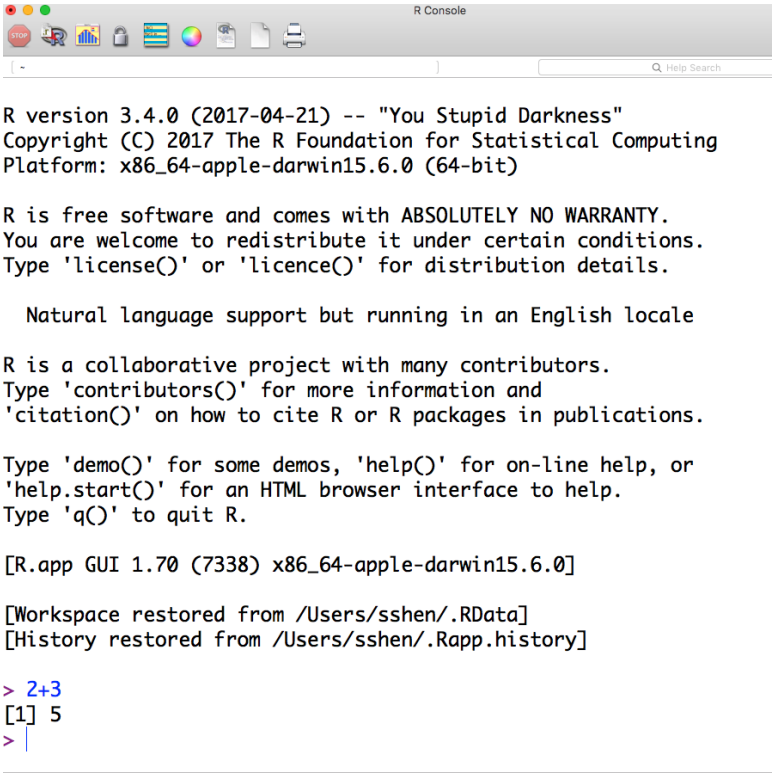

Figure 1.1 R Console window after opening R.

After both R and RStudio are installed, one can use either R or RStudio, or both, depending on his interest. However, RStudio will not work without R. Thus, always install R first.

When opening RStudio, four windows will appear as shown in Fig. 1.2: The top left window is called R script, for writing the R code. The green arrow on top of the window can be clicked to run the code. Each run is shown in the lower left R Console window, and recorded on the upper right R History window. When plotting, the figure will appear in the lower right R Plots window. For example, `plot(x,x*x)` renders the eight points in the Plots window, because `x=1:8` defines a sequence of numbers from 1 to 8. `x*x` yields a sequence from 1^2 to 8^2.

1.2 R Tutorial

There are many excellent tutorials for a quick learn of R programming, using a few hours or a few evenings, are available online and in Youtube, such as

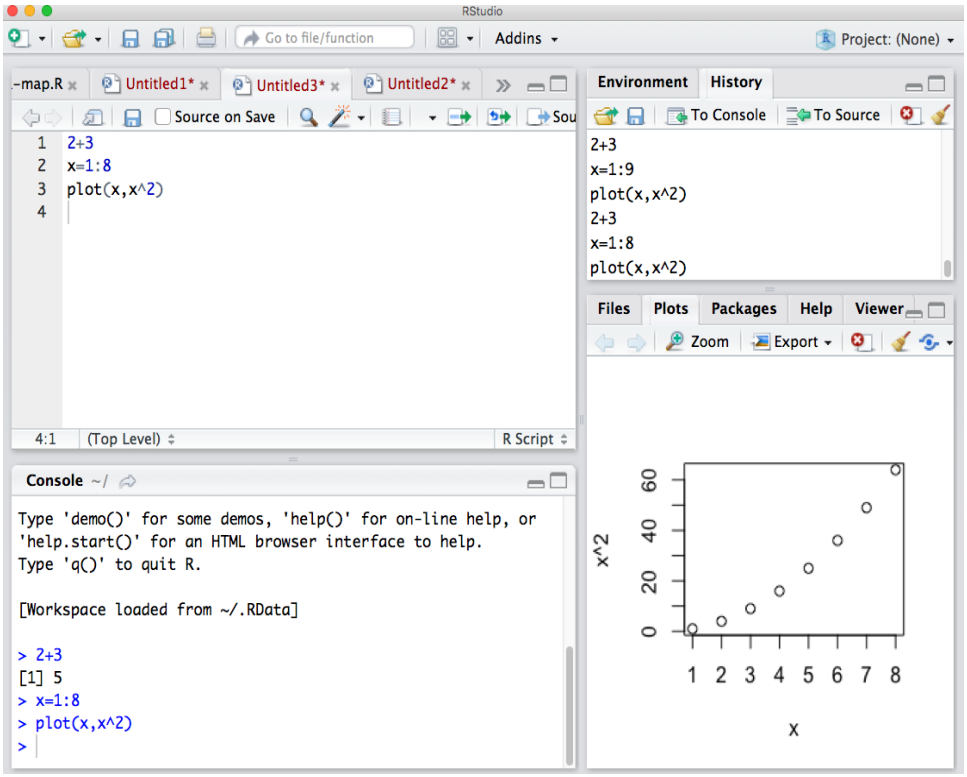

Figure 1.2 R Studio windows.

`http://ww2.coastal.edu/kingw/statistics/R-tutorials/`.
You can google around and find your preferred tutorials.

It is very hard for the beginners of R to navigate through the official, formal, detailed, and massive R-Project documentation:
`https://www.r-project.org/`

1.2.1 R as a smart calculator

R can be used like a smart calculator that allows fancier calculations than those done on regular calculators.

```
1+4
[1] 5
2+pi/4-0.8
[1] 1.985398
x<-1
y<-2
z<-4
t<-2*x^y-z
t
[1] -2
```

```
u=2         # "=" sign and "<-" is almost equivalent
v=3         # The text behind the "#" sign is comments
u+v
[1] 5
sin(u*v)     # u*v = 6 is considered radian
[1] -0.2794155
```

R programming uses assignment operator a¡- b+ to assign b to a. Often the equal operator a=b can do the same job or vice versa. The two operators are equivalent in general. However, certain R formulas have specific meanings for = and cannot be replaced by <−+. Most veteran R users use <−+ for assignment and = for defined R formulas.

1.2.2 Define a sequence in R

Directly enter a sequence of daily maximum temperature data at San Diego International Airport during 1-7 May 2017 [unit: $^{\circ}F$].
`tmax <- c(77, 72, 75, 73,66,64,59)`
The data are from the United States Historical Climatology Network (USHCN)
`www.ncdc.noaa.gov/cdo-web/quickdata`
The command `c( )` is used to hold a data sequence and is named `tmax`. Entering the `tmax` command will render temperature data sequence:

```
tmax
[1]  4.5  4.1 -2.1  3.4  2.5  6.0  4.3
```

You can generate different sequences using R, e.g.,
`1: 8 #Generates a sequence 1,2,...,8`
Here the pound sign # begins R comments which are not executed by R calculations. The same sequence can be generated by different commands, such as

```
seq(1,8)
seq(8)
seq(1,8, by=1)
seq(1,8, length=8)
seq(1,8, length.out =8)
```

The most useful sequence commands are `seq(1,8, by=1)` and `seq(1,8, length=8)` or `seq(1,8, len=8)`. The former is determined by a begin value, end value, and step size, and the latter by a begin value, end value, and number of values in the sequence. For example, `seq(1951,2016, len=66*12)` renders a sequence of all the months from January 1951 to December 2016.

1.2.3 Define a function in R

The function command is
`name <- function(var1, var2, ...) expression of the function.`
For example,

```
samfctn <- function(x) x*x
samfctn(4)
[1] 16
```

```
fctn2 <- function(x,y,z) x+y-z/2
fctn2(1,2,3)
[1] 1.5
```

1.2.4 Plot with R

R can can plot all kinds of curves, surfaces, statistical plots, and maps. Below are a few very simple examples for R beginners. For adding labels, ticks, color, and other features to a plot, you learn them from later parts of the book and can also google R plot to find the commands for the proper inclusion of the desired features.

R plotting is based on the coordinate data. The following command plots the seven days of San Diego Tmax data above:

```
plot(1:7, tmax)
```

The result figure is shown in Fig. 1.3.

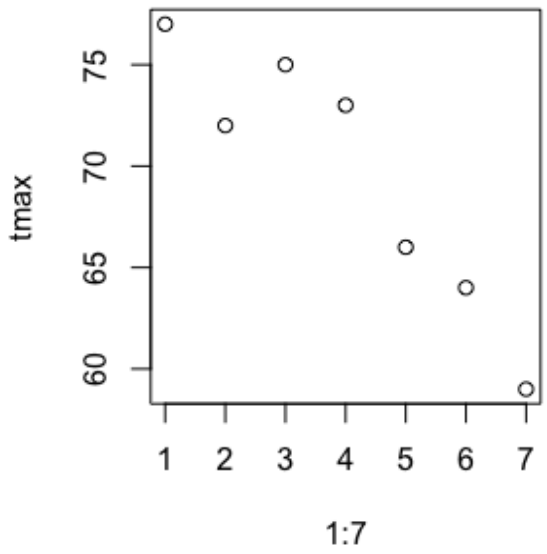

Figure 1.3 The daily maximum temperature during 1-7 May 2017 of the San Diego International Airport.

```
plot(sin, -pi, 2*pi)   #plot the curve of y=sin(x) from -pi to 2 pi

square <- function(x) x*x   #Define a function
plot(square, -3,2)   # Plot the defined function

 # Plot a 3D surface
x <- seq(-1, 1, length=100)
y <- seq(-1, 1, length=100)
z <- outer(x, y, function(x, y)(1-x^2-y^2))
#outer (x,y, function) renders z function on the x, y grid
persp(x,y,z, theta=330)
```

```
# yields a 3D surface with perspective angle 330 deg

#Contour plot
contour(x,y,z) #lined contours
filled.contour(x,y,z) #color map of contours
```

The color map of contours resulted from the last command is shown in Fig. 1.4.

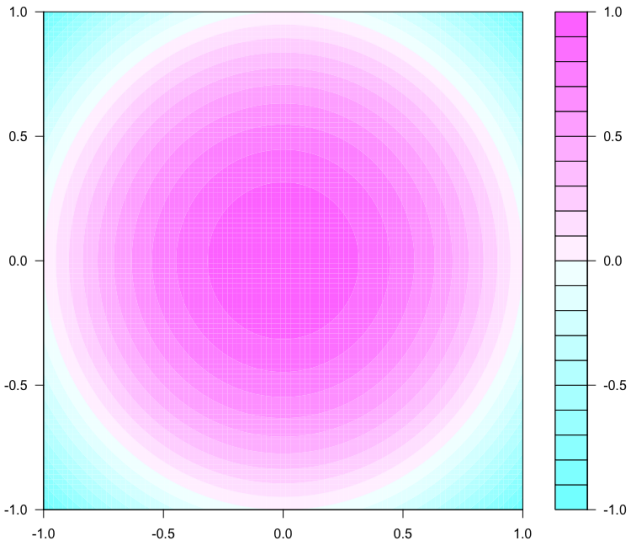

Figure 1.4 The color map of contours for the function $z = 1 - x^2 - y^2$.

1.2.5 Symbolic calculations by R

People used to think that R can only handle numbers. Actually R can also do symbolic calculations, such as finding a derivative, although, up to now R is not the best symbolic calculation tool. One can use WolframAlpha, SymPy, and Yacas for free symbolic calculations or use the paid software package Maple or Mathematica. Google symbolic calculation for calculus to find a long list of symbolic calculation software packages, e.g., https://en.wikipedia.org/wiki/List_of_computer_algebra_systems.

```
D(expression(x^2,'x'), 'x')
# Take derivative of x^2 w.r.t. x
2 * x #The answer is 2x

fx= expression(x^2,'x')  #assign a function
D(fx,'x') #differentiate the function w.r.t. x
2 * x  #The answer is 2x

fx= expression(x^2*sin(x),'x')
#Change the expression and use the same derivative command
```

```
D(fx,'x')
2 * x * sin(x) + x^2 * cos(x)

 fxy = expression(x^2+y^2, 'x','y')
#One can define a function of 2 or more variables
 fxy #renders an expression of the function in terms of x and y
#expression(x^2 + y^2, "x", "y")
D(fxy,'x') #yields the partial derivative with respect to x: 2 * x
D(fxy,'y') #yields the partial derivative with respect to y: 2 * y

square = function(x) x^2
integrate (square, 0,1)
#Integrate x^2 from 0 to 1 equals to 1/3 with details below
#0.3333333 with absolute error < 3.7e-15

integrate(cos,0,pi/2)
#Integrate cos(x) from 0 to pi/2 equals to 1 with details below
#1 with absolute error < 1.1e-14
```

The above two integration examples are for definite integral. It seems that no efficient R packages are available for finding antiderivatives, or indefinite integrals.

1.2.6 Vectors and matrices

R can handle all kinds of operations vectors and matrices.

```
c(1,6,3,pi,-3) #c() gives a vector and is considered a 4X1 column vector
#[1]  1.000000  6.000000  3.000000  3.141593 -3.000000
seq(2,6) #Generate a sequence from 2 to 6
#[1] 2 3 4 5 6
seq(1,10,2) # Generate a sequence from 1 to 10 with 2 increment
#[1] 1 3 5 7 9
x=c(1,-1,1,-1)
x+1 #1 is added to each element of x
#[1] 2 0 2 0
2*x #2 multiplies each element of x
#[1]  2 -2  2 -2
x/2 # Each element of x is divided by 2
#[1]  0.5 -0.5  0.5 -0.5
y=seq(1,4)
x*y  # This multiplication * multiples each pair of elements
#[1]  1 -2  3 -4
x%*%y #This is the dot product of two vectors and yields
#     [,1]
#[1,]   -2
t(x)  # Transforms x into a row 1X4 vector
#     [,1] [,2] [,3] [,4]
#[1,]    1   -1    1   -1
t(x)%*%y #This is equivalent to dot product and forms 1X1 matrix
```

```
#     [,1]
#[1,]   -2
x%*%t(y) #This column times row yields a 4X4 matrix
#     [,1] [,2] [,3] [,4]
#[1,]    1    2    3    4
#[2,]   -1   -2   -3   -4
#[3,]    1    2    3    4
#[4,]   -1   -2   -3   -4
my=matrix(y,ncol=2)
#Convert a vector into a matrix of the same number of elements
#The matrix elements go by column, first column, second, etc
#Commands matrix(y,ncol=2, nrow=2)  or matrix(y,2)
#or matrix(y,2,2) does the same job
my
#     [,1] [,2]
#[1,]    1    3
#[2,]    2    4
dim(my)  #find dimensions of a matrix
#[1] 2 2
as.vector(my) #Convert a matrix to a vector, again via columns
#[1] 1 2 3 4
mx <- matrix(c(1,1,-1,-1), byrow=TRUE,nrow=2)
mx*my #multiplication between each pair of elements
#     [,1] [,2]
#[1,]    1    3
#[2,]   -2   -4
mx/my #division between each pair of elements
#     [,1]       [,2]
#[1,]  1.0  0.3333333
#[2,] -0.5 -0.2500000
mx-2*my
#     [,1] [,2]
#[1,]   -1   -5
#[2,]   -5   -9
mx%*%my #This is the real matrix multiplication in matrix theory
#     [,1] [,2]
#[1,]    3    7
#[2,]   -3   -7
det(my) #determinant
#[1] -2
myinv = solve(my) #yields the inverse of a matrix
myinv
#     [,1] [,2]
#[1,]   -2  1.5
#[2,]    1 -0.5
myinv%*%my #verifies the inverse of a matrix
#     [,1] [,2]
#[1,]    1    0
#[2,]    0    1
```

```
diag(my) #yields the diagonal vector of a matrix
#[1] 1 4
myeig=eigen(my) #yields eigenvalues and unit eigenvectors
myeig
myeig$values
#[1]  5.3722813 -0.3722813
myeig$vectors
#           [,1]       [,2]
#[1,] -0.5657675 -0.9093767
#[2,] -0.8245648  0.4159736
mysvd = svd(my) #SVD decomposition of a matrix M=UDV'
#SVD can be done for a rectangular matrix of mXn
mysvd$d
#[1] 5.4649857 0.3659662
mysvd$u
#           [,1]       [,2]
#[1,] -0.5760484 -0.8174156
#[2,] -0.8174156  0.5760484
mysvd$v
#           [,1]       [,2]
#[1,] -0.4045536  0.9145143
#[2,] -0.9145143 -0.4045536

ysol=solve(my,c(1,3))
#solve linear equations matrix %*% x = b
ysol  #solve(matrix, b)
#[1]  2.5 -0.5
my%*%ysol #verifies the solution
#     [,1]
#[1,]    1
#[2,]    3
```

1.2.7 Simple statistics by R

R was originally designed to do statistical calculations. Thus, R has a comprehensive set of statistics functions and software packages. This sub-section gives a few basic commands. More will be described in the statistics chapter of this book.

```
x=rnorm(10) #generate 10 normally distributed numbers
x
 #[1]  2.8322260 -1.2187118  0.4690320 -0.2112469  0.1870511
 #[6]  0.2275427 -1.2619005  0.2855896  1.7492474 -0.1640900
 mean(x)
#[1] 0.289474
var(x)
#[1] 1.531215
sd(x)
#[1] 1.237423
median(x)
```

```
#[1] 0.2072969
quantile(x)
#        0%        25%        50%        75%       100%
#-1.2619005 -0.1994577  0.2072969  0.4231714  2.8322260
range(x) #yields the min and max of x
#[1] -1.261900  2.832226
 max(x)
#[1] 2.832226

boxplot(x) #yields the box plot of x
w=rnorm(1000)

summary(rnorm(12)) #statistical summary of the data sequence
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#-1.9250 -0.6068  0.3366  0.2309  1.1840  2.5750

hist(w)
#yields the histogram of 1000 random numbers with a normal distribution
```

1.3 Online Tutorials

1.3.1 Youtube tutorial: for true beginners

This is a very good and slow paced 22 minutes youtube tutorial: Chapter 1. An Introduction to R
https://www.youtube.com/watch?v=suVFuGET-0U

1.3.2 Youtube tutorial: for some basic statistical summaries

This is a 9 minutes tutorial by Layth Alwan.
https://www.youtube.com/watch?v=XjOZQN-Nre4

1.3.3 Youtube tutorial: Input data by reading a csv file into R

An excel file can be saved as csv file: xxxx.csv. This 15 minutes youtube video shows how to read a csv file into R by Layth Alwan. He also shows linear regression.
https://www.youtube.com/watch?v=QkE8cp0B9gg

R can input all kinds of data files, including xlsx, netCDF, fortran data, and sas data. Some commands are below. One can google to find proper data reading command for your particular data format.

```
mydata <- read.csv("mydata.csv")
# read csv file named "mydata.csv"

mydata <- read.table("mydata.txt")
# read text file named "my data.txt"

library(gdata)                  # load gdata package
```

```
mydata = read.xls("mydata.xls")  # read an excel file

library(foreign)                 # load the foreign package
mydata = read.mtp("mydata.mtp")  # read from .mtp file

library(foreign)                 # load the foreign package
mydata = read.spss("myfile", to.data.frame=TRUE)

ff <- tempfile()
cat(file = ff, "123456", "987654", sep = "\n")
read.fortran(ff, c("F2.1","F2.0","I2")) #read a fotran file

library(ncdf)
ncin <- open.ncdf(ncfname)  # open a NetCDF file
lon <- get.var.ncdf(ncin, "lon") #read a netCDF file into R
```

Many more details of reading and reformatting of .nc file will be discussed later when dealing with NCEP/NCAR Reanalysis data.

Some libraries are not in the R project anymore. For example,

```
library(ncdf) #The following error message pops up
Error in library(ncdf) : there is no package called ncdf
```

One can then google r data reading netcdf R-project and go to the R-project website. The following can be found.

```
Package ncdf was removed from the CRAN repository.
Formerly available versions can be obtained from the archive.
Archived on 2016-01-11: use 'RNetCDF' or 'ncdf4' instead.
```

This means that one should use RNetCDF, which can be downloaded from internet. Thus, if a library gives an error message, then google the library package, download and install the package, and finally read the data with a specified format.

CHAPTER 2

R ANALYSIS OF INCOMPLETE DATA: NOAAGLOBALTEMP

2.1 The missing data problem

Unlike the climate model data which are space-time complete, the observed data are often space-time incomplete, i.e., some space-time grid points or boxes do not have data. We often call this the missing data problem.

Missing data problems can be many kinds and very complicated. Here we use the NOAAGlobalTemp dataset to illustrate a few methods often used in analyzing the datasets with missing data. NOAAGlobalTemp is the merged land and oceanic observed surface temperature anomalies with respect to the 1970-2000 base period climatology, produced by the United States National Center of Environmental Information in 2015.
`https://www.ncdc.noaa.gov/data-access/marineocean-data/noaa-global-surface-temperature-noaaglobaltemp`
It is a monthly data from January 1880 to present with $5^\circ \times 5^\circ$ latitude-longitude spatial resolution. The earlier years had many missing data while the recent years are better covered. Figure 2.1 shows the history of the percentage of area covered by the data. One hundred minus the percentage is the percentage of missing data. The minimum coverage is nearly 60%, much of which is due to the good coverage due to reconstruction, i.e., NOAA ERSST (extended reconstructed sea surface temperature). The actual raw observations have much less coverage of the Earth surface.

Using software 4DVD (4-dimensional visual delivery of big climate data) developed at San Diego State University, one can see where and when data are missing. Figure 2.2 shows the NOAAGlobalTemp data distribution over the globe for January 1917. The data

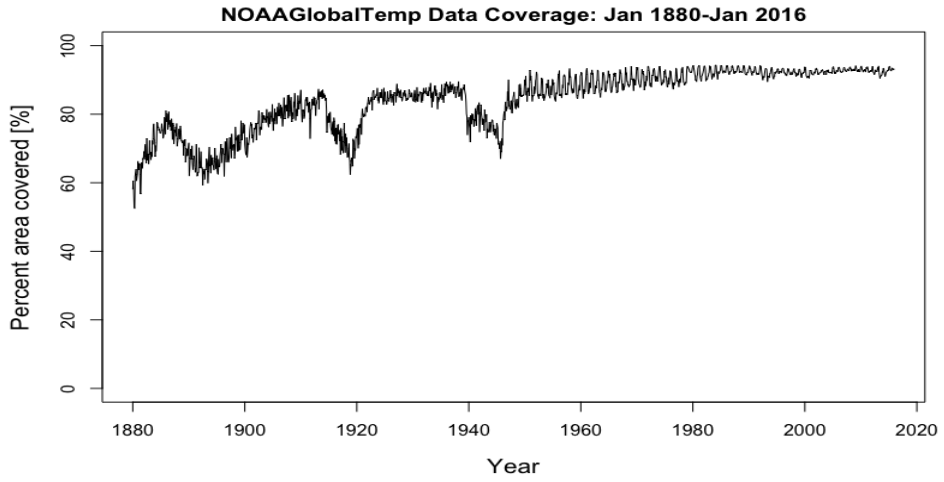

Figure 2.1 Percentage of the global surface area covered by the NOAAGlobalTemp dataset.

cover 72% of the global area. The black region has 28% of the global area and has missing data. The data void regions are the polar areas which could not be accessed at that time, the central tropical Pacific which were not on the tracks of commercial ships, central Asia, part of Africa, and Amazon. Figure 2.3 shows that the grid box (12.5S, 117.5W) in the Amazon region did not begin its data until 1918, and the data time series is discontinuous with missing data around 1921 and 1922.

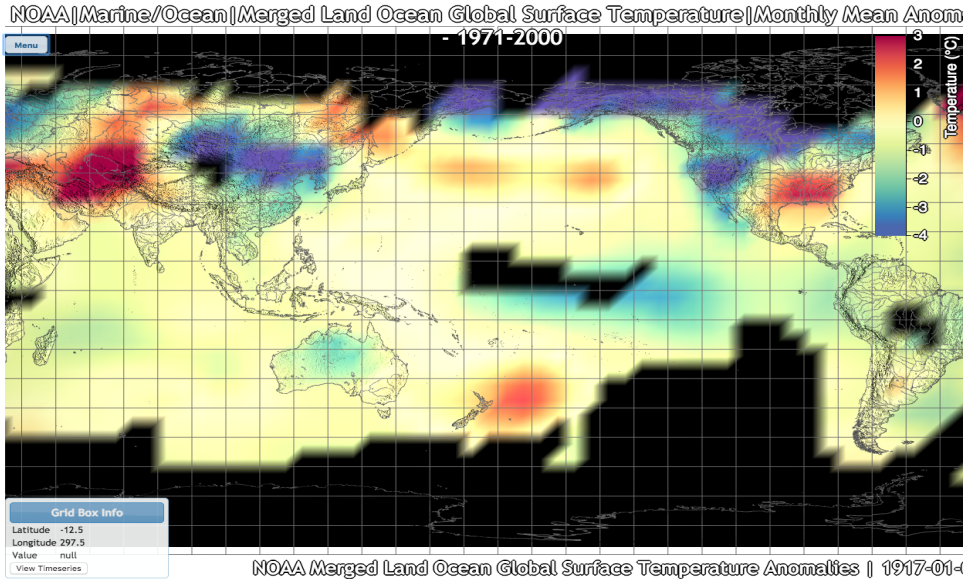

Figure 2.2 The January 1917 distribution of the NOAAGlobalTemp data. The black region means missing data.

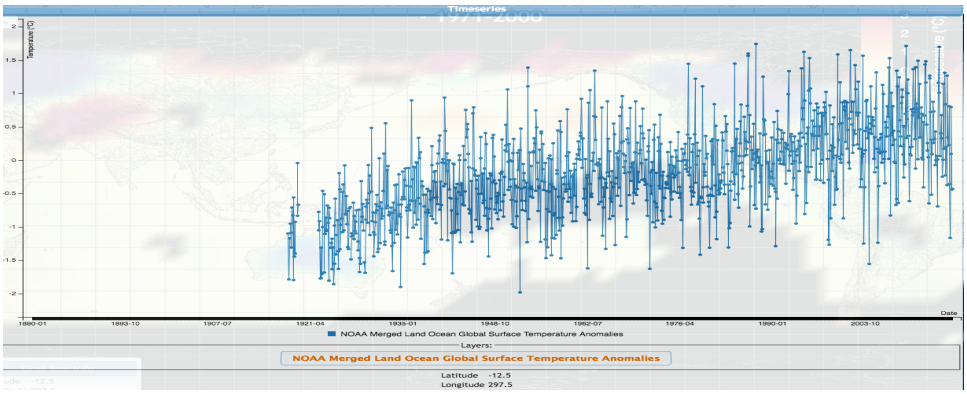

Figure 2.3 Time series of the monthly temperature anomalies for a grid box over the Amazon region.

2.2 Read NOAAGlobalTemp and form the space-time data matrix

This section describes how to use R to read the data and convert the data into a standard space-time matrix for various of kinds of analyses and visualizations.

2.2.1 Read the downloaded data

First, we download the NOAAGlobalTemp gridded data from its ftp site
`ftp://ftp.ncdc.noaa.gov/pub/data/noaaglobaltemp/operational`
The anomalies are with respect to the 1971-2000 climatology.

The ftp site has two data formats: asc and bin. We use asc as example to describe the R analysis. The following R codes read the asc data and makes the conversion.

```
rm(list=ls(all=TRUE))
# Download .asc file
setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2016/Rcodes/NOAAGlobalTemp")
da1=scan("NOAAGlobalTemp.gridded.v4.0.1.201701.asc")
length(da1)
#[1] 4267130
da1[1:3]
#[1]    1.0 1880.0 -999.9 #means mon, year, temp
#data in 72 rows (2.5, ..., 357.5) and
#data in 36 columns (-87.5, ..., 87.5)
tm1=seq(1,4267129, by=2594)
tm2=seq(2,4267130, by=2594)
length(tm1)
length(tm2)
mm1=da1[tm1] #Extract months
yy1=da1[tm2] #Extract years
head(mm1)
head(yy1)
length(mm1)
length(yy1)
```

```
rw1<-paste(yy1, sep="-", mm1) #Combine YYYY with MM
head(tm1)
head(tm2)
tm3=cbind(tm1,tm2)
tm4=as.vector(t(tm3))
head(tm4)
#[1]    1    2 2595 2596 5189 5190
da2<-da1[-tm4] #Remote the months and years data from the scanned data
length(da2)/(36*72)
#[1] 1645 #months, 137 yrs 1 mon: Jan 1880-Jan 2017
da3<-matrix(da2,ncol=1645) #Generate the space-time data
#2592 (=36*72) rows and 1645 months (=137 yrs 1 mon)
```

To facilitate the use of space-time data, we add the latitude and longitude coordinates for each grid box as the first two columns, and the time mark for each month as the first row. This can be done by the following R code.

```
colnames(da3)<-rw1
lat1=seq(-87.5, 87.5, length=36)
lon1=seq(2.5, 357.5,  length=72)
LAT=rep(lat1, each=72)
LON=rep(lon1,36)
gpcpst=cbind(LAT, LON, da3)
head(gpcpst)
dim(gpcpst)
#[1] 2592 1647 #The first two columns are Lat and Lon
#-87.5 to 87.5 and then 2.5 to 375.5
#The first row for time is header, not counted as data.
write.csv(gpcpst,file="NOAAGlobalT.csv")
#Output the data as a csv file
```

2.2.2 Plot the temperature data map of a given month

With this space-time data, one can plot a data map for a given month or a data time series for a given location. For example, the following R code plots the temperature data map for December 2015, an El Nino month (See Fig. 2.4).

```
#Install maps package if not done before
#install.packages("maps")
library(maps)
Lat= seq(-87.5, 87.5, length=36)
Lon=seq(2.5, 357.5, length=72)
mapmat=matrix(gpcpst[,1634],nrow=72)
#column 1634 corresponding to Dec 2015
#Covert the vector into a lon-lat matrix for R map plotting
mapmat=pmax(pmin(mapmat,6),-6)
#This command compresses numbers larger than 6 to 6
plot.new()
par(mar=c(4,5,3,0))
int=seq(-6,6,length.out=81)
```

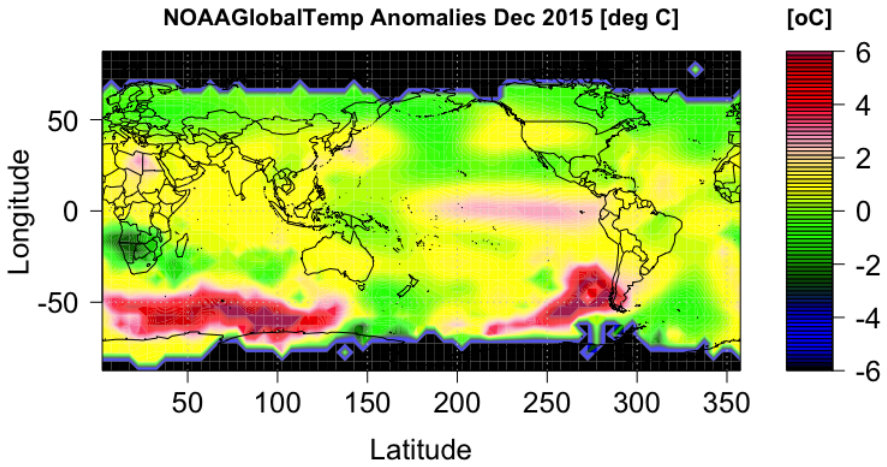

Figure 2.4 Monthly mean temperature anomalies of December 2015 based on the NOAAGlobalTemp data.

```
rgb.palette=colorRampPalette(c('black','blue', 'darkgreen','green',
'yellow','pink','red','maroon'),interpolate='spline')
mapmat= mapmat[,seq(length(mapmat[1,]),1)]
filled.contour(Lon, Lat, mapmat, color.palette=rgb.palette, levels=int,
plot.title=title(main="NOAAGlobalTemp Anomalies Dec 2015 [deg C]",
                                xlab="Latitude",ylab="Longitude", cex.lab=1.5),
               plot.axes={axis(1, cex.axis=1.5);
               axis(2, cex.axis=1.5);map('world2', add=TRUE);grid()},
               key.title=title(main="[oC]"),
               key.axes={axis(4, cex.axis=1.5)})
```

Figure 2.4 should appear in the Plots window. One can click `Export` button in the Plots window to save the figure in different formats, including png, jpg, tiff and eps, and different aspect ratios.

2.2.3 Extract the data for a specified region

If one wishes to study the data over a particular region, say, the tropical Pacific for El Nino characteristics, he can extract the data for the region for a given time interval. The following code extract the space-time data for the tropical Pacific region (20S-20N, 160E-120W) from 1951-2000.

```
#Keep only the data for the Pacific region
n2<-which(gpcpst[,1]>-20&gpcpst[,1]<20&gpcpst[,2]>160&gpcpst[,2]<260)
dim(gpcpst)
length(n2)
#[1] 160 #4 latitude bends and 20 longitude bends
pacificdat=gpcpst[n2,855:1454]  #from 1951-2000
#(1951-1880)*12 + lat col + lon col =854
#Thus, Jan 1951 data from column 855
```

Here, we have used a powerful and convenient `which` search command. This very useful command is easier to program and faster than `if` conditions.

Despite the good coverage of ERSST, it still has a few missing data in this tropical Pacific area. Because the missing data are assigned -999.00, they can significantly impact the computing results, such as SVD, when they are used in computing. We assign the missing data to zero, instead of -999.00. The following code plots the December 1997 temperature data for the tropical Pacific region (20S-20N, 160E-120W) (see Fig. 2.5).

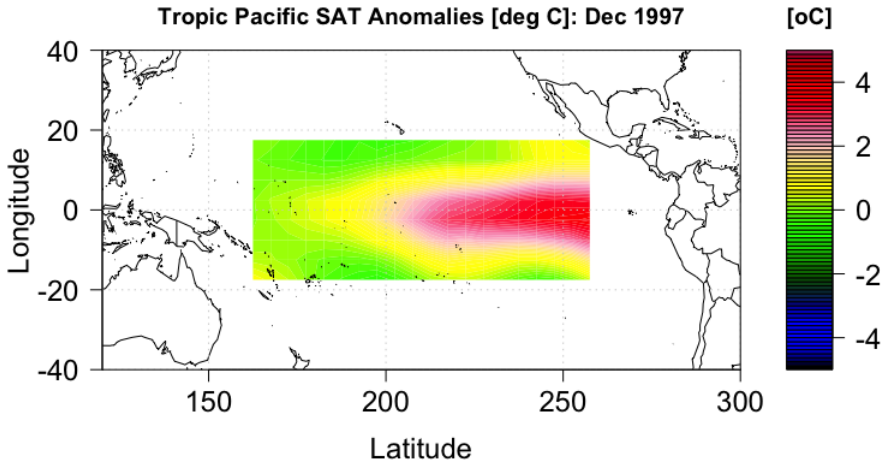

Figure 2.5 Tropical Pacific SST anomalies of December 1997 based on the NOAAGlobalTemp data.

```
Lat=seq(-17.5,17.5, by=5)
Lon=seq(162.5, 257.5, by=5)
plot.new()
par(mar=c(4,5,3,0))
mapmat=matrix(pacificdat[,564], nrow=20)
int=seq(-5,5,length.out=81)
rgb.palette=colorRampPalette(c('black','blue', 'darkgreen',
'green', 'yellow','pink','red','maroon'),interpolate='spline')
#mapmat= mapmat[,seq(length(mapmat[1,]),1)]
filled.contour(Lon, Lat, mapmat, color.palette=rgb.palette, levels=int,
               xlim=c(120,300),ylim=c(-40,40),
plot.title=title(main="Tropic Pacific SAT Anomalies [deg C]: Dec 1997",
                                xlab="Latitude",ylab="Longitude", cex.lab=1.5),
               plot.axes={axis(1, cex.axis=1.5); axis(2, cex.axis=1.5);
               map('world2', add=TRUE);grid()},
               key.title=title(main="[oC]"),
               key.axes={axis(4, cex.axis=1.5)})
```

2.2.4 Extract data from only one grid box

A special case is to extract data for a specified grid box with given latitude and longitude, e.g., the San Diego box (32.5N, 117.5W) or (+32.5, 242.5). This can be easily done by the following R code with a simple plotting command.

```
#Extract data for a specified box with given lat and lon
n2 <- which(gpcpst[,1]==32.5&gpcpst[,2]==242.5)
SanDiegoData <- gpcpst[n2,855:1454]
plot(seq(1880,2017, len=length(SanDiegoData)),
     SanDiegoData, type="l",
     xlab="Year", ylab="Temp [oC]",
     main="San Diego temperature anomalies history")
```

2.3 Spatial averages and their trends

2.3.1 Compute and plot the global area-weighted average of monthly data

The area-weighted average, also called spatial average, of a temperature field $T(\phi, \theta, t)$ on a sphere is mathematically defined as follows

$$\bar{T}(t) = \frac{1}{4\pi} \iint T(\phi, \theta, t) \cos(\phi) d\phi d\theta, \tag{2.1}$$

where ϕ is latitude and θ is longitude, and t is time. The above formula's discrete form for a grid of resolution $\Delta\phi \times \Delta\theta$ is

$$\hat{\bar{T}}(t) = \sum_{i,j} T(i, j, t) \frac{\cos(\phi_{ij}) \Delta\phi \Delta\theta}{4\pi}, \tag{2.2}$$

where (i, j) are coordinate indices for the grid box (i,j), and $\Delta\phi$ and $\Delta\theta$ are latitude and longitude resolution in radians. If it is a 5° resolution, then $\Delta\phi = \Delta\theta = (5/180)\pi$.

If NOAAGlobalTemp had data in every box, then the global average would be easy to calculate according to the above formula:

$$\hat{\bar{T}}(t) = \sum_{i,j} T(i, j, t) \frac{\cos(\phi_{ij}) (5/180)^2}{4}. \tag{2.3}$$

However, NOAAGlobalTemp has missing data. We thus should not average the data-void regions. An approach is to consider the spatial average problem as a weighted average, which assigns a data box with weight proportional to $\cos \phi_{ij}$ and a data-void box with zero weight. We thus generate a weight matrix `areaw` corresponding to the data matrix `temp` by the following R code.

```
#36-by-72 boxes and Jan1880-Jan2017=1645 months + lat and lon
temp=gpcpst
areaw=matrix(0,nrow=2592,ncol = 1647)
dim(areaw)
#[1] 2592 1647
areaw[,1]=temp[,1]
```

```
areaw[,2]=temp[,2]
veca=cos(temp[,1]*pi/180)
#Here, don't forget to convert degrees to radians
#Area-weight matrix equal to cosine lat of the boxes with data
#and to zero for the boxes of missing data -999.9
for(j in 3:1647)
{
for (i in 1:2592)
  {if(temp[i,j]> -290.0) {areaw[i,j]=veca[i]} }
}
```

Then compute an area-weighted temperature data matrix and its average:

```
#area-weight data matrixs first two columns as lat-lon
tempw=areaw*temp
tempw[,1:2]=temp[,1:2]
#create monthly global average vector for 1645 months
#Jan 1880- Jan 2017
avev=colSums(tempw[,3:1647])/colSums(areaw[,3:1647])
```

Figure 2.6 shows the spatial average of the monthly temperature data from NOAAGlobalTemp from January 1880 -January 2017 and can be generated by the following R code.

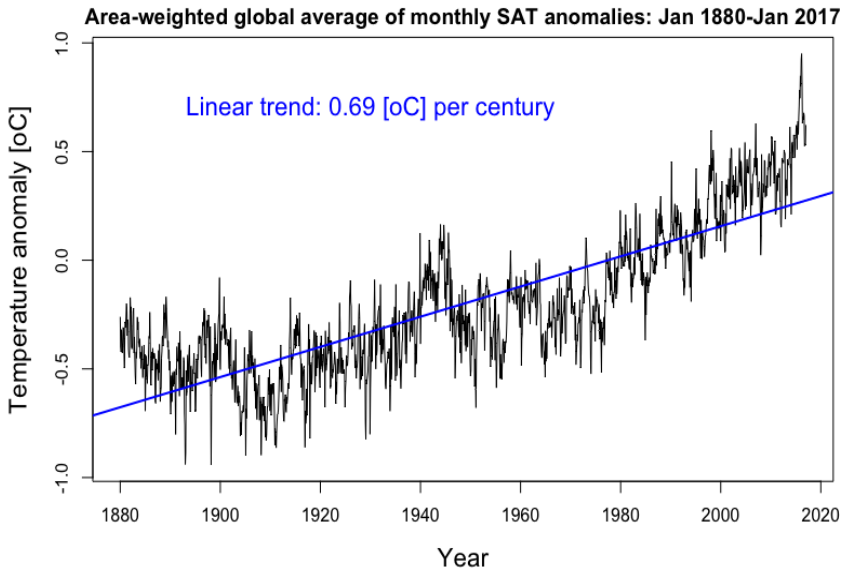

Figure 2.6 Spatial average of monthly temperature anomalies with respect to 1971-2000 climatology based on the NOAAGlobalTemp data.

```
timemo=seq(1880,2017,length=1645)
plot(timemo,avev,type="l", cex.lab=1.4,
     xlab="Year", ylab="Temperature anomaly [oC]",
main="Area-weighted global average of
```

```
monthly SAT anomalies: Jan 1880-Jan 2017")
abline(lm(avev ~ timemo),col="blue",lwd=2)
text(1930,0.7, "Linear trend: 0.69 [oC] per century",
     cex=1.4, col="blue")
```

2.3.2 Percent coverage of the NOAAGlobalTemp

As a byproduct of the above weighted average, the matrix *areaw* can be used to calculate the percentage of area covered by the data.

```
rcover=100*colSums(areaw[,3:1647])/sum(veca)
```

The following R code can plot this time series against time, which is the percentage of data covered area with respect to the entire sphere, shown in Fig. 2.1 at the beginning of this chapter.

```
#Plot this time series
motime=seq(1880, 2017, length=1645)
plot(motime,rcover,type="l",ylim=c(0,100),
     main="NOAAGlobalTemp Data Coverage: Jan 1880-Jan 2017",
     xlab="Year",ylab="Percent area covered [%]")
```

2.3.3 Difference between above averages and the NOAA NCEI monthly mean global averages

NOAA NCEI also computed the monthly mean global averages, which can be downloaded from the same NOAAGlobalTemp website. The differences between our monthly means and the NCEI's monthly means are less than 0.02°C. Figure 2.7 shows our data minus the NCEI data, and can be generated by the following R code.

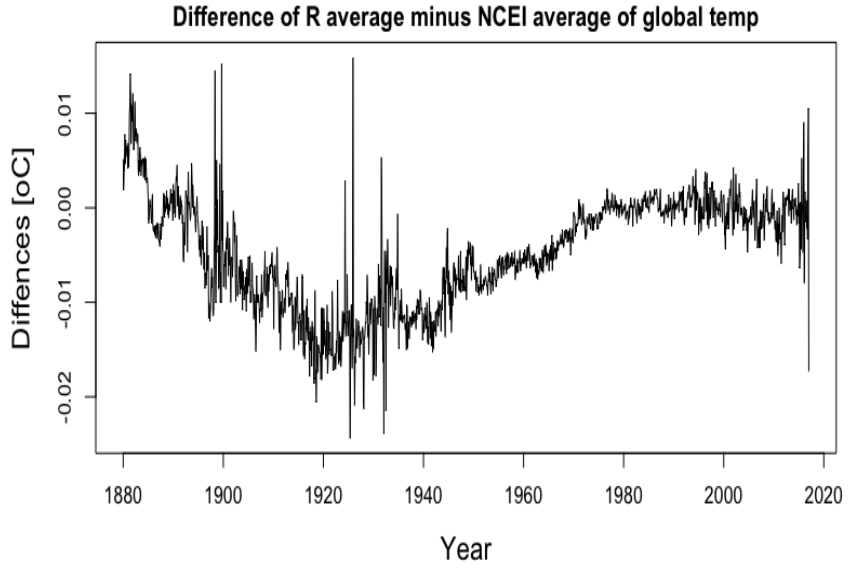

Figure 2.7 Shen's spatial average of monthly anomalies minus the NCEI time series.

```
#Download the NCEI spatial average time series of monthly data
#https://www1.ncdc.noaa.gov/pub/data/noaaglobaltemp/operational/
#timeseries/aravg.mon.land_ocean.90S.90N.v4.0.1.201702.asc
setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2016/Rcodes/Ch12-13-Rgraphics"
aveNCEI <- read.table("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2016/Rcodes/Ch
                      header=FALSE)
dim(aveNCEI) #Jan 1880-Feb 2017 #an extra month to be deleted
#[1] 1646   10
avediff<-avev-aveNCEI[1:1645,3]
par(mar=c(4,5,2,1))
plot(timemo,avediff,type="l",
     cex.lab=1.4,
     xlab="Year",
     ylab="Diffences [oC]",
     main="Difference of R average minus NCEI average of global temp")
```

The small difference might be caused by the different round off errors in the computer programs. The .asc data have four decimal places. The differences are in the third or more decimal places. Double procession might help to eliminate the differences.

Climatologically, these small differences of less than 0.02 $^{\circ}C$ do not alter scientific conclusions implied by the NOAAGlobalTemp data.

2.3.4 Which month has the strongest trend?

It is known that climate changes are not uniform across a year. We thus plot the trends of each month from January to December in the period of 1880-2016. Figure 2.8 shows the strongest trend 0.75°/century in March, and the weakest trend 0.656°/century in September. This method of study is even more meaningful for hemispheric averages or regional averages, such as the United States. Figure 2.8 can be produced by the following R code.

```
#create matrix of 136 yrs of data matrix
# row=year from 1880 to 2016, col=mon 1 to 12
avem = matrix(avev[1:1644], ncol=12, byrow=TRUE)

#compute annual average
  annv=seq(0,length=137)
  for(y in 1:137){annv[y]=mean(avem[y,])}

#Put the monthly averages and annual ave in a matrix
  avemy=cbind(avem,annv)

#Plot 12 panels on the same figure
  plot.new()
  timeyr=seq(1880, 2016, len=137)
  par(mfrow = c(4, 3))  # 4 rows and 3 columns
  par(mgp=c(2,1,0))
  for (i in 1:12) {
    plot(timeyr, avemy[,i],type="l", ylim=c(-1.0,1.0),
         xlab="Year",ylab="Temp [oC]",
         main = paste("Month is", i, split = ""))
```

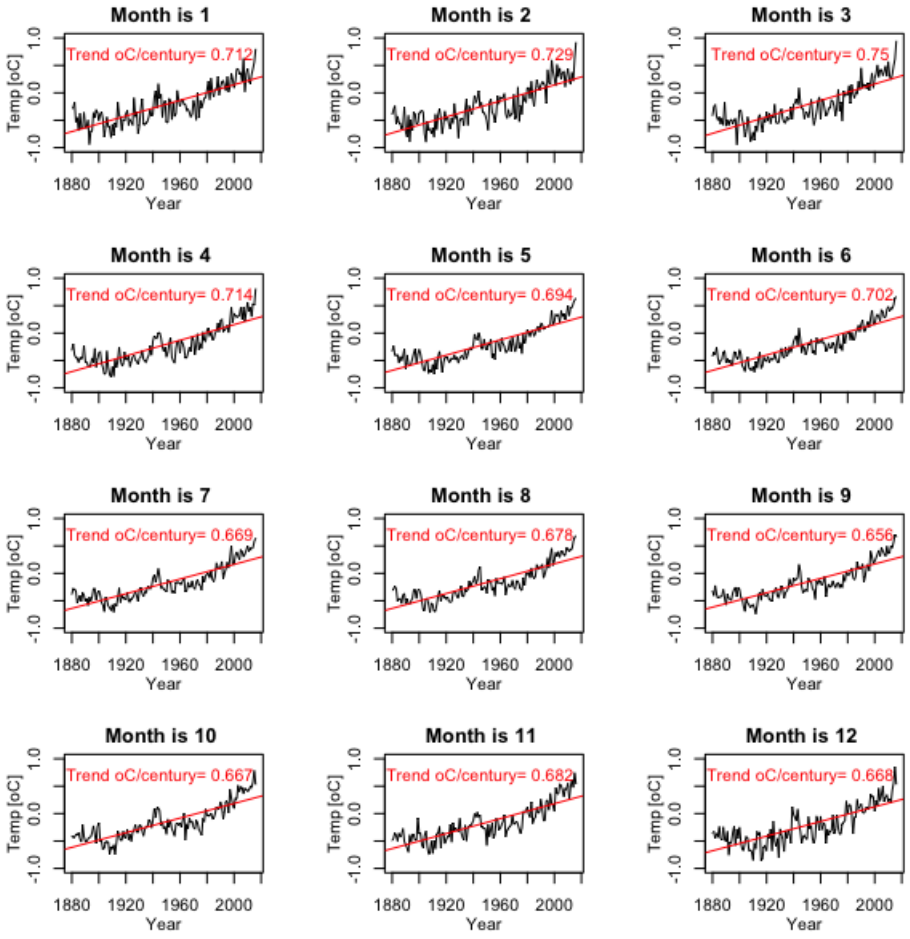

Figure 2.8 The trend of the spatial average of each month based on the NOAAGlobalTemp data from 1880-2016.

```
    abline(lm(avemy[,i]~yrtime),col="red")
    text(1945,0.7, paste("Trend oC/century=",
          round(digits=3,
          (100*coefficients(lm(avemy[,i]~yrtime))[2]))),
         col="red")
}
```

2.3.5 Spatial average of annual data

The following R code can compute and plot the annual mean. It first convert the vector data of monthly spatial averages to a 12-column matrix. Each column is a month. The row mean yields the annual mean.

```
plot.new()
```

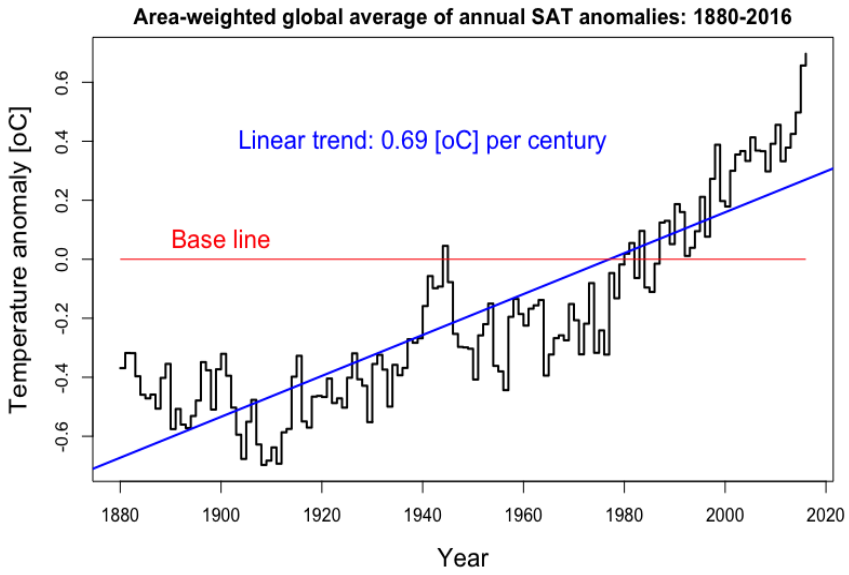

Figure 2.9 Annual mean of the monthly spatial average anomalies from the NOAAGlobalTemp data.

```
  avem = matrix(avev[1:1644], ncol=12, byrow=TRUE)
  #compute annual average
  annv=rowMeans(avem)
  #Plot the annual mean global average temp
  timeyr<-seq(1880, 2016)
  plot(timeyr,annv,type="s",
       cex.lab=1.4, lwd=2,
       xlab="Year", ylab="Temperature anomaly [oC]",
       main="Area-weighted global average of annual SAT anomalies: 1880-2016")
  abline(lm(annv ~ timeyr),col="blue",lwd=2)
  text(1940,0.4, "Linear trend: 0.69 [oC] per century",
       cex=1.4, col="blue")
  text(1900,0.07, "Base line",cex=1.4, col="red")
  lines(timeyr,rep(0,137), type="l",col="red")

#One can compare with the NOAAGlobalTemp annual time series
#There are some small differences
aveannNCEI <- read.table("http://shen.sdsu.edu/data/aravg.ann.land_ocean.90S.90N.v4.0.1.20
dim(aveannNCEI)
#[1] 138   6
diff2=annv-aveannNCEI[1:137,2]
range(diff2)
#[1] -0.016094969  0.007508731
```

2.3.6 Nonlinear trend of the global average annual mean data

The global average annual mean temperature apparently does not vary linearly. It is thus useful to examine nonlinear underlining variation of the annual temperature time series. The simplest nonlinear trend exploration is thorough polynomial fit. Usually, orthogonal polynomial fits are more efficient and have better fidelity to data. Figure 2.10 shows two fits by the 9th order and 20th order orthogonal polynomials. The choice of 9th order is because it is the lowest order polynomial which can reflect the oscillation of temperature from high in the 1880s to the low in the1910s, rising until the 1940s, decreasing in the 1960s and 1970s. The choice os the 20th order polynomial fit is because it is the lowest order orthogonal polynomial that can mimic the detailed climate variations, such as the local highs around 1900 and 1945. We have tried higher order polynomials which often show unphysical overfit.

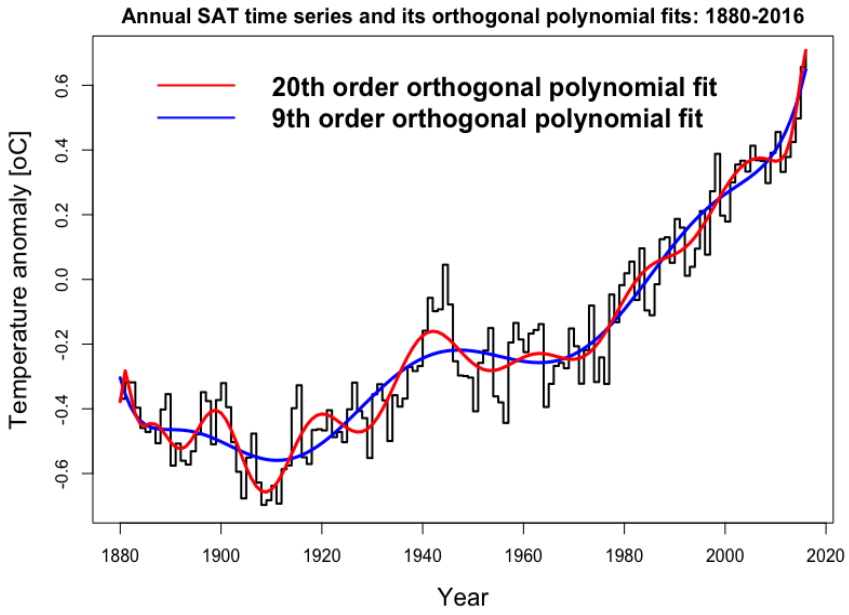

Figure 2.10 Annual mean time series and its fit by orthogonal polynomials.

Figure 2.10 can be produced by the following R code.

```
#Polynomial fitting to the global average annual mean
#poly9<-lm(annv ~ poly(timeyr,9, raw= TRUE))
#raw=TRUE means regular polynomial a0+a1x^2+..., non-orthogonal
polyor9<-lm(annv ~ poly(timeyr,9, raw= FALSE))
polyor20<-lm(annv ~ poly(timeyr,20, raw= FALSE))
#raw=FALSE means orthongonal polynomial of 9th order
#Orthogonal polynomial fitting is usually better
plot(timeyr,annv,type="s",
     cex.lab=1.4, lwd=2,
     xlab="Year", ylab="Temperature anomaly [oC]",
     main="Annual SAT time series and its orthogonal polynomial fits: 1880-2016")
lines(timeyr,predict(polyor9),col="blue", lwd=3)
```

```
legend(1880, 0.6,  col=c("blue"),lty=1,lwd=2.0,
       legend=c("9th order orthogonal polynomial fit"),
       bty="n",text.font=2,cex=1.5)
lines(timeyr,predict(polyor20),col="red", lwd=3)
legend(1880, 0.7,  col=c("red"),lty=1,lwd=2.0,
       legend=c("20th order orthogonal polynomial fit"),
       bty="n",text.font=2,cex=1.5)
```

A popular non-parametric fit is the LOWESS (locally weighted scatterplot smoothing), often referred to as the Loess fit. It is basically a weighted piecewise local polynomial fitting. The local fitting property requires many data points to make a reasonable fit. One can use the following one-line R code to generate a nonlinear fit which has a shape similar to the 20th order polynomial fit.

```
scatter.smooth(annv   timeyr, span=2/18, cex=0.6)
```

2.4 Spatial characteristics of the temperature change trends

It has been widely shown the global temperature has increased. However, the increase is non-uniform, and a few places even experience cooling, such as the 1900-1999 cooling over the North Atlantic, off the coast of Canada. Figure 2.11 shows the uneven spatial distribution of the linear trend of the monthly SAT anomalies in the NOAAGlobalTemp data from January 1900-December 1999. Most parts of the world experienced warming particularly over the land areas. Canada and Russia experienced more warming in the 20th century, compared to other regions over the world.

Many grid boxes do not have complete data stream from January 1900-December 1999. Our trend calculation's R code allows some missing data in the mid of the data stream, but requires data at the beginning month (January 1900) and the end month (December 1999). When a grid box does not satisfy the requirement, the trend for the box is not calculated. Figure 2.11's large white areas over the polar regions, Pacific, Africa, and Central America do not satisfy the requirement. For the missing data in the middle of a data stream for a grid box, our linear regression omits the missing data and carries out the regression with a shorter temperature data stream, and correspondingly with a shorter time data stream.

Figure 2.11 can be produced by the following R code.

```
#Compute the trend for each box for the 20th century
timemo1=seq(1900,2000, len=1200)
temp1=temp
temp1[temp1 < -490.00] <- NA
trendgl=rep(0,2592)
for (i in 1:2592){
  if(is.na(temp1[i,243])==FALSE & is.na(temp1[i,1442])==FALSE)
  {trendgl[i]=lm(temp1[i,243: 1442] ~ timemo1, na.action=na.omit)$coefficients[2]}
  else
    {trendgl[i]=NA}
}
library(maps)
Lat= seq(-87.5, 87.5, length=36)
Lon=seq(2.5, 357.5, length=72)
```

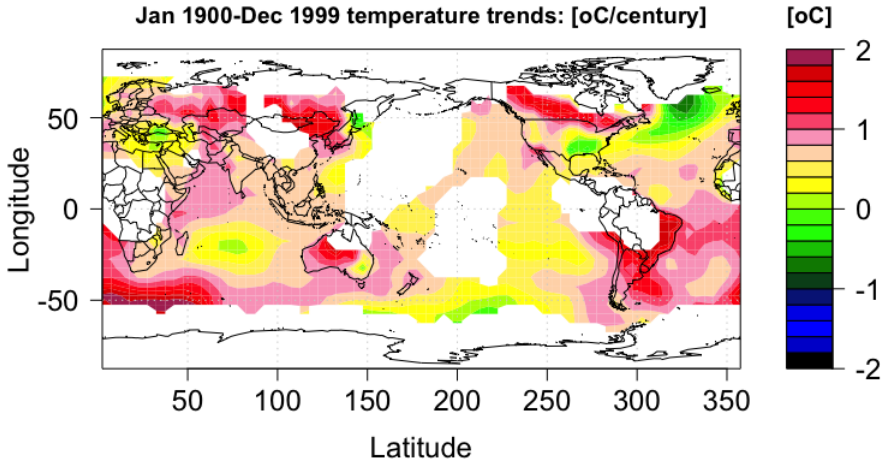

Figure 2.11 Linear trend of SAT from January 1900-December 1999. The trend was calculated for each grid box using the NOAAGlobalTemp data and required that the box did not have missing data for the first month (January 1900) and the last month (December 1999). The white regions mean that the data did not satisfy our calculation conditions. Namely, the regions are of insufficient amount of data.

```
mapmat=matrix(100*trendgl,nrow=72)
mapmat=pmax(pmin(mapmat,2),-2)
#Matrix flipping is not needed since the data goes from 2.5 to 375.5
plot.new()
par(mar=c(4,5,3,0))
int=seq(-2,2,length.out=21)
rgb.palette=colorRampPalette(c('black','blue', 'darkgreen','green',
'yellow','pink','red','maroon'),interpolate='spline')
#mapmat= mapmat[,seq(length(mapmat[1,]),1)]
filled.contour(Lon, Lat, mapmat, color.palette=rgb.palette, levels=int,
               plot.title=title(main="Jan 1900-Dec 1999 temperature trends: [oC/century]",
                                xlab="Latitude",ylab="Longitude", cex.lab=1.5),
plot.axes={axis(1, cex.axis=1.5); axis(2, cex.axis=1.5);map('world2', add=TRUE);grid()},
               key.title=title(main="[oC]"),
               key.axes={axis(4, cex.axis=1.5)})
```

If the data requirement is relaxed, saying allowing one-third data missing, then the trend can be computed almost everywhere and the white region will be reduced.

```
#Compute trend for each box for the 20th century
#Version 2: Allow 2/3 of data, i.e., 1/3 missing
#Compute the trend
timemo1=seq(1900,2000, len=1200)
temp1=temp[,243:1442]
temp1[temp1 < -490.00] <- NA
temptf=is.na(temp1)
bt=et=rep(0,2592)
for (i in 1:2592) {
```

```
  if (length(which(temptf[i,]==FALSE)) !=0)
  {
    bt[i]=min(which(temptf[i,]==FALSE))
    et[i]=max(which(temptf[i,]==FALSE))
  }
}

trend20c=rep(0,2592)
for (i in 1:2592){
  if(et[i]-bt[i] > 800)
  {trend20c[i]=lm(temp1[i,bt[i]:et[i]] ~ seq(bt[i],et[i]), na.action=na.omit)$coefficients
  else
  {trend20c[i]=NA}
}
#plot the 20C V2 trend map
plot.new()
#par(mar=c(4,5,3,0))
mapmat=matrix(120*trend20c,nrow=72)
mapmat=pmax(pmin(mapmat,0.2),-0.2)
int=seq(-0.2,0.2,length.out=41)
rgb.palette=colorRampPalette(c('black','blue', 'darkgreen','green', 'yellow','pink','red',
filled.contour(Lon, Lat, mapmat, color.palette=rgb.palette, levels=int,
               plot.title=title(main="Jan 1900-Dec 1999 temperature trends: [oC/decade]",
                                xlab="Latitude",ylab="Longitude", cex.lab=1.5),
               plot.axes={axis(1, cex.axis=1.5); axis(2, cex.axis=1.5);map('world2', add=T
               key.title=title(main="[oC]"),
               key.axes={axis(4, cex.axis=1.5)})
```

Our recent period of long-term rapid warming (four decades from 1976-2016) is stronger than the last long-term warming from the 1910s to the early 1950s, which was also about four decades. Figure 2.12 shows the strong warming trend from January 1976 -December 2016. It shows that the world became warmer on every continent except Antarctic.

The trend data for Fig. 2.12 can be calculated using the following R code.

```
timemo2=seq(1976,2017, len=492)
temp1=temp
temp1[temp1 < -490.00] <- NA
trend7616=rep(0,2592)
for (i in 1:2592){
  if(is.na(temp1[i,1155])==FALSE & is.na(temp1[i,1646])==FALSE)
  {trend7616[i]=lm(temp1[i,1155: 1646] ~ timemo2, na.action=na.omit)$coefficients[2]}
  else
  {trend7616[i]=NA}
}
```

The R code for plotting Fig. 2.12) is almost identical to that for the 20th century trend and is omitted here.

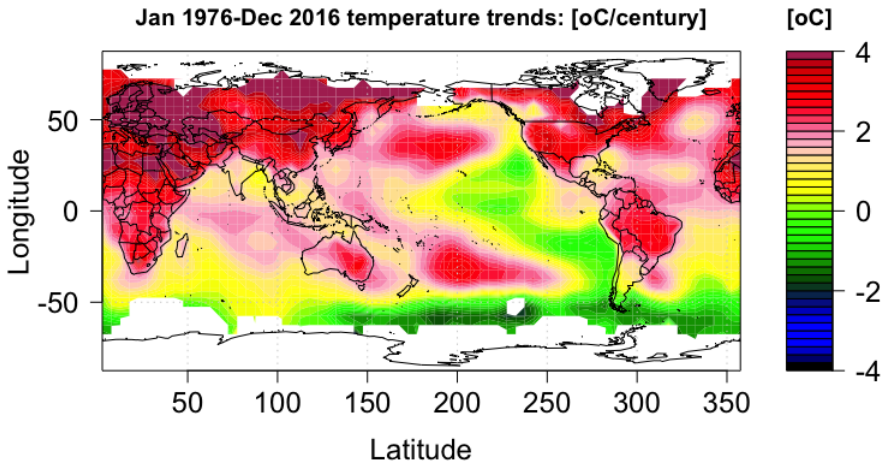

Figure 2.12 Linear trend of SAT from January 1976-December 2016. The white regions mean insufficient amount of data.

CHAPTER 3

R GRAPHICS FOR CLIMATE SCIENCE

This chapter provides basic skills of R graphics for climate science. These skills are sufficient to meet most needs for climate science research, teaching and publications. The skills are divided into the following categories:

(i) Multiple data time series on the same figure, multiple panels of a figure, adjustment of margin, and fonts of text, labels, and axes;

(ii) Color maps of a climate parameter, such as the surface air temperature on the globe or over a given region; and

(iii) Animation.

3.1 Two dimensional line plots and setups of margins and labels

3.1.1 Plot two different time series on the same plot

We have already showed how to plot a simple time series using `plot(xtime, ydata)`. Climate science often requires to plot two different quantities time series on the same plot so that direct comparisons can be made. For example, to see whether a hot year is also a dry year, one may plot the temperature data on the same figure as the precipitation data. The left side of the y-axis shows temperature and the right side shows precipitation. The temperature and precipitation share the same time axis. The following code plots the

figure for the contiguous United States (CONUS) annual mean temperature and annual total precipitation from 2001-2010 (see Fig. 3.1).

```
#Plot US temp and prec times series on the same figure
plot.new()
Time <- 2001:2010
Tmean <- c(12.06, 11.78,11.81,11.72,12.02,12.36,12.03,11.27,11.33,11.66)
Prec <- c(737.11,737.87,774.95,844.55,764.03,757.43,741.17,793.50,820.42,796.80)
plot(Time,Tmean,type="o",col="red",xlab="Year", ylab="Tmean [dec C]",lwd=1.5,
     main="Contiguous U.S. Annual Mean Temperature and Total Precipitation")
legend(2000.5,12.42, col=c("red"),lty=1,lwd=2.0,
       legend=c("Tmean"),bty="n",text.font=2,cex=1.0)
#Allows a figure to be overlaid on the first plot
par(new=TRUE)
plot(Time, Prec,type="o",col="blue",lwd=1.5,axes=FALSE,xlab="",ylab="")
legend(2000.5,839, col=c("blue"),lty=1,lwd=2.0,
       legend=c("Prec"),bty="n",text.font=2,cex=1.0)
#Suppress the axes and assign the y-axis to side 4
axis(4)
mtext("Precipitation [mm]",side=4,line=3)
#legend("topleft",col=c("red","blue"),lty=1,legend=c("Tmean","Prec"),cex=0.6)
#Plotting two legends at the same time is difficult to adjust the font size
#because of different scales
```

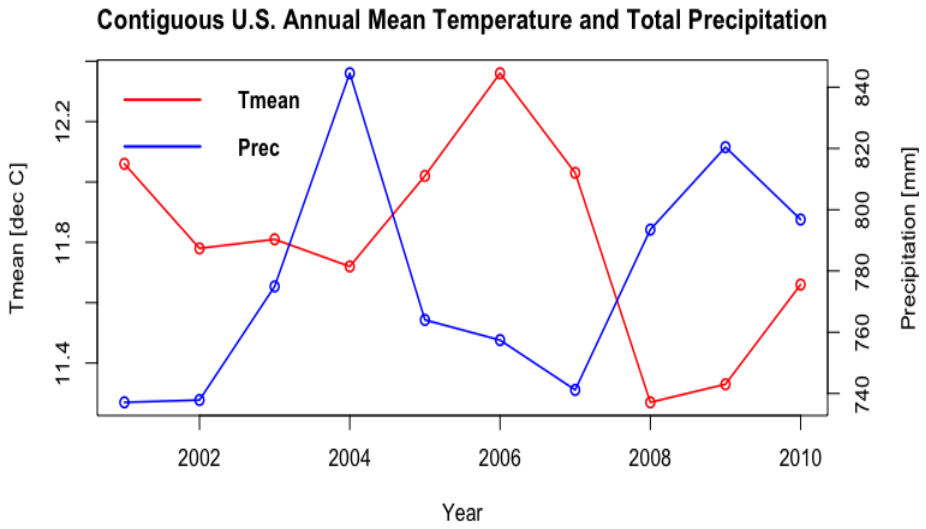

Figure 3.1 Contiguous United States annual mean temperature and annual total precipitation.

Figure 3.1 shows that during the ten years from 2001-2010, the CONUS precipitation and temperature are in opposite phase: higher temperature corresponding to dry years with less precipitation, and lower temperature to wet years with more precipitation.

3.1.2 Figure setups: margins, fonts, mathematical symbols, and more

R has flexibility to make specific margins, mathematical symbols for text and labels, text fonts, text size, and more. R also allows merge of multiple figures. These flexibilities are often useful to produce a high quality figure for presentation or publication.

`par(mar=c(2,5,3,1))` specifies the four margins of a figure. The first margin 2 (i.e., two line space) is the x-axis, the second 5 is for y-axis, 3 for top, and 1 for right. One can change the numbers in `par(mar=c(2,5,3,1))` to adjust the margins. A simple example is shown in Fig. 3.2, which may be generated by the following R program

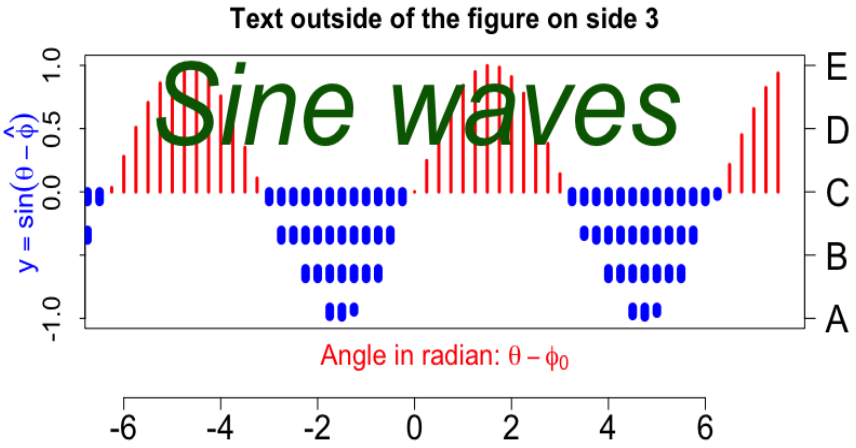

Figure 3.2 Set margins, insert mathematical symbols, and write text outside of a figure.

```
#Margins, math symbol, and figure setups
plot.new()
par(mar=c(6,4,3,3))
x<-0.25*(-30:30)
y<-sin(x)
x1<-x[which(sin(x) >=0)]
y1<-sin(x1)
x2<-x[which(sin(x) < 0)]
y2<-sin(x2)
plot(x1,y1,xaxt="n", xlab="",ylab="",lty=1,type="h",
     lwd=3, tck=-0.02, ylim=c(-1,1), col="red",
     col.lab="purple",cex.axis=1.4)
lines(x2,y2,xaxt="n", xlab="",ylab="",lty=3,type="h",
      col="blue",lwd=8, tck=-0.02)
axis(1, at=seq(-6,6,2),line=3, cex.axis=1.8)
axis(4, at=seq(-1,1,0.5), lab=c("A", "B", "C", "D","E"),
     cex.axis=2,las=2)
text(0,0.7,font=3,cex=6, "Sine waves", col="darkgreen") #Itatlic font size 2
mtext(side=2,line=2, expression(y==sin(theta-hat(phi))),cex=1.5, col="blue")
mtext(font=2,"Text outside of the figure on side 3",side=3,line=1, cex=1.5)#Bold font
mtext(font=1, side=1,line=1,
```

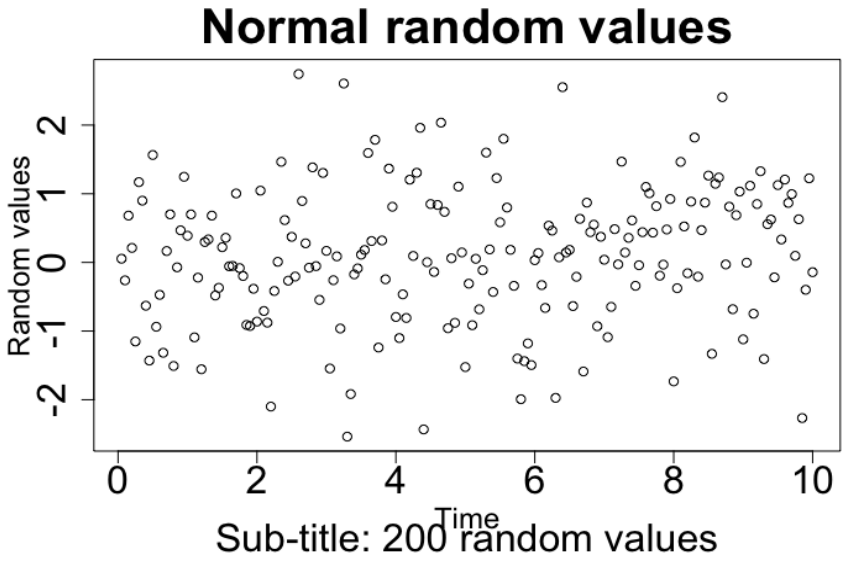

Figure 3.3 Adjust font size, axis labels space, and margins.

```
      expression(paste("Angle in radian: ",
                       theta-phi[0])),cex=1.5, col="red")
```

Similar to `cex.axis=1.8` that changes the font size of the tick values, one can use `cex.lab=1.5, cex.main=1.5, cex.sub=1.5` to change the font sizes for axis labels, main title, and sub-title. An example is shown in Fig. 3.3 generated by the R code below.

```
par(mar=c(8,6,3,2))
par(mgp=c(2.5,1,0))
plot(1:200/20, rnorm(200),sub="Sub-title: 200 random values",
     xlab= "Time", ylab="Random values", main="Normal random values",
     cex.lab=1.5, cex.axis=2, cex.main=2.5, cex.sub=2.0)
```

Here `par(mgp=c(2.5,1,0))` is used to adjust the positions of axis labels, ticks values, and tick bars, where 2.5 means the xlab is two and half lines away from the figure's lower and left borders, 1 means the x-axis tick values are one line away from the borders, 0 means the tick bars are on the border lines. The default mgp values are 3,1,0. Another simple example is below.

```
par(mgp=c(2,1,0))
plot(sin,xlim=c(10,20))
```

The above R codes used many R plot functions. An actual climate science line plot is often simpler than this. One can simply remove the redundant functions in the above R code to produce the desired figure.

Let us reproduce the global average annual mean surface air temperature (SAT) time series figure in the Inter-governmental Panel for Climate Change (IPCC). See the result Fig. 3.4 from 1880 - 2016 using the above plot functions. The data are from the NOAA-GlobalTemp dataset

https://www.ncdc.noaa.gov/data-access/marineocean-data/noaa-global-surface-temperature-noaaglobaltemp
We write the data into two columns in a file named NOAATemp. The first column is year, and the second the temperature anomalies.

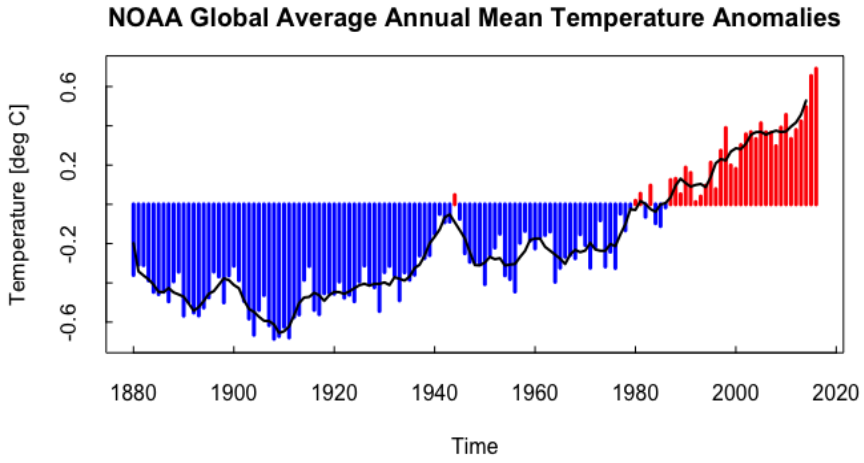

Figure 3.4 Global average annual mean SAT based on the United States' NOAAGlobalTemp data
.

Figure 3.4 can be generated by the following R code.

```
#A fancy IPCC plot of the NOAAGlobalTemp time series
NOAATemp=read.table("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2016/Rcodes/Ch12
plot.new()
par(mar=c(4,4,3,1))
x<-NOAATemp[,1]
y<-NOAATemp[,2]
z<-rep(-99,length(x))
for (i in 3:length(x)-2) z[i]=mean(c(y[i-2],y[i-1],y[i],y[i+1],y[i+2]))
n1<-which(y>=0)
x1<-x[n1]
y1<-y[n1]
n2<-which(y<0)
x2<-x[n2]
y2<-y[n2]
x3<-x[2:length(x)-2]
y3<-z[2:length(x)-2]
plot(x1,y1,type="h",xlim=c(1880,2016),lwd=3,
     tck=0.02, ylim=c(-0.7,0.7), #tck>0 makes ticks inside the plot
     ylab="Temperature [deg C]",
     xlab="Time",col="red",
     main="NOAA Global Average Annual Mean Temperature Anomalies")
lines(x2,y2,type="h",
      lwd=3, tck=-0.02,  col="blue")
lines(x3,y3,lwd=2)
```

3.1.3 Plot two or more panels on the same figure

Another way to compare the temperature and precipitation time series is to plot them in different panels and display them in one figure, as shown in Fig. 3.5.

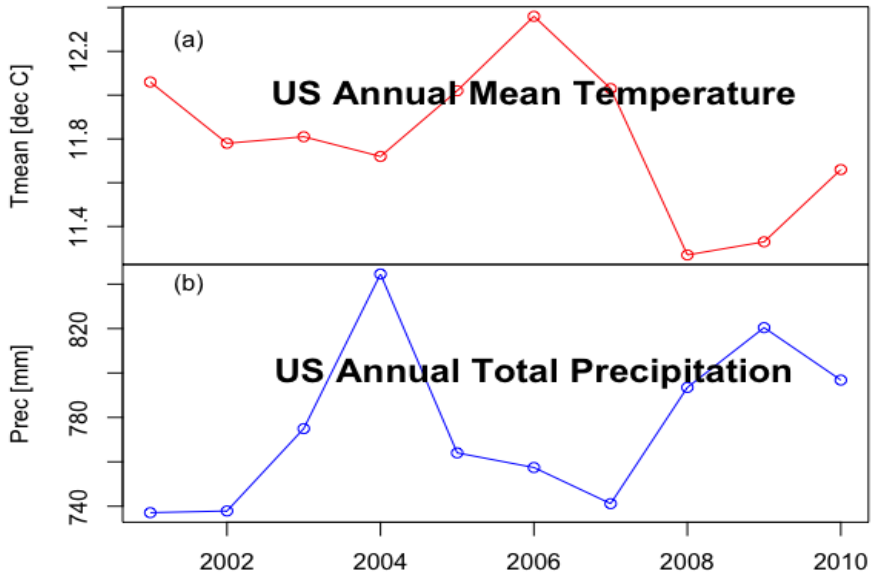

Figure 3.5 (a) Contiguous United States annual mean temperature; and (b) annual total precipitation.

Figure 3.5 can be generated by the following R code. This figure's arrangement has used the setups described in the above sub-section.

```
#Plot US temp and prec times series on the same figure
par(mfrow=c(2,1))
par(mar=c(0,5,3,1)) #Zero space between (a) and (b)
Time <- 2001:2010
Tmean <- c(12.06, 11.78,11.81,11.72,12.02,12.36,12.03,11.27,11.33,11.66)
Prec <- c(737.11,737.87,774.95,844.55,764.03,757.43,741.17,793.50,820.42,796.80)
plot(Time,Tmean,type="o",col="red",xaxt="n", xlab="",ylab="Tmean [dec C]")
text(2006, 12,font=2,"US Annual Mean Temperature", cex=1.5)
text(2001.5,12.25,"(a)")
#Plot the panel on row 2
par(mar=c(3,5,0,1))
plot(Time, Prec,type="o",col="blue",xlab="Time",ylab="Prec [mm]")
text(2006, 800, font=2, "US Annual Total Precipitation", cex=1.5)
text(2001.5,840,"(b)")
```

After completing this figure, the R console may remember the setup. When you plot the next figure for default setup, R may still use the previous setup. One can remove the R memory by clicking `Plots` and selecting `Clear All...`

Sometimes `plot.new()` can also help. Or use `rm(list=ls()` to remove all the commands and variables in this R session.

A more flexible way to stack multiple panels together as a single figure is to use the `layout` matrix. The following example has three panels on a 2-by-2 matrix space. The first panel occupies the first row's two positions. Panels 2 and 3 occupies the second row's two positions.

```
plot.new()
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE),
         widths=c(3,3), heights=c(2,2))
  plot(sin,xlim=c(0,20))
  plot(sin,xlim=c(0,10))
  plot(sin,xlim=c(10,20))
```

This layout setup does not work for the plot function `filled.contour` described in the next section, since it has already used an layout and overwrites any other layout.

3.2 Contour color maps

3.2.1 Basic principles for a R contour plot

The basic principles for a R contour plot are below.

(i) The main purpose of a contour plot is to show a 3D surface with contours or filled contours, or simply called a color map for a climate parameter;

(ii) (x, y, z) coordinates data or function $z = f(x, y)$ should be given; and

(iii) A color scheme should be defined, such as `color.palette = heat.colors`.

A few simple examples are below.

```
x <- y <- seq(-1, 1, len=25)
z <- matrix(rnorm(25*25),nrow=25)
contour(x,y,z, main="Contour Plot of Normal Random Values")
filled.contour(x,y,z, main="Filled Contour Plot of Normal Random Values")
filled.contour(x,y,z, color.palette = heat.colors)
filled.contour(x,y,z, color.palette = colorRampPalette(c("red", "white", "blue")))
```

3.2.2 Plot contour color maps for random values on a map

For climate applications, a contour plot is often overlaid on a geography map, such as a world map or a map of country or a region. Our first example is to show a very simple color plot over the world: plotting the standard normal random values on a $5^\circ \times 5^\circ$ grid over the globe.

```
#Plot a 5-by-5 grid global map of standard normal random values
library(maps)
plot.new()
#Step 1: Generate a 5-by-5 grid (pole-to-pole, lon 0 to 355)
Lat<-seq(-90,90,length=37) #Must increasing
```

```
Lon<-seq(0,355,length=72) #Must increasing
#Generate the random values
mapdat<-matrix(rnorm(72*37),nrow=72)
#The matrix uses lon as row going and lat as column
#Each row includes data from south to north
#Define color
int=seq(-3,3,length.out=81)
rgb.palette=colorRampPalette(c('black','purple','blue','white',
                               'green', 'yellow','pink','red','maroon'),
                             interpolate='spline')
#Plot the values on the world map
filled.contour(Lon, Lat, mapdat, color.palette=rgb.palette, levels=int,
               plot.title=title(xlab="Longitude", ylab="Latitude",
main="Standard Normal Random Values on a World Map: 5 Lat-Lon Grid"),
               plot.axes={ axis(1); axis(2);map('world2', add=TRUE);grid()}
               )
#filled.contour() is a contour plot on an x-y grid.
#Background maps are added later in plot.axes={}
#axis(1) means ticks on the lower side
#axis(2) means ticks on the left side
#Save image with width=800, maintain aspect ratio
```

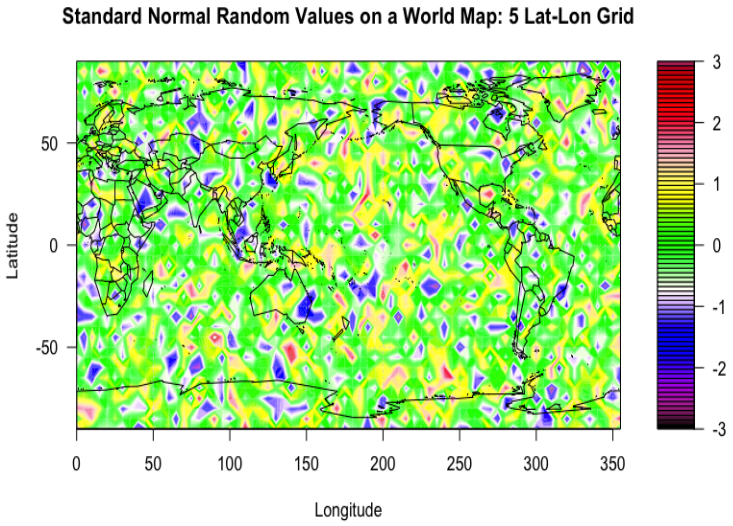

Figure 3.6 Color maps of standard normal random values $5^\circ \times 5^\circ$ grid over the globe.

Similarly one can plot a regional map.

```
#Plot a 5-by-5 grid regional map to cover USA and Canada
Lat3<-seq(10,70,length=13)
Lon3<-seq(230,295,length=14)
mapdat<-matrix(rnorm(13*14),nrow=14)
```

```
int=seq(-3,3,length.out=81)
rgb.palette=colorRampPalette(c('black','purple','blue','white',
                               'green', 'yellow','pink','red','maroon'),
                             interpolate='spline')
filled.contour(Lon3, Lat3, mapdat, color.palette=rgb.palette, levels=int,
               plot.title=title(main="Standard Normal Random Values on a World Map: 5-deg
               xlab="Lon", ylab="Lat"),
               plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()})
```

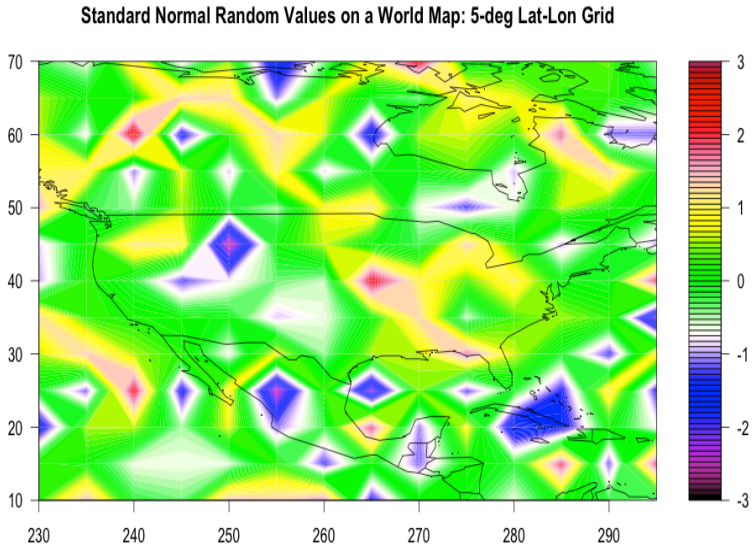

Figure 3.7 Color maps of standard normal random values $5^\circ \times 5^\circ$ grid over Canada and USA.

3.2.3 Plot contour maps from climate model data in NetCDF files

Here we show how to plot the downloaded netCDF NCEP/NCAR Reanalysis data of surface air temperature.
`https://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.derived.surface.html`
The reanalysis data are generated by climate models assimilated (i.e., constrained) by observed data. The reanalysis output is the complete space-time gridded data. Reanalysis is still model data, although some people regard the reanalysis data as dynamically interpolated observational data because of data assimilation. The modern gridded observational data are usually referred to the interpolated results from observational data with assistance of climate models. Spectral optimal gridding (SOG) via EOF is a model-assisted method for gridding observations.

3.2.3.1 Read .nc file We first download the Reanalysis data, which gives a .nc data file: `air.mon.mean.nc`. The R package `ncdf` can read the data into R.

```
#R plot of NCEP/NCAR Reanalysis PSD monthly temp data .nc file
```

```
#http://www.esrl.noaa.gov/psd/data/gridded/data.ncep.
#reanalysis.derived.surface.html

rm(list=ls(all=TRUE))
setwd("/Users/sshen/Desktop/Papers/KarlTom/Recon2016/Test-with-Gregori-prec-data")

# Download netCDF file
# Library
install.packages("ncdf")
library(ncdf4)

# 4 dimensions: lon,lat,level,time
nc=ncdf4::nc_open("air.mon.mean.nc")
nc
nc$dim$lon$vals # output values 0.0->357.5
nc$dim$lat$vals #output values 90->-90
nc$dim$time$vals
#nc$dim$time$units
#nc$dim$level$vals
Lon <- ncvar_get(nc, "lon")
Lat1 <- ncvar_get(nc, "lat")
Time<- ncvar_get(nc, "time")
head(Time)
#[1] 65378 65409 65437 65468 65498 65529
#install.packages("ncdf")
library(chron)
month.day.year(1297320/24,c(month = 1, day = 1, year = 1800))
#1948-01-01
precnc<- ncvar_get(nc, "air")
dim(precnc)
#[1] 144  73 826, i.e., 826 months=1948-01 to 2016-10, 68 years 10 mons
#plot a 90S-90N precip along a meridional line at 160E over Pacific
plot(seq(-90,90,length=73),precnc[15,,1],
     type="l", xlab="Lat", ylab="Temp [oC]",
     main="90S-90N temperature [mm/day]
     along a meridional line at 35E: Jan 1948",
     lwd=3)
```

Here, our first example is to plot the temperature variation along a meridional direction from pole to pole, for a given longitude

Next we plot the global color contour map for the January temperature climatology as the average of the January temperature from 1948- 2015, the surface air temperature of January 1998, and for its anomaly as the difference between January 1998 minus the January climatology. The R code is below and the figures are shown in Figs. 3.9 - 3.11 .

```
#Compute and plot climatology and standard deviation Jan 1948-Dec 2015
library(maps)
climmat=matrix(0,nrow=144,ncol=73)
sdmat=matrix(0,nrow=144,ncol=73)
Jmon<-12*seq(0,67,1)
```

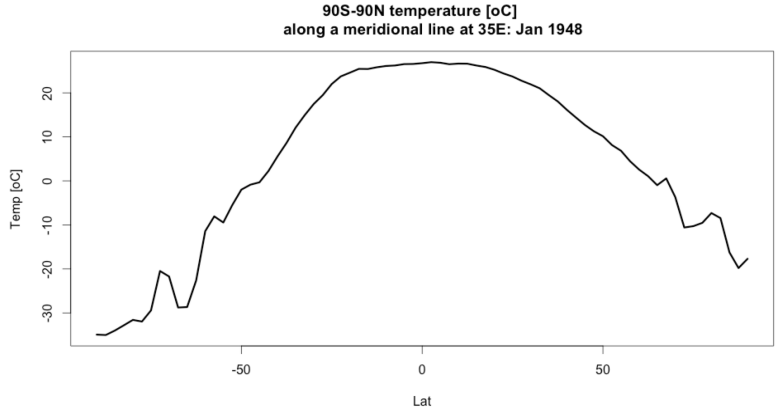

Figure 3.8 The surface air temperature along a meridional line at 160°E over the Pacific.

```
for (i in 1:144){
  for (j in 1:73) {climmat[i,j]=mean(precnc[i,j,Jmon]);
  sdmat[i,j]=sd(precnc[i,j,])
  }
}
mapmat=climmat
#Note that R requires coordinates increasing from south to north -90->90
#and from west to east from 0->360. We have to make Lat and Lon this way.
#Correpondingly, we have to flip the data matrix left to right according to
#the data matrix precnc[i,j,]: 360 (i.e. 180W) lon and from North Pole
#and South Pole, then lon 178.75W, 176.75W, ..., 0E. This puts Greenwich
#at center, China on the right, and USA on the left. But our map should
#have Pacific at center, and USA on teh right. Thus, we make a flip.
Lat=-Lat1
mapmat= mapmat[,length(mapmat[1,]):1]#Matrix flip around a column
#mapmat= t(apply(t(mapmat),2,rev))
int=seq(-50,50,length.out=81)
rgb.palette=colorRampPalette(c('black','blue','darkgreen','green',
               'white','yellow','pink','red','maroon'),interpolate='spline')
filled.contour(Lon, Lat, mapmat, color.palette=rgb.palette, levels=int,
               plot.title=title(main="NCEP RA 1948-2015 January climatology [deg C]",
                                xlab="Longitude",ylab="Latitude"),
               plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()},
               key.title=title(main="[oC]"))

#plot standard deviation
plot.new()
par(mgp=c(2,1,0))
par(mar=c(3,3,2,2))
mapmat= sdmat[,seq(length(sdmat[1,]),1)]
int=seq(0,20,length.out=81)
rgb.palette=colorRampPalette(c('black','blue', 'green','yellow','pink','red','maroon'),
```

```
                                  interpolate='spline')
filled.contour(Lon, Lat, mapmat, color.palette=rgb.palette, levels=int,
 plot.title=title(main="NCEP 1948-2015 Jan SAT RA Standard Deviation [deg C]",
                                   xlab="Longitude", ylab="Latitude"),
               plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()},
               key.title=title(main="[oC]"))
```

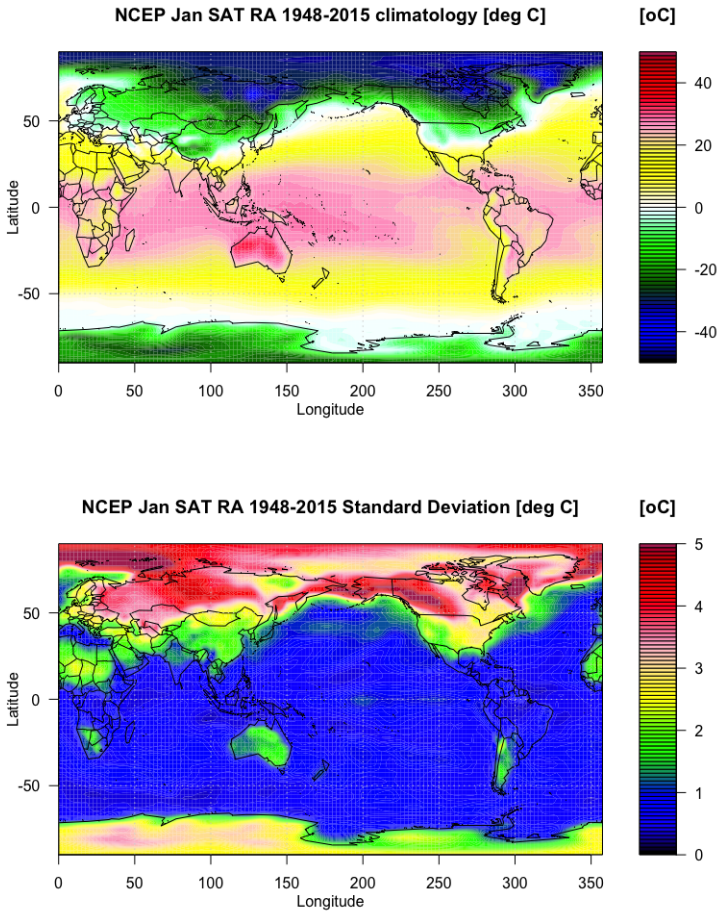

Figure 3.9 NCEP Reanalysis January climatology (upper panel) computed as the January temperature mean from 1948-2015. Lower panel shows the corresponding standard deviation.

3.2.3.2 Plot data for displaying climate features Next figure is the January 1983 temperature. The 1982-83 winter had a strong El Nino event. However, the full temperature field shown in Fig. 3.10 cannot display the El Nino feature: the warming of the eastern tropical Pacific. This is due to that the full temperature field is dominated by its annual cycle: hot in the equator area and cold in the polar areas. El Nino is a phenomenon of climate anomalies: the temperature over the eastern tropical Pacific is warmer than normal by as high as 6°C.

```
#Plot the January 1983 temperature using the above setup
mapmat83J=precnc[,,421]
mapmat83J= mapmat83J[,length(mapmat83J[1,]):1]
int=seq(-50,50,length.out=81)
rgb.palette=colorRampPalette(c('black','blue','darkgreen',
'green', 'white','yellow','pink','red','maroon'),interpolate='spline')
filled.contour(Lon, Lat, mapmat83J, color.palette=rgb.palette, levels=int,
               plot.title=title(main="January 1983 surface air temperature [deg C]",
                                xlab="Longitude",ylab="Latitude"),
               plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()},
               key.title=title(main="[oC]"))
```

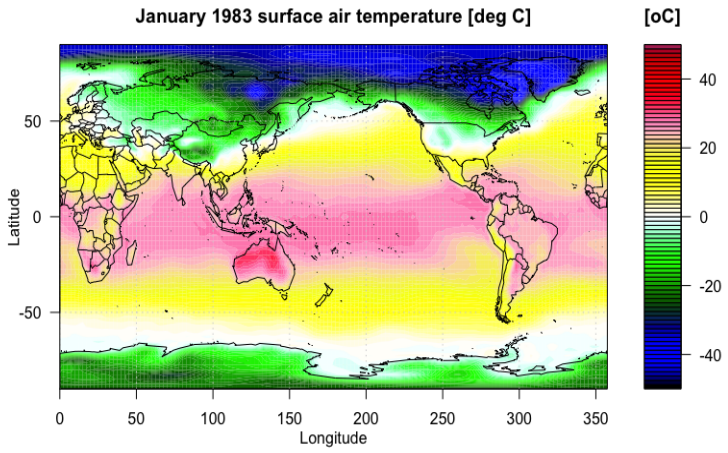

Figure 3.10 NCEP Reanalysis temperature of January 1983: an El Nino event.

To see the El Nino, we compute the temperature anomaly, which is January 1983 temperature minus the January climatology. A large tongue-shape region over the eastern tropical Pacific appears with up to almost 6°C warmer than normal temperature (Fig. 3.11). This is the typical El Nino signal.

```
#Plot the January 1983 temperature anomaly from NCEP data
plot.new()
anomat=precnc[,,421]-climmat
anomat=pmin(anomat,6)
anomat=pmax(anomat,-6)
anomat= anomat[,seq(length(anomat[1,]),1)]
int=seq(-6,6,length.out=81)
rgb.palette=colorRampPalette(c('black','blue','darkgreen','green',
 'white','yellow','pink','red','maroon'),interpolate='spline')
filled.contour(Lon, Lat, anomat, color.palette=rgb.palette, levels=int,
plot.title=title(main="January 1983 surface air temperature anomaly [deg C]",
                                xlab="Longitude",ylab="Latitude"),
               plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()},
               key.title=title(main="[oC]"))
```

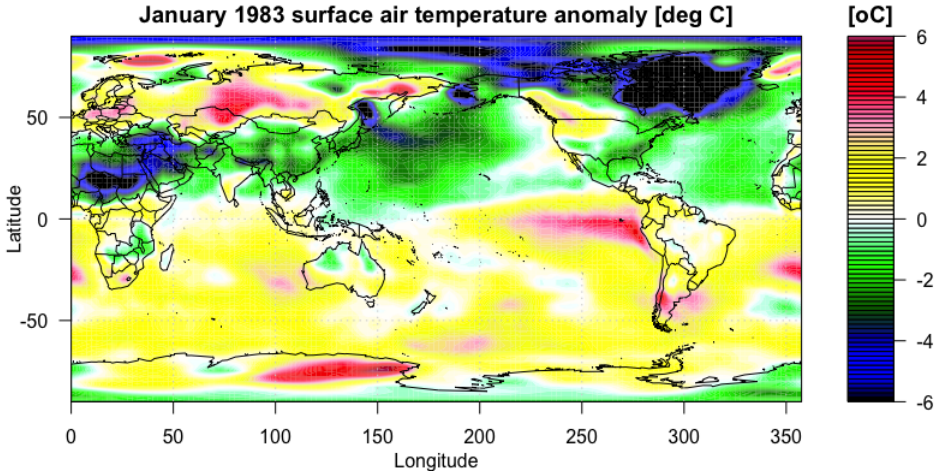

Figure 3.11 NCEP Reanalysis temperature anomaly of January 1983, showing the eastern tropical Pacific's El Nino warming tongue.

Sometimes one needs to zoom in to a given latitude-longitude box of the above maps, in order to see the spatial climate pattern over the region. For example, Fig.3.12 shows the January 1983's SAT anomalies over Pacific and North America. The El Nino pattern over the Pacific and El Nino's influence over North America are much more clear than the global map shown in Fig. 3.11.

The top panel for the Pacific region in Fig. 3.12 may be generated by the following code, which is a minor change from the global map generation: specifying the `xlmin` and `ylim` to the desired region Pacific region (100E, 60W) and (50S,50N).

```
#Zoom in to a specific lat-lon region: Pacific
int=seq(-6,6,length.out=81)
rgb.palette=colorRampPalette(c('black','blue','darkgreen','green',
              'white','yellow','pink','red','maroon'), interpolate='spline')
filled.contour(Lon, Lat, anomat,
              xlim=c(100,300),ylim=c(-50,50),zlim=c(-6,6),
              color.palette=rgb.palette, levels=int,
              plot.title=title(
              main="January 1983 surface air temperature anomaly [deg C]",
              xlab="Longitude",ylab="Latitude"),
              plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()},
              key.title=title(main="[oC]"))
```

The bottom panel for the Northern America region can be generated in the similar way by changing the `xlim` and `ylim`: (130W, 60W) and (20N,60N).

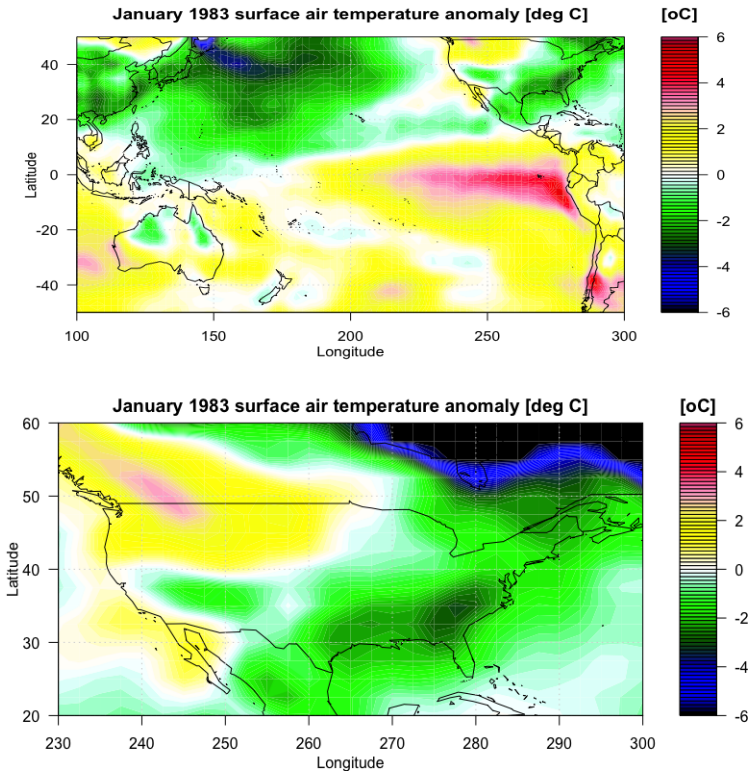

Figure 3.12 NCEP Reanalysis temperature anomaly of January 1983: the Pacific region (the top panel), and the North America region (the bottom panel).

3.3 Plot wind velocity field on a map

3.3.1 Plot a wind field using `arrow.plot`

To describe the use of `arrow.plot`, we use the ideal geostrophic wind field as an example to plot a vector field on a map (see Fig.3.13). The wind field is a result of the balance between the pressure gradient force (PGF) and the Coriolis force (CF).

Figure 3.13 can be generated by the following R code.

```
#Wind directions due to the balance between PGF and Coriolis force
#using an arrow plot for vector fields on a map
library(fields)
library(maps)
library(mapproj)

lat<-rep(seq(-75,75,len=6),12)
lon<-rep(seq(-165,165,len=12),each=6)
x<-lon
y<-lat
#u<- rep(c(-1,1,-1,-1,1,-1), each=12)
```

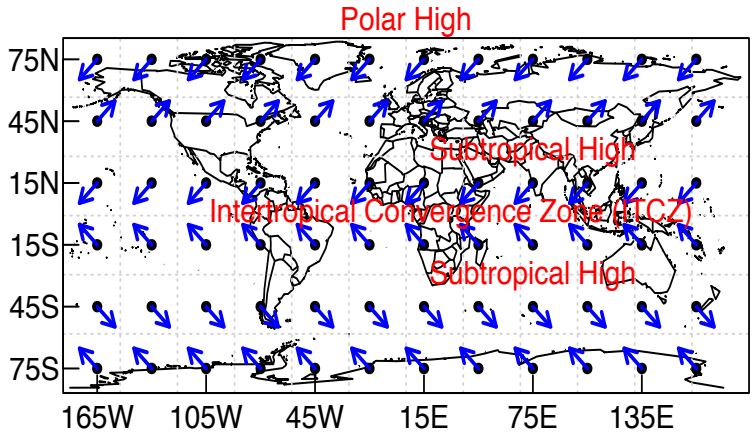

Figure 3.13 Vector field of the ideal geostrophic wind field.

```
#v<- rep(c(1,-1,1,-1,1,-1), each=12)
u<- rep(c(-1,1,-1,-1,1,-1), 12)
v<- rep(c(1,-1,1,-1,1,-1), 12)
wmap<-map(database="world", boundary=TRUE, interior=TRUE)
grid(nx=12,ny=6)
#map.grid(wmap,col=3,nx=12,ny=6,label=TRUE,lty=2)
points(lon, lat,pch=16,cex=0.8)
arrow.plot(lon,lat,u,v, arrow.ex=.08, length=.08, col='blue', lwd=2)
box()
axis(1, at=seq(-165,135,60), lab=c("165W","105W","45W","15E","75E","135E"),
     col.axis="black",tck = -0.05, las=1, line=-0.9,lwd=0)
axis(1, at=seq(-165,135,60),
     col.axis="black",tck = 0.05, las=1, labels = NA)
axis(2, at=seq(-75,75,30),lab=c("75S","45S","15S","15N","45N","75N"),
     col.axis="black", tck = -0.05, las=2, line=-0.9,lwd=0)
axis(2, at=seq(-75,75,30),
     col.axis="black", tck = 0.05, las=1, labels = NA)
text(30, 0, "Intertropical Convergence Zone (ITCZ)", col="red")
text(75, 30, "Subtropical High", col="red")
text(75, -30, "Subtropical High", col="red")
mtext(side=3, "Polar High", col="red", line=0.0)
```

3.3.2 Plot a sea wind field from netCDF data

This sub-section uses `vectorplot` in `rasterVis` to plot the wind velocity field. The NOAA sea wind field is used as an example. The procedure is described from data download to final product of wind field. The NOAA wind data were generated from multiple satellites observations, such as QuikSCAT, SSMIs, TMI, and AMSR-E , on a global at $1/4^\circ \times 1/4^\circ$ grid with a time resolution of 6 hours.

```
#Plot the wind field over the ocean
#Ref: https://rpubs.com/alobo/vectorplot
```

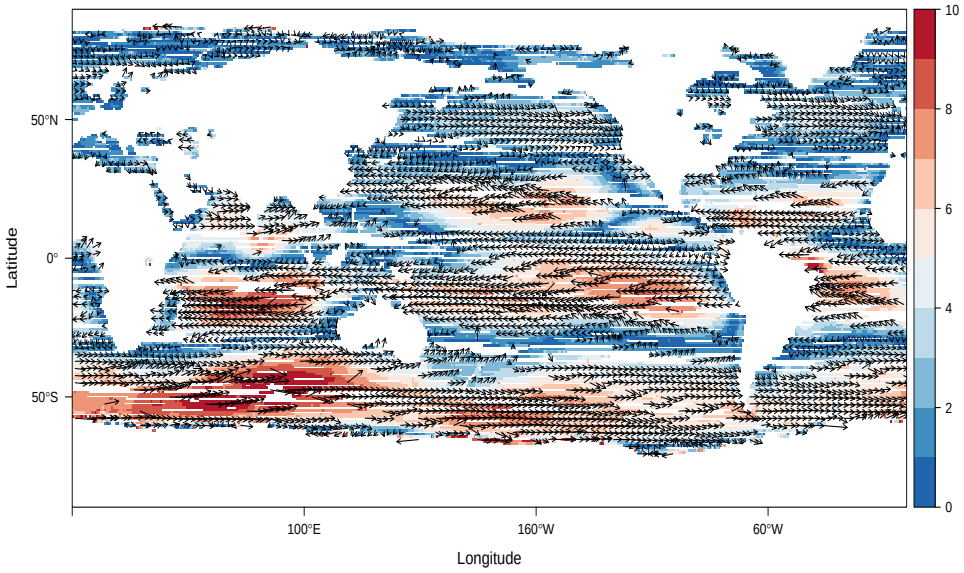

Figure 3.14 The NOAA sea wind field of 1 January 1995: UTC00Z at $1/4^\circ \times 1/4^\circ$ resolution.

```
#Agustin.Lobo@ictja.csic.es
#20140428

library(ncdf4)
library(chron)
library(RColorBrewer)
library(lattice)
download.file("ftp://eclipse.ncdc.noaa.gov/pub/seawinds/SI/uv/clm/uvclm95to05.nc",
              "uvclm95to05.nc", method = "curl")
mincwind <- nc_open("uvclm95to05.nc")
dim(mincwind)
#print.nc(mincwind)
u <- ncvar_get(mincwind, "u")
class(u)
dim(u)
v <- ncvar_get(mincwind, "v")
class(v)
dim(v)
u9 <- raster(t(u[, , 9])[ncol(u):1, ])
v9 <- raster(t(v[, , 9])[ncol(v):1, ])

filled.contour(u[, , 9])
filled.contour(u[, , 9],color.palette = heat.colors)
filled.contour(u[, , 9],color.palette = colorRampPalette(c("red", "white", "blue")))
contourplot(u[, , 9])

install.packages("raster")
```

```
library(raster)
library(sp)
library(rgdal)
u9 <- raster(t(u[, , 9])[ncol(u):1, ])
v9 <- raster(t(v[, , 9])[ncol(v):1, ])
w <- brick(u9, v9)
wlon <- ncvar_get(mincwind, "lon")
wlat <- ncvar_get(mincwind, "lat")
range(wlon)
range(wlat)

projection(w) <- CRS("+init=epsg:4326")
extent(w) <- c(min(wlon), max(wlon), min(wlat), max(wlat))
w

plot(w[[1]])

plot(w[[2]])

install.packages("rasterVis")
install.packages("latticeExtra")
library(latticeEtra)
library(rasterVis)

vectorplot(w * 10, isField = "dXY", region = FALSE, margin = FALSE, narrows = 10000)

slope <- sqrt(w[[1]]^2 + w[[2]]^2)
aspect <- atan2(w[[1]], w[[2]])
vectorplot(w * 10, isField = "dXY", region = slope,
           margin = FALSE,
           par.settings=BuRdTheme,
           narrows = 10000, at = 0:10)

vectorplot(stack(slope * 10, aspect), isField = TRUE, region = FALSE, margin = FALSE)
```

Also see the following websites for more vector field plots `https://www.r-bloggers.com/vectorplot-in-rastervis`
`https://rpubs.com/alobo/vectorplot`

3.4 ggplot for data

`ggplot` is a data-oriented R plot tool developed by Hadley Wickham based on Leland Wilkinsons landmark 1999 book entitled "The Grammar of Graphicsthe" (gg). `ggplot` can generally produce graphic-artist quality default output and make plotting complicated data with a relatively simple code. For example, ggplot can save plots as objects, which allows superposition of different layers in a figure and hence enables one to see the evolution of a figure from an initial framework to the final product. The `ggplot2` library was built on a logical mapping between data and graphical elements and has many maps and datasets useful for climate science.

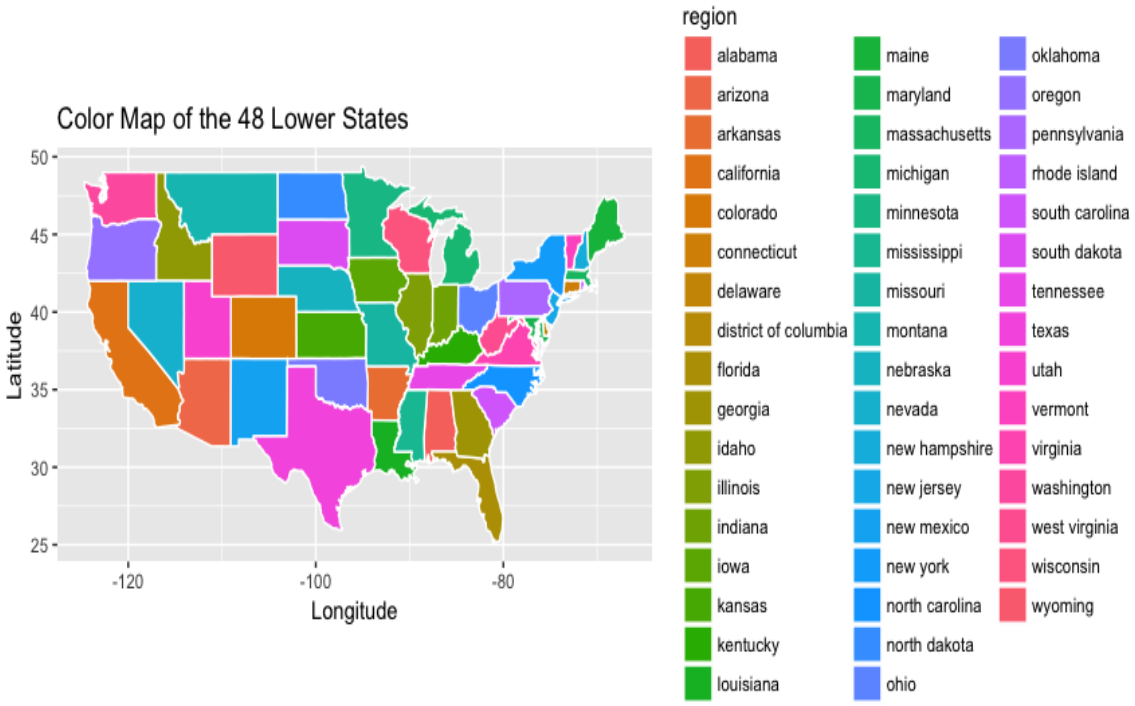

Figure 3.15 Lower 48 states of the United States.

However, ggplot syntax is not the same as the conventional R plot. There is a learning curve.

A simple example is given here for plotting the contiguous lower 48 states of the United States shown in Fig. 3.15. The figure may be generated by the following ggplot code.

```
#ggplot for USA States
library(ggplot2)
states <- map_data("state")
p<- ggplot(data = states) +
  geom_polygon(aes(x = long, y = lat, fill = region, group = group),
               color = "white") +
  coord_fixed(1.3)
#guides(fill=TRUE)  # This leaves off the color legend on the right
p<- p + xlab("Longitude")+ ylab("Latitude")
p<- p + ggtitle("Color Map of the 48 Lower States")
p
```

Although some R users highly advocate the use of ggplot, a non-expert of R may stay with the regular plot that might be sufficient for her applications. So ggplot is always a good resource if a figure cannot be generated by the regular R plot code. Many good tutorial materials and examples are online and can be searched by google. The following is a list of online R graphics tutorials.

(i) GeoR `http://geog.uoregon.edu/GeogR/index.html`
: This is a "Geographic Data Analysis Using R Maps in R" written by Pat Bartlein of the University of Oregon for a geographic data analysis course in 2016. For map plotting examples, visit `http://geog.uoregon.edu/GeogR/topics/maps01.html`
`http://geog.uoregon.edu/GeogR/topics/maps02.html`
`http://geog.uoregon.edu/GeogR/topics/maps03.html`

(ii) Introduction to R Graphics with ggplot2
`http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html`: This is a good ggplot2 tutorial, starting from the beginning and ending with relatively complex plots.

CHAPTER 4

CALCULATIONS AND PLOTTING OF EOFS, PCS, AND VARIANCES

4.1 Ideas of EOF, PC and variances from SVD

SVD decomposes space-time climate data matrix A into three parts: spatial patterns U, temporal patterns V, and energy D, i.e.,

$$A_{N\times Y} = UDV^t. \tag{4.1}$$

Here, the data has N rows for N spatial locations, Y columns for Y length of temporal observations, and the superscript t means matrix transpose. Both U and V are orthogonal matrices, meaning that each column has length equal to one and is orthogonal to a different column vector. The first column of U determines the spatial pattern of mode one. The pattern is called an empirical orthogonal function (EOF), coined in the in the 1950s by Edward Lorenz, who is well known for his contributions to the theories of chaos and the butterfly effect. The corresponding first column of V determines the temporal pattern, which is called a principal component (PC). The variation magnitude corresponding to the EOF and PC is measured by the first element d_1 in the diagonal matrix D. The energy or variance of the observed system for data A corresponding to mode one is measured by d_1^2/Y, where Y is the temporal length of the observation. Thus, the nth mode has an EOF u_n for spatial pattern , a PC v_n for temporal pattern, and d_n^2/Y for energy.

For example, for the standardized monthly SLP data of Darwin and Tahiti from January 1951-December 2015, the EOFs are the SLP patterns at the two locations, and the PCs are the corresponding time series. EOF1 shows that Darwin and Tahiti have opposite SLP anomalies. When Darwin's SLP anomaly is positive and Tahiti's negative, it is an El Nino.

The corresponding PC1's positive peaks shows the time when El Nino actually happened. PC1's negative peaks indicates the time of La Nina. EOF1 and PC1 are both referred to mode 1 and has "energy": 31.35^2, which is in fact the variance of $A\mathbf{u}_1$, i.e., the data's projection on the first EOF pattern. The variance of a mode relative to the total variance is a more useful piece of information. In Darwin-Tahiti SLP case, the relative variance of EOF1 is

$$\frac{d_1^2}{d_1^2 + d_2^2} = \frac{31.35^2}{31.35^2 + 22.25^2} = 67\%. \tag{4.2}$$

EOF1 thus accounts 2/3 of the total variance. EOF2's variance relative to the total variance is thus 33%, or 1/3.

4.2 2Dim spatial domain EOFs and 1-Dim temporal PCs

The spatial fields of many climate data applications are two dimensional. The corresponding EOF are over a 2-Dim domain the Earth surface, and the corresponding PCs are on a time interval. This section shows the basics to use SVD to compute EOFs and PCs and use R graphics to show them. We use a simple synthetic data to illustrate the procedures. Next section will show the real climate data.

4.2.1 Generate synthetic data by R

The spatial domain is $\Omega = [0, 2\pi] \times [0, 2\pi]$, and time interval is $T = [1, 10]$. The synthetic data are generated by the following function

$$z(x, y, t) = c_1(t)\psi_1(x, y) + c_2(t)\psi_2(x, y), \tag{4.3}$$

where $\psi_1(x, y)$ and $\psi_2(x, y)$ are two orthonormal basis functions given below

$$\psi_1(x, y) = (1/\pi) \sin x \sin y, \tag{4.4}$$
$$\psi_2(x, y) = (1/\pi) \sin(8x) \sin(8y), \tag{4.5}$$

with

$$\int_\Omega d\Omega \psi_k^2(x, y) = 1, \;\; k = 1, 2 \tag{4.6}$$
$$\int_\Omega d\Omega \psi_1(x, y)\psi_2(x, y) = 0 \tag{4.7}$$

The corresponding basis' expansion coefficients are

$$c_1(t) = \sin(t), \tag{4.8}$$
$$c_2(t) = \exp(-0.3t). \tag{4.9}$$

These are not orthogonal. Thus, the generating function for $z(x, y, t)$ in eq. (4.3) is not an SVD decomposition.

The spatial domain Ω is divided into a 100×100 grid. The time grid is $1, 2, \cdots, 10$. There are 10,000 spatial grid points $(100 \times 100 = 10,000)$ and 10 temporal grid points. The space-time data may be generated by the following R commands.

```
x<-seq(0, 2*pi, len=100)
```

```
y<-seq(0, 2*pi, len=100)
#t<-seq(1,20,by=1)
mydat<-array(0,dim=c(100,100,10))
for(t in 1:10){z<-function(x,y){z=sin(t)*(1/pi)*sin(x)*sin(y) +
                          exp(-0.3*t)*(1/pi)*sin(8*x)*sin(8*y)}
                    mydat[,,t]=outer(x,y,z)
                    }
```

The $z(x, y, t)$ is a superposition of a large scale spatial wave $\psi_1(x, y)$ (see Fig. 4.2) with a small scale wave $\psi_2(x, y)$. The first wave's coefficient $c_1(t)$ varies periodically, while the second wave's coefficient $c_2(t)$ decays exponentially. Thus, for a large time, the z pattern will be dominated by the large scale spatial wave $\psi_1(x, y)$. Figure 4.1 shows the $z(x, y, 1)$ and $z(x, y, 10)$. When time is 1, the figure shows the superposition of a large scale wave to a small scale wave, while the time is 10, the fire shows only the large scale wave and small scale's influence is negligible.

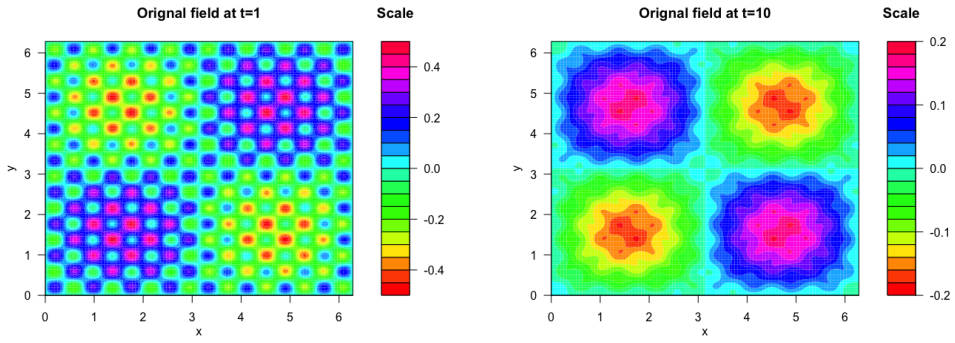

Figure 4.1 The $z(x, y, t)$ function as $t = 1$ and $t = 10$.

Figure 4.1 can be generated by the following `filled.contour` command for a matrix data.

```
#Plot the original z(x,y,t) waves for a given t
plot.new()
int=seq(-0.5,0.5,length.out=61)
filled.contour(x,y,mydat[,,1], color.palette =rainbow, levels=int,
               plot.title=title(main="Orignal field at t=10",
                                xlab="x", ylab="y", cex.lab=1.0),
               key.title = title(main = "Scale"),
               plot.axes =  {axis(1,seq(0,3*pi, by = 1), cex=1.0)
               axis(2,seq(0, 2*pi, by = 1), cex=1.0)}
                 )
```

4.2.2 SVD for the synthetic data: EOFs, variances, and PCs

We first convert the synthetic data into a 1000×10 dimensional space-time data. Then SVD can be applied to the space-time data to generate EOFs, variances and PCs.

The following code coverts the 3-Dim array `mydat(,,,)` into a 2Dim space-time data matrix `da1`.

```
da1<- matrix(0,nrow=length(x)*length(y),ncol=10)
for (i in 1:10) {da1[,i]=c(t(mydat[,,i]))}
```

SVD on this space-time data is below.

```
da2<-svd(da1)
uda2<-da2$u  #EOFs
vda2<-da2$v  #PCs
dda2<-da2$d  #Energy for the corresponding EOFs
dda2
#[1] 3.589047e+01 1.596154e+01 7.764115e-14 6.081008e-14
dda2^2/sum(dda2^2)
# [1] 8.348751e-01 1.651249e-01 1.939476e-29
#The first mode counts for variance 83% variance
#The second 17%, and almost zero for higher modes
```

The EOFs shown in Fig. 4.2 can be plotted by the following R code.

```
par(mgp=c(2,1,0))
filled.contour(x,y,matrix(-uda2[,1],nrow=100), color.palette =rainbow,
               plot.title=title(main="SVD Mode 1: EOF1", xlab="x", ylab="y", cex.lab=1.0),
               key.title = title(main = "Scale"),
               plot.axes =  {axis(1,seq(0,2*pi, by = 1), cex=1.0)
                 axis(2,seq(0, 2*pi, by = 1), cex=1.0)})
```

Figure 4.2 shows that the EOF patterns from SVD are similar to the original orthonormal basis $\psi_1(x, y)$ and $\psi_2(x, y)$. It means that SVD has recovered the original orthonormal basis functions. However, this is not always true, when the variances of the two modes are close to each other. The first two SVD eigenvalues will be close to each other, then the EOFs, as eigenfunctions, will have large differences from the original orthonormal basis functions. North's rule-of-thumb shows the reason of the large differences. In linear algebra terms, this means that when two eigenvalues are close to each other, the corresponding eigenspaces tend to be close to each other, and hence to be mixed to form a 2-dimensional eigenspace, which has infinitely many eigenvectors, rather than one or two fixed eigenvectors. These uncertain eigenvectors have a large chance to be very different from the original orthonormal basis functions.

The original orthonormal basis functions can be plotted by the following R codes.

```
#Accurate spatial patterns from functions that generate data
z1 <- function(x,y){(1/pi)*sin(x)*sin(y)}
z2 <- function(x,y){(1/pi)*sin(5*x)*sin(5*y)}
fcn1<-outer(x,y,z1)
fcn2<-outer(x,y,z2)
par(mgp=c(2,1,0))
filled.contour(x,y,fcn1, color.palette =rainbow,
               plot.title=title(main="Accurate Mode 1",
                                xlab="x", ylab="y", cex.lab=1.0),
               key.title = title(main = "Scale"),
               plot.axes =  {axis(1,seq(0,3*pi, by = 1), cex=1.0)
```

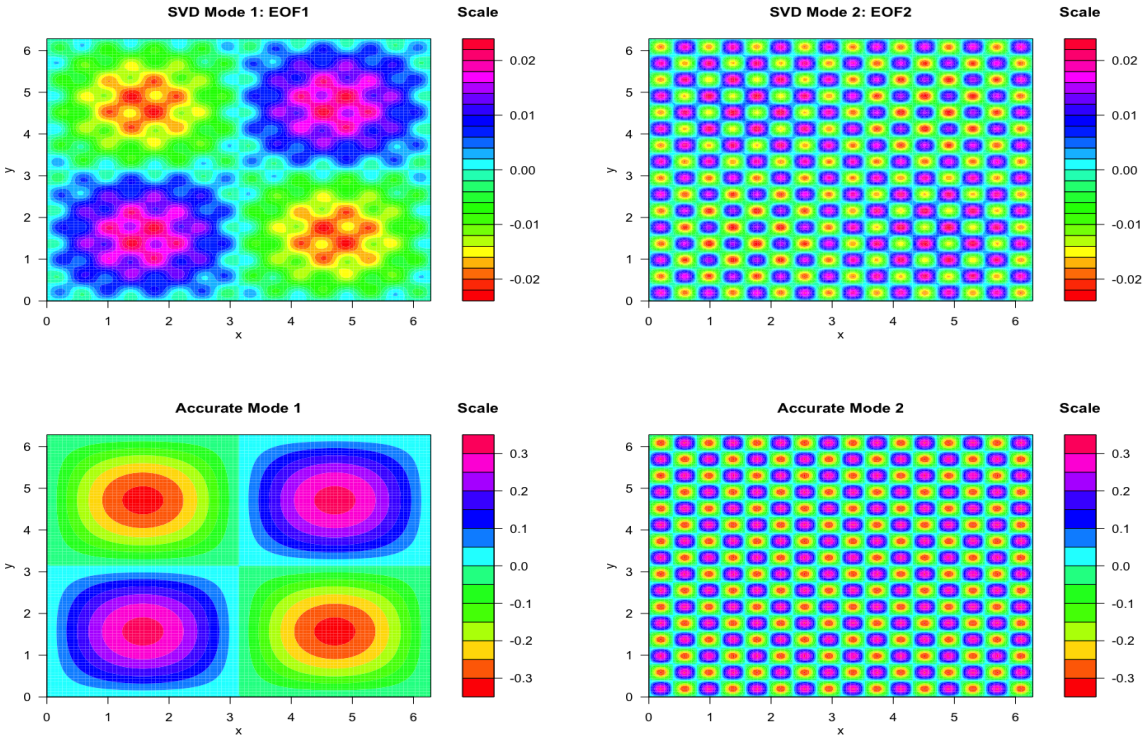

Figure 4.2 The first row shows two EOFs from SVD, and the second row shows two orthonormal basis functions on xy-domain: $\phi_1(x,y) = -(1/\pi)\sin x \sin y$, and $\phi_2(x,y) = (1/\pi)\sin 8x \sin 8y$.

```
                axis(2,seq(0, 2*pi, by = 1), cex=1.0)}
)
```

The first two principal components (PCs) are shown in Fig. 4.3, which can be generated by the following code.

```
#Plot PCs and coefficients of the functional patterns
t=1:10
plot(1:10, vda2[,1],type="o", ylim=c(-1,1), lwd=2,
     ylab="PC or Coefficient", xlab="Time",
     main="SVD PCs vs. Accurate Temporal Coefficients")
legend(0.5,1.15, lty=1, legend=c("PC1: 83% variance"),
       bty="n",col=c("black"))
lines(1:10, vda2[,2],type="o", col="red", lwd=2)
legend(0.5,1.0, lty=1, legend=c("PC2: 17% varance"),
       col="red", bty="n",text.col=c("red"))
lines(t, -sin(t), col="blue", type="o")
legend(0.5,0.85, lty=1, legend=c("Original mode 1 coefficient -sin(t): 91% variance"),
       col="blue", bty="n",text.col="blue")
lines(t, -exp(-0.3*t), type="o",col="purple")
legend(0.5,0.70, lty=1, legend=c("Original mode 2 coefficient -exp(-0.3t): 9% variance"),
```

```
col="purple", bty="n",text.col="purple")
```

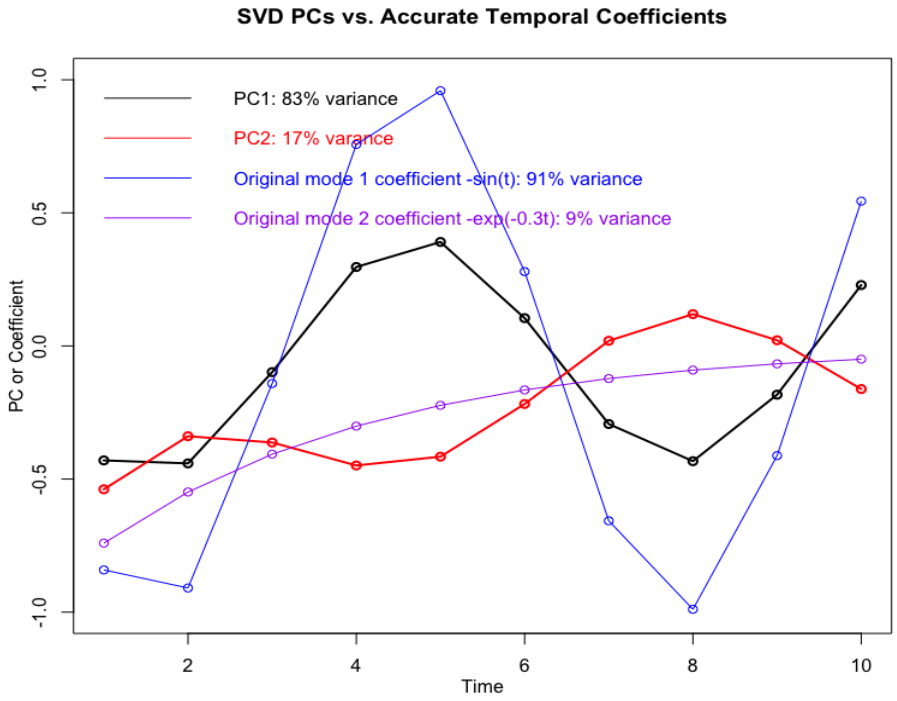

Figure 4.3 Two principal components (PCs) from SVD approximation and two accurate time coefficients: $-\sin t$ and $-\exp(-0.3t)$.

PC1 demonstrates sinusoidal oscillation, while PC2 shows an wavy increase. These two temporal patterns are similar to the original time coefficients $-\sin(t)$ and $-\exp(-0.3t)$. Here, the negative signs are added to make the EOF patterns have the same sign as the original basis functions, because the EOFs are determined up to the sign. The variances accounted by the original time coefficients are 91% and 9%, computed below by R:

```
t=1:10
t(-sin(t))%*%(-exp(-0.3*t))
v1=var(-sin(t))
v2=var(-exp(-0.3*t))
v1/(v1+v2)
#[1] 0.9098719
```

PC1 and PC2 are orthogonal, but coefficients $c_1(t) = sin(t)$ and $c_2(t) = -exp(0.3 * t)$ are not orthogonal. These can be verified by the following code.

```
#Verify orthogonality of PCs
t(vda2[,1])%*%vda2[,2]
#   [1,] -5.551115e-17
t=1:10
```

```
t(-sin(t))%*%(-exp(-0.3*t))
#[1,] 0.8625048
```

The SVD theory means that the original data can be recovered from EOFs, PCs and the variances by the following formula

$$z = UDV^t. \tag{4.10}$$

Because the eigenvalues, except the first two, for this problem are close to be zero, the reconstruction can be very accurate by using only the first two EOFs, PCs and their corresponding eigenvalues. The R code for both 2-mode approximation and all-mode recovery are below.

```
B<-uda2[,1:2]%*%diag(dda2)[1:2,1:2]%*%t(vda2[,1:2])
B1<-uda2%*%diag(dda2)%*%t(vda2)
```

Figure 4.4 shows the recovered z at time $t = 5$ using only two EOF modes, and can be plotted by the following R code.

```
plot.new()
filled.contour(x,y,matrix(B[,5],nrow=100), color.palette =rainbow,
               plot.title=title(main="2-mode SVD reduced field t=5",
                                xlab="x", ylab="y", cex.lab=1.0),
               key.title = title(main = "Scale"),
               plot.axes =  {axis(1,seq(0,3*pi, by = 1), cex=1.0)
                 axis(2,seq(0, 2*pi, by = 1), cex=1.0)})
```

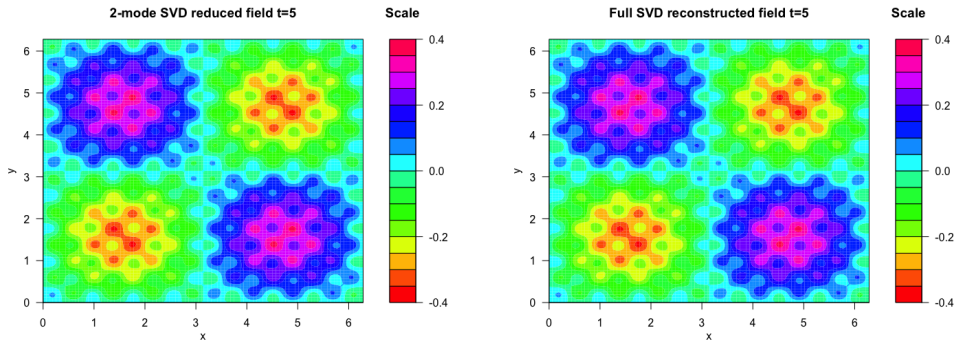

Figure 4.4 Recovery of the original data using two modes (the left panel) and using all modes (the right panel).

The full recovery or the original field at time $t = 5$, shown in the right panel of Fig. 4.4, has virtually no difference with the 2-mode approximation. The difference is less than 10^{-10} at any given points. This high level of accuracy may not always be achieved even with the full recovery UDV^t when high spatial variability appears. Namely, the full recovery UDV^t may have truncation errors, which can cause large errors when high spatial variability is involved.

4.3 From climate data download to EOF and PC visualization

This section presents an example of computing EOFs and PCs from a netCDF file downloaded from internet.

Climate research and teaching often involve the process of downloading data and making EOF analysis. We present an example to illustrate this process. Our example is for the popular surface air temperature (SAT) data from the NCEP/NCAR Reanalysis I, which outputs the 2.5 degree monthly data from January 1948 to present.

4.3.1 Download and visualize the NCEP temperature data

We downloaded the monthly surface air temperature (SAT) data from `https://www.esrl.noaa.gov/psd/data/gridded/data.ncep.reanalysis.surface.html`. The datafile is called `air.month.mean.nc`, and has 26MB from January 1948-October 2016. The spatial resolution is a 2.5 lat-lon grid.

```
# Download netCDF file
# Library
install.packages("ncdf")
library(ncdf4)

# 4 dimensions: lon,lat,level,time
nc=ncdf4::nc_open("/Users/sshen/Desktop/Papers/KarlTom/
Recon2016/Test-with-Gregori-prec-data/air.mon.mean.nc")
nc
nc$dim$lon$vals #output lon values 0.0->357.5
nc$dim$lat$vals #output lat values 90->-90
nc$dim$time$vals #output time values in GMT hours: 1297320, 1298064
nc$dim$time$units
#[1] "hours since 1800-01-01 00:00:0.0"
#nc$dim$level$vals
Lon <- ncvar_get(nc, "lon")
Lat <- ncvar_get(nc, "lat")
Time<- ncvar_get(nc, "time")
#Time is the same as nc$dim$time$vals
head(Time)
#[1] 1297320 1298064 1298760 1299504 1300224 1300968
library(chron)
Tymd<-month.day.year(Time[1]/24,c(month = 1, day = 1, year = 1800))
#c(month = 1, day = 1, year = 1800) is the reference time
Tymd
#$month
#[1] 1
#$day
#[1] 1
#$year
#[1] 1948
#1948-01-01
precnc<- ncvar_get(nc, "air")
```

```
dim(precnc)
#[1] 144  73 826, i.e., 826 months=1948-01 to 2016-10, 68 years 10 mons
```

To verify whether our downloaded data are reasonable, we plot the first month's temperature data at longitude 180°E from the south pole to the north pole (see Fig. 4.5). The figure shows a reasonable temperature distribution: a high temperature nearly 30°C over the tropics, a lower temperature below -30°C over the Antarctic region, and between -20°C and -10°C over the Arctic region. We are reasonably confident that our downloaded data are correct and have the right correspondence to the latitude-longitude grid.

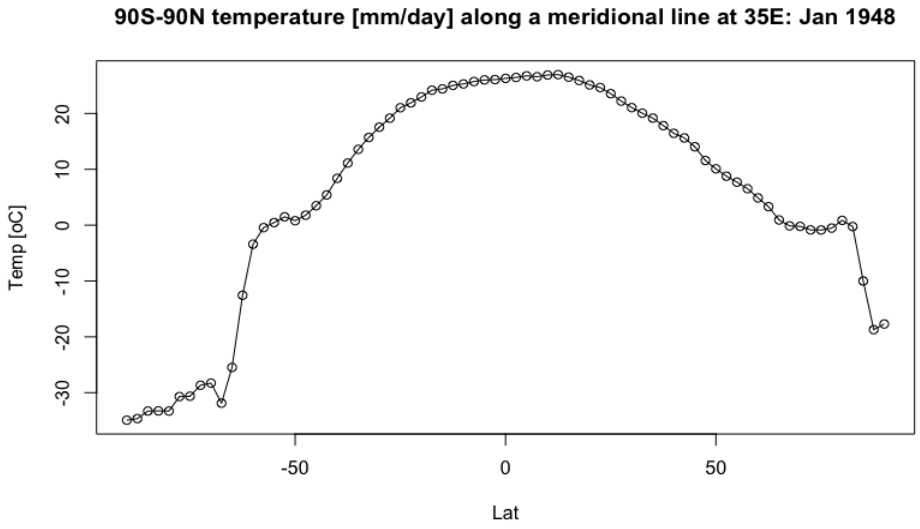

Figure 4.5 The north-south distribution of SAT along the meridional line at 180°E for January 1948.

Figure 4.5 may be generated by the following R code:

```
#plot a 90S-90N temp along a meridional line at 180E
plot(seq(-90,90,length=73),precnc[72,,1], type="l",
xlab="Lat", ylab="Temp [oC]",
main="90S-90N temperature [mm/day] along a meridional line at 35E: Jan 1948")
```

4.3.2 Space-time data matrix and SVD

4.3.2.1 Reformat the data into a space-time data matrix We convert the downloaded nc file into a space-time matrix and write it into a csv file which is easy for users to read and apply their simple computer programs. The R code for this procedure is below.

```
#Write the data as space-time matrix with a header
precst=matrix(0,nrow=10512,ncol=826)
temp=as.vector(precnc[,,1])
head(temp)
```

```
for (i in 1:826) {precst[,i]=as.vector(precnc[ , , i])}
dim(precst)
#[1] 10512   826
#Build lat and lon for 10512 spatial positions using rep
LAT=rep(Lat, 144)
LON=rep(Lon[1],73)
for (i in 2:144){LON=c(LON,rep(Lon[i],73))}
gpcpst=cbind(LAT, LON, precst)
dim(gpcpst)
#[1] 10512   828
#The first two columns are lat and lon. 826 mons: 1948.01-2016.10
#Convert the Julian day and hour into calendar mons for time
tm=month.day.year(Time/24, c(month = 1, day = 1, year = 1800))
tm1=paste(tm$year,"-",tm$month)
#tm1=data.frame(tm1)
tm2=c("Lat","Lon",tm1)
colnames(gpcpst) <- tm2
setwd("/Users/sshen/Desktop/MyDocs/teach/
SIOC290-ClimateMath2016/Rcodes/Ch12-RGraphics")
#setwd routes the desired csv file to a given directory
write.csv(gpcpst,file="ncepJan1948_Oct2016.csv")
```

The resulted csv file's first column is latitude, the second longitude, the first row is time mark for each month from January 1948 to October 2016. The csv file can be read and computed by Excel, R, Matlab, Python, and almost all the computer programs.

4.3.2.2 Climatology and standard deviation Here we show a way to compute the climatology and standard deviation using the space-time data matrix. With this matrix, computing the climatology and standard deviation becomes very easy. Graphically showing the spatial distribution of climatology and standard deviation (see Fig.3.9) can further verify the correctness of the downloaded data, such as the cold temperature over the Himalayas and Tibetan Plateau regions over Asia, and the region of Andes over South America. These regions have relatively low latitude but have very low temperature due to their altitude or more 4,000 meters.

The R code for computing the January climatology and standard deviation is below.

```
monJ=seq(1,816,12) #Select each January
gpcpdat=gpcpst[,3:818]
gpcpJ=gpcpdat[,monJ]
climJ<-rowMeans(gpcpJ)
library(matrixStats)# rowSds command is in the matrixStats package
sdJ<-rowSds(gpcpJ)
```

The climatology and standard deviation are shown in Fig.3.9, which can be plotted by the following R code.

```
#Plot Jan climatology
library(maps)
Lat1=seq(90,-90, len=73)
Lat=-Lat1
mapmat=matrix(climJ,nrow=144)
```

```
mapmat= mapmat[,seq(length(mapmat[1,]),1)]
plot.new()
int=seq(-50,50,length.out=81)
rgb.palette=colorRampPalette(c('black','blue', 'darkgreen','green',
                               'white','yellow','pink','red','maroon'),interpolate='spline
filled.contour(Lon, Lat, mapmat, color.palette=rgb.palette, levels=int,
               plot.title=title(main="NCEP Jan SAT RA 1948-2015 climatology [deg C]",
                                xlab="Longitude", ylab="Latitude"),
               plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()},
               key.title=title(main="[oC]"))
#--------------------
#Plot Jan Standard Deviation
Lat=-Lat1
mapmat=matrix(sdJ,nrow=144)
mapmat= mapmat[,seq(length(mapmat[1,]),1)]
plot.new()
mapmat=pmin(mapmat,5) #Compress the values >5 to 5
int=seq(0,5,length.out=81)
rgb.palette=colorRampPalette(c('black','blue', 'green',
                               'yellow','pink','red','maroon'),interpolate='spline')
filled.contour(Lon, Lat, mapmat, color.palette=rgb.palette, levels=int,
               plot.title=title(main="NCEP Jan SAT RA 1948-2015 Standard Deviation [deg C]
                                xlab="Longitude", ylab="Latitude"),
               plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()},
               key.title=title(main="[oC]"))
```

The very large standard deviation, more than $5°$C over the high latitude Northern Hemisphere shown in Fig.3.9 may be artificially amplified by the climate model that was used to do the NCEP/NCAR reanalysis. This may be due to the model's handling of sea ice and albedo feedback. The actually standard deviation might be smaller over the same region.

4.3.2.3 Plot EOFs, PCs, and variances The EOFs, PCs, and variances for a month can be easily calculated by SVD for the space-time matrix for the given month, by the following R code:

```
#Compute the Jan EOFs
monJ=seq(1,816,12)
gpcpdat=gpcpst[,3:818]
gpcpJ=gpcpdat[,monJ]
climJ<-rowMeans(gpcpJ)
library(matrixStats)
sdJ<-rowSds(gpcpJ)
anomJ=(gpcpdat[,monJ]-climJ)/sdJ #standardized anomalies
anomAW=sqrt(cos(gpcpst[,1]*pi/180))*anomJ #Area weighted anormalies
svdJ=svd(anomAW)  #execute SVD
```

The eigenvalues of a covariance matrix are variances, and the SVD eigenvalues from a data matrix correspond to standard deviations. Climate science often uses variance to measure the strength of a climate signal. We assess the relative importance of a variance by the percentage of a mode's variance relative to the sum of the variances of all the modes.

We thus plot the percentage of the square of each SVD eigenvalue, and the cumulative percentage from the first mode to the last mode. See Fig. 4.6 for the plot. The figure can be plotted by the following R code.

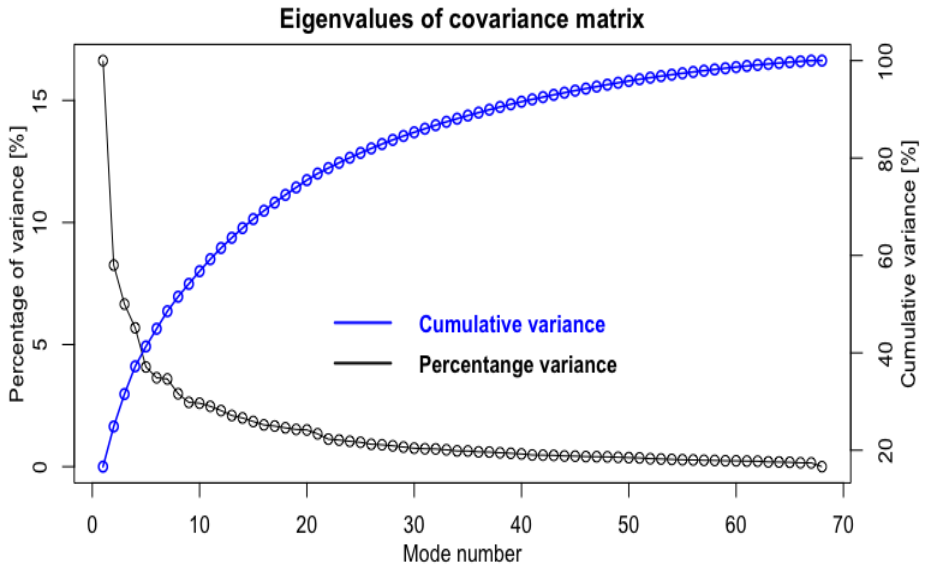

Figure 4.6 Percentage variances and their cumulative of the covariance matrix of the January SAT from 1948-2015.

```
#plot eigenvalues
par(mar=c(3,4,2,4))
plot(100*(svdJ$d)^2/sum((svdJ$d)^2), type="o", ylab="Percentage of variance [%]",
     xlab="Mode number", main="Eigenvalues of covariance matrix")
legend(20,5, col=c("black"),lty=1,lwd=2.0,
       legend=c("Percentange variance"),bty="n",
       text.font=2,cex=1.0, text.col="black")
par(new=TRUE)
plot(cumsum(100*(svdJ$d)^2/sum((svdJ$d)^2)),type="o",
col="blue",lwd=1.5,axes=FALSE,xlab="",ylab="")
legend(20,50, col=c("blue"),lty=1,lwd=2.0,
       legend=c("Cumulative variance"),bty="n",
       text.font=2,cex=1.0, text.col="blue")
axis(4)
mtext("Cumulative variance [%]",side=4,line=2)
```

The EOFs are from the column vectors of the SVD's U matrix and the PCs are the SVD's V matrix' columns. The first three EOFs and PCs are shown in Figs. 4.7-4.9, which may be generated by the following R code.

```
#plot EOF1: The physical EOF= eigenvector divided by area factor
mapmat=matrix(svdJ$u[,1]/sqrt(cos(gpcpst[,1]*pi/180)),nrow=144)
rgb.palette=colorRampPalette(c('blue','green','white',
```

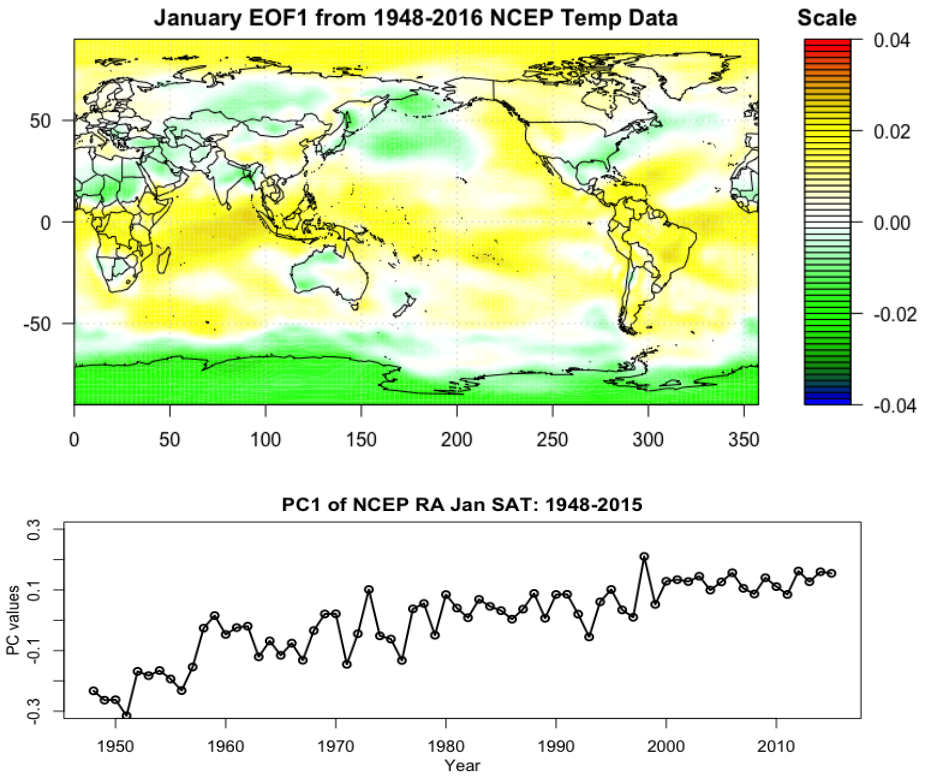

Figure 4.7 The first EOF and PCs= from the January SAT's standardized area-weighted anomalies.

```
               'yellow','red'),interpolate='spline')
int=seq(-0.04,0.04,length.out=61)
mapmat=mapmat[, seq(length(mapmat[1,]),1)]
filled.contour(Lon, Lat, -mapmat, color.palette=rgb.palette, levels=int,
 plot.title=title(main="January EOF1 from 1948-2016 NCEP Temp Data"),
               plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()},
               key.title=title(main="Scale"))
#
#plot PC1
pcdat<-svdJ$v[,1]
Time<-seq(1948,2015)
plot(Time, -pcdat, type="o", main="PC1 of NCEP RA Jan SAT: 1948-2015",
xlab="Year",ylab="PC values",
     lwd=2, ylim=c(-0.3,0.3))
```

Usually, the first 2-3 EOFs and PCs have some physical interpretations. In this case, PC1 shows an increasing trend. The corresponding EOF1 shows the spatially non-uniform pattern of temperature increasing.

EOF2 shows an El Nino pattern, with a warm tongue over the eastern tropical Pacific. PC2 has the El Nino correspondence. For example, the January 1983 and 1998 peaks correspond to two strong El Ninos.

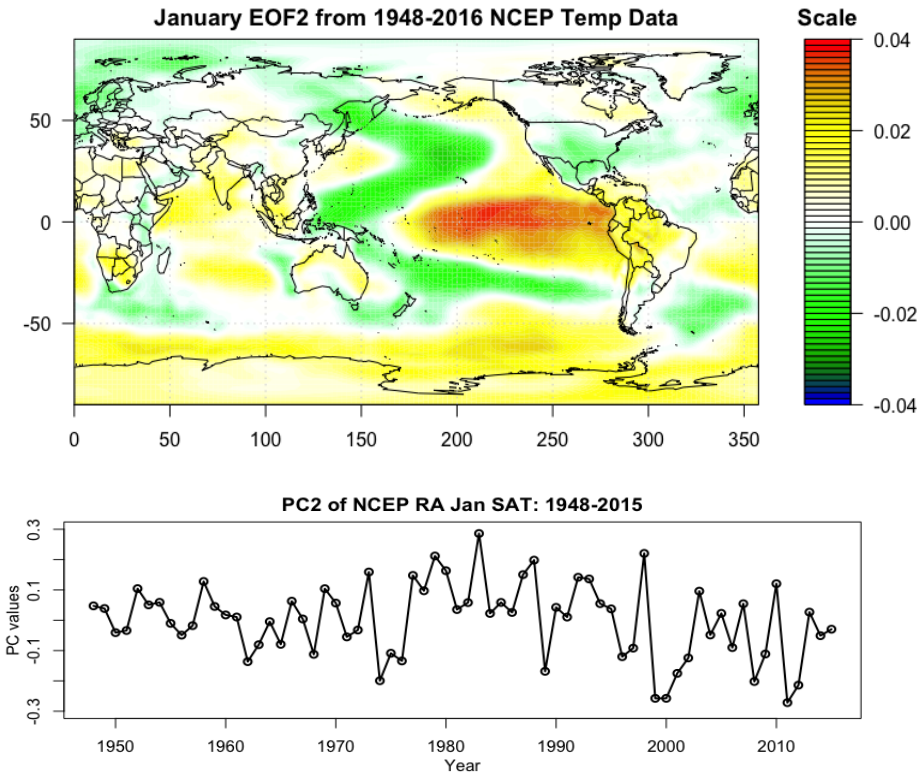

Figure 4.8 The second EOF and PCs= from the January SAT's standardized area-weighted anomalies.

EOF3 appears corresponding to the Pacific Decadal Oscillation. It shows a dipole pattern over the Northern Pacific. PC3 shows quasi-periodicity about 20-30 years.

4.3.2.4 EOFs from the de-trended standardized data We can de-trend the standardized anomaly data first and then compute the EOFs and PCs. As expected, the new EOF1 is not be the trend pattern anymore, rather it is the El Nino mode, i.e., the EOF2 of the non-detrended anomalies (see Figs. 4.10 and 4.11).

The de-trend and SVD procedures can carried out by the following R code.

```
#EOF from de-trended data
monJ=seq(1,816,12)
gpcpdat=gpcpst[,3:818]
gpcpJ=gpcpdat[,monJ]
climJ<-rowMeans(gpcpJ)
library(matrixStats)
sdJ<-rowSds(gpcpJ)
anomJ=(gpcpdat[,monJ]-climJ)/sdJ
trendM<-matrix(0,nrow=10512, ncol=68)#trend field matrix
trendV<-rep(0,len=10512)#trend for each grid box: a vector
for (i in 1:10512) {
 trendM[i,] = (lm(anomJ[i,] ~ Time))$fitted.values
```

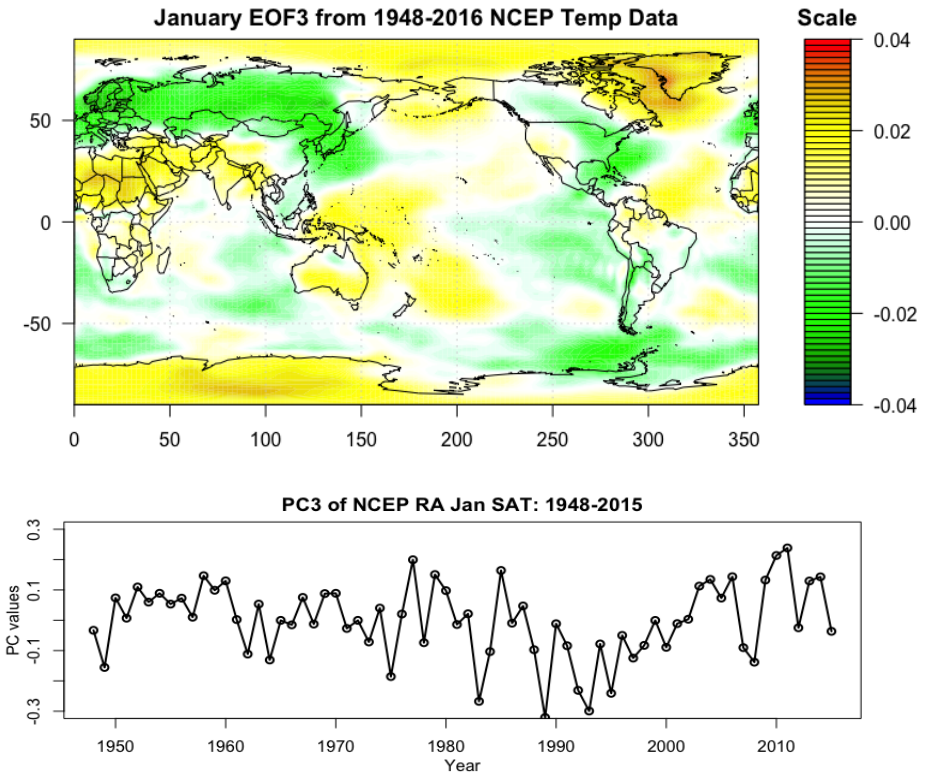

Figure 4.9 The third EOF and PC from the January SAT's standardized area-weighted anomalies.

```
 trendV[i]<-lm(anomJ[i,] ~ Time)$coefficients[2]
}
dtanomJ = anomJ - trendM
dim(dtanomJ)
dtanomAW=sqrt(cos(gpcpst[,1]*pi/180))*dtanomJ
svdJ=svd(dtanomAW)
```

One can then use the same EOF plotting code in the above sub-subsection to make the plots of eigenvalues, EOFs, and PCs. Comparing with the EOFs and PCs of the previous sub-sub-section, it is clear that the de-trended EOF1 here is similar to non-de-trended EOF2. However, they are not exactly the same. Thus, the de-trend process is an approximately the same as the EOF1 filtering, although not exactly the same. A similar statement can be made for other EOFs and PCs.

The first eigenvalue of the de-trended anomalies explains about 10% of variance, equivalent to the 8% variance explained by EOF2 of the non-de-trended anomalies (see Fig. 4.6). This can be derived from the non-de-trended SVD results. Let

$$c_i = 100\frac{d_i^2}{\sum_{i=1}^{K} d_i^2}, \quad i = i, 2, \cdots, K \tag{4.11}$$

be the percentage of variance explained by the ith mode, where K is the total number of modes available. In our case of January temperature from 1948-2015, $K = 68$. The SVD

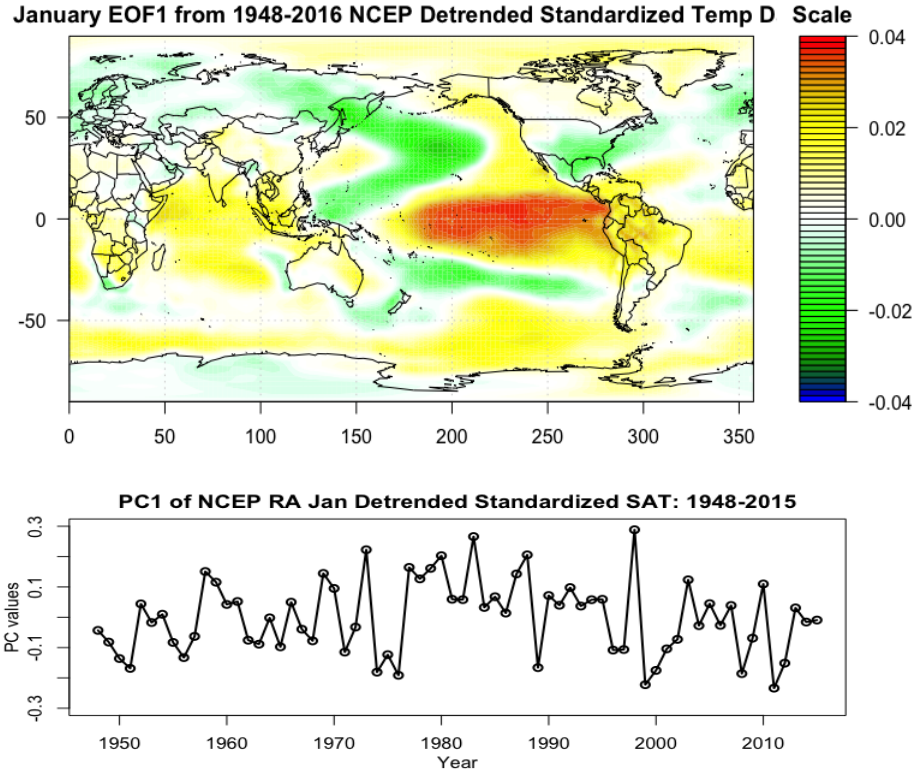

Figure 4.10 The first EOF and PC from the January SAT's de-trended standardized area-weighted anomalies.

calculation of the previous sub-sub-section found that

$$c_1 = 16.63[\%], c_2 = 8.25[\%]. \tag{4.12}$$

Figure 4.6 shows these values. From the following formula

$$c_2 = 100\frac{d_2^2}{\sum_{i=1}^{K} d_i^2} = 100\frac{d_2^2}{d_1^2 + \sum_{i=2}^{K} d_i^2}, \tag{4.13}$$

one can derive that

$$\begin{aligned} 100\frac{d_2^2}{\sum_{i=2}^{K} d_i^2} &= 100\frac{1}{1/c_2 - d_1^2/d_2^2} \\ = \quad 100\frac{c_2}{1-c_1} &= 100\frac{0.0825}{1-0.1663} = 9.9[\%] \end{aligned} \tag{4.14}$$

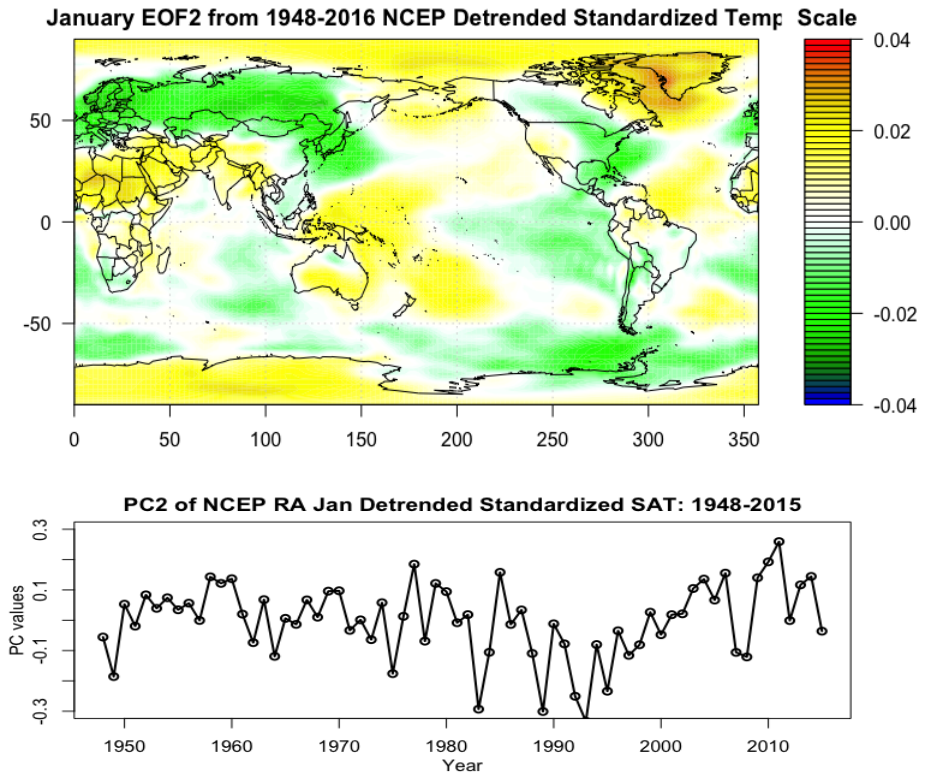

Figure 4.11 The second EOF and PC from the January SAT's de-trended standardized area-weighted anomalies.

4.4 Area-weighted average and spatial distribution of linear trends

4.4.1 Global average and PC1

To verify that PC1 of the non-de-trended anomalies represents the trend, we compute and plot the area-weighted SAT (see Fig. 4.12) from the NCEP/NCAR RA1 data using the following R code

```
#Plot the area-weighted global average Jan temp from 1948-2015
#Begin from the space-time data matrix gpcpst[,1]
vArea=cos(gpcpst[,1]*pi/180)
anomA=vArea*anomJ
dim(anomA)
JanSAT<-colSums(anomA)/sum(vArea)
plot(Time, JanSAT, type="o", lwd=2,
     main="Global Average Jan SAT Anomalies from NCEP RA",
     xlab="Year",ylab="Temperature [oC]")
regSAT<-lm(JanSAT ~ Time)
#0.61oC/100a trend
abline(regSAT, col="red", lwd=4)
```

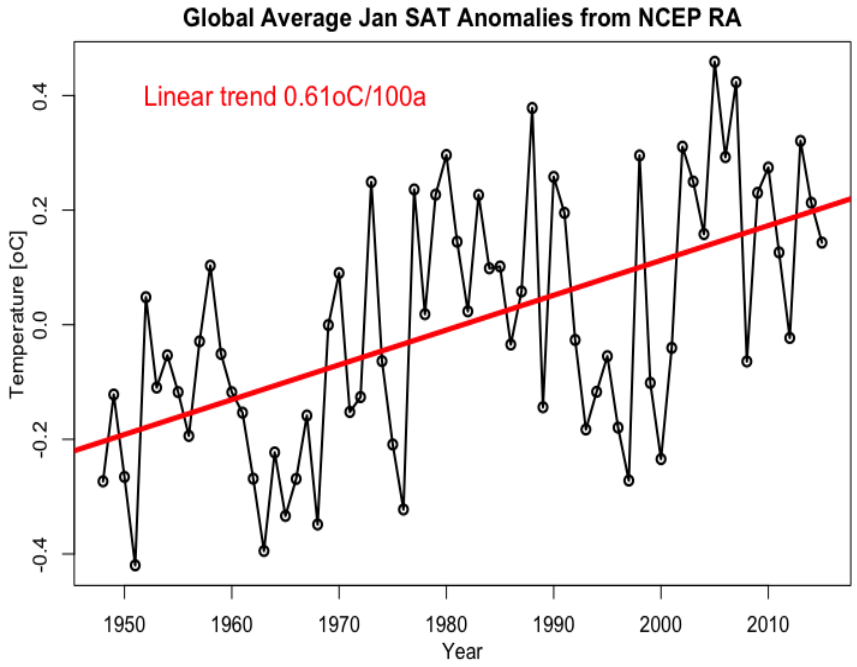

Figure 4.12 The global area-weighted January SAT anomalies from 1948-2015 based on the NCEP/NCAR RA1 data.

```
text(1965,0.4,"Linear trend 0.61oC/100a", col="red", cex=1.3)
```

Figures 4.12 and 4.7 clearly show that this global average is similar to PC1 of the non-de-trended data. Their trends are very close: 0.61°C/100a for the global average, and 0.54°C/100a for PC1. Their correlation is 0.69, not as strong as one may expect, because the global average has more extremes and large temporal variances. PC1 may be understood as the nonlinear trend of large scale spatial patterns. In other words, PC1 and EOF1 may be regarded as a low frequency and small wave number filter of the temperature anomaly field.

4.4.2 Spatial pattern of linear trends

Next we compute and plot the temperature trend from 1948-2015 based on the NCEP/NCAR Reanalysis' January temperature data. Each grid box has a time series of 68 years January SAT anomalies from 1948-2015 and has a linear trend. These trends form a spatial pattern (see Fig. 4.13, which is similar to that of EOF1 for the non-de-trended SAT anomaly data. The trend value for each grid box is computed by R's linear model command: `lm(anomJ[i,] ~ Time)$coefficients[2]`. The anomaly data are assumed to be written in the space-time data matrix `gpcpst` with the first two columns as latitude and longitude. The following R code makes the trend calculation and plots the figure.

```
#plot the trend of Jan SAT non-standardized anomaly data
#Begin with the space-time data matrix
```

```
monJ=seq(1,816,12)
gpcpdat=gpcpst[,3:818]
gpcpJ=gpcpdat[,monJ]
plot(gpcpJ[,23])
climJ<-rowMeans(gpcpJ)
anomJ=(gpcpdat[,monJ]-climJ)
trendV<-rep(0,len=10512)#trend for each grid box: a vector
for (i in 1:10512) {
  trendV[i]<-lm(anomJ[i,] ~ Time)$coefficients[2]
}
mapmat1=matrix(10*trendV,nrow=144)
mapv1=pmin(mapmat1,1) #Compress the values >5 to 5
mapmat=pmax(mapv1,-1) #compress the values <-5 t -5
rgb.palette=colorRampPalette(c('blue','green','white', 'yellow','red'),
interpolate='spline')
int=seq(-1,1,length.out=61)
mapmat=mapmat[, seq(length(mapmat[1,]),1)]
filled.contour(Lon, Lat, mapmat, color.palette=rgb.palette, levels=int,
               plot.title=title(
                 main="Trend of the NCEP RA1 Jan 1948-2015 Anom Temp",
                 xlab="Latitude",ylab="Longitude"),
               plot.axes={axis(1); axis(2);map('world2', add=TRUE);grid()},
               key.title=title(main="oC/10a"))
```

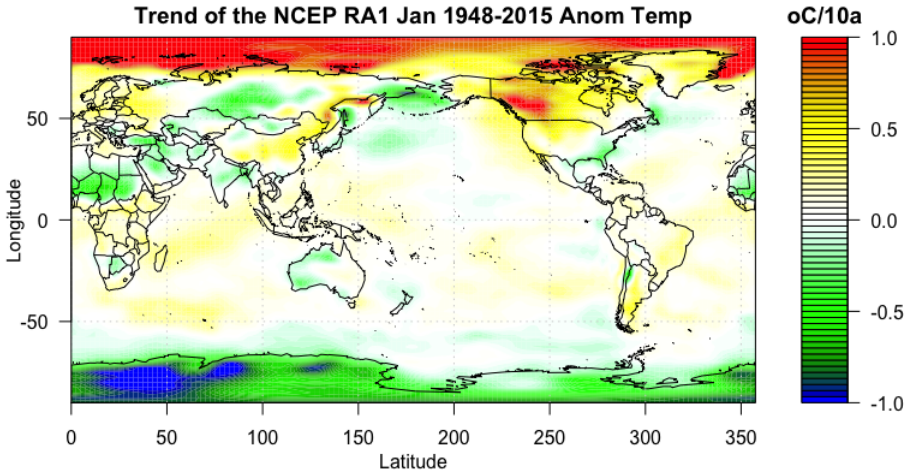

Figure 4.13 Linear trend of NCEP Reanalysis' January SAT from 1948-2015.

Figure 4.13 shows that the trends are non-uniform. The trend magnitudes over land are larger than ocean. The largest positive trends are over the Arctic region, while the Antarctic region has negative trends. However, these trends over the polar regions may not be reliable since the reanalysis climate model has an amplified variance over the polar regions.

Indeed, the spatial distribution of the trends appears similar to EOF1 of the non-detrended data (see Fig. 4.7). The spatial correlation between the trend map of Fig. 4.7

and the EOF1 map in Fig. 4.7 is very high and is equal to 0.97. This result implies that temperature's spatial patterns are more coherent than temporal patterns, consistent with the existence of large spatial correlation scales for the monthly temperature.

CHAPTER 5

MATRICES, MATRIX ALGEBRA, AND MULTIVARIATE REGRESSION

A matrix is a table such as that shown in Fig. 5.1, consisting of N-rows and Y-columns of numbers, which are called elements. Figure 5.1 shows the 10 years' annual precipitation anomalies from 1900-1909 for the 5 lat-lon degree boxes centered at $2.5^\circ E$ for different latitude over the Northern Hemisphere from $2.5^\circ N$ to $72.5^\circ N$.

Lat	Lon	1900	1901	1902	1903	1904	1905	1906	1907	1908	1909
2.5	2.5	0.283240	-0.131860	-0.190500	0.160040	-0.878110	0.080356	0.059193	-0.136900	0.200420	0.822600
7.5	2.5	0.172670	0.830550	-0.180350	-0.203630	-0.238590	0.425310	0.002805	0.102780	0.254050	0.516200
12.5	2.5	0.024392	0.152030	-0.034115	-0.062696	-0.192070	0.074360	0.201970	-0.011311	0.035259	0.272010
17.5	2.5	0.006780	0.066783	-0.084581	-0.008636	-0.038109	-0.001092	0.088250	0.011047	0.029358	0.082329
22.5	2.5	0.021162	0.079977	0.020016	-0.022142	-0.027032	0.065704	0.012937	-0.003823	0.032545	0.028636
27.5	2.5	0.049846	0.057413	0.026621	0.019914	-0.002651	0.071242	0.012837	0.001567	0.051857	0.099650
32.5	2.5	0.107740	0.143510	0.061613	0.076137	0.147760	0.137890	-0.074612	0.110300	0.087752	0.126920
37.5	2.5	0.128250	0.211940	0.113010	0.027472	0.183710	0.125550	-0.267500	0.215980	0.007609	0.055573
42.5	2.5	0.158490	0.800950	0.292690	0.172930	0.272010	0.126370	-0.017183	0.184880	0.118980	0.200520
47.5	2.5	-0.112800	0.243130	-0.121630	-0.076247	-0.047231	0.110160	0.080978	-0.091371	0.016172	-0.060487
52.5	2.5	-0.199840	-0.381070	-0.217570	-0.107760	-0.124700	-0.117470	-0.062448	-0.171070	-0.277650	-0.132690
57.5	2.5	-0.076619	-0.515070	0.005342	0.016647	0.137820	0.038041	0.131370	-0.196490	-0.132480	0.014887
62.5	2.5	-0.261760	-0.402600	0.137200	-0.214960	0.249210	0.147550	0.866120	-0.453910	-0.026134	0.053409
67.5	2.5	0.034079	0.223610	0.314090	-0.044832	0.130470	0.201260	0.554170	-0.054434	0.185870	0.308950
72.5	2.5	-0.119680	0.022949	0.004324	-0.050248	0.251330	-0.233080	-1.043800	0.363850	-0.315400	-0.113080

Figure 5.1 Annual precipitation anomalies data of the Northern Hemisphere at longitude $2.5^\circ E$ [unit: mm/day]. The annual total of the anomalies should be multiplied by 365.

Precipitation data [unit: mm/day] at multiple stations and multiple days also form a matrix, normally with stations [marked by station ID] counted in rows and time [unit: day] counted in columns. The daily minimum surface air temperature (Tmin) data for the same stations and the same period of time form another matrix. In general, the space-time climate data table always forms a matrix. Conventionally, the spatial location determines the row, and the time coordinate determines the column.

Another daily life matrix example is that the ages data of the audience sitting in a movie theater's rows and columns of chairs form a matrix. The audience weights form another matrix. Their bank account balance still another, and so on. So, matrix is a data table. Mathematical theories have been developed to study data matrices in the 20th century. Computer programming languages, such as R, MatLab, and Python, are very convenient and efficient in handling matrices.

A slightly higher level of matrix concept is the coefficient matrix of a group of linear equations. Solving linear equations is very common in science and engineering. Any numerical solutions of a partial differential equation climate model will have to solve a set of linear equations. The famous Leontief's economic supply-demand balance model is a set of linear equations that can be written in the matrix form.

A simple elementary school kid's problem reads like this: The sum of two brothers' age is 20 years and their difference is 4. What are the ages of the brothers? One can easily guess that the older brother is 12 and the younger one is 8. An eight-year old kid can most likely figure this out. The idea can be extended to a more general form of linear equations. The linear equations for this problem are below:

$$\begin{aligned} x_1 + x_2 &= 20 \\ x_1 - x_2 &= 4, \end{aligned} \tag{5.1}$$

where x_1 and x_2 stand for the brothers ages.

The matrix form of these equations would be

$$\mathbf{Ax} = \mathbf{b} \tag{5.2}$$

which involves three matrices:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} 20 \\ 4 \end{bmatrix}. \tag{5.3}$$

Most linear algebra textbooks introduce matrix in this way by describing a group of linear equations, which is less intuitive for in the field of climate science and policy, which emphasizes data.

The single column n-row matrix is often called an n-dimensional vector.

Although one can easily guess that the solution to the matrix equation is $x_1 = 12$ and $x_2 = 8$, a more consistent computing may be done by R using the following commands

```
A<-matrix(seq(1:4),2)
b<-seq(1:2)
A[1,1]=1
A[1,2]=1
A[2,1]=1
A[2,2]=-1
b[1]=20
```

```
b[2]=4
solve(A,b)
#[1] 12  8 This is the result x1=12, and x2=8.
```

This kind of R computer program can solve much more complicated linear equations, such as an equation of 1,000 unknowns rather than 2 in this example.

This solution may be represented as

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}, \tag{5.4}$$

where $\mathbf{A}^{-1}$ is the inverse matrix of $\mathbf{A}$, i.e.,

$$\mathbf{A}^{-1}\mathbf{A} = \mathbf{I} \tag{5.5}$$

where

$$\mathbf{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \tag{5.6}$$

is called identity matrix, which is like value 1.0 in our commonly used real number system.

This chapter will discuss following topics: (i) matrix algebra of addition, subtraction, multiplication and division (i.e., inverse matrix) and linear transform), (ii) space-time decomposition of a space-time climate data matrix, (iii) interpretation of the space-time decomposition using empirical orthogonal functions (EOFs) and principal components (PCs), (iv) matrix application in balancing the mass in chemical reaction equation, and (v) the matrix application in multivariate linear regression.

5.1 Matrix algebra and echelon form of a matrix

5.1.1 Matrix algebra

Addition, subtraction, multiply a matrix by a scalar constant, multiply a matrix by a matrix, a matrix divided by a matrix, and matrix inverse.

5.1.2 Independent row vectors and row echelon form

Example 1:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \tag{5.7}$$

Example 2: 5-deg global precipitation data

`http://www.ats.ucla.edu/stat/r/library/matrix_alg.htm`

5.2 Eigenvalues and eigenvectors of a square space matrix

Eigenvalue is a German word, meaning “self-value” or “proper value”. ”Eigen” means ”self”, ”my”.

A square matrix means that the number of rows is equal to the number of columns. The matrix is thus a square. Other matrices may be called rectangular matrices, or tall matrices.

Climate science often considers the covariance or correlation among N stations or grid boxes. The covariance matrix is thus a square matrix. The ij-th element is equal to the covariance of the i-th station with the j-th station. If Y is the time length, say Y years, of the anomaly data $A_{N\times Y}$, then the covariance matrix is

$$C = AA'/Y. \tag{5.8}$$

EXAMPLE 5.1

```
A=matrix(c(1,-1,2,0,3,1),nrow=2)
A
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]   -1    0    1
covm=(1/(dim(A)[2]))*A%*%t(A)
covm #is the covariance matrix.
          [,1]      [,2]
[1,] 4.6666667 0.6666667
[2,] 0.6666667 0.6666667
u=c(1,-1)
v=covm%*%x
v
     [,1]
[1,]    4
[2,]    0
#u and v are in different directions.
```

In general, a given square matrix C and a given vector x, the product Cx is usually not parallel to x, as shown in the above example. A special case is the matrix's self-vector: w whose Cw is parallel to w, i.e.,

$$Cw = \lambda w, \tag{5.9}$$

where λ is a scalar, which scales w so that the above equation holds and is called eigenvalue, and w is called eigenvector.

R can calculate the eigenvalues and eigenvectors of the above covariance matrix `covm` with a command

```
eigen(covm)
$values
[1] 4.7748518 0.5584816
$vectors
           [,1]       [,2]
[1,] -0.9870875  0.1601822
[2,] -0.1601822 -0.9870875
```

A 2-by-2 order covariance has two eigenvalues, and two eigenvectors: (λ_1, w_1) and (λ_2, w_2):

$$\lambda_1 = 4.7748518, \quad \mathbf{w_1} = \begin{bmatrix} -0.9870875 \\ -0.1601822 \end{bmatrix}, \tag{5.10}$$

$$\lambda_2 = 0.5584816, \quad \mathbf{w_2} = \begin{bmatrix} 0.1601822 \\ -0.9870875 \end{bmatrix}. \tag{5.11}$$

The eigenvectors are called modes, or empirical orthogonal functions (EOFs). The first a few eigenvectors of large climate covariance matrix often represent some typical climate dynamic patterns, such as El Nino Southern Oscillation (ENSO), North America Oscillation (NAO), and Pacific Decadal Oscillation (PDO).

It is usually that the first mode's components have the same sign, all positive or all negative. The second mode's components have half negative and half positive. Exceptions can happen.

The eigenvalues for a climate covariance matrix are always positive. The sum of all the eigenvalues represent the total variance of the climate system of these N stations.

However, in the climate data analysis, one can find the climate dynamic patterns as eigenvectors more directly from the anomaly data matrix A without computing the covariance matrix C explicitly. This is the singular value decomposition (SVD) approach that separates the space-time anomaly data into space part, time part, and variation energy part. This mathematical law of space-time decomposition is universally applicable to all data we sample in space-time and can help explore the insight physics or science recorded by the data. Efficient computing methods of SVD were extensively researched and developed since 1960s. The leading figure of the field was Gene H. Golub (1932-2007).

5.3 An SVD representation model for space-time data

We encounter space-time data every day, such as the air temperature at different locations at different time: the temperature at San Diego in the morning and that at New York at night after your arrival. We may need to examine the precipitation conditions around the world at different days to monitor the agricultural yield. Cellphone companies may need to monitor its market share and its temporal variations at different countries. A doctor may need to monitor a patient's temperature change at different parts: hands, feet, forehead, and mouth. The observed data form a space-time data matrix with the row position corresponding to the spatial location and the column position corresponding to time. See Table 5.1.

Graphically, the space-time data are usually plotted in time series according to each given spatial position, or a spatial map according to each given time. Although these straight forward graphical representation can sometimes provide very useful information for signal detection, such as abnormal conditions indicating a certain decease of a patient, the signals are often buried inside the data and need to be detected by different linear combinations in space and time. Sometimes the data matrix are very big, millions of dimension in either space or time. Then what is the essential information in this big data matrix? Can we distill the most important information and represent the data in a simpler way but more useful way? A very useful way is the space-time

Table 5.1 Space-time data table

	Time 1	Time 2	Time 3	Time 4
Space 1	D11	D12	D13	$D14$
Space 2	D21	D22	D23	$D24$
Space 3	D31	D32	D33	$D34$
Space 4	D41	D42	D43	$D44$
Space 5	D51	D52	D53	$D54$

separation. Singular value decomposition (SVD) is a designed for this purpose. SVD decomposes a space-time data matrix into a spatial pattern matrix U, a diagonal energy level matrix D, and a temporal matrix V', i.e., the data matrix A is decomposed into

$$A_{n\times t} = U_{n\times m} D_{m\times m} (V')_{m\times t}. \tag{5.12}$$

where n is the spatial dimension, t is the temporal length, $m = min(n, t)$, and V' the transpose of V.

EXAMPLE 5.2

```
#Develop a 2-by-3 space-time data matrix
A=matrix(c(1,-1,2,0,3,1))
A
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]   -1    0    1
#Perform SVD calculation
msvd=svd(A)
msvd
$d
[1] 3.784779 1.294390
$u
           [,1]       [,2]
[1,] -0.9870875 -0.1601822
[2,] -0.1601822  0.9870875
$v
           [,1]       [,2]
[1,] -0.2184817 -0.8863403
[2,] -0.5216090 -0.2475023
[3,] -0.8247362  0.3913356
#One can verify that A=UDV'.
verim=msvd$u%*%diag(msvd$d)%*%t(msvd$v)
verim
     [,1]         [,2] [,3]
[1,]    1 2.000000e+00    3
[2,]   -1 1.665335e-16    1
round(verim)
     [,1] [,2] [,3]
```

```
[1,]    1    2    3
[2,]   -1    0    1
#This is the original data matrix A
```

The covariance of the space-time matrix A is a spatial matrix:

$$C = \frac{1}{Y}AA', \tag{5.13}$$

where Y is the number of columns of A and is the time length.

```
covm=(1/(dim(A)[2])*A%*%t(A)
eigcov=eigen(covm)
eigcov
$values
[1] 4.7748518 0.5584816
$vectors
           [,1]       [,2]
[1,] -0.9870875  0.1601822
[2,] -0.1601822 -0.9870875
```

Thus, the covariance matrix' eigenvectors are the same as the SVD eigenvectors of the anomaly matrix. The eigenvalues of covariance matrix and the SVD have following relationship

```
((msvd$d)^2)/(dim(A)[2])=eigcov$values
[1] 4.7748518 0.5584816
```

This is formally stated and proved below.

Theorem 5.1 *The covariance's matrix $C = AA'/Y$'s eigenvectors are the same as the spatial modes:*

$$\mathbf{C}\mathbf{u}_k = \lambda_i \mathbf{u}_k, \ \ (k = 1, 2, \cdots, N), \tag{5.14}$$

and the eigenvalues $lambda_k$ of C and those d_k of A from SVD have the following relationship

$$\lambda_k = d_k^2/Y \ \ (k = 1, 2, \cdots, N), \tag{5.15}$$

where Y is the total time length (i.e., time dimension) of the anomaly data matrix A, and N is the total number of stations for A (i.e., space dimension): $A_{N\times Y}$ and $C_{N\times N}$

Proof:

$$\begin{aligned} A &= UDV' && (5.16)\\ C &= \frac{1}{Y}AA' = \frac{1}{Y}UDV'(UDV')' \\ &= \frac{1}{Y}UDV'(VDU') = \frac{1}{Y}UDDU' = \frac{1}{Y}D^2UU' && (5.17)\\ CU &= \frac{1}{Y}D^2UU'U = \frac{1}{Y}D^2U = \Lambda U && (5.18)\\ \Lambda &= \frac{1}{Y}D^2. && (5.19) \end{aligned}$$

In the above, D^2 is allowed to be moved in front of U because D is a diagonal matrix.

Thus

$$\lambda_k = \frac{d_k^2}{Y} \ \ (k = 1, 2 \cdots, N). \tag{5.20}$$

5.4 SVD analysis of Southern Oscillation Index

This section is an SVD approach to construct an optimally weighted Southern Oscillation Index (WSOI).

SOI is an index that monitors ENSO and is the standardized Tahiti (18S, 149W) sea level pressure (SLP) minus that of Darwin (12S, 131E). It measures the SLP difference between the eastern tropical Pacific and the western tropical Pacific. During a normal year, Darwin's anomaly pressure is lower than that of Tahiti, which maintains the easterlies trade wind and supports the western Pacific warm pool. When the wind reverses, the pressure anomalies have opposite order, leading to the westerlies trade wind and the accumulation of warm water over the eastern tropical Pacific or central tropical Pacific. This is El Nino.

The SLP data of these two stations can be downloaded from
`http://www.cpc.ncep.noaa.gov/data/indices/`

This section uses SVD approach to analyzing the standardized Tahiti and Darwin SLP and developing WSOI.

```
# Read the txt data
Pta<-read.table("~/Desktop/MyDocs/teach/336MathModel-2016SP/
BookMathModeling2016/R-code4MathModelBook/Ch5-SOI/PSTANDtahiti", header=F)
# Remove the first column that is the year
ptamon<-Pta[,seq(2,13)]
#Convert the matrix into a vector according to mon: Jan 1951, Feb 1951, ..., Dec 2015
ptamonv<-c(t(ptamon))
xtime<-seq(1951, 2016-1/12, 1/12)
# Plot the Tahiti standardized SLP anomalies
plot(xtime, ptamonv,type="l",xlab="Year",ylab="Presure",
 main="Standardized Tahiti SLP Anomalies", col="red",
 xlim=range(xtime), ylim=range(ptamonv))
# Do the same for Darwin SLP
Pda<-read.table("~/Desktop/MyDocs/teach/336MathModel-2016SP/
BookMathModeling2016/R-code4MathModelBook/Ch5-SOI/PSTANDdarwin.txt", header=F)
pdamon<-Pda[,seq(2,13)]
pdamonv<-c(t(pdamon))
plot(xtime, pdamonv,type="l",xlab="Year",ylab="Presure",
 main="Standardized Darwin SLP Anomalies", col="blue",
 xlim=range(xtime), ylim=range(pdamonv))
#Plot the SOI index
plot(xtime, ptamonv-pdamonv,type="l",xlab="Year",
ylab="SOI index", col="black",xlim=range(xtime), ylim=c(-4,4), lwd=1)
#Add ticks on top edge of the plot box
axis (3, at=seq(1951,2015,4), labels=seq(1951,2015,4))
#Add ticks on the right edge of the plot box
axis (4, at=seq(-4,4,2), labels=seq(-4,4,2))
# If put a line on a plot, use the command below
lines(xtime,ptamonv-pdamonv,col="black", lwd=1)
```

The accumulative SOI, denoted by CSOI, has a nonlinear trend similar to that of SST over North Atlantic (80W-0, 30-60N). See CPC report on Feb 9, 2016.

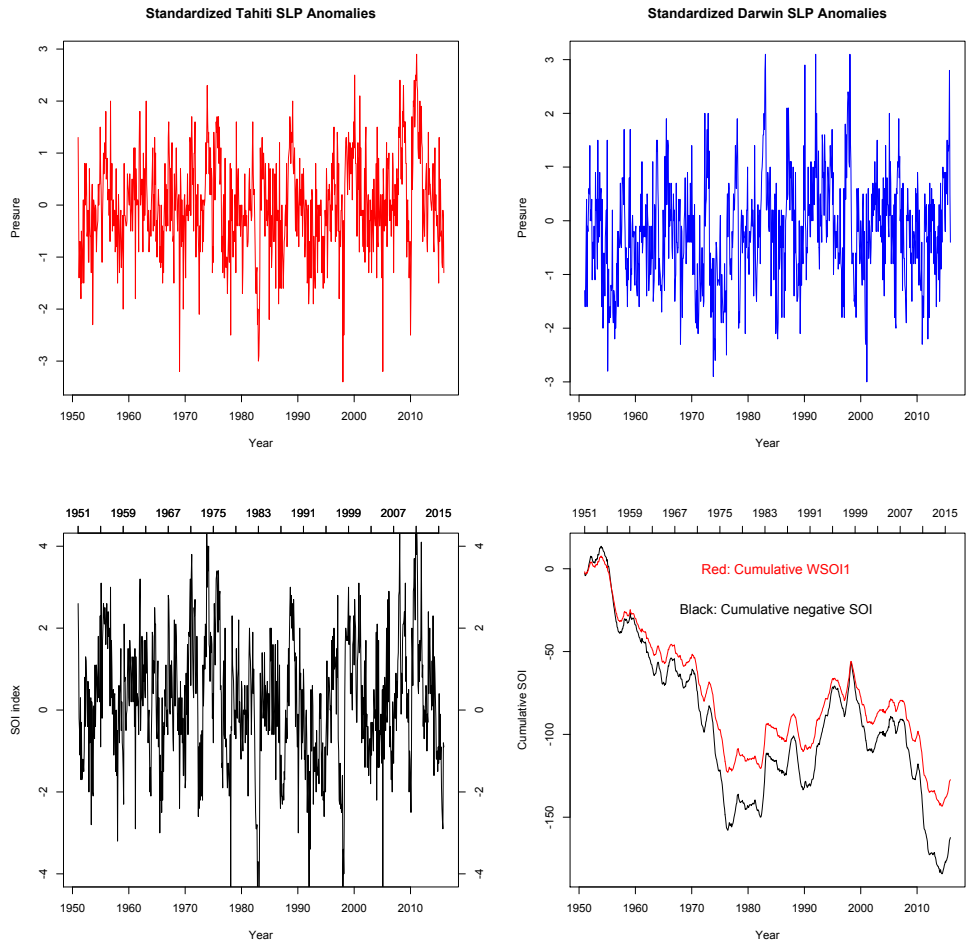

Figure 5.2 Standardized sea level pressure anomalies of Tahiti (up-left panel), that of Darwin (up-right). SOI time series (low-left), and the cumulative of the negative SOI time series (low-right).

```
cnegsoi<--cumsum(ptamonv-pdamonv)
plot(xtime, cnegsoi,type="l",xlab="Year",ylab="Negative CSOI index",
col="black",xlim=range(xtime), ylim=range(cnegsoi), lwd=1)
```

The space-time data matrix of the SLP at Tahiti and Darwin from January 1951-December 2015 can be obtained from

`ptada <-cbind(ptamonv,pdamonv)`

This is a matrix of two columns: the first column is the Tahiti SLP and the second the Darwin. Because normally the spatial position is by row and time by column, we transpose the matrix `ptada<-t(ptada)` This is the 1951-2015 standardized SLP data for Tahiti and Darwin: 2 rows and 780 columns.

```
dim(ptada)
[1]   2 780
```

Make the SVD space-time separation: `svdptd<-svd(ptada)`
Verify this separation by reconstructing the original space-time data matrix using the SVD results
`recontd=svdptd$u%*%diag(svdptd$d[1:2])%*%t(svdptd$v)`
One can verify that `recontd=ptada`.

The spatial matrix U is a 2×2 orthogonal matrix since there are only two points. Each column is an eigenvector of the covariance matrix $C = AA'/t$, where $A_{n\times t}$ is the original data matrix of n spatial dimension and t temporal dimension. These eigenvectors are spatial patterns, called empirical orthogonal function (EOF) in atmospheric sciences. Our U matrix is

```
U=svdptd$u
U
            [,1]      [,2]
[1,] -0.6146784 0.7887779
[2,]  0.7887779 0.6146784
```

The first column is the first spatial mode is $\mathbf{u}_1 = (-0.61, 0.79)$, meaning opposite signs of Tahiti and Darwin, which justifies the SOI index as one pressure minus another. This result further suggests that a better index should be the weighted SOI:

$$WSOI1 = -0.6147P_{Tahiti} + 0.7888P_{Darwin} \tag{5.21}$$

This mode's energy level, i.e., the temporal variance, is $d_1 = 31.35$ given by

```
svdptd$d
[1] 31.34582 22.25421
D=diag(svdptd$d)
D
         [,1]     [,2]
[1,] 31.34582  0.00000
[2,]  0.00000 22.25421
```

which forms the diagonal matrix D in the SVD formula. In the nature, the second eigenvalue is often much smaller than the first, but not this one. The second mode's energy level is $d_2 = 22.25$, equal to 71% of the first energy level.

The second weighted SOI mode, i.e. the second column $\mathbf{u}_2$ of U, is thus

$$WSOI2 = 0.7888P_{Tahiti} + 0.6147P_{Darwin} \tag{5.22}$$

From the SVD formula $A = UDV'$, the above two weighted SOIs are $U'A$:

$$U'A = DV', \tag{5.23}$$

because U is a an orthogonal matrix and $U^{-1} = U'$.

The V matrix is given by

```
V=svdptd$v
V
                 [,1]           [,2]
  [1,] -5.820531e-02  1.017018e-02
```

```
[2,] -4.026198e-02 -4.419324e-02
[3,] -2.743069e-03 -8.276652e-02
......
```

The first temporal mode $\mathbf{v}_1$ is the first row of V' and is called the first principal component (PC1). The above formulas imply that

$$\mathbf{v}_1 = WSOI1/d_1 \tag{5.24}$$

$$\mathbf{v}_2 = WSOI2/d_2 \tag{5.25}$$

The two PCs are orthonormal vectors. So are the two EOFs. Thus, the SLP data at Tahiti and Darwin have been decomposed into a set of spatially and temporally orthonormal vectors: EOFs and PCs, together with energy levels.

The WSOIs' standard deviations are d_1 and d_2, reflecting the WSOI's oscillation magnitude and frequency.

We also have the relations

$$d_k PC_k = WSOI_k \ \ (k = 1, 2). \tag{5.26}$$

The two WSOIs are shown in Fig.5.3.

```
#Plot WSOI1
xtime<-seq(1951, 2016-1/12, 1/12)
wsoi1=D[1,1]*t(V)[1,]
plot(xtime, wsoi1,type="l",xlab="Year",ylab="Weighted SOI 1",
col="black",xlim=range(xtime), ylim=range(wsoi1), lwd=1)
axis (3, at=seq(1951,2015,4), labels=seq(1951,2015,4))
#Plot WSOI2
wsoi2=D[2,2]*t(V)[2,]
plot(xtime, wsoi2,type="l",xlab="Year",ylab="Weighted SOI 2",
 col="black",xlim=range(xtime), ylim=c(-2,2), lwd=1)
axis (3, at=seq(1951,2015,4), labels=seq(1951,2015,4))
```

The cumulative WSOIs can be plotted by the following R commands

```
%Plot cumulative WSOI1
cwsoi1=cumsum(wsoi1)
plot(xtime, cwsoi1,type="l",xlab="Year",ylab="Cumulated weighted SOI 1",
col="black",xlim=range(xtime), ylim=range(cwsoi1), lwd=1)
axis (3, at=seq(1951,2015,4), labels=seq(1951,2015,4))
%Plot cumulative WSOI2
cwsoi2=cumsum(wsoi2)
plot(xtime, cwsoi2,type="l",xlab="Year",ylab="Cumulated weighted SOI 2",
col="black",xlim=range(xtime), ylim=range(cwsoi2), lwd=1)
axis (3, at=seq(1951,2015,4), labels=seq(1951,2015,4))
```

The cumulative WSOI1 appears to trace the southern hemisphere's surface air temperature history, according to Jones' data

`http://cdiac.ornl.gov/ftp/trends/temp/jonescru/sh.txt`

`http://cdiac.ornl.gov/trends/temp/jonescru/graphics/glnhsh.png`

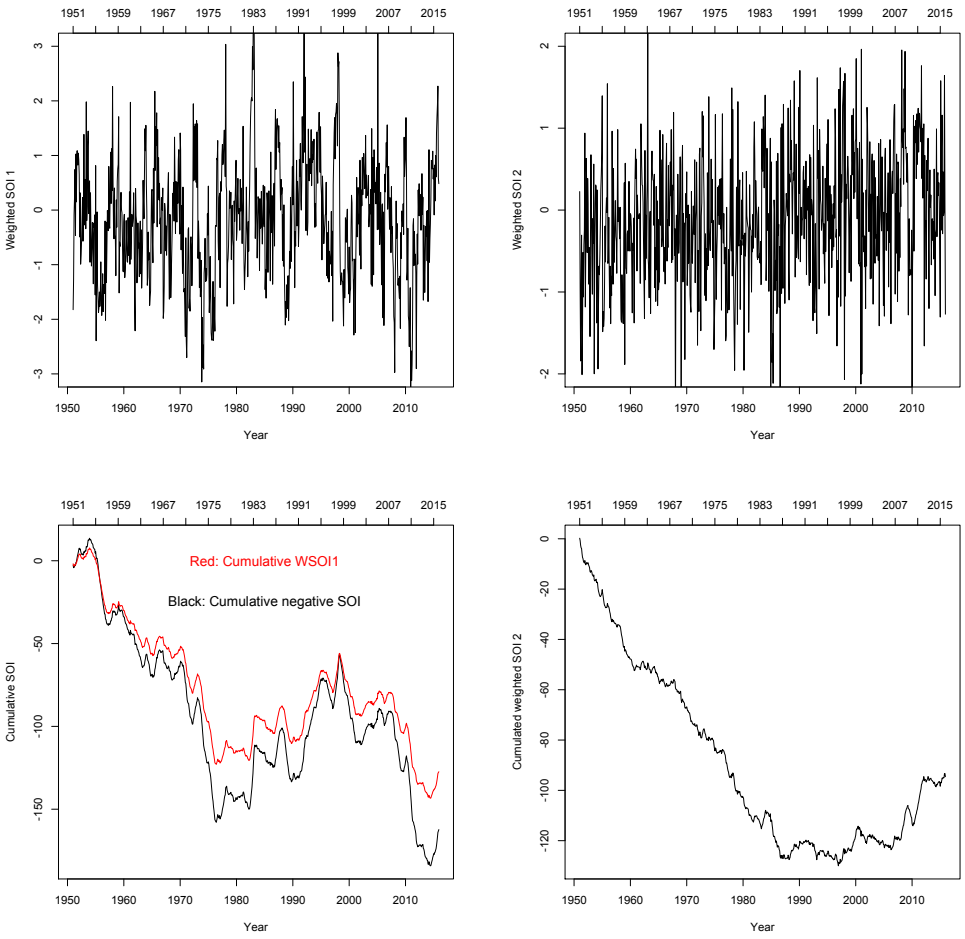

Figure 5.3 Weighted SOI1 (up-left panel), weighted SOI2 (up-right), cumulative WSOI1 (low-left), and cumulative WSOI2 (low-right).

When the cumulative WSOI decreases, so does the SH surface air temperature from 1951 to 1980. When the cumulative WSOI increases, so does the temperature from the 1980s to the peak 1998. Then cumulative WSOI1 decreases to a plateau from 1998 to 2002, another plateau until 2007, then decreases again. This also agrees with the SH surface air temperature trend.

Therefore, SVD results may lead to physical meanings and is a convenient tool to use.

An integration of the R codes above for SLP SVD is below.

```
 #SVD Analysis for the Darwin-Tahiti Standardized SLP
#By Sam Shen, August 2015, Revised May 2017

setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2016/Rcodes/Ch5-SOI")
```

```
Pta<-read.table("PSTANDtahiti", header=F)
# Remove the first column that is the year
ptamon<-Pta[,seq(2,13)]
#Convert the matrix into a vector according to mon: Jan 1951, Feb 1951, ..., Dec 2015
ptamonv<-c(t(ptamon))
#Generate time ticks from Jan 1951 to Dec 2015
xtime<-seq(1951, 2016-1/12, 1/12)
# Plot the Tahiti standardized SLP anomalies
plot(xtime, ptamonv,type="l",xlab="Year",ylab="Presure",
     main="Standardized Tahiti SLP Anomalies", col="red",
     xlim=range(xtime), ylim=range(ptamonv))
# Do the same for Darwin SLP
Pda<-read.table("PSTANDdarwin.txt", header=F)
pdamon<-Pda[,seq(2,13)]
pdamonv<-c(t(pdamon))
plot(xtime, pdamonv,type="l",xlab="Year",ylab="Presure",
     main="Standardized Darwin SLP Anomalies", col="blue",
     xlim=range(xtime), ylim=range(pdamonv))
#Plot the SOI index
SOI <- ptamonv-pdamonv
plot(xtime, SOI,type="l",xlab="Year",main="Southern Oscillation Index",
     ylab="SOI index", col="black",xlim=range(xtime), ylim=c(-6,6), lwd=1)
#Add ticks on top edge of the plot box
axis (3, at=seq(1951,2015,4), labels=seq(1951,2015,4))
#Add ticks on the right edge of the plot box
axis (4, at=seq(-4,4,2), labels=seq(-4,4,2))
lines(xtime, rep(0,length(xtime)))
abline(lm(SOI ~ xtime), col="red", lwd=2)

#Cumulative negative SOI
cnegsoi<--cumsum(SOI)
plot(xtime, cnegsoi,type="l",xlab="Year",ylab="Negative CSOI index",
     col="black",xlim=range(xtime), ylim=range(cnegsoi), lwd=2,
     main="Cumulative southern oscillation index")

#Space-time standardized SLP data matrix
ptada <-cbind(ptamonv,pdamonv)
ptada <- t(ptada)
dim(ptada)
#[1]   2 780

#SVD analysis of the space-time data
svdptd <- svd(ptada)
EOF = svdptd$u
EOF
#       [,1]      [,2]
#[1,] -0.6146784 0.7887779
#[2,]  0.7887779 0.6146784
#Weighted SOI
```

```
WSOI=t(EOF)%*%ptada
dim(WSOI)
#The two weighted SOI indices
WSOI1=WSOI[1,]
WSOI2=WSOI[2,]
plot(xtime, WSOI1,type="l",xlab="Year",
     main="Weighted Southern Oscillation Indices",
     ylab="WSOI index", col="black",xlim=range(xtime),
     ylim=c(-4,4), lwd=1)
legend(1945,5.5, lty=1, legend=c("WSOI1: 67% variance"),
       bty="n",col=c("black"),cex=1.3)
lines(xtime, WSOI2,type="l", col="blue")
legend(1945,4.7, lty=1, legend=c("WSOI2: 33% variance"),
       bty="n",col=c("blue"), cex=1.3)

#Energy for each mode
Energy=svdptd$d
Energy
#[1] 31.34582 22.25421
#Energy ratio
Energy^2/sum(Energy^2)

#Principal components
PC=svdptd$v
PC1=PC[,1]
PC2=PC[,2]
#Plot the PCs
plot(xtime, PC1,type="l",xlab="Year",
     main="Tahiti-Darwin Principal Components",
     ylab="PC index", col="black",xlim=range(xtime),
     ylim=c(-0.15,0.15), lwd=1)
legend(1945,0.20, lty=1, legend=c("PC1: 67% variance"),
       bty="n",col=c("black"),cex=1.3)
lines(xtime, PC2,type="l", col="blue")
legend(1945,0.17, lty=1, legend=c("PC2: 33% variance"),
       bty="n",col=c("blue"), cex=1.3)

#Plot cumulative PCs
plot(xtime, cumsum(PC1),type="l",xlab="Year",
     main="Cumulative Tahiti-Darwin Principal Components",
     ylab="Cumulative PC index", col="black",xlim=range(xtime),
     ylim=c(-6,1),lwd=1)
legend(1960,2, lty=1, legend=c("Cumulative PC1: 67% variance"),
       bty="n",col=c("black"),cex=1.3)
lines(xtime, cumsum(PC2),type="l", col="blue")
legend(1960,1.5, lty=1, legend=c("Cumulative PC2: 33% variance"),
       bty="n",col=c("blue"), cex=1.3)
```

5.5 Visualization of SVD results: EOFs and PCs

We use three examples the visualization of EOFs and PCs from SVD results by `ggplot`.

EXAMPLE 5.3

The space-time data matrix `ptada` of the SLP at Tahiti and Darwin from January 1951-December 2015 has 2 rows for space and 780 columns for time. The U matrix from the SVD is a 2×2 matrix. Its first column is the El Nino mode. Note that the eigenvectors are determined except a positive or negative sign. Because Tahiti has a positive SST anomaly during an El Nino, we thus choose Tahiti 0.61 and hence make Darwin -0.79. This is the negative first eigenvector from the SVD. The second mode is Tahiti 0.79 and Darwin 0.61. These two modes are orthogonal because $(-0.79, 0.61)\cdot(0.61, 0.79) = 0$ and are displayed in Fig. 5.4, which may be generated by the following R code.

```
#Display the two ENSO modes on a world map
library(maps)
library(mapdata)

plot.new()
par(mfrow=c(2,1))

par(mar=c(0,0,0,0)) #Zero space between (a) and (b)
map(database="world2Hires",ylim=c(-70,70), mar = c(0,0,0,0))
grid(nx=12,ny=6)
points(231, -18,pch=16,cex=2, col="red")
text(231, -30, "Tahiti 0.61", col="red")
points(131, -12,pch=16,cex=2.6, col="blue")
text(131, -24, "Darwin -0.79", col="blue")
axis(2, at=seq(-70,70,20),
     col.axis="black", tck = -0.05, las=2, line=-0.9,lwd=0)
axis(1, at=seq(0,360,60),
     col.axis="black",tck = -0.05, las=1, line=-0.9,lwd=0)
text(180,30, "El Nino Southern Oscillation Mode 1",col="purple",cex=1.3)
text(10,-60,"(a)", cex=1.4)
box()

par(mar=c(0,0,0,0)) #Plot mode 2
map(database="world2Hires",ylim=c(-70,70),  mar = c(0,0,0,0))
grid(nx=12,ny=6)
points(231, -18,pch=16,cex=2.6, col="red")
text(231, -30, "Tahiti 0.79", col="red")
points(131, -12,pch=16,cex=2, col="red")
text(131, -24, "Darwin 0.61", col="red")
text(180,30, "El Nino Southern Oscillation Mode 2",col="purple",cex=1.3)
axis(2, at=seq(-70,70,20),
     col.axis="black", tck = -0.05, las=2, line=-0.9,lwd=0)
axis(1, at=seq(0,360,60),
```

```
     col.axis="black",tck = -0.05, las=1, line=-0.9,lwd=0)
text(10,-60,"(b)", cex=1.4)
box()
```

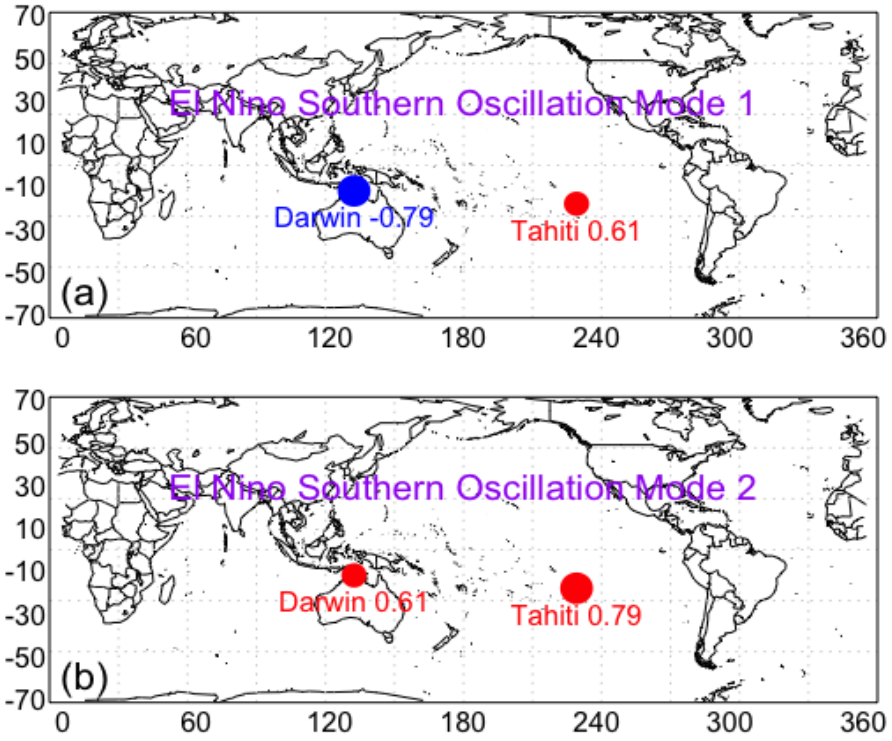

Figure 5.4 The two orthogonal ENSO modes from the Tahiti and Darwin standardized SLP data. The relative data sizes are proportional to the component values of each eigenvector in the U matrix. Red color means positive value, and blue negative value.

The EOFs and PC time series from the SVD can be computed and plotted in another way using the following R code

```
#Plot principal components of the Darwin and Tahiti SLP
tdsvd<-svd(ptada)
dat=provideDimnames(tdsvd$u,base=list(c("Tahiti", "Darwin"),
            +    c("EOF1","EOF2")))
ft=as.data.frame(dat)
dp=ggplot(ft,aes(lon,lat))
dp1=dp+geom_point(aes(colour=factor(EOF1)),cex=9)
+ xlim(-180,180) + ylim(-90,90)
dp1#Show the plot of EOF1
plot(seq(1951, 2015, length=780), tdsvd$v[1,],
  +  xlab="Year", ylab="WSOI1", col="red",
  + main="PC1 as the weighted SOI")
```

EXAMPLE 5.4

NASA Global Precipitation Climatology Project (GPCP) precipitation data.

EXAMPLE 5.5

NCEP/NCAR Reanalysis data: wind data.

Read the following website for more about the SVD R code:

https://stat.ethz.ch/R-manual/R-devel/library/base/html/svd.html

5.6 Multivariate linear regression using matrix notations

The linear regression in Chapter 4 discussed the fitting of $y = ax + b$ to a pair of data vectors: x_d, y_d. It resulted in correlation, trend and other regression quantities. An example is the correlation between the January SOI as x and U.S. temperature as y. The non-trivial correlation can quantitatively support the physical mechanism how the January SOI influences the U.S. January temperature.

However, the U.S. January temperature can be influenced by multiple factors: SOI, SST over Atlantic, North Pacific SST, Arctic pressure conditions, etc. Then, the linear model becomes

$$y = b_0 + b_1 x_1 + b_2 x_2 + \cdots + b_n x_n. \tag{5.27}$$

When $n = 1$, this degenerates into the single variable regression.

Let us look at two examples before proceeding to the mathematical theory of multivariate regression.

EXAMPLE 5.6

This example shows a two-variable regression.

$$y = b_0 + b_1 x_1 + b_2 x_2. \tag{5.28}$$

Geometrically, this is an equation of a plane in a 3D space (x_1, x_2, y). Given three points not on a straight line, a plane can be determined uniquely. This means specifying three x_1 coordinate values, three x_2 coordinate values and three y coordinate values, which can be done in the following R code:

```
x1=c(1,2,3) #Given the coordinates of the 3 points
x2=c(2,1,3)
y=c(-1,2,1)
df=data.frame(x1,x1,y) #Put data into the data.frame format
df=data.frame(x1,x2,y)
fit <- lm(y ~ x1 + x2, data=df)
fit#Show the regression results
Call:
lm(formula = y ~ x1 + x2, data = df)
Coefficients:
(Intercept)           x1           x2
 -5.128e-16    1.667e+00   -1.333e+00
```

```
 1.667*x1-1.333*x2  #Verify that 3 points determining a plane
[1] -0.999  2.001  1.002
```

EXAMPLE 5.7

This example will show that four arbitrary points cannot be all on a plane. The fitted plane has the shortest distance squares, i.e., the least squares (LS), or minimal mean square error (MMSE). Thus, the residuals are non-zero, in contrast to the zero residuals in the previous example.

```
u=c(1,2,3,1)
v=c(2,4,3,-1)
w=c(1,-2,3,4)
mydata=data.frame(u,v,w)
myfit <- lm(w ~ u + v, data=mydata)
summary(myfit)#Show the result
Call:
lm(formula = w ~ u + v, data = mydata)

Residuals:
   1    2    3    4
 1.0 -1.0  0.5 -0.5

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   1.0000     1.8708   0.535    0.687
u1            2.0000     1.2472   1.604    0.355
v1           -1.5000     0.5528  -2.714    0.225

Residual standard error: 1.581 on 1 degrees of freedom
Multiple R-squared:  0.881,Adjusted R-squared:  0.6429
F-statistic:   3.7 on 2 and 1 DF,  p-value: 0.345
```

EXAMPLE 5.8

This example will show a general multivariate linear regression using R. It has three independent variables, one dependent variable, and ten data points. For R program simplicity, the data are generated by an R random number generator. Again, R requires that the data be put into data frame format so that a user can clearly specify which are independent variables and which is dependent.

```
dat=matrix(rnorm(40),nrow=10, dimnames=list(c(letters[1:10]), c(LETTERS[23:26])))
fdat=data.frame(dat)
fit=lm(Z~ W + X + Y, data=fdat)
summary(fit)
Call:
lm(formula = Z ~ W + X + Y, data = fdat)
Residuals:
```

```
     Min       1Q   Median       3Q      Max
-0.52997 -0.22259 -0.01266  0.22771  0.74387
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.36680    0.16529   2.219   0.0683 .
W            0.11977    0.20782   0.576   0.5853
X           -0.53277    0.19378  -2.749   0.0333 *
Y           -0.04389    0.14601  -0.301   0.7739
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1
Residual standard error: 0.4805 on 6 degrees of freedom
Multiple R-squared:  0.5883,Adjusted R-squared:  0.3824
F-statistic: 2.857 on 3 and 6 DF,  p-value: 0.1267
```

Thus, the linear model is

$$Z = 0.37 + 0.12W - 0.53X - 0.04Y. \tag{5.29}$$

The 95% confidence interval for W's coefficient is $0.12 \pm 2 \times 0.21$, that for X's coefficient is $-0.53 \pm 2 \times 0.19$,Y's coefficient is $-0.04389 \pm 2 \times 0.15$. Each confidence interval includes zero. Thus, there is no significant no-zero trend for the Z data with respect to W, X, Y. This is expected, since the data are randomly generated and should not have a trend.

In practical applications, a user can just make the data into the same data frame format as shown here. Then, R command

```
lm(formula = Z   W + X + Y, data = fdat)
```

can do the regression job.

R can also do nonlinear regression with specified functions, such as quadratic functions and exponential functions. See examples from the URLs

```
https://www.zoology.ubc.ca/~schluter/R/fit-model/
https://stat.ethz.ch/R-manual/R-devel/library/stats/html/nls.html
```

EXERCISES

5.1 The following is the SVD results

```
mat
     [,1] [,2]
[1,]    1    1
[2,]    1   -1

svd(mat)
$d
[1] 1.414214 1.414214
```

```
$u
           [,1]       [,2]
[1,] -0.7071068 -0.7071068
[2,] -0.7071068  0.7071068

$v
     [,1] [,2]
[1,]   -1    0
[2,]    0   -1
```

Use A=UDV' to recover the first column of

```
mat
     [,1] [,2]
[1,]    1    1
[2,]    1   -1
```

Show detailed calculations of all the relevant matrices and vectors. Use space-time decomposition to describe your results. Extra 5 bonus points: Describe the space and temporal modes, and their corresponding variances or energies.

5.2 SVD space-time decompositions and data reconstruction.

SVD decomposition: Use R and SVD method to decompose the 5-degree reconstructed annual precipitation data (PrcpRecon.csv) into space, variance, and time matrices: $u, d,$ and v.

EOF-PC reconstruction: Use the SVD's $u, d,$ and v matrices to reconstruct the original data matrix, $m = u \cdot d \cdot v^T$, where v^T is the transpose matrix of v. Here, u's column vectors are EOFs which represent spatial variations; the diagonal matrix's elements are eigenvalues representing variances; and v's row vectors are principal components (PCs) which represent the temporal variation.

a) Choose a case of 4 grid boxes and 2 years.
b) Choose a case of 4 grid boxes and 3 years.
c) Discuss the above results, using plain text or figures or both.

5.3 A covariance matrix C can be computed from a space-time observed anomaly data matrix X which has N rows for spatial grid boxes and Y columns for time in years:

$$C = X \cdot X^T / Y \tag{5.30}$$

This is an $N \times N$ matrix.

a) Choose a Y matrix from the reconstructed 5-degree annual precipitation data and calculate a covariance matrix for $N = 5$ and $Y = 6$.
b) Use R to find the inverse matrix of the covariance matrix C.

CHAPTER 6

BASIC STATISTICAL METHODS FOR CLIMATE DATA ANALYSIS

Statistics originated from Latin “status” meaning ”state” and is a suite of scientific methods that analyze data and make credible conclusions. Statistical methods are routinely used for climate data, such as calculating the climate normal of precipitation at a weather station, claiming the global warming based on a significant positive linear trend of the surface air temperature (SAT) anomalies, and inferring a significant shift from a lower North Pacific sea level pressure (SLP) state to a higher state. The list of questions such as the above can be infinitely long. The purpose of this chapter is to provide basic concepts and a user manual on the commonly used statistical methods in climate data analysis, so that the users can make credible conclusions with a given error probability.

R codes will be supplied for examples in this chapter. Users can easily apply these codes and the given formulas in this book for their data analyses and do not need prerequisite of calculus and much statistics. To interpret the statistics results in a meaningful way, domain knowledge of climate science should be very useful when using the statistical concepts and calculations results to state the conclusions from specific climate datasets.

The statistical methods in this chapter focusing on making credible inference about the climate state with a given error based on the analysis of climate data, so that the observed data can form the basis of making objective and reliable conclusions. We will describe a list of statistical indices, such as mean, variance and quantiles, for climate data, and then look into the probability distributions and statistical inferences.

6.1 A list of statistical indices for a set of temperature data

Below is data of the global average annual mean temperature anomalies from 1880-2015 (Karl et al. 2015, NOAA GlobalTemp dataset at NCDC http://www1.ncdc.noaa.gov/pub/data/noaaglobaltemp/operational/). In the data list, the first datum corresponds to 1880, and the last 2015. These 136 years of data are used to illustrate the following statistical concepts: mean, variance, standard deviation, skewness, kurtosis, median, 5th percentile, 95th percentile, and other quantiles. The anomalies are with respect to the 1971-2000 mean, i.e., the 1971-2000 climatology period.

```
#Read the data online
d1=read.table("https://www1.ncdc.noaa.gov/pub/data/noaaglobaltemp/operational/timeseries/a
dim(d1)
#[1] 138   6
d1[1:3,]
#V1        V2       V3       V4       V5       V6
#1 1880 -0.366861 0.009744 0.001402 0.000850 0.007492
#2 1881 -0.316453 0.009777 0.001402 0.000877 0.007498
#3 1882 -0.317072 0.009767 0.001402 0.000897 0.007468
tmean15=d1[1:136,2] #Extract the temperature data from 1880-2015
tmean15 #Show the data of the second column below

  [1] -0.366861 -0.316453 -0.317072 -0.391987 -0.452519
  [6] -0.464505 -0.448412 -0.497834 -0.393168 -0.345842
 [11] -0.567892 -0.494397 -0.549249 -0.564002 -0.523957
 [16] -0.467993 -0.334248 -0.361843 -0.504724 -0.364227
 [21] -0.313154 -0.386735 -0.490564 -0.581421 -0.664936
 [26] -0.537632 -0.463620 -0.621444 -0.691783 -0.679401
 [31] -0.635055 -0.687750 -0.583232 -0.572698 -0.395108
 [36] -0.324717 -0.545935 -0.569831 -0.458031 -0.455610
 [41] -0.462709 -0.400354 -0.482141 -0.466976 -0.502474
 [46] -0.399837 -0.314040 -0.401816 -0.423511 -0.548603
 [51] -0.354206 -0.324295 -0.369086 -0.495110 -0.353120
 [56] -0.387715 -0.362240 -0.263512 -0.276404 -0.260986
 [61] -0.151611 -0.048452 -0.092399 -0.091062  0.046776
 [66] -0.075250 -0.250403 -0.294178 -0.293862 -0.302982
 [71] -0.406029 -0.256716 -0.218683 -0.148851 -0.359614
 [76] -0.379019 -0.443277 -0.193774 -0.133514 -0.185089
 [81] -0.224220 -0.166323 -0.154321 -0.135753 -0.392847
 [86] -0.321024 -0.266600 -0.258192 -0.275150 -0.152131
 [91] -0.207871 -0.323882 -0.218903 -0.081980 -0.318040
 [96] -0.242315 -0.324682 -0.047364 -0.132460 -0.018256
[101]  0.018821  0.054134 -0.063736  0.096249 -0.096068
[106] -0.110964 -0.015905  0.124013  0.130747  0.051752
[111]  0.186930  0.160061  0.010888  0.039230  0.096167
[116]  0.212830  0.077362  0.273566  0.389591  0.198839
[121]  0.181156  0.302444  0.358044  0.368890  0.334530
[126]  0.416214  0.370507  0.365983  0.296281  0.392659
```

```
[131]  0.457878  0.334962  0.379798  0.424233  0.495303
[136]  0.655707
```

We use R to calculate all the statistical parameters. The data is read as `tmean15`.

```
 mean(tmean15)
#[1] -0.2044572
sd(tmean15)
#[1] 0.3069892
var(tmean15)
#[1] 0.09424238
library(e1071)
skewness(tmean15)
#[1] 0.7160457
kurtosis(tmean15) #kurtosis(normal)=0
#[1] -0.3652721 #less peaked than standard normal
median(tmean15)
#[1] -0.29402
quantile(tmean15,probs= c(0.05,0.25, 0.75, 0.95))
#       5%        25%        75%        95%
#-0.58187375 -0.42845250 -0.01766825  0.38224625
```

The following R commands can plot the time series of the temperature data with a linear trend (see Fig. 6.1).

```
yrtime15=seq(1880,2015)
reg8015<-lm(tmean15 ~ yrtime15)
# Display regression results
reg8015
#Call:
lm(formula = tmean15 ~ yrtime15)
#Coefficients:
#(Intercept)     yrtime15
# -13.299762     0.006724
# Plot the temperature time series and its trend line
plot(yrtime15,tmean15,xlab="Year",ylab="Temperature deg C",
     main="Global Annual Mean Land and Ocean Surface
     Temperature Anomalies 1880-2015", type="l")
abline(reg8015, col="red")
text(1930, 0.4, "Linear temperature trend 0.67 oC per century",
     col="red",cex=1.2)
```

The global average of the 20th century mean is 12.7 °C, according to NCEI (https://www.ncdc.noaa.gov/sotc/global/201503). The 2015 anomaly was 0.65 °C with respect to 1971-2000 climatology, which is 0.26°C higher than 1900-1999 climatology. Thus, the 2015's global average annual mean temperature is 13.62°C.

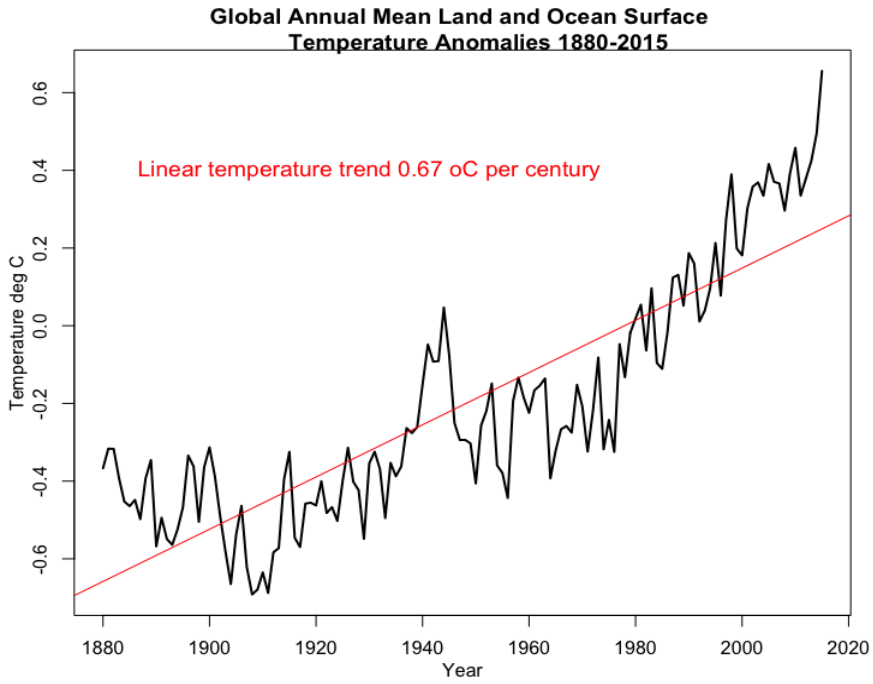

Figure 6.1 Time series of the global average annual mean temperature with respect to 1971-2000 climatology.

The mathematics formulas for the above statistical parameters are below. Let $x = \{x_1, x_2, \cdots, x_n\}$ be the sampling data for a time series. Then,

$$\text{mean: } \mu(x) = \frac{1}{n}\sum_{k=1}^{n} x_k, \tag{6.1}$$

$$\text{variance by unbiased estimate: } \sigma^2(x) = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \mu(x))^2, \tag{6.2}$$

$$\text{standard deviation: } \sigma(x) = (\sigma^2(x))^{1/2}, \tag{6.3}$$

$$\text{skewness: } \gamma_3(x) = \frac{1}{n}\sum_{k=1}^{n}\left(\frac{x_k - \mu(x)}{\sigma}\right)^3, \tag{6.4}$$

$$\text{kurtosis: } \gamma_4(x) = \frac{1}{n}\sum_{k=1}^{n}\left(\frac{x_k - \mu(x)}{\sigma}\right)^4 - 3. \tag{6.5}$$

Mean gives the average of samples. Variance and standard deviation measure the spread of samples. It is large when the samples have a wide spread. Skewness is dimensionless and measures the asymmetry of samples. Zero skewness means symmetric distribution. For example, a normal distribution's skewness is zero. Negative skewness is skew to the left, meaning that the long distribution tail is on the left. Positive skewness has a long tail on the right. Kurtosis is also dimensionless and measures the peakedness of a distribution. The normal distribution's kurtosis is zero. Positive kurtosis means a high peak at the mean, a slim and tall distribution. This is referred

to as leptokurtic. ”Lepto” is Greek and means thin or fine. Negative kurtosis means a low peak at the mean, a fat and short distribution, referred to as platykurtic. ”Platy” is also Greek and means flat or broad. ”Kurtic” and ”kurtosis” are Greek and mean peakedness.

For the 136 years of global average annual mean temperature data, the skewness is 0.72, meaning skew to the right with tail to on the right with more extreme high temperatures, as shown in the histogram Fig. 6.2. The kurtosis is -0.37, meaning flatter than a normal distribution also shown by the histogram.

Median is a number sample set such that 50% of the samples are less than the median, and another 50% greater than the median. Sort the samples from the smallest to the largest. The median is the number in the middle. If the number of the samples is even, then median is equal to the mean of the two middle samples.

Quantiles are defined in the same way by sorting. For example, 25-percentile is a sample that 25% of samples are less than this sample. By definition, 75-percentile is larger than 40-percentile. 100-percentile is the largest sample, and 0-percentile is the smallest sample. Usually, people use a box plot to show the typical quantiles. See Fig. 6.3 for the box plot of the 136 years of temperature data.

The 50-percentile is called median. If the distribution is symmetric, then median is equal to mean. Otherwise they are not. If skew to the right, then mean is on the right of median: mean greater than median. If skew to the left, then mean is on the left of median: mean less than median. Our 136 years of temperature data are right skewed and have mean equal to -0.20°C, greater than their median equal to -0.29°C.

6.2 A set of commonly used statistical figures

We will use the 136 years of temperature data and R to illustrate the following commonly used statistical figures: histogram, box plot, scatter plot, qq-plot, and linear regression trend line.

6.2.1 Histogram of a set of data

```
h<-hist(tmean15, main="Histogram of 1880-2015 Temperature
Anomalies",xlab="Temperature anomalies") #Plot historgram
xfit<-seq(min(tmean15),max(tmean15), length=30)
areat=diff(h$mids[1:2])*length(tmean15) #Normalization area
yfit<-areat*dnorm(xfit, mean=mean(tmean15), sd=sd(tmean15))
lines(xfit,yfit,col="blue",lwd=2) #Plot the normal fit
```

Figure 6.2 shows the result of the above R commands.

One can also plot the probability density function based on the R’s kernel estimate.

```
plot(density(tmean15), main="Kernel estimate of density",
          xlab="Temperature") #Kernel estimate density
lines(xfit,dnorm(xfit, mean=mean(tmean15),
       sd=sd(tmean15)), col="blue") #Moment estimated normal
```

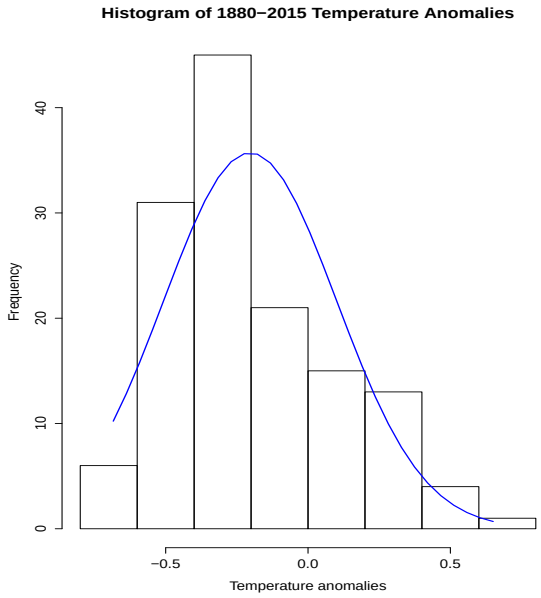

Figure 6.2 Histogram of the global average annual mean temperature anomalies from 1880-2015.

6.2.2 Box plot

Figure 6.3 is the box plot of the 136 years of temperature and can be made from the following R command

```
boxplot(tmean15, ylab="Temperature anomalies")
```

The rectangular box's mid line indicates the level of media, which is -0.30°C. The rectangular box's lower boundary is the first quartile, i.e., 25-percentile. The box's upper boundary is the third quartile, i.e., the 75-percentile. The box's height is the third quartile minus the first quartile, and is called interquartile range (IQR). The upper whisker is the third quartile plus 1.5 IQR. The lower whisker is supposed to be at the first quartile minus 1.5 IQR. However, this whisker is lower than the lower extreme. Thus, the lower whisker takes the lower extreme, which is -0.69 °C. The points outside of the two whiskers are considered outliers. Our dataset has one outlier, which is 0.66 °C. This is the hottest year happened in 2015.

Sometimes, one may need to plot multiple box plots on the same figure, which can be done by R. One can follow an example in R-project document

```
http://www.inside-r.org/r-doc/graphics/boxplot
```

6.2.3 Scatter plot

Scatter plot is used to display if two datasets are correlated. We use the southern oscillation index (SOI) and the contiguous United States temperature as an example to describe the scatter plot. The data can be downloaded from

```
www.ncdc.noaa.gov/teleconnections/enso/indicators/soi/
www.ncdc.noaa.gov/temp-and-precip/
```

The following R commands can produce the scatter plot shown in Fig. 6.4.

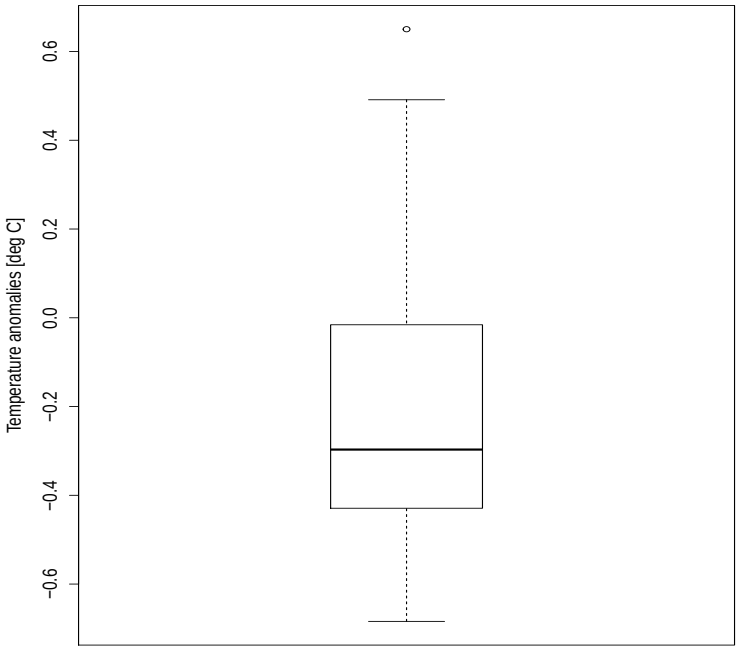

Figure 6.3 Box plot of the global average annual mean temperature anomalies from 1880-2015.

```
#Change directory to where the datasets are
setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2016/chap4data-refs")
ust=read.csv("USJantemp1951-2016-nohead.csv",header=FALSE)
soi=read.csv("soi-data-nohead.csv", header=FALSE) #Read data
soid=soi[,2] #Take the second column SOI data
soim=matrix(soid,ncol=12,byrow=TRUE)
#Make the SOI into a matrix with each month as a column
soij=soim[,1] #Take the first column for Jan SOI
ustj=ust[,3] #Take the third column: Jan US temp data
plot(soij,ustj,main="Scatter plot between Jan SOI
     and US temp",xlab="SOI[dimensionless]",
     ylab="US Temp degF", pch=19)
# Plot the scatter plot
soiust=lm(ustj ~ soij) #Linear regression
abline(soiust, col="red") #Linear trend line
```

The correlation between the two datasets is 0. Thus, the slope is also zero.

The scatter plot shows that the nearly zero correlation is mainly due to the five negative SOI values, which are El Nino Januarys: 1983 (-3.5), 1992 (-2.9), 1998 (-2.7), 2016 (-2.2), 1958 (-1.9). When these strong El Nino Januarys are removed, then the correlation is 0.2. The slope is then 0.64, compared with 1.0 for perfect correlation.

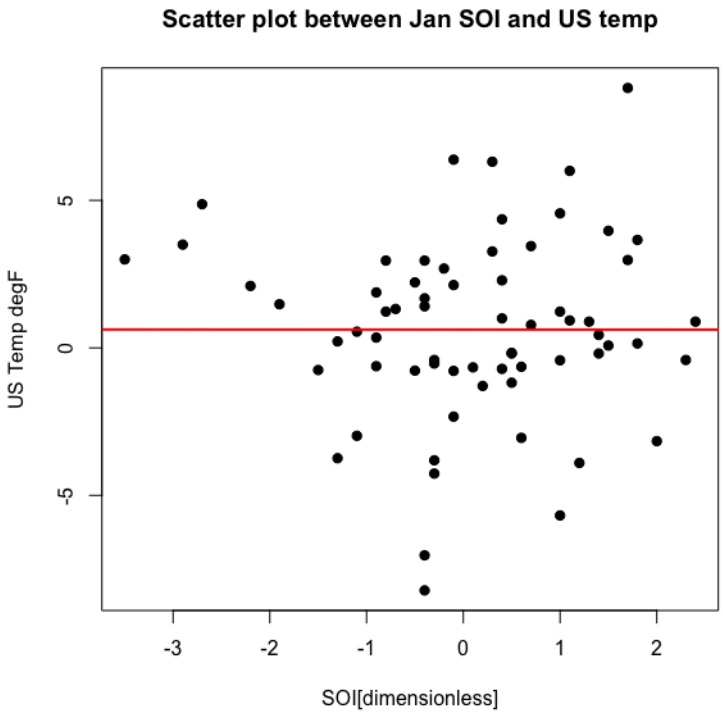

Figure 6.4 Scatter plot of the US January temperature vs. SOI from 1951-2016.

The R commands to retain the data without the above five El Nino years are below

```
soijc=soij[c(1:7,9:32,34:41,43:47,49:65)]
ustjc=ustj[c(1:7,9:32,34:41,43:47,49:65)]
```

With these data, the scatter plot and trend line can be produced in the same way.

We thus may say that the SOI has some prediction skill for the contiguous U.S.' January temperature for the non-El Nino years. This correlation is stronger for particular regions of the U.S. since temperature field over the U.S. is inhomogeneous and is related to the tropical ocean dynamics in different ways. This gives us a hint to find out the prediction skill for an objective field: to plot a scatter plot between the objective field and the field used for prediction. The objective field is called predicant and the field used for prediction is called predictor. A very useful prediction skill is that predictor leads predicant by a time, say a month. Then the scatter plot will be made from the pairs between predictor and predicant data with one month lead. The absolute value of correlation can then be used as a measure of prediction skill. Since 1980s, the U.S. Climate Prediction Center has been using sea surface temperature (SST) and sea level pressure (SLP) as predictors for the U.S. temperature and precipitation via the canonical correlation analysis method. Therefore, before a prediction is made, it is a good idea to examine the prediction skill via scatter plots, which can help identify the best predictors.

However, the scatter plot approach above for maximum correlation is only applicable for linear prediction or weakly nonlinear relationships. Nature can sometimes be

very nonlinear, which require more sophisticated assessments of prediction skill, such as neural networks and time-frequency analysis.

Our short chapter on statistics ends here. Many free statistics tutorials are available online for search and reading. The NIST materials are for basic statistics

http://www.itl.nist.gov/div898/handbook/eda/section3/eda35.htm

The manuscript by David Stephenson of the University of Reading, and another by the working group statistics at the Climate Service Center (Hamburg) are particularly for weather and climate data.

http://empslocal.ex.ac.uk/people/staff/dbs202/cag/courses/MT37C/course-d.pdf

http://www.climate-service-center.de/imperia/md/content/csc/projekte/csc-report13_englisch_final-mit_umschlag.pdf

Eric Gilleland of NCAR authored a slide for using R to do climate statistics:

http://www.maths.lth.se/seamocs/meetings/Malta_Posters_and_Talks/MaltaShortCourseSlides4.pdf

EXERCISES

6.1 Before NOAAGlobalTemp, the two most commonly used datasets of global average annual mean surface air temperature (SAT) anomalies are those credited to the research groups of Jim Hansen (relative to 1951-1980 climatology period) and Phil Jones (relative to 1961-1990 climatology period):

http://cdiac.ornl.gov/trends/temp/hansen/hansen.html

http://cdiac.ornl.gov/trends/temp/jonescru/jones.html

a) Find the average anomalies of each 15 years, starting at 1880. Use t-distribution to find the confidence interval of each 15-year SAT average at 95% confidence level using t-distribution. You can use either Hansen's data or Jones' data. Figure SPM.1(a) of IPCC 2013 (AR4) is a helpful reference.

b) Find the hottest and the coldest 15-year periods from 1880-2014, which has nine 15-year periods. Use t-distribution to check whether the hottest and the coldest 15-year periods are different from zero.

c) Discuss the differences between the Hansen and Jones datasets.

6.2 To test if the average of temperature in Period 1 is significantly different from that in Period 2, one can use t-statistic

$$t^* = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \tag{6.6}$$

where $\bar{x}_i$ and s_i^2 are the sample mean and variance of the Period i ($i = 1, 2$). The degree of freedom (i.e., d.f.) of the relevant t-distribution is equal to the smaller $n_1 - 1$ and $n_2 - 1$. The null hypothesis is that the two averages do not have significant differences, i.e., their difference is zero (in statistical sense with a confidence interval). The alternative hypothesis is that the difference is significantly different from zero. Now you can choose to use one-sided test when the difference is positive. Use a significance level of 5% or 1%, or another level at your own choice.

a) Choose two 15-year periods which have very different average anomalies. Use the higher one minus the lower one. Use this t-test method for a one-sided test to check if the difference is significantly greater than zero.

b) Choose two 15-year periods which have very close average anomalies. Use the higher one minus the lower one. Use this t-test method for a two-sided test to check if the difference is not significantly different from zero.