Citation of this work: Shen, S.S.P., 2017: R Programming for Climate Data Analysis and Visualization: Computing and plotting for NOAA data applications. The first revised edition. San Diego State University, San Diego, USA, 152pp.

R Programming for Climate Data Analysis and Visualization:

Computing and plotting for NOAA data applications

SAMUEL S.P. SHEN

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

Contents

Preface

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, which are taken from the book entitled "Climate Mathematics with R" authored by Samuel Shen and Richard Somerville to be published by the Cambridge University Press. 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.

The book's typeset follows a Cambridge University Press LaTex template.

SSPS at San Diego, California, USA
May 2017
The first revised version in October 2017

Acknowledgements

The project was supported in part by the U.S. National Oceanographic and Atmospheric Administration (Award No. 13342-Z7812001). The students in the U.S. National Center for Environmental Information (NCEI), Asheville, North Carolina, 30 May- 2 June 2017 helped clarify some R codes and correct some typos.

1 Basics of Computer Program R

It is popular in today's applied mathematics books to use computing tools for complex and tedious algebras so that students can focus on correct usage of the mathematical tools with accurate statement of assumptions and precise interpretation of the results. Among many software packages used in climate community, R's popularity has dramatically increased in the last a few years due to its enormous power of handling big data. We thus have chosen to use R in this book. This chapter shows installation of R and R Studio, and some very basic mathematical operations and plotting by R. A student who has mastered the R examples used in this book should have sufficient skills to develop R projects independently.

1.1 Download and install R and R-Studio

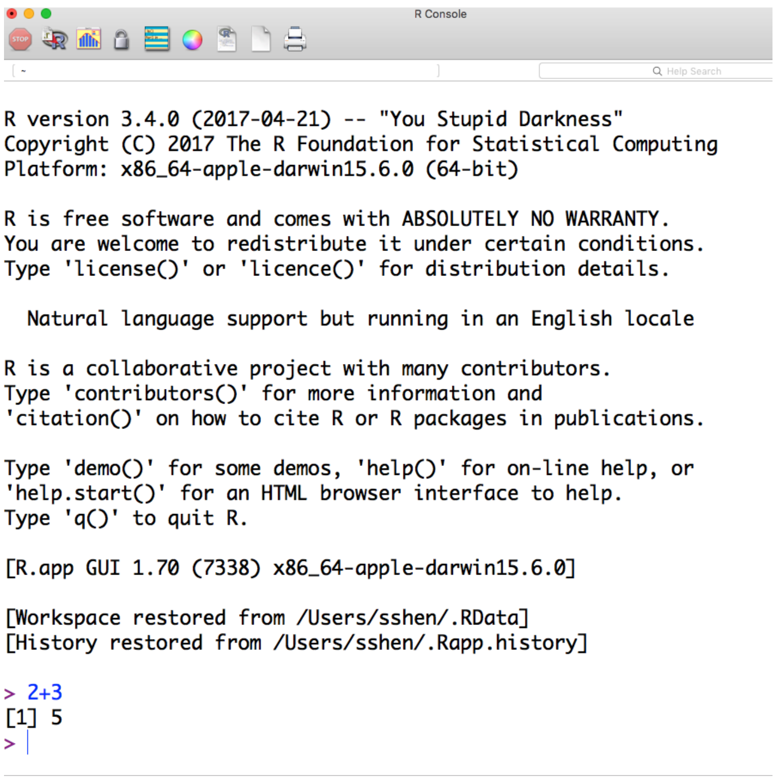
For Windows users, visit the website
`https://cran.r-project.org/bin/windows/base/`
to find the instructions of R program download and installations.

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.



```
R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[R.app GUI 1.70 (7338) x86_64-apple-darwin15.6.0]

[Workspace restored from /Users/sshen/.RData]
[History restored from /Users/sshen/.Rapp.history]

> 2+3
[1] 5
>
```

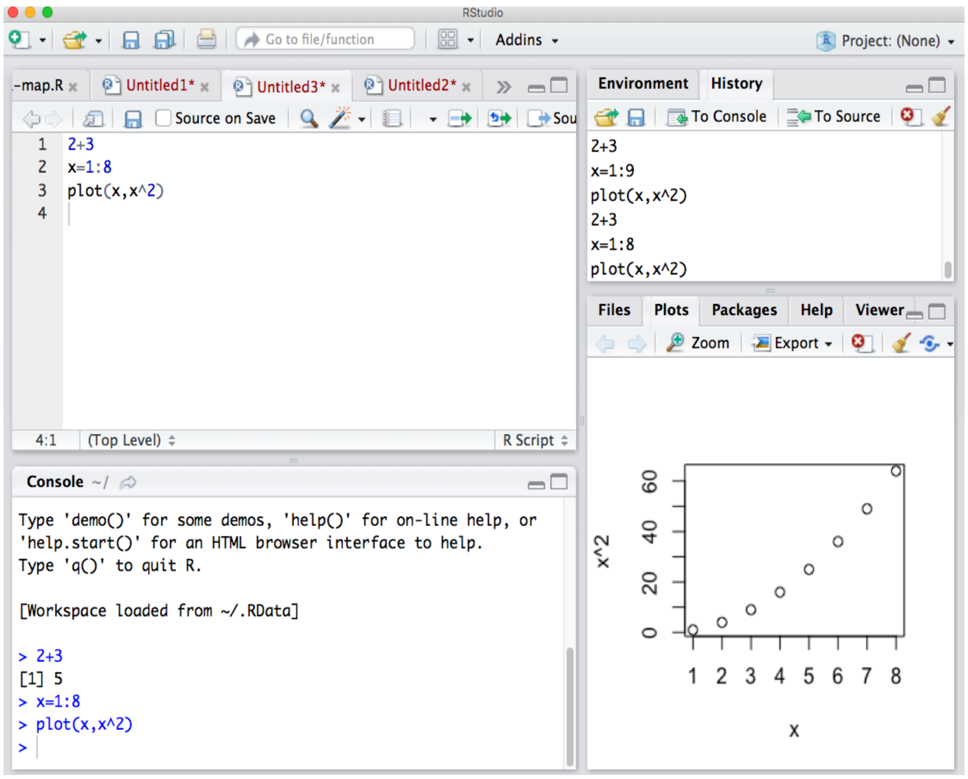
Fig. 1.1 R Console window after opening R.

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



```
2+3
x=1:8
plot(x,x^2)
```

```
2+3
x=1:9
plot(x,x^2)
2+3
x=1:8
plot(x,x^2)
```

```
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

> 2+3
[1] 5
> x=1:8
> plot(x,x^2)
>
```

Fig. 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.

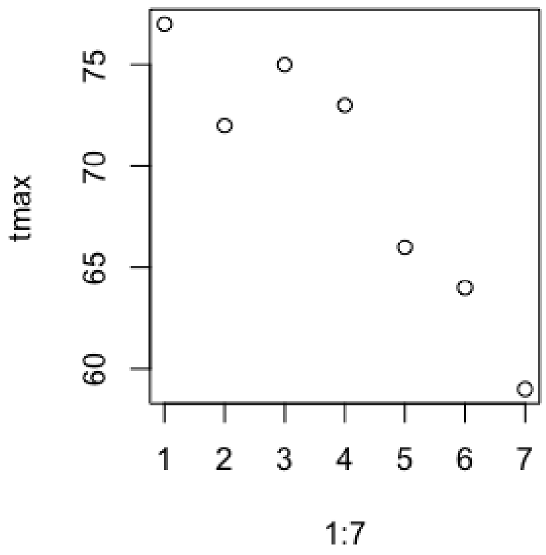


Fig. 1.3 The daily maximum temperature during 1-7 May 2017 of the San Diego International Airport.

```
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.

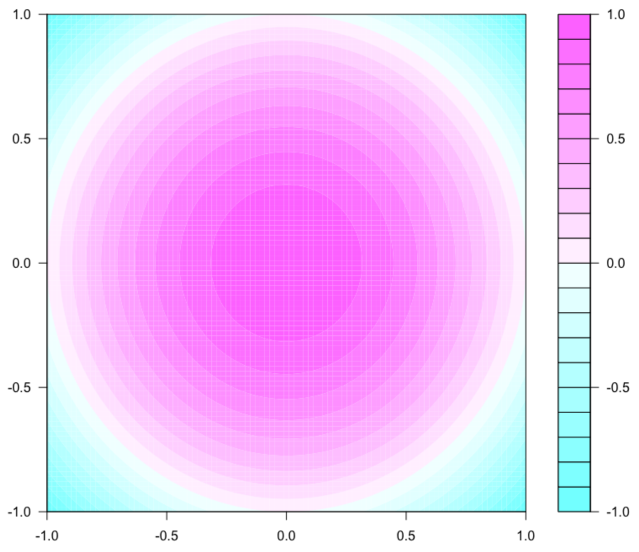


Fig. 1.4 The color map of contours for the function $z = 1 - x^2 - y^2$.

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

#Linear regression and linear trend line
#2007-2016 data of the global temperature anomalies
#Source: NOAAGlobalTemp data
t=2007:2016
T=c(.36,.30, .39, .46, .33, .38, .42, .50, .66, .70)
lm(T ~ t) #Linear regression model of temp vs time
#(Intercept)            t
#-73.42691      0.03673
#Tempearture change rate is 0.03673 oC/yr or 0.37 oC/decade
plot(t,T, type="o",xlab="Year",ylab="Temperature [deg C]",
     main="2007-2016 Global Temperature Anomalies
          and Their Linear Trend [0.37 oC/decade] ")
abline(lm(T ~ t), lwd=2, col="red") #Regression line
```

The global temperature data from 2007-2016 in the above R code example are displayed in Fig. 1.5, together with their linear trend line

$$T = -73.42691 + 0.03673t. \tag{1.1}$$

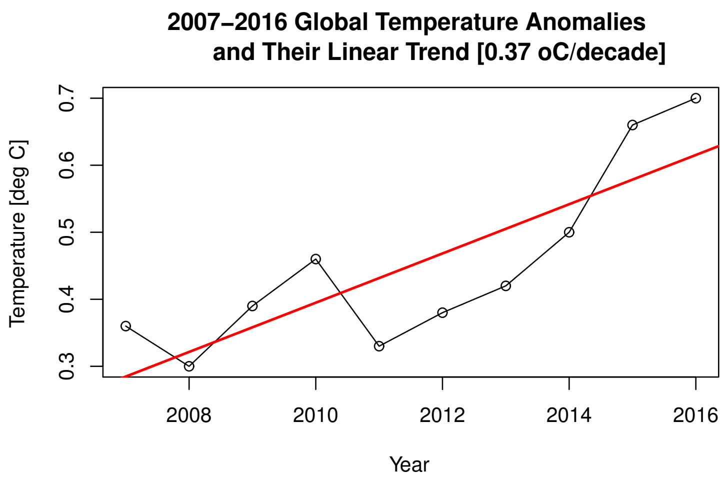


Fig. 1.5 The 2007-2016 global average annual mean surface air temperature anomalies with respect to the 1971-2000 climate normal. The red is a linear trend line computed from a linear regression model.

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.

References

[1] R tutorials by William B. King, Coastal Carolina University,

`http://ww2.coastal.edu/kingw/statistics/R-tutorials/`

[2] R tutorial by Steve Jost, De Paul University,

`http://facweb.cs.depaul.edu/sjost/csc423/`

Exercises

1.1 For some purposes, climatology or climate is dened as the mean state, or normal state, of a climate parameter, and is calculated from data over a period of time called the climatology period (e.g., 1961-1990). Thus the surface air temperature climate or climatology at a given location may be calculated by averaging observational temperature data over a period such as 1961 through 1990. Thirty years are often considered in the climate science community as the standard length of a climatology period. Due to the relatively high density of weather stations in 1961-1990, compared to earlier periods, investigators have often used 1961-1990 as their climatology period, although some may now choose 1971-2000 or 1981-2010. Surface air temperature (SAT) is often dened as the temperature inside a white-painted louvered instrument container or box, known as a Stevenson screen, located on a stand about 2 meters above the ground. The purpose of the Stevenson screen is to shelter the instruments from radiation, precipitation, animals, leaves, etc, while allowing the air to circulate freely inside the box. Daily maximum temperature (Tmax) is the maximum temperature measured inside the screen box by a maximum temperature thermometer within 24 hours.

Go to the United States Historical Climatology Network (USHCN) website

`http://cdiac.ornl.gov/epubs/ndp/ushcn/ushcn_map_interface.html`

and download the monthly Tmax, Tmin, and Tmean data of the Cuyamaca station (USHCN Site No. 042239) near San Diego, California. Use R to calculate the climatology of the August , California, USA according to the 1961-1990 climatology period.

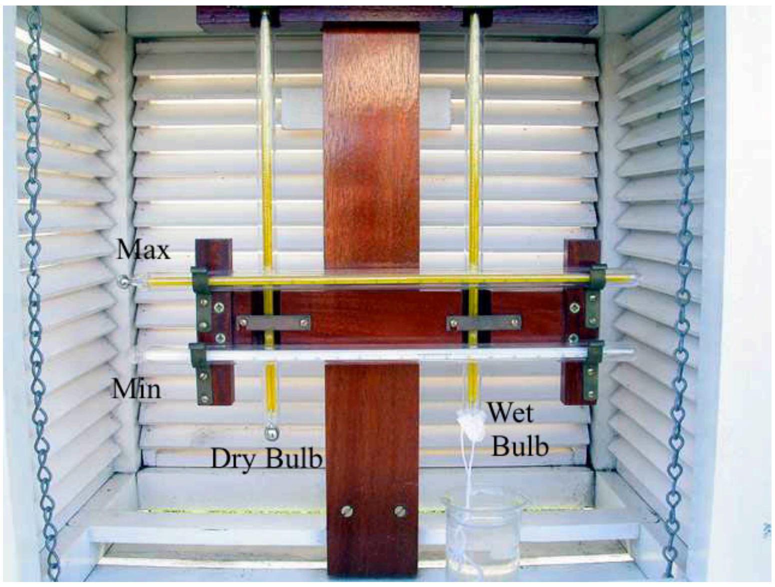


Fig. 1.6 Inside a Stevenson screen, invented by Thomas Stevenson in 1864, and recommended by the World Meteorological Organization (WMO) to measure Tmax and Tmin using two thermometers. The data were recorded every 24 hours. Tmax and Tmin are the temperature extremes over the previous 24 hours and depend on the time of observation. Thus, the observations have the time of observation bias (TOB) due to the inconsistent time of data recording. A much-used dataset called the USHCN dataset includes data corrected for TOB, as well as the raw (uncorrected) data.

1.2 Express the Tmax climatology as an integral when regarding Tmax as a function of time t, using the definition of an integral from the statistics perspective.

1.3 Use R (a) to plot the the Cuyamaca January Tmin data from 1951 to 2010 with continuous curve, and (b) to plot the linear trend lines of Tmin on the same plot as (a) in the following time periods:
(i) 1951-2010,
(ii) 1961-2010,
(iii) 1971-2010, and
(iv) 1981-2010.

Finally, what is the temporal trend per decade for each of the four periods above?

1.4 Trend and derivative:

(a) Use the derivative to explain the trends of the above exercise problem, and

(b) Treat the time series of the Cuyamaca January Tmin in above exercise problem as a smooth function from 1951 to 2010. Use the curve and its derivative to explain the instantaneous rate of change. Use the concept of derivative.

(c) Use the average rate of change for a given period of time to explain the linear trend in each of the four periods. Use the concept of mean value theorem in the integral form.

1.5 Use the integral concept to describe the rainfall deficit or surplus history of San Diego since January 1 of this year according to the USHCN daily precipitation data, or do this for another location you are familiar with. You may use the integral to describe the precipitation deficit or surplus. The daily data can be found and downloaded from

`http://cdiac.ornl.gov/epubs/ndp/ushcn/ushcn_map_interface.html`

Requirements: You should use at least one figure. Your English text must be longer than 100 words.

1.6 Time series and trend line plots for the NOAA global average annual mean temperature anomaly data:

`https://www.ncdc.noaa.gov/data-access/marineocean-data/noaa-global-surface-temperature-noaaglobaltemp`

(a) Plot the global average annual mean temperature from 1880 to 2015.

(b) Find the linear trend of the temperature from 1880 to 2015. Plot the trend line on the same figure as a).

(c) Find the linear trend from 1900 to 1999. Plot the trend line on the same figure as a).

1.7 Use the gridded NOAA global monthly temperature data from the following website or another data source

`https://www.ncdc.noaa.gov/data-access/marineocean-data/noaa-global-surface-temperature-noaaglobaltemp`

(a) Choose two 5-by-5 degrees lat-lon grid boxes of your interest. Plot the temperature anomaly time series of the two boxes on the same figure using two different colors.

(b) Choose sufficiently many grid boxes that cover the state of Texas. Compute the average temperature of these boxes. Then plot the monthly average temperature of these anomalies. Show the trend line on the same figure.

1.8 Research problem: Use the integral of temperature with respect to time to interpret the concept of cumulative degree-days in agriculture. Consider the energy needed by plants to grow.
Requirements: You must use at least one figure and one table. Your English text must be longer than 100 words.

2 R Analysis of Incomplete Climate Data

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 call this the missing data problem.

Missing data problems can be of many kinds and can be very complicated. Here we use the NOAAGlobalTemp dataset to illustrate a few methods often used in analyzing 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 for Environmental Information in 2015.
`https://www.ncdc.noaa.gov/data-access/marineocean-data/`
`noaa-global-surface-temperature-noaaglobaltemp`
This dataset is a monthly data from January 1880 to the present with $5 \times 5°$ 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 this percentage is the percentage of missing data. The minimum coverage is nearly 60%, much of which is due to the good coverage provided by NOAA ERSST (extended reconstructed sea surface temperature).

Using software 4DVD (4-dimensional visual delivery of big climate data) developed at San Diego State University, one can easily see where and when data are missing. Figure 2.2 shows the NOAAGlobalTemp data distribution over the globe for January 1917. The data cover 72% of the global area. The black region includes 28% of the global area and has missing data. The data void regions include the polar areas which could not be accessed at that time, the central tropical Pacific regions which were not on the tracks of commercial ships, central Asia, part of Africa, and the Amazon region. Figure 2.3 shows that the grid box (12.5S, 117.5W) in the Amazon region did not begin to have data until 1918, and the data time series after 1918 is discontinuous with missing data around 1921 and 1922.

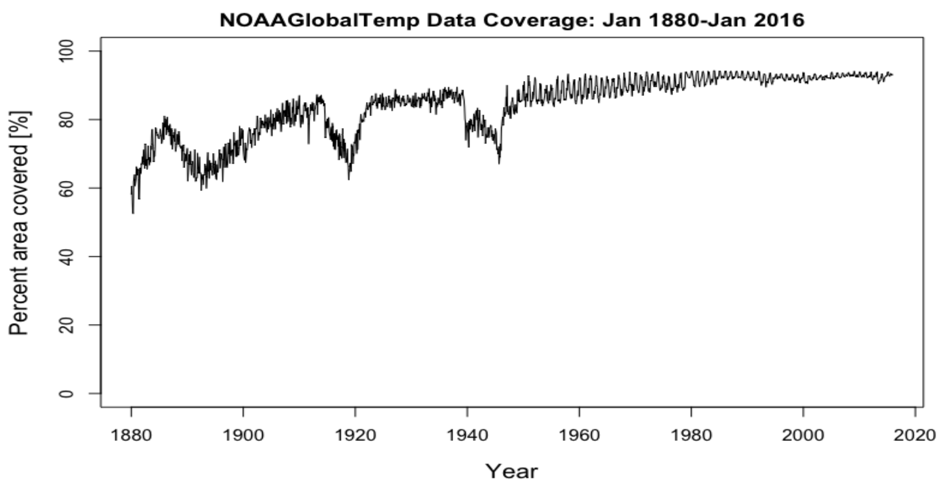
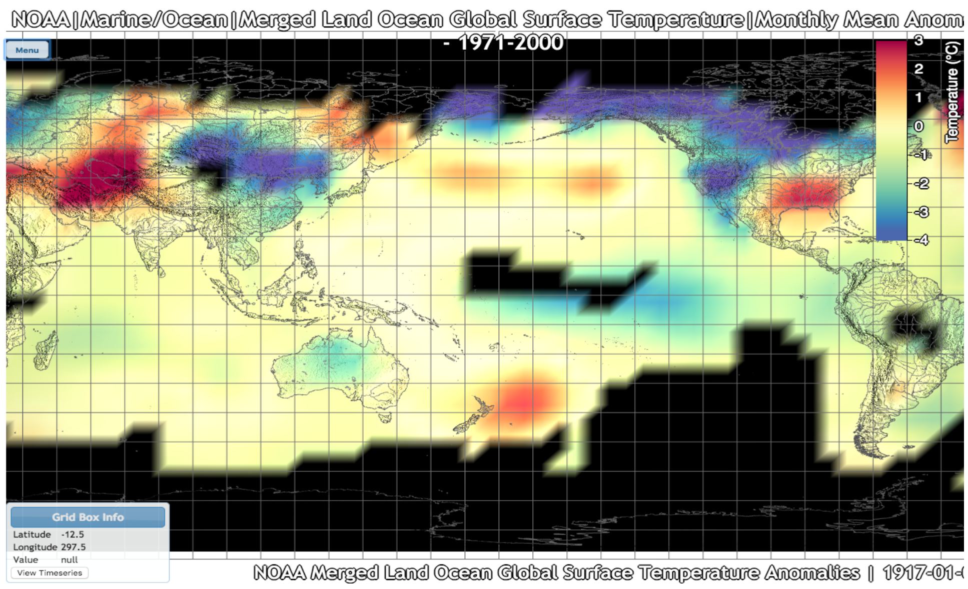


Fig. 2.1 Percentage of the global surface area covered by the NOAAGlobalTemp dataset.


NOAA | Marine/Ocean | Merged Land Ocean Global Surface Temperature | Monthly Mean Anom
- 1971-2000
Menu
3
2
1
0
-1
-2
-3
-4
Temperature (°C)
Grid Box Info
View Timeseries
NOAA Merged Land Ocean Global Surface Temperature Anomalies | 1917-01-

Latitude -12.5
Longitude 297.5
Value null

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

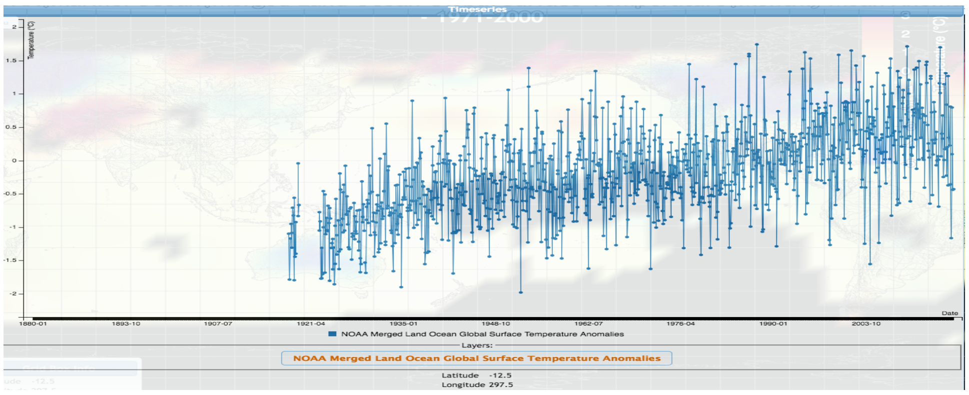


Fig. 2.3 Time series of the monthly temperature anomalies for a grid box over the Amazon region.

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.

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 the asc format as example to describe the R analysis. The following R code reads 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 Niño month (See Fig. 2.4).

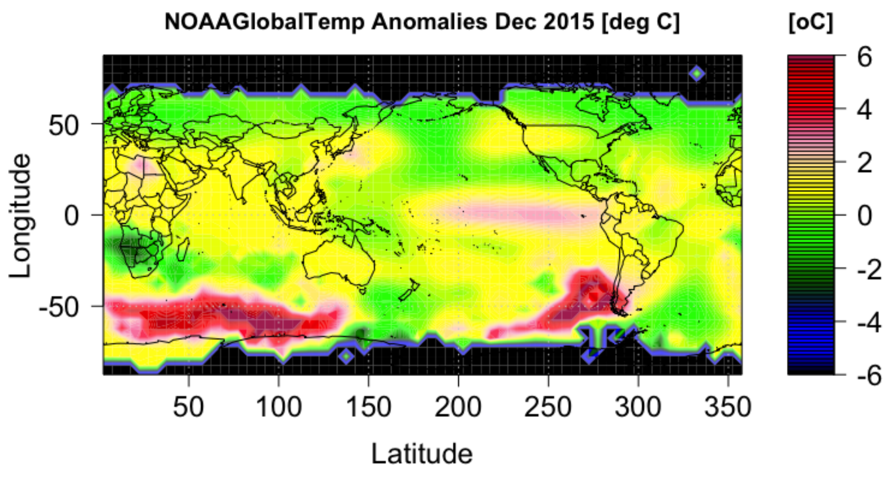


Fig. 2.4 Monthly mean temperature anomalies of December 2015 based on the NOAAGlobalTemp data.

```
library(maps)#Install maps package if not done before
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)
#Matrix flipping is not needed since the data go from 2.5 to 375.5
plot.new()
par(mar=c(4,5,3,0))
int=seq(-6,6,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,
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)})
```

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 Niño characteristics, he can extract the data for the region for a given time interval. The following code extracts the space-time data for the tropical Pacific region (20S-20N, 160E-120W) from 1951 to 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]
```

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 be 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).

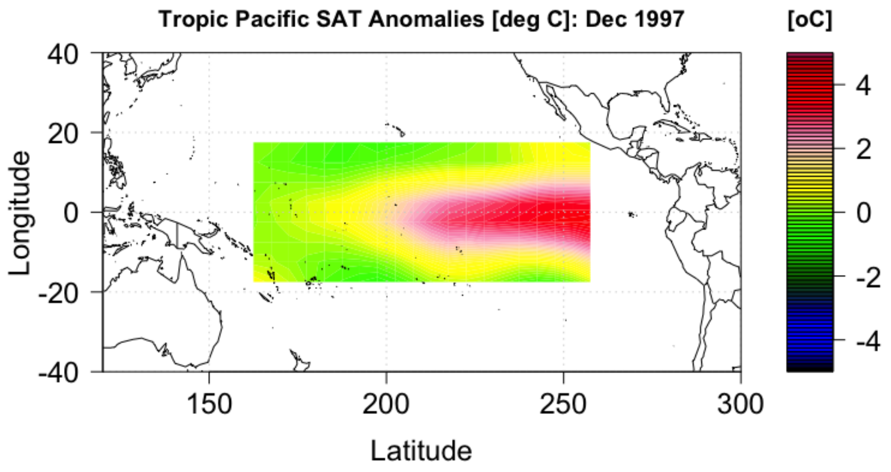


Fig. 2.5 Tropical Pacific SST anomalies of December 1997 based on the NOAAGlobalTemp data.

```
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 that includes 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 in radian. 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 formua:

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

However, NOAAGlobalTemp has missing data. We thus should not average the data-void region. A method 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-Jan2016=1633 months + lat and lon
areaw=matrix(0,nrow=2592,ncol = 1647)
dim(areaw)
#[1] 2592 1647
areaw[,1]=temp[,1]
areaw[,2]=temp[,2]
#create an area-weight matrix equal to cosine box with data and zero for missing
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 to January 2017 and can be generated by the following R code.

```
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.

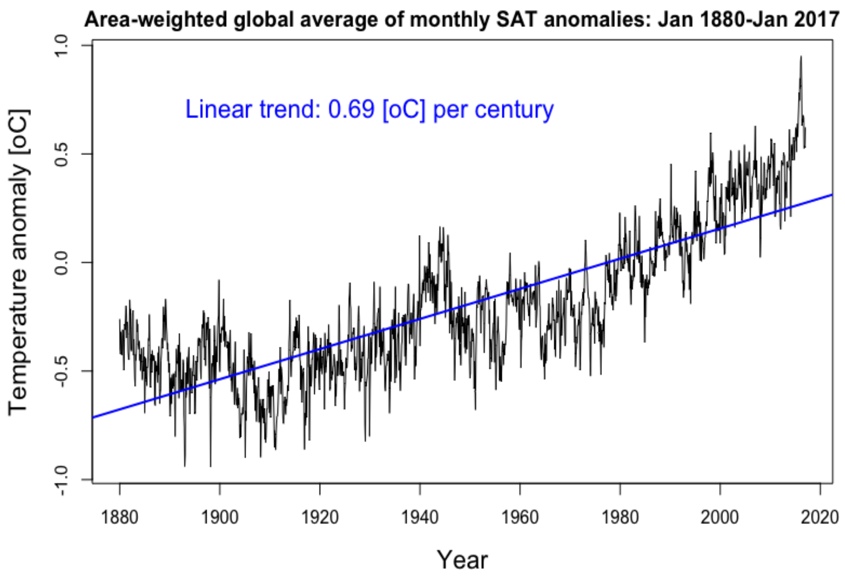


Fig. 2.6 Spatial average of monthly temperature anomalies with respect to 1971-2000 climatology based on the NOAAGlobalTemp data.

```
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 from the NOAA NCEI monthly mean global averages

The NOAA National Centers for Environmental Information (NCEI) also computed the monthly mean global averages, which can also be downloaded from the NOAA-GlobalTemp 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.

```
#Download the NCEI spatial average time series of monthly data
#https://www1.ncdc.noaa.gov/pub/data/noaaglobaltemp/operational/timeseries/aravg.mon.land_
setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2016/Rcodes/Ch15-Rgraphics")
aveNCEI<-read.table("aravg.mon.land_ocean.90S.90N.v4.0.1.201702.asc.txt", header=FALSE)
```

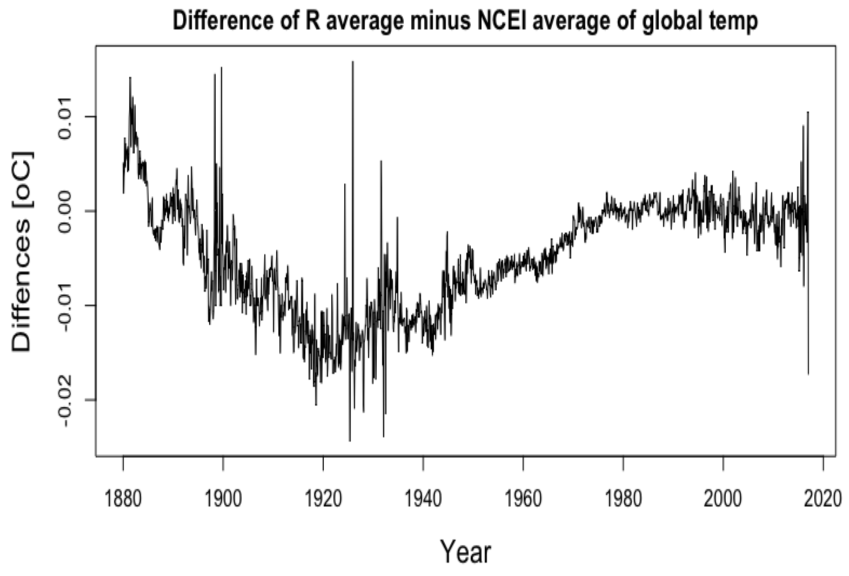


Fig. 2.7 Shen's spatial average of monthly anomalies minus the NCEI time series.

```
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. These small differences do not alter any scientific conclusions based on 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.

```
#Plot the each month's anomalies with trend in 12 panels
plot.new()
```

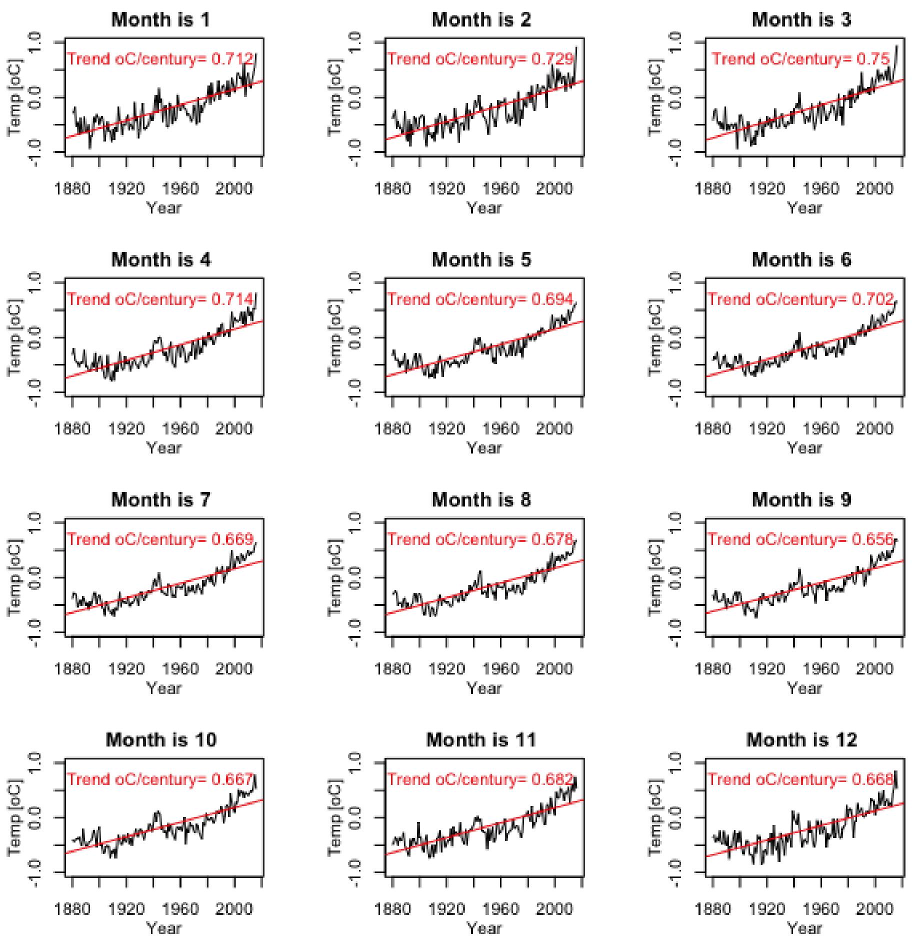


Fig. 2.8 The trend of the spatial average of each month based on the NOAAGlobalTemp data from 1880-2016.

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

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.

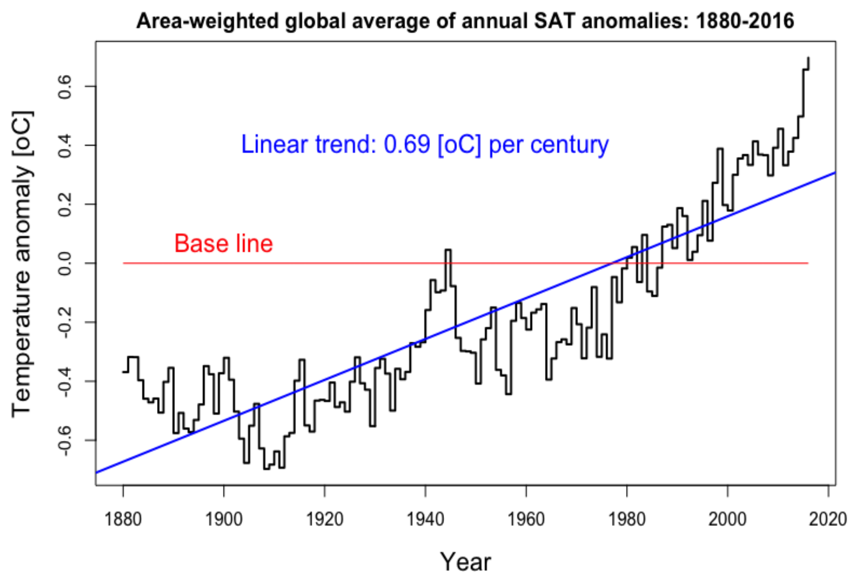


Fig. 2.9 Annual mean of the monthly spatial average anomalies from the NOAAGlobalTemp data.

```
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")
```

2.3.6 Nonlinear trend of the global average annual mean data

The global average annual mean temperature apparently does not vary linearly with time. It is thus useful to examine the underlying nonlinear variation of the annual temperature time series. The simplest nonlinear trend exploration is thorough a polynomial fit. Usually, orthogonal polynomial fits are more efficient and have better fidelity to the 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 the high in the 1880s to the low in the 1910s, then rising until the 1940s, decreasing in the 1960s and 1970s. The choice of 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 an unphysical overfit.

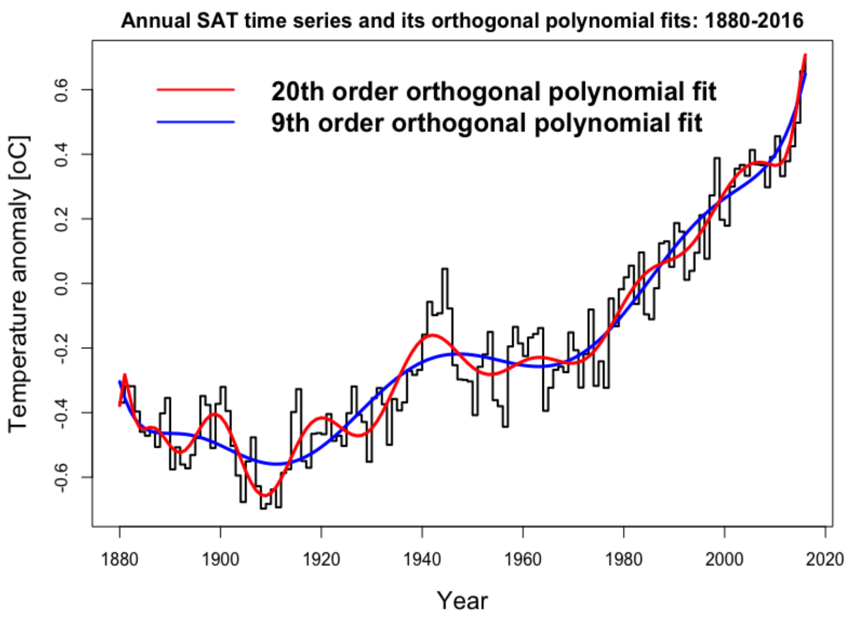


Fig. 2.10 Annual mean time series and its fit by orthogonal polynomials.

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

2.4.1 20th century temperature trend

It is widely known that the global average temperature has increased, especially in recent decades since the 1970s. This is known to the general public as “global warming.” However, the increase is non-uniform, and a few area have even experienced cooling, such as the 1900- 1999 cooling over the North Atlantic off the coast of Greenland. Figure 2.11 shows the uneven spatial distribution of the linear trend of the monthly SAT anomalies from January 1900 to December 1999. Most parts of the world experienced warming particularly over the land areas. Canada and Russia thus experienced more warming in the 20th century, compared to other regions around 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 middle of the data streams, but it requires data at both 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.

We used `lm(temp1[i,243: 1442]   timemo1, na.action=na.omit)` to treat the missing data between the beginning month and the end month. The missing data have been replaced by NA. The R command `na.action=na.omit` means that the missing data are omitted in the regression, and the fitted data at the missing data's time locations are omitted too and are not outputted. One can use another command `lm(temp1[i,243: 1442]   timemo1, na.action=na.exclude)` to do linear regression with missing data. The slope and intercept results computed by the two commands are the same. The only difference is that the latter outputs NA for the fitted data at the missing data's time locations. For example,

```
x=1:8
y=c(2,4,NA,3,6.8,NA,NA,9)
fitted(lm(y ~ x, na.action=na.exclude))
#   1    2    3    4    5    6    7    8
#2.08 3.04   NA 4.96 5.92   NA   NA 8.80
##
fitted(lm(y ~ x, na.action=na.omit))
#   1    2    4    5    8
#2.08 3.04 4.96 5.92 8.80
```

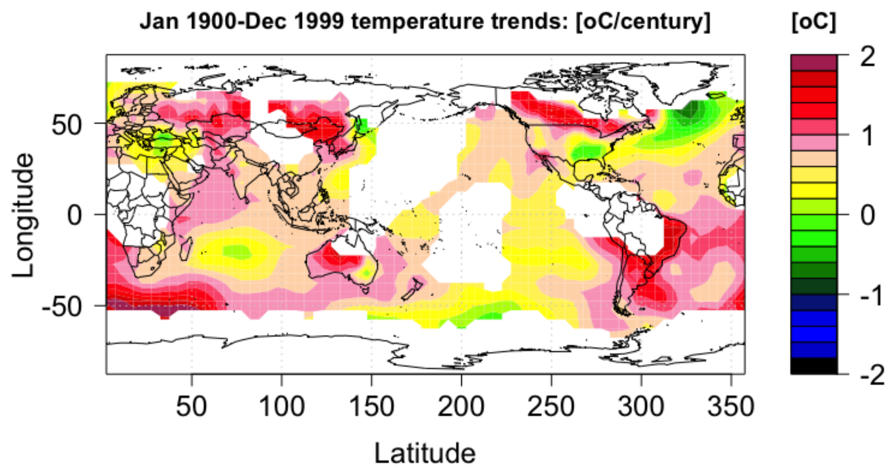


Fig. 2.11 Linear trend of SAT from January 1900 to December 1999. The trend was calculated for each grid box using the NOAAGlobalTemp data, and the procedure required that the box did not have missing data for the first month (January 1900) and the last month (December 1999). The white regions mean that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

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)
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)})
```

2.4.2 20th century temperature trend computed under a relaxed condition

If we relax our trend calculation condition and allow a trend to be computed for a grid box when the box has less than one third ot its data missing, then the trends can be computed for more grid boxes. Figure 2.12 shows the trend map computed under this relaxed condition.

Figure 2.12 uses °C per decade as the unit, while Fig. 2.11 uses °C per century. The patterns of the two figures are consistent, which implies that the relaxed condition for trend calculation has not led to spatially inconsistent trends. Thus, Fig. 2.12 can be regarded as an accurate spatial extension of Fig. 2.11.

Figure 2.12 can be generated by the following R code.

```
#Trend for each box for the 20th century: Version 2: Allow 2/3 of data
```

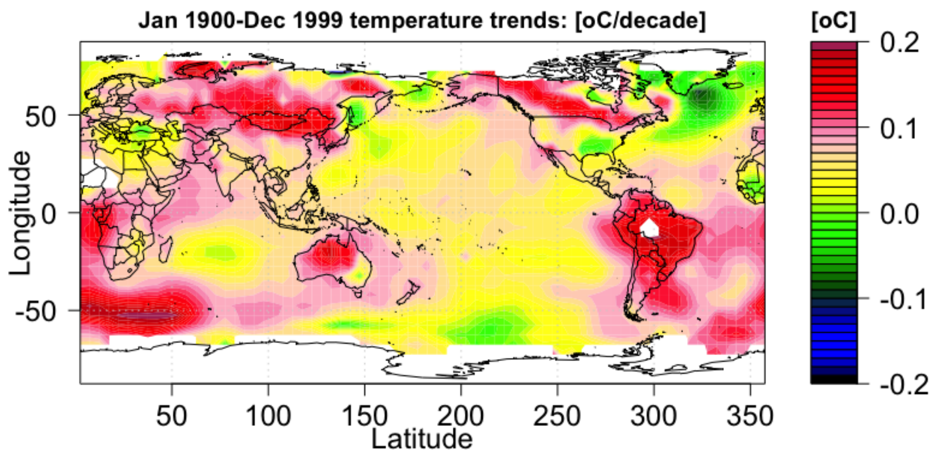


Fig. 2.12 Linear trend of SAT from January 1900-December 1999. The trend was calculated for each grid box using the NOAAGlobalTemp data when the box has less than 1/3 of data missing.

```
#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','maroon'),interpolate='spline')
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=TRUE);grid()},
               key.title=title(main="[oC]"),
               key.axes={axis(4, cex.axis=1.5)})
```

2.4.3 Trend pattern for the four decades of consecutive warming: 1976-2016

Our recent period of long-term rapid warming (four decades from 1976-2016) exhibits a warming that is greater than the last long-term warming from the 1910s to the early 1950s, which also lasted about four decades. Figure 2.13 shows the strong global warming trend from January 1976 to December 2016. It shows that during this period, the world became warmer on every continent except Antarctica.

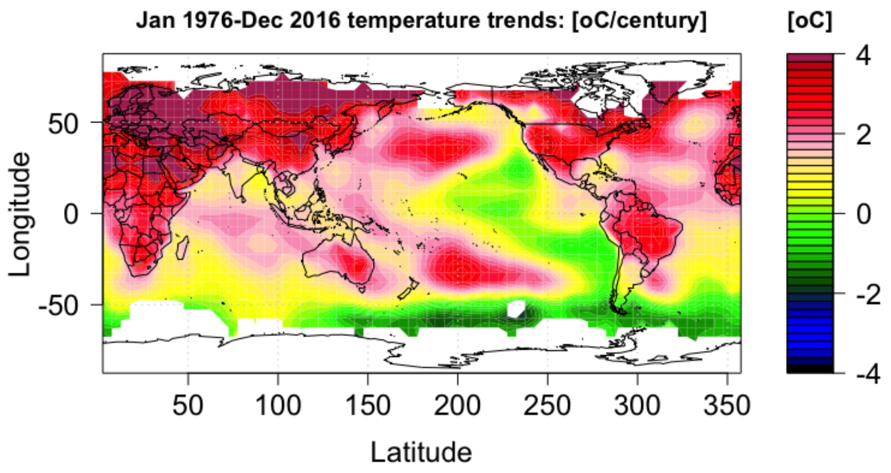


Fig. 2.13 Linear trend of SAT from January 1976-December 2016. The white regions mean insufficient amount of data.

The trend data for Fig. 2.13 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. 2.13 is almost identical to that for the 20th century trend of Fig. 2.12 and is omitted here.

References

[1] Huang, B., V.F. Banzon, E. Freeman, J. Lawrimore, W. Liu, T.C. Peterson, T.M. Smith, P.W. Thorne, S.D. Woodruff, H.M. and Zhang (2015): Extended reconstructed sea surface temperature version 4 (ERSST. v4). Part I: upgrades and intercomparisons. Journal of Climate, 28, 911-930.

[2] Karl, T.R., A. Arguez, B. Huang, J.H. Lawrimore, J.R. McMahon, M.J. Menne, T.C.Peterson, R.S. Vose, H.M. and Zhang (2015): Possible artifacts of data biases in the recent global surface warming hiatus. Science, 348,1469-1472.

[3] Smith, T.M. and R.W. Reynolds (2003): Extended reconstruction of global sea surface temperatures based on COADS data (18541997). Journal of Climate, 16,1495-1510.

Exercises

2.1 Following the R code for generating Fig. 2.6 for the monthly global average SAT anomalies, write an R code to generate a similar figure but for the Northern Hemisphere's SAT anomalies from January 1880 to December 2016, based on the gridded 5-deg NOAAGlobalTemp.

2.2 Compute and plot the spatial average of the annual mean SAT for the Northern Hemisphere from 1880 to December 2016.

2.3 Do the same as the previous problem, but for the Southern Hemisphere.

2.4 Plot and compare the maps of the January SAT anomalies' linear trends from 1948 to 2016 based on the gridded January SAT anomalies around the 1971-2000 climatology period for two datasets: the NCEP/NCAR Reanalysis data and the NOAAGlobalTemp data. Use 200-500 words to describe your results.

2.5 (a) Plot the time series of the spatially averaged annual mean SAT anomalies for the contiguous United States using the NOAAGlobalTemp data from 1880 to 2016.
(b) Add a linear trend line to the time series plot in (a). Mark the trend value on the figure with the unit [$^\circ$C per century].

3 R Graphics for Climate Science

This chapter is an introduction to the basic skills needed to use R graphics for climate science. These skills are sufficient to meet most needs for climate science research, teaching and publications. We have divided these skills into the following categories:

(i) Plotting multiple data time series in the same figure, including multiple panels in a figure, adjusting margins, and using proper fonts for text, labels, and axes;
(ii) Creating 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

Chapter 3 already showed how to plot a simple time series using `plot(xtime, ydata)`. Climate science often requires one to plot two different quantities, such as two 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 following code plots a figure containing 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)
#Plot two legends at the same time make it difficult to adjust the font size
#because of different scale
```

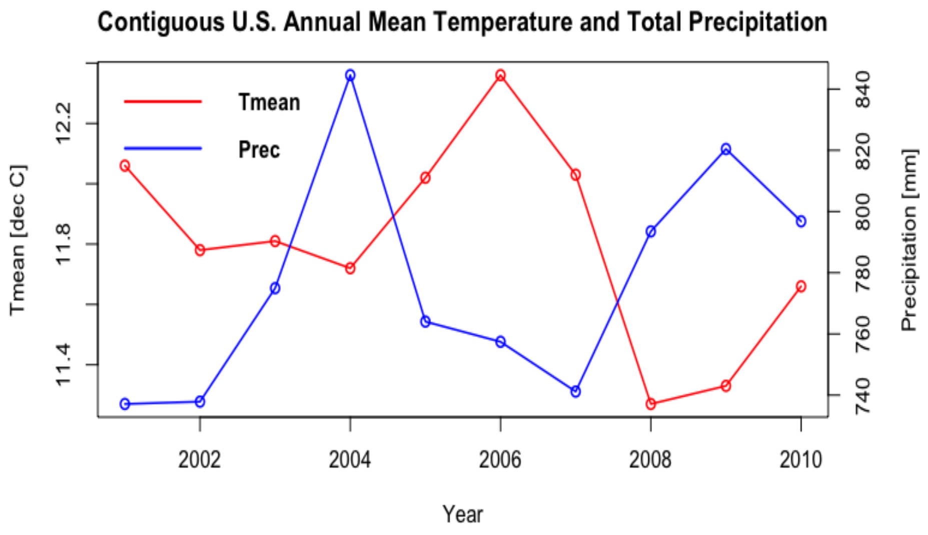


Fig. 3.1

Contiguous United States annual mean temperature and annual total precipitation.

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

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

R has the flexibility to create plots with specific margins, mathematical symbols for text and labels, text fonts, text size, and more. R also allows one to merge multiple figures. These capabilities are often useful in producing a high-quality figure for presentations 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 the y-axis, 3 is for the top, and 1 is for the 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.

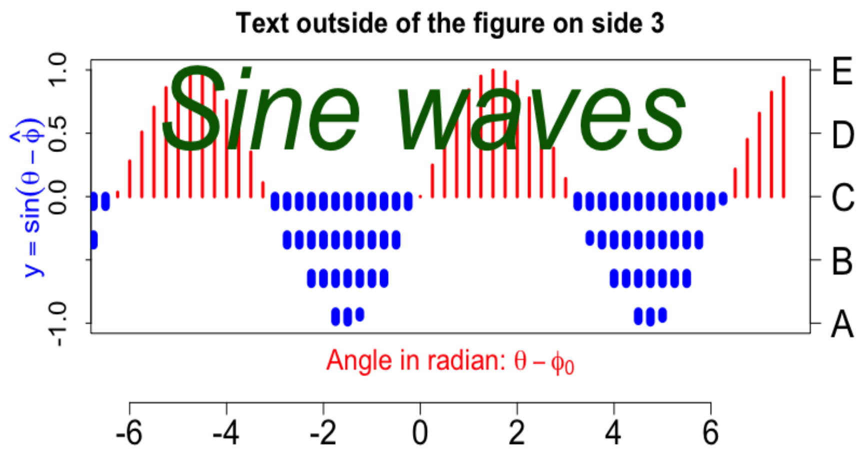


Fig. 3.2 Set margins, insert mathematical symbols, and write text outside a figure.

```
#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")
```

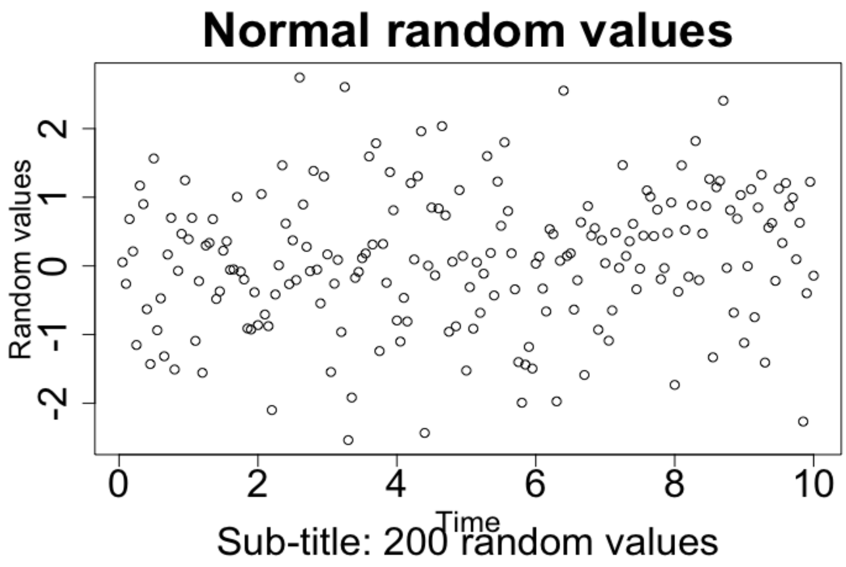


Fig. 3.3 Adjust font size, axis labels space, and margins.

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

Similar to using `cex.axis=1.8` to change 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, the main title, and the 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, tick 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 code 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 plot the global average annual mean surface air temperature (SAT) from 1880 - 2016 using the above plot functions (see Fig. 3.4). The data is from the NOAAGlobalTemp dataset
`https://www.ncdc.noaa.gov/data-access/marineocean-data/`
`noaa-global-surface-temperature-noaaglobaltemp`
We write the data in two columns in a file named `NOAATemp`. The first column is the year, and the second is the temperature anomalies.

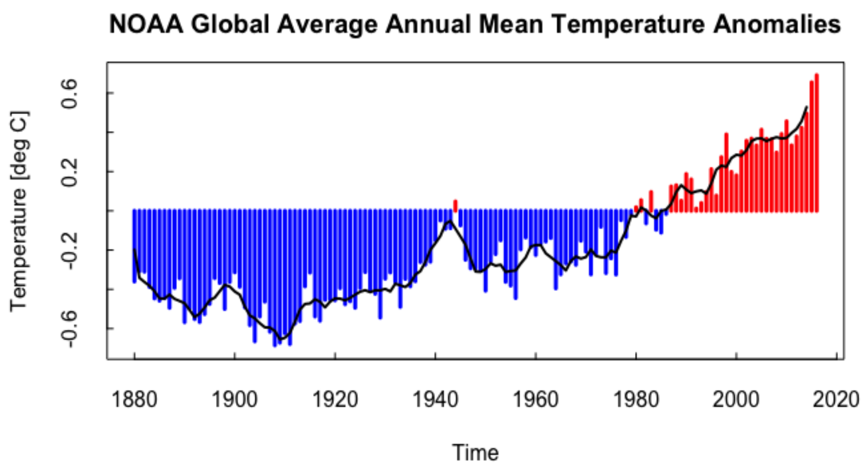


Fig. 3.4 Global average annual mean SAT based on the United States' NOAAGlobalTemp data
.

Figure 3.4 can be generated by the following R code.

```
#A fancy plot of the NOAAGlobalTemp time series
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.

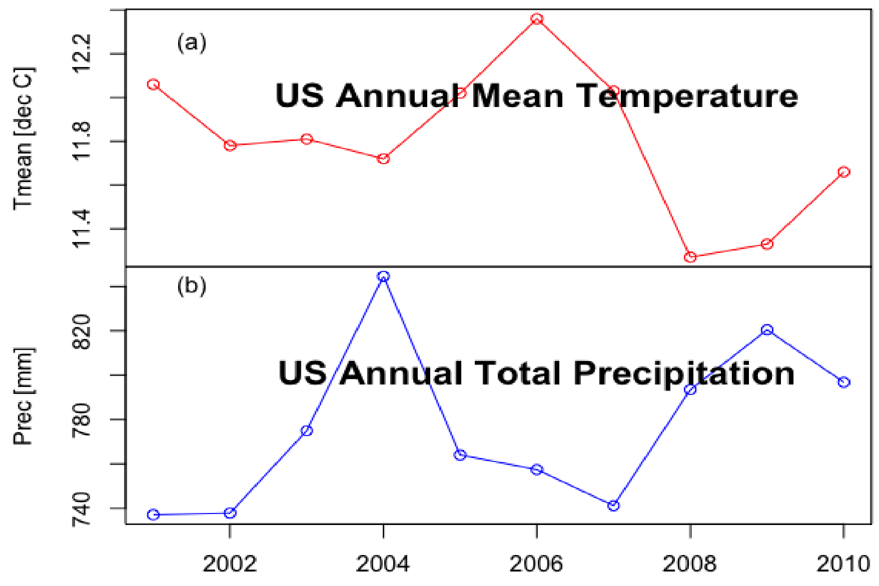


Fig. 3.5 (a) Contiguous United States annual mean temperature; and (b) annual total precipitation.

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 expecting the default setup, R may still use the previous setup. One can remove the R "memory" by

```
rm(list=ls())
plot.new()
```

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.

```
layout(matrix(c(1,1,2,3), 2, 2, byrow = TRUE),
       widths=c(3,3), heights=c(2,2))
plot(sin,type="l", 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 a layout and overwrites any other layout.

3.2 Contour color maps

3.2.1 Basic principles for an R contour plot

The basic principles for an 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 a color map for a climate parameter;
(ii) (x, y, z) coordinates data or a 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 $5° \times 5°$ 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
```

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)
```

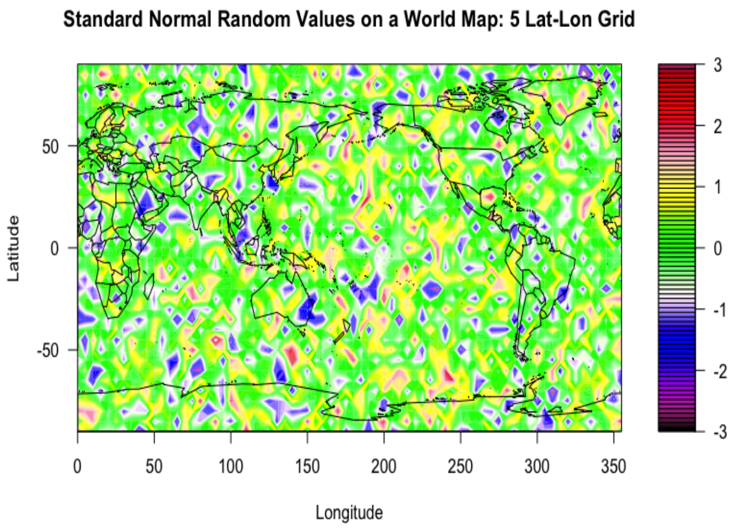

Standard Normal Random Values on a World Map: 5 Lat-Lon Grid
Latitude
Longitude
50
0
-50
0
50
100
150
200
250
300
350
3
2
1
0
-1
-2
-3

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

```
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()})
```

3.2.3 Plot contour maps from climate model data in NetCDF files

Here we show how to plot a downloaded netCDF NCEP/NCAR Reanalysis dataset 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 that have "assimilated" (i.e., been constrained by) observed data. The reanalysis output is the complete space-time gridded data. Reanalysis data in a sense is still model data, although some scientists prefer to regard the reanalysis data as dynamically interpolated observational data because the assimilation of observational data has taken place. Gridded observational data in this context may thus be the interpolated results from ob-

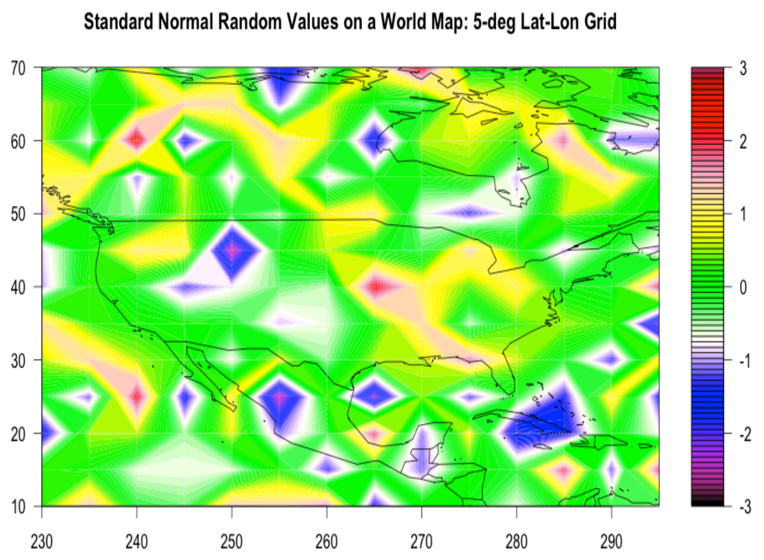

Standard Normal Random Values on a World Map: 5-deg Lat-Lon Grid
70
60
50
40
30
20
10
230
240
250
260
270
280
290
3
2
1
0
-1
-2
-3

Fig. 3.7 Color maps of standard normal random values $5° \times 5°$ grid over Canada and USA.

servational data which have been adjusted in a physically consistent way with the assistance of climate models. The data assimilation system is a tool to accomplish such a data adjustment process correctly.

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
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)
```

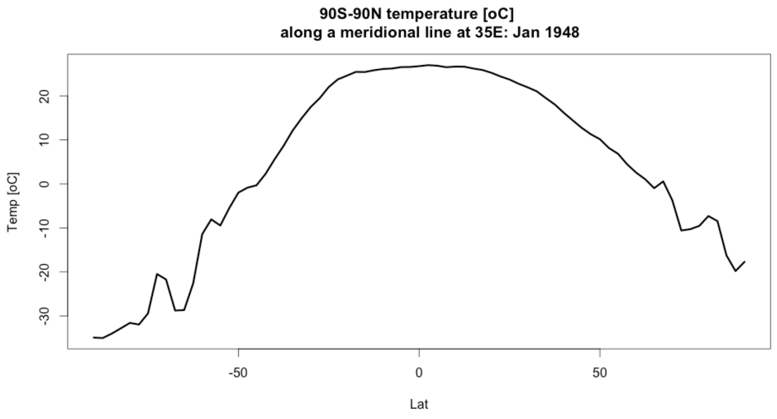


Fig. 3.8 The surface air temperature along a meridional line at 160°E over the Pacific.

Here, our first example is to plot the temperature variation in the meridional (i.e., north-south) direction from pole to pole, for a given longitude.

Next we plot the global color contour map showing the January temperature climatology as the average of the January temperature from 1948 to 2015, plus the surface air temperature of January 1983, and finally its anomaly calculated as the difference defined as the January 1983 data minus the January climatology. The R code is below, and the results 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)
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 must arrange Lat and Lon this way.
#Correspondingly, 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 the center, China on the right, and USA on the left. However, our map should
#have the Pacific at the center, and USA on the 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]"))
```

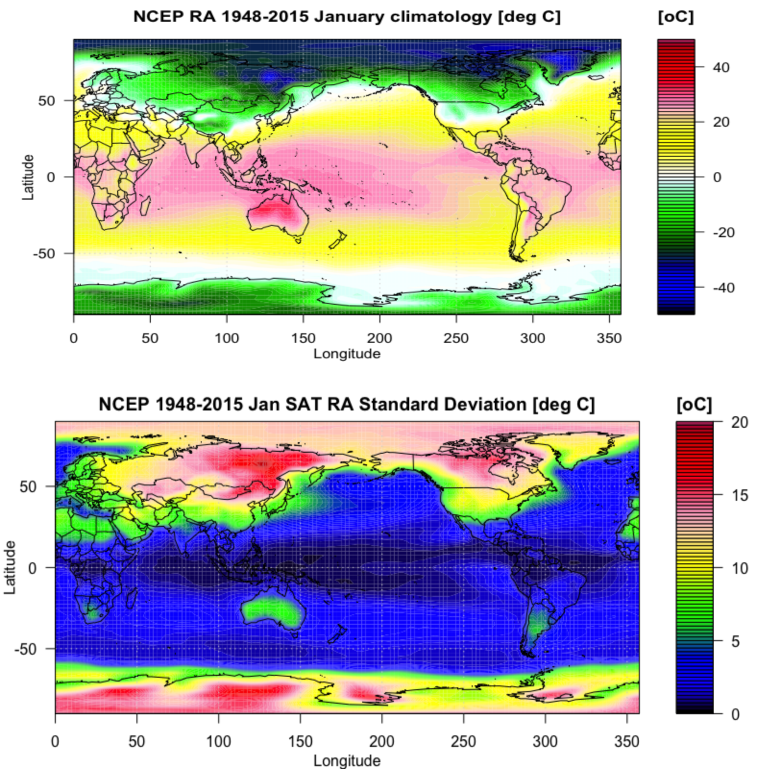

NCEP RA 1948-2015 January climatology [deg C]
[oC]
Latitude
Longitude
NCEP 1948-2015 Jan SAT RA Standard Deviation [deg C]
[oC]
Latitude
Longitude

Fig. 3.9 NCEP Reanalysis January climatology (upper panel) computed as the January temperature mean from 1948-2015. Lower panel shows the standard deviation of the same 1948-2015 January temperature data.

3.2.3.2 Plot data for displaying climate features

The next figure is the January 1983 temperature. The 1982-83 winter is noteworthy because of a strong El Niño event. However, the full temperature field depicted in Fig. 3.10 cannot show the El Niño 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 equatorial area and cold in the polar areas. El Niño is a phenomenon of climate anomalies: the temperature over the eastern tropical Pacific is warmer than normal, sometimes by as much 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]"))
```

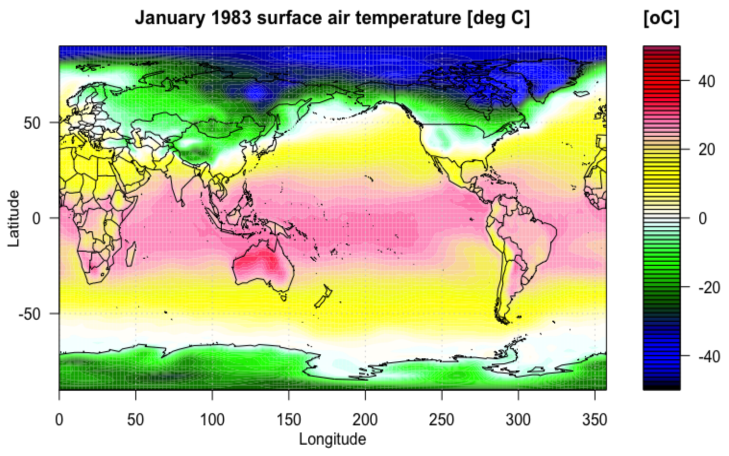


Fig. 3.10 NCEP Reanalysis temperature of January 1983: an El Niño event.

To see the El Niño, we compute the temperature anomaly, which is the January 1983 temperature minus the January climatology. A large tongue-shaped region over the eastern tropical Pacific appears with temperatures up to almost 6°C warmer than the climatological average temperatures (Fig. 3.11). This is the typical El Niño 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]"))
```

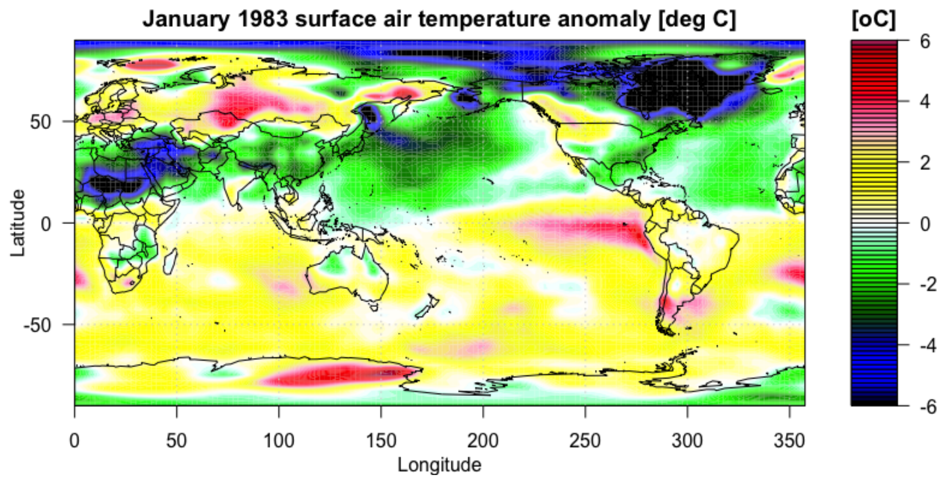


Fig. 3.11 NCEP Reanalysis temperature anomaly of January 1983, showing the eastern tropical Pacific's El Niño warming tongue.

Sometimes one needs to zoom in to a given latitude-longitude box of the above maps, in order to see the detailed spatial climate pattern over the region. For example, Fig.3.12 shows the January 1983 SAT anomalies over Pacific and North America. The El Niño pattern over the Pacific and El Niño's influence over North America are much more clear than in 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, mapdiff,
                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 American region can be generated in a similar way by changing the `xlim` and `ylim`: (130W, 60W) and (20N,60N).

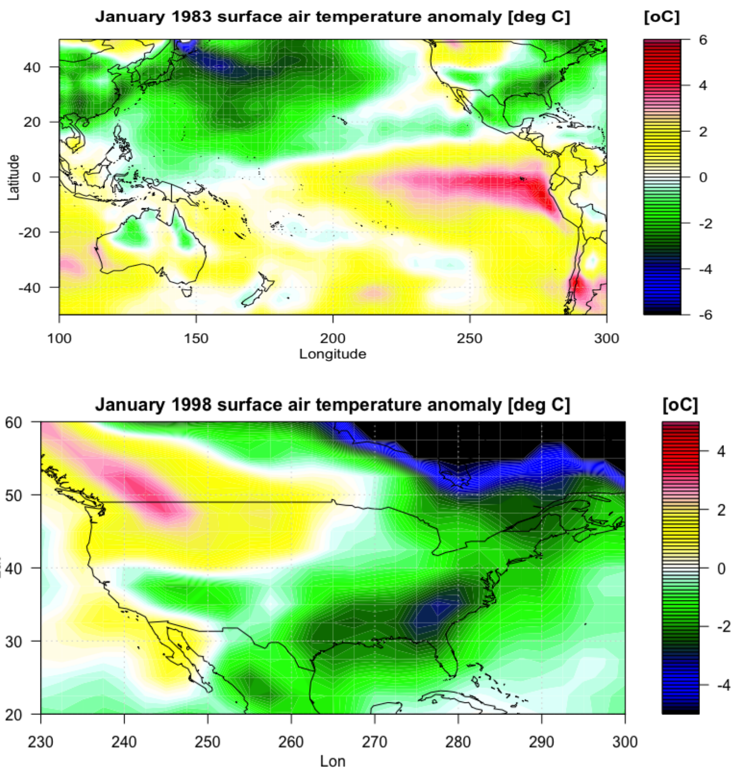

January 1983 surface air temperature anomaly [deg C]
[oC]
Latitude
Longitude
January 1998 surface air temperature anomaly [deg C]
[oC]
Lat
Lon

Fig. 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 geostrophic 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)
```

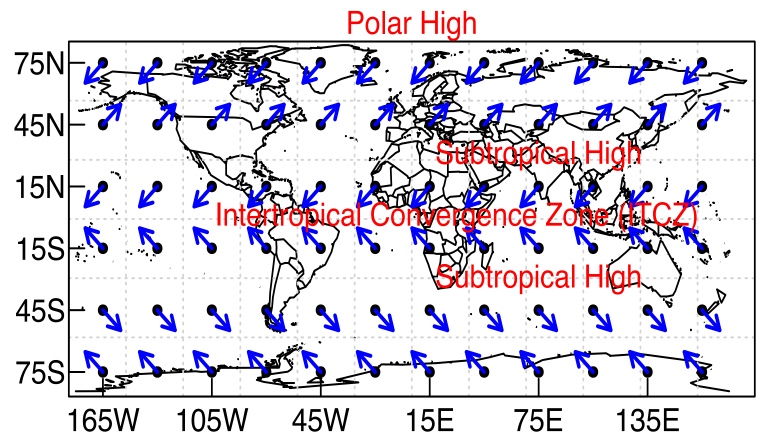


Fig. 3.13 Vector field of the ideal geostrophic wind field.

```
lon<-rep(seq(-165,165,len=12),each=6)
x<-lon
y<-lat
#u<- rep(c(-1,1,-1,-1,1,-1), each=12)
#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 (i.e., the surface wind data over the global ocean) is used as an example. The procedure is described from the data download to the final product of a wind field. The NOAA wind data were generated from multiple satellites observations, such as QuikSCAT, SSMIs, TMI, and AMSR-E , on a global at $1/4° \times 1/4°$ grid with a time resolution of 6 hours.

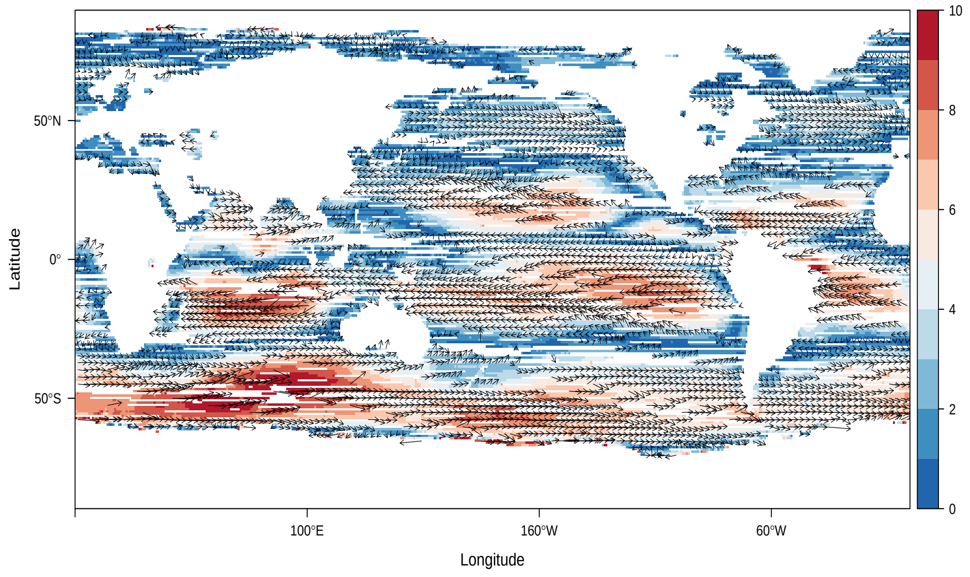


Fig. 3.14 The NOAA sea wind field of 1 January 1995: UTC00Z at $1/4° \times 1/4°$ resolution.

```
#Plot the wind field over the ocean
#Ref: https://rpubs.com/alobo/vectorplot
#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` `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 Graphics" (gg). `ggplot` can generally produce graphic-artist-quality default output and can make plotting complicated data easy 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 includes many maps and datasets that are useful in climate science.

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 strongly advocate the use of ggplot, a non-expert in R may remain with the regular R codes to produce plots that might be sufficient for his or her applications. However, ggplot is always a good resource if a figure cannot be generated by the usual R plot. Many good ggplot tutorial materials and examples are online and can be easily found with a search engine such as Google.

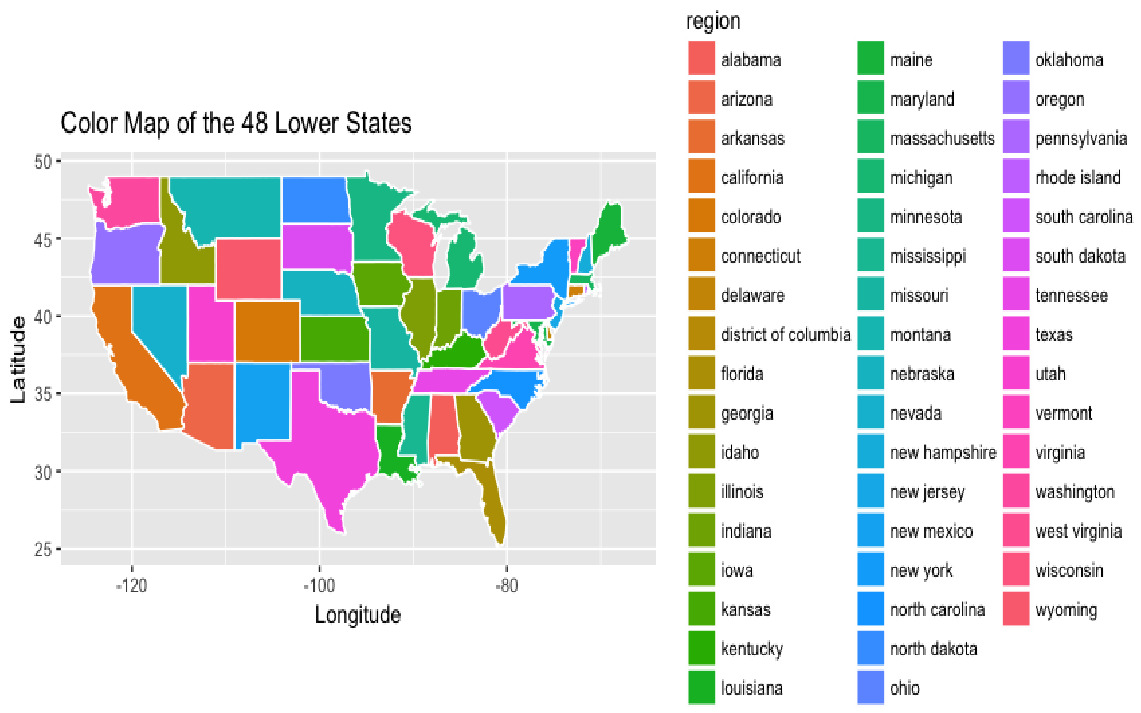


Fig. 3.15 "Lower 48" contiguous states of the United States.

References

[1] 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`

[2] 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.

Exercises

3.1 Use R to plot the temperature and precipitation anomaly time series from the NCEP Reanalysis data for the grid boxes of Tahiti and Darwin. Put the four time series on the same figure, and explain the their behaviors during the El Niño and La Niña periods.

3.2 Use R and NCEP Reanalysis data to display the El Niño temperature anomaly for January 2016. Find the latitude and longitude of the grid box on which the maximum temperature anomaly of the month occurred? What is the maximum anomaly?

3.3 Use R to compute the 1971-2000 climatology from the NCEP Reanalysis' annual mean temperature data for each grid box. Plot the climatology map.

3.4 Use R to compute the 1948-2010 standard deviation from the NCEP Reanalysis' annual mean temperature data for each grid box. Plot the standard deviation map.

3.5 Use R to plot the map of the North America and use arrows to indicate the Alaska Jet Stream.

4 Advanced R Analysis and Plotting for Climate Data

The empirical orthogonal function (EOF) method is a commonly used tool for climate data analysis in modern days. EOFs show spatial patterns of climate data, such as the El Niño warm anomaly pattern of the eastern Tropical Pacific. The corresponding temporal patterns are called principal components (PC). Thus, the EOF analysis is also called the PC analysis. We describe the EOFs and PCs as a natural space-time decomposition of a space-time data matrix using the singular value decomposition (SVD) method and a simple R command `svd(datamatrix)`. This is different from traditional approach of an eigenvalue problem of a covariance matrix. This chapter provides recipe-like R codes and their explanations for EOF and PC calculations. It also describes temporal trend calculations of climate data and the trend influences on the first a few EOFs.

4.1 Ideas of EOF, PC and variances from SVD

SVD described in Chapter 5 decomposes a space-time climate data matrix A into three parts: spatial patterns U, temporal patterns V, and energy D, i.e.,

$$A = UDV^t. \tag{4.1}$$

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 1. The pattern is called an empirical orthogonal function (EOF), and the method was introduced into meteorology in the 1950s by Edward Lorenz, who is well known for his contributions to theoretical meteorology, 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). For example, for the SLP data of Darwin and Tahiti analyzed in Chapter 5, the EOFs are the SLP patterns at the two locations, and the PCs are the 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 Niño. The corresponding PC1's positive peaks shows the time when El Niño actually happened. PC1's negative peaks indicates the time of La Niña.

EOF1 and PC1 are both referred to mode 1 and has "energy": 31.35^2, which is the variance of $A\mathbf{u}_1$, i.e., the data's projection onto 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.

The elements of U and V are dimensionless and consist of othonormal vectors. The dimension of $d_i (i = 1, 2)$ is the same as that of the elements of the data matrix A, and measures the "amplitude" that is proportional to the system's variance or "energy." The dimension of d_1^2 is the square of the dimension of A, i.e., the dimension of the variance. Variance is a measure of "energy" of the El Niño Southern Oscillation system.

4.2 2Dim spatial domain EOFs and 1Dim temporal PCs

The spatial fields of many climate data applications are two-dimensional, or 2Dim for short. The corresponding EOFs are over a 2Dim domain on the Earth's surface, and the corresponding PCs are on a time interval. This section describes the basic concepts of using SVD to compute EOFs and PCs and using R graphics to show them. We use a simple synthetic data set to illustrate the procedures. The next section will feature using real climate data.

4.2.1 Generate synthetic data by R

The spatial domain is $\Omega = [0, 2\pi] \times [0, 2\pi]$, and the 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 $\psi_1(x, y)$ 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

$$\int_\Omega d\Omega \psi_k^2(x, y) = 1, \quad k = 1, 2 \tag{4.6}$$

$$\int_\Omega d\Omega \psi_1(x, y)\psi_2(x, y) = 0 \tag{4.7}$$

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 value of time t, 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 and a small-scale wave. When the time is 10, the figure shows only the large scale wave, and the small-scale's influence is negligible.

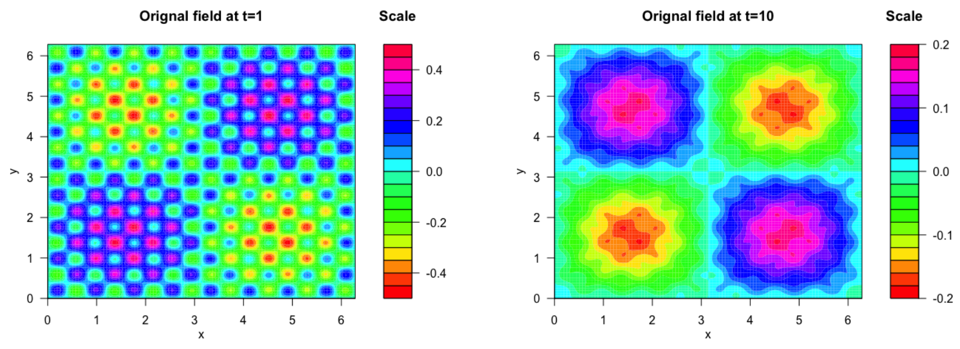


Fig. 4.1 The $z(x, y, t)$ function as $t = 1$ and $t = 10$.

This figure 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
filled.contour(x,y,mydat[,,1], color.palette =rainbow,
               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 matrix 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 3Dim 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]))}
```

Applying SVD on this space-time data is shown below.

```
da2<-svd(da1)
uda2<-da2$u
vda2<-da2$v
dda2<-da2$d
dda2
#[1] 3.589047e+01 1.596154e+01 7.764115e-14 6.081008e-14
#The first mode variance 36/(36+16)= 69%
```

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 $\psi_2(x, y)$. This 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. These two SVD eigenvalues will then be close to each other. Consequently, the EOFs, as eigenfunctions, will have large differences from the original true orthonormal basis functions. This is quantified by the North's rule-of-thumb, which states that both EOFs will have large errors, which are inversely proportional to the difference between the two eigenvalues. Thus, when the two eigenvalues have a small difference, the two corresponding eigenfunctions will have large errors due to mode mixing. 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. They form a 2-dimensional eigenspace, which has infinitely many eigenvectors due to the mixture of the two eigenvectors. A physically meaningful eigenvector should have no ambiguity, and infinitely many eigenvectors imply uncertainties, large errors, and no physical interpretation.

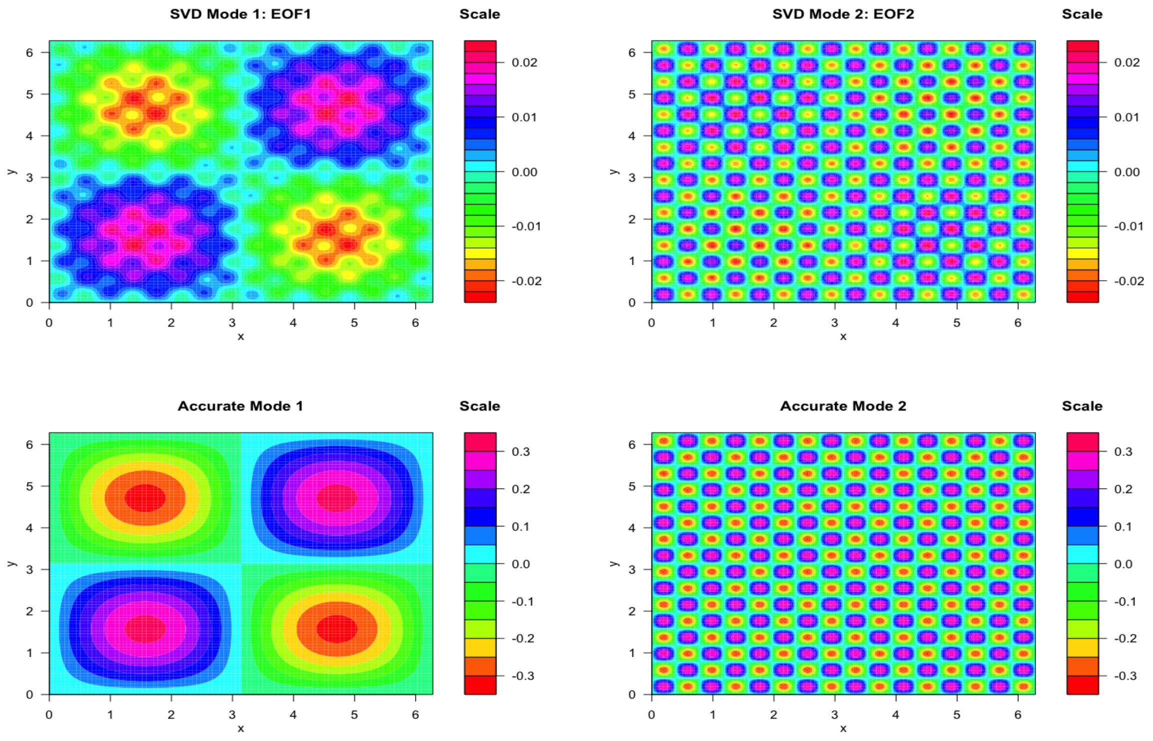


Fig. 4.2 The first row shows two EOFs from SVD, and the second row shows two orthonormal basis functions on the xy-domain: $\phi_1(x, y) = -(1/\pi) \sin x \sin y$, and $\phi_2(x, y) = (1/\pi) \sin 8x \sin 8y$.

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)
                 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: 69% 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: 31% 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("Mode 1 coefficient: 80% 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("Mode 2 coefficient: 20% variance"),
  col="purple", bty="n",text.col="purple")
```

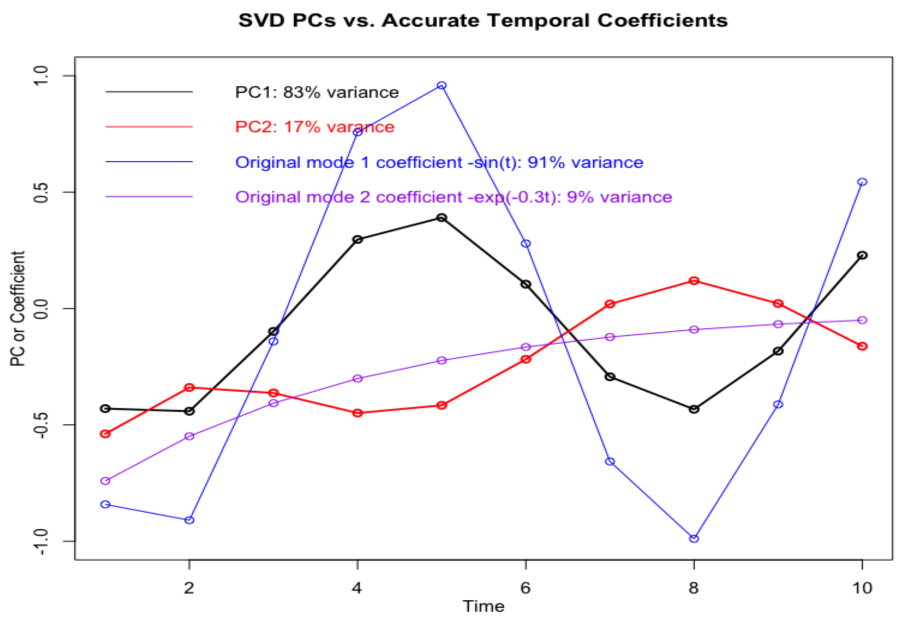


Fig. 4.3 Two principal components (PCs) from SVD approximation and two accurate time coefficients: $-\sin t$ and $\exp(-0.3t)$.

PC1 demonstrates sinusoidal oscillation, while PC2 shows a 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, i. e., the plus or minus sign is indeterminate.

PC1 and PC2 are orthogonal, but coefficients $c_1(t)$ and $c_2(t)$ are not orthogonal. This 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 tells us that the original data can be recovered from the EOFs, PCs and the variances by the following formula

$$z = UDV'. \tag{4.10}$$

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

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

The left panel of Fig. 4.2.2 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 reconstructed 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)})
```

The full recovery or the original field at time $t = 5$, shown in the right panel of Fig. 4.2.2, is virtually identical to 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. To be specific, the full recovery UDV^t may have non-negligible numerical digits truncation errors (single precision or double precision), which can cause large errors in the recovery results, when the high spatial variability is involved.

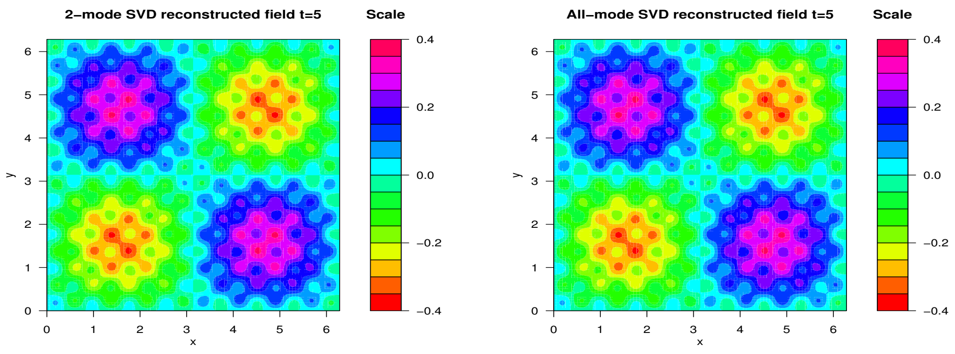


Fig. 4.4 Recovery of the original data using two modes (the left panel) and using all modes (the right panel).

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 the Internet. In climate research and teaching, data from netCDF files are often encountered. We shall download the data and make an EOF analysis. The example is the surface temperature data from the NCEP/NCAR Reanalysis I, which outputs the 2.5 degree monthly data from January 1948 to the present. We choose the most frequently used surface air temperature (SAT) field.

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 its size is 26MB for the data extending from January 1948 to October 2016. The spatial resolution is a 2.5-degree latitude-longitude (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")
Lat1 <- 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 check whether our downloaded data appear to be 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 at the left, and between -20°C and -10°C over the Arctic region at the right. We are thus reasonably confident that our downloaded data are correct and that the data values correctly correspond to their assigned positions on the latitude-longitude grid.

Figure **??** 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="o",
     xlab="Latitude", ylab="Temperature [oC]",
     main="90S-90N temperature [mm/day] along a meridional line at 180E: Jan 1948")
```

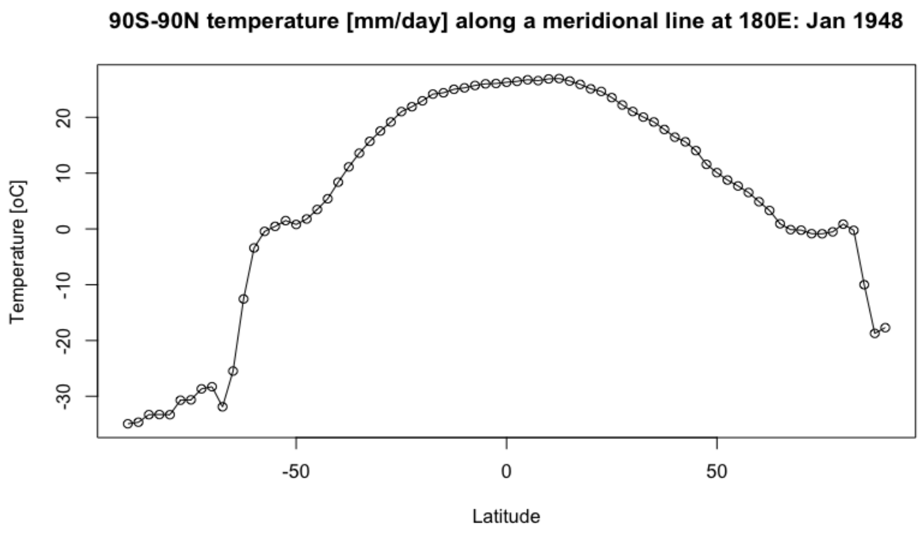


Fig. 4.5 The north-south distribution of SAT along the meridional line at 180°E for January 1948.

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 first to read the data and then to use it their 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 usig 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 resulting csv file's first column is latitude, the second is longitude, and the first row is the time mark for each month from January 1948 to October 2016. The csv file can be read and then used by Excel, R, Matlab, Python, and almost all other commonly encountered computer programs.

4.3.2.2 Climatology and standard deviation

Here we show a different way to compute the climatology and standard deviation, which were already described in the last chapter. The method described here is to use 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 of Asia, and the Andes region of South America. These regions are at relatively low latitudes but they experience very low temperatures due to their high altitudes, which may be 4,000 or more meters.

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

```
monJ=seq(1,816,12)
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 generated by the following R code.

```
#Plot Jan climatology
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]"),
               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()
int=seq(0,3,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]"),
               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 employed to carry out the NCEP/NCAR reanalysis. This error may be due to the models handling of a physically complex phenomenon, namely the sea ice and albedo feedback. The actually standard deviation might thus be smaller over the same region. This type of error highlights a caution that should be kept in mind when using reanalysis datasets. Combining a complex climate model with observational data can improve the realism of data sets, but it can also introduce a type of error that could not exist if no model were used.

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 data 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 for measuring a signal strength that may be attributed to a particular mode or modes. Further, what is often most important is the relative variance, i.e., the percentage of the variance attributable to a specific mode, relative to the sum of the variances in 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. This figure can be plotted by the following R code.

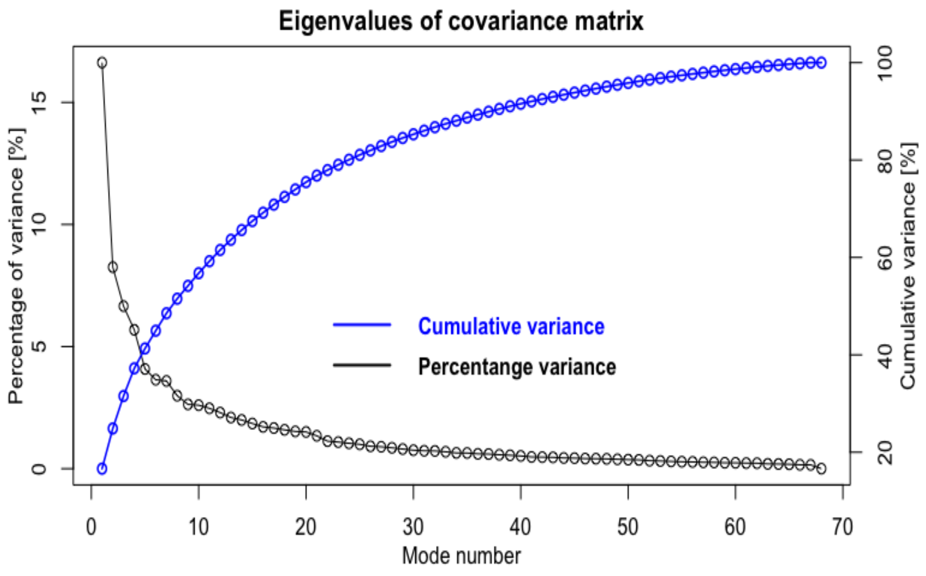


Fig. 4.6

Percentage variance, and the cumulative variance 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 columns. The first three EOFs and PCs are shown in Figs. 4.7-4.9, which may be generated by the following R code.

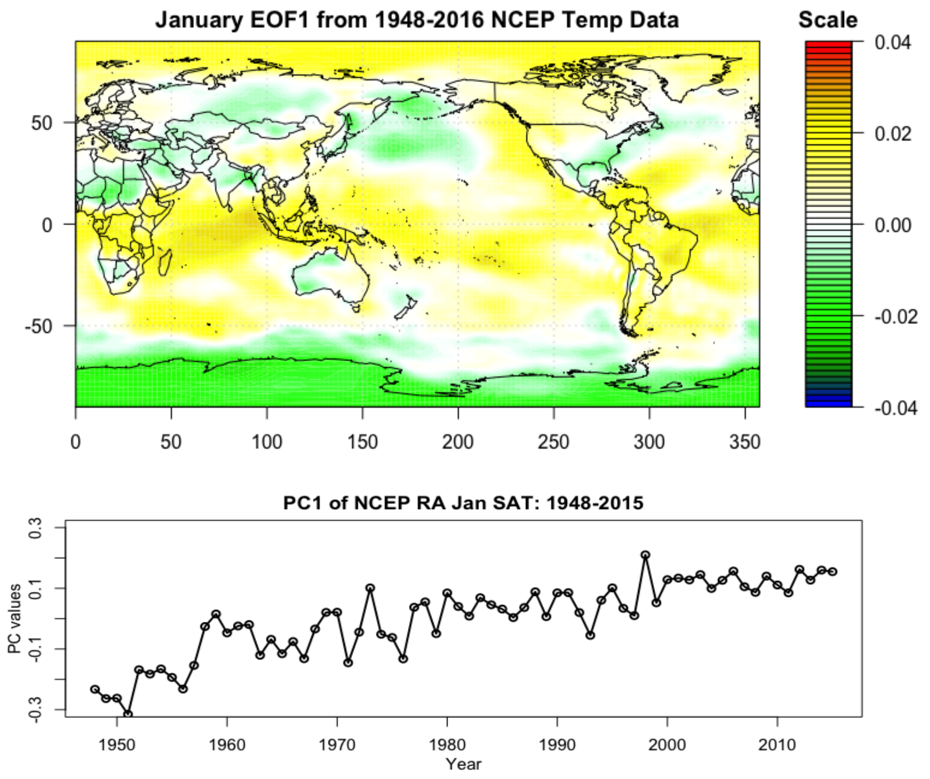


Fig. 4.7 The first EOF and PCs= from the January SAT's standardized area-weighted anomalies.

```
#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',
               '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"))
#
```

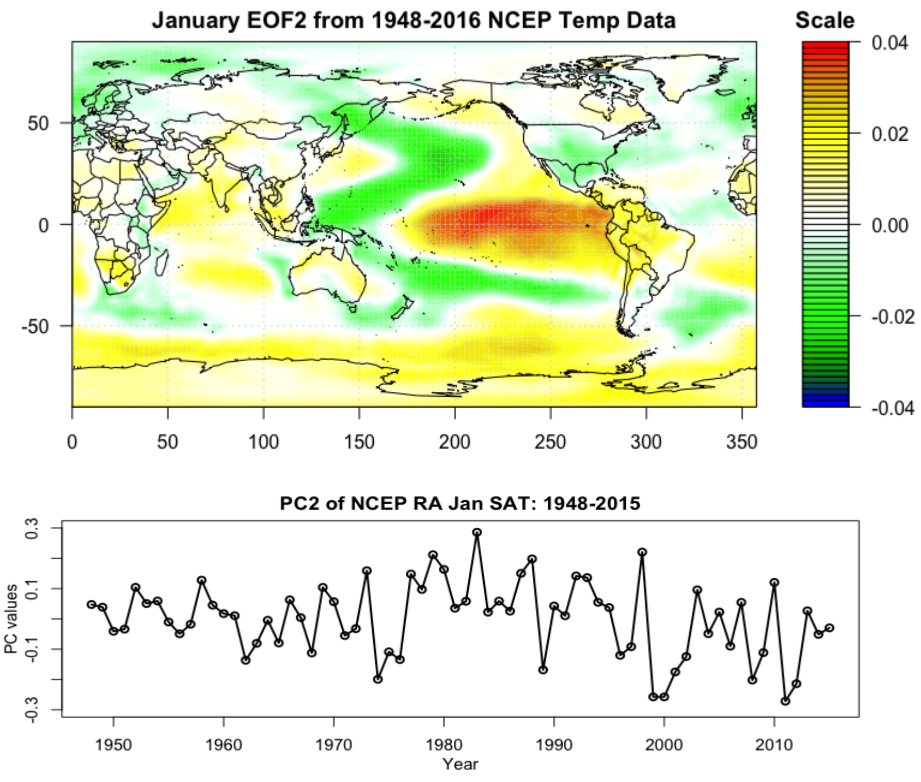


Fig. 4.8

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

```
#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))
```

Often, the first two or three EOFs and PCs will have some physical interpretations, because the higher modes' eigenvalues are too close to each other, and hence have a large amount of uncertainty.

In the case of the January SAT Reanalysis data here, PC1 shows an increasing trend. The corresponding EOF1 shows the spatially non-uniform pattern of temperature increasing.

EOF2 shows an El Niño pattern, with a warm tongue over the eastern tropical Pacific. PC2 shows the timpoeral variation of the the El Niño signal. For example, the January 1983 and 1998 peaks correspond to two strong El Niños.

EOF3 appears to correspond to a mode known as the Pacific Decadal Oscillation.

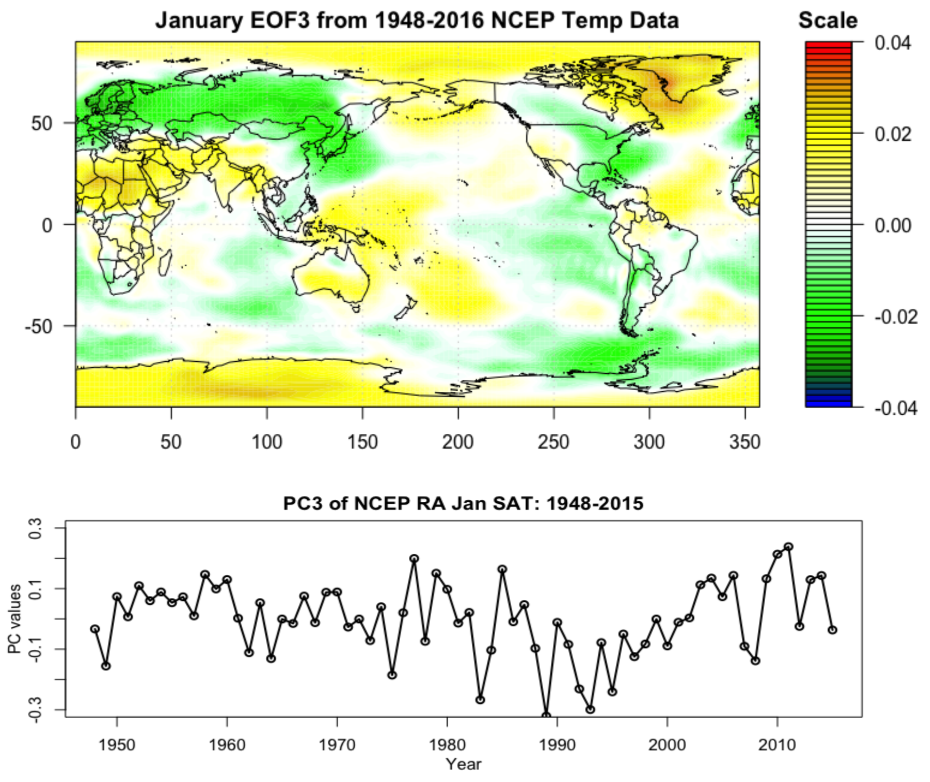


Fig. 4.9 The third EOF and PC from the January SAT's standardized area-weighted anomalies.

It shows a dipole pattern over the Northern Pacific. PC3 shows quasi-periodicity in this mode with a period of about 20 to 30 years.

4.3.2.4 EOFs from the de-trended standardized data

We can also de-trend the standardized anomaly data first and then compute the EOFs and PCs. As expected, the new EOF1 then will no longer be the trend pattern, rather it is the El Niño mode, i.e. the EOF2 of the non-detrended anomalies (see Figs. 4.10 and 4.11). This implies that the de-trending process has removed the EOF1 mode in the original non-de-trended data.

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

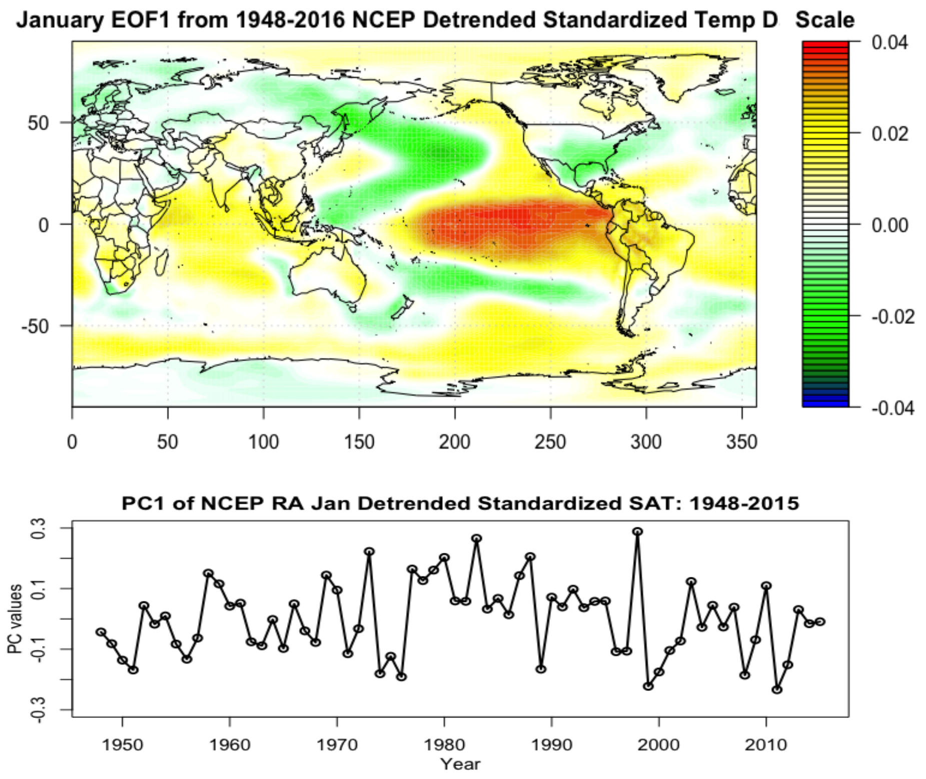


Fig. 4.10 The first EOF and PC from the January SAT's de-trended standardized area-weighted anomalies.

One can then use the EOF plotting code described in the previous 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 the non-de-trended EOF2. However, they are not exactly the same. Thus, the de-trending process is approximately similar to the EOF1 filtering, although not exactly the same. Similar statements can be made for the other EOFs and PCs.

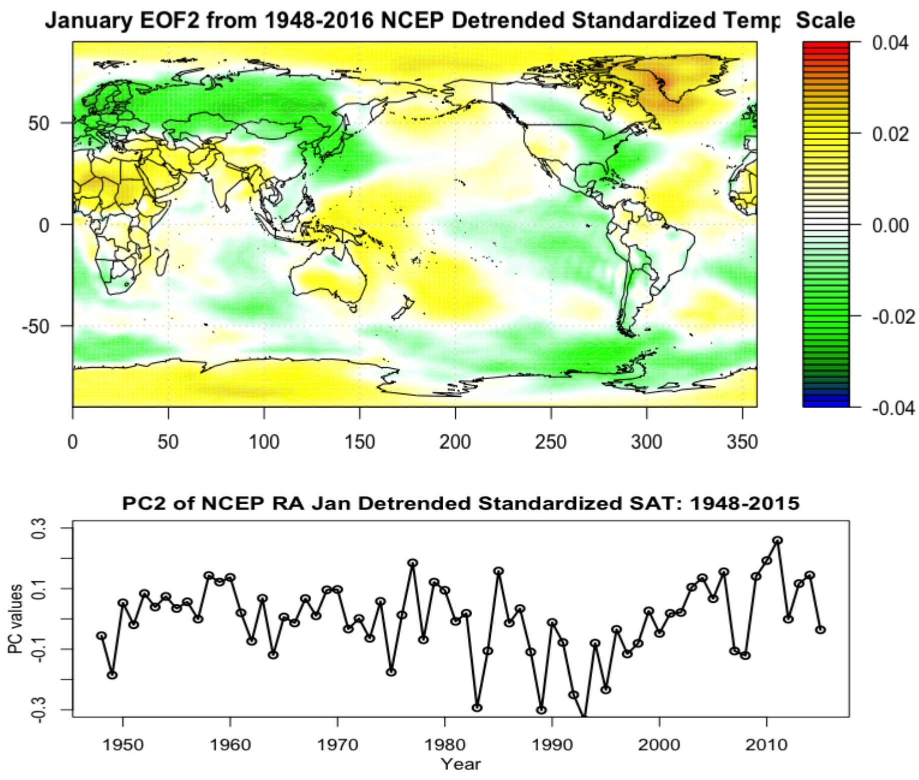


Fig. 4.11 The second EOF and PC from the January SAT's de-trended standardized area-weighted anomalies.

The first eigenvalue of the de-trended anomalies explains about 10% of the total variance, approximately equivalent to the 8% of the total 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 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

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

4.4 Area-weighted average and spatial distribution of trend

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

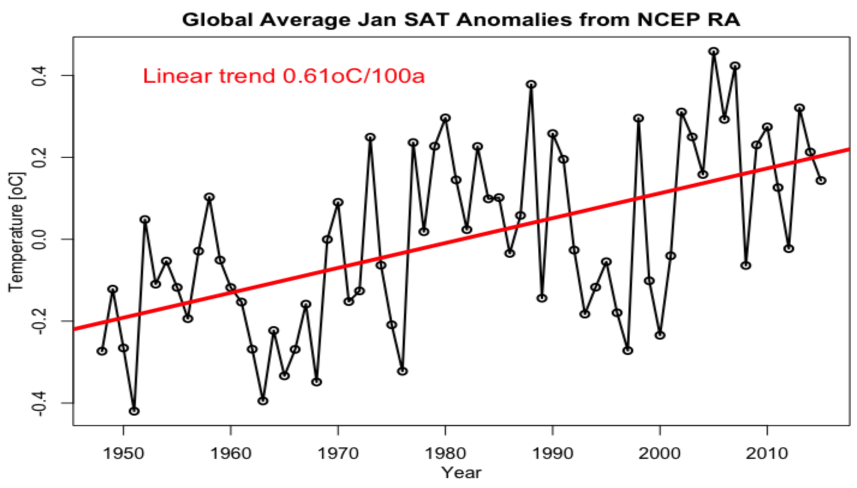


Fig. 4.12 The global area-weighted January SAT anomalies from 1948-2015 based on the NCEP/NCAR RA1 data.

```
#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)
#0.48oC/100a trend
abline(regSAT, col="red", lwd=4)
text(1965,0.35,"Linear trend 0.48oC/100a", col="red", cex=1.3)
```

Figs. 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: 0.53°C/100a for the global average, and 0.48°C/100a for PC1. Their correlation is 0.61, not as large 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 to 2015 based on the NCEP/NCAR Reanalysis' January temperature data. Each grid box has a time series of 68 years of January SAT anomalies from 1948 to 2015, and each grid box 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 codes make the trend calculation and plot.

```
#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"))
```

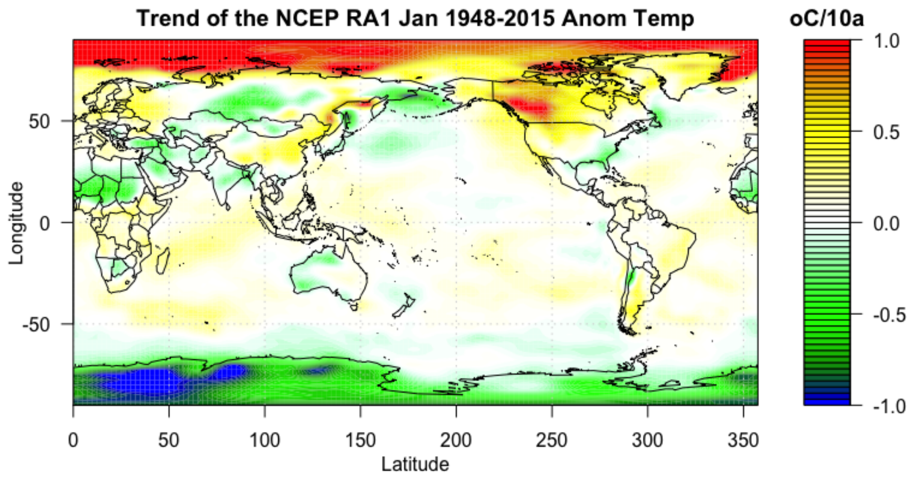


Fig. 4.13 Linear trend of NCEP Reanalysis' January SAT from 1948-2015.

Figure 4.13 shows that the trends are non-uniform. The trend magnitudes over land are larger than those over ocean. The largest positive trends are over the Arctic region, while the Antarctic region has negative trends. These trends may not be reliable since the reanalysis climate model has an amplified variance over the polar regions, another example of model deficiencies leading to erroneous information in the reanalysis produced by using that model.

Indeed, the spatial distribution of the trends appears similar to EOF1 of the non-de-trended 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 in fact is 0.97. This result implies that temperature's spatial patterns are more coherent than the temporal patterns, which is consistent with the existence of large spatial correlation scales for the monthly mean temperature field.

References

[1] Monahan, A.H., J.C. Fyfe, M.H. Ambaum, D.B. Stephenson, G.R. and North, 2009: Empirical orthogonal functions: The medium is the message. Journal of Climate, 22, 6501-6514.

[2] North, G. R., F. J. Moeng, T. J. Bell and R. F. Cahalan, 1982: Sampling Errors in the Estimation of Empirical Orthogonal Functions. Mon. Wea. Rev., 110, 699-706.

[3] Strang, G., 2016: Introduction to Linear Algebra, 5th edition, Wellesley-Cambridge Press, Wellesley, U.S.A.

Exercises

4.1 Use the SVD approach to find the EOFs and PCs for the Northern Hemisphere's January SAT based on the anomaly data from the NCEP/NCAR Reanalysis. Use the 1981-2010 January mean as the January climatology for each grid box. Plot the squared SVD eigenvalues against mode number for the first 30 modes. The suggested steps are below.
(a) Convert the Reanalysis data into a space-time data matrix.
(b) Extract the Northern Hemisphere's January data using proper row and column indices.
(c) Apply the SVD to the extract space-time matrix.
(d) Plot the squared eigenvalues.

4.2 Use R to plot the first three EOFs and PCs from the previous problem. Interpret the climatic meaning as much as you can, but limited 200-500 words.

4.3 Compute and plot the first three EOFs and PCs for the January SAT anomalies for the contiguous United States. Use these EOFs and PCs to describe the U.S. climate patterns, but limited 200-500 words. *[Hint: You may first use internet to find a grid mask table for the contiguous U.S. Use the table and R's* `which` *command to extract the U.S. data out of the January Northern Hemispheric data.]*

5 Climate Data Matrices and Linear Algebra

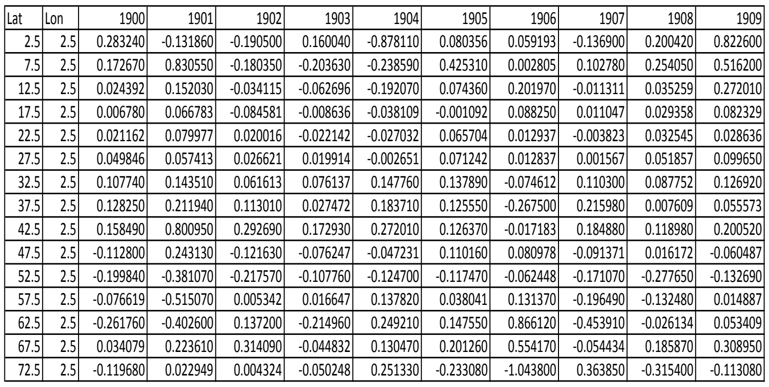
This chapter introduces matrix from the perspective of space-time climate data and emphasizes the singular value decomposition (SVD) that decomposes a space-time data matrix into three matrices: the spatial pattern matrix, "energy" matrix, and temporal pattern matrix. An extensive analysis is made for the sea level pressure data of Darwin and Tahiti and their optimal formation of a weighted Southern Oscillation Index. The chapter also contains the conventional and basic materials of linear algebra: matrix operations, linear equations, multivariate regression by R, and various applications, such as balancing the number of molecules for a chemical reaction equation.

5.1 Matrix as a data array

In general, a matrix is a rectangular array of numbers or symbols or expressions, which are called elements, arranged in rows or columns. A table such as that shown in Fig. 5.1 is a matrix, consisting of N rows and Y columns of numbers. Figure 5.1 shows the 10 ten-year annual precipitation anomalies from the year 1900 to 1909 for the 15 five-degree latitude-longitude boxes centered at $2.5°E$ longitude for different latitudes over the Northern Hemisphere ranging from latitudes $2.5°N$ to $72.5°N$. The matrix shown in Fig. 5.1 thus has 15 rows ($N = 15$), and 10 columns ($Y = 10$). An anomaly of a climate parameter is defined as its real value minus its normal value that is an average of 30 or more years.

Many other types of climate data can also be represented as matrices (which is the plural of matrix). Precipitation data [units: mm/day] at multiple stations and multiple days can also form a matrix, normally with stations [each marked by a station identifier, or station ID] represented in rows, and time [units: day] represented in columns. The daily minimum surface air temperature (Tmin) data for the same stations and the same days form another matrix. In general, a space-time climate data table always forms a matrix. Conventionally, the spatial locations correspond to the rows, and the time coordinate corresponds to the columns.

Another matrix example, taken from everyday life, is that the ages of members of an audience, sitting in a movie theater in chairs arranged in rows and columns, also form a matrix. The weights of these audience members form another matrix. Their bank account balances form still another matrix, and so on. Thus, a matrix is a data table, and extensive mathematical methods have been developed in the



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

Fig. 5.1

Annual precipitation anomalies data of the Northern Hemisphere at longitude $2.5°E$ [units: mm/day]. The annual total of the anomalies should be multiplied by 365.

20th century to study matrices. Computer software systems, such as R, have also been developed in recent years that greatly facilitate working with matrices.

This chapter will discuss following topics:

(i) Matrix algebra of addition, subtraction, multiplication, and division (i.e., inverse matrix));

(ii) Linear equations;

(iii) Space-time decomposition, eigenvalues, eigenvectors, and the climate dynamics interpretation of a space-time climate data matrix;

(iv) A matrix application example in balancing the mass in a chemical reaction equation by solving a set of linear equations; and

(v) A matrix application in multivariate linear regression.

5.2 Matrix algebra

Matrix algebra is quite different from the algebra for a few scalars of x, y, z as we learned in high school. For example, matrix multiplication does not have the commutative property, i.e., matrix A times matrix B is not always the same as matrix B times matrix A. This section describes a set of rules for doing matrix algebra.

5.2.1 Matrix equality, addition and subtraction

An $m \times n$ matrix A has m rows and n columns and can be written as

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, \tag{5.1}$$

or

$$A_{m\times n} = [a_{ij}], \tag{5.2}$$

or simply

$$A = [a_{ij}], \tag{5.3}$$

where $a_{ij}, i = 1, \cdots, m, j = 1, \cdots n$ are the mn elements of the rectangular matrix A, and $m \times n$ is often called the size or order of a matrix. We say that A is an m times n matrix. The elements of the matrix shown in Fig. 5.1 are the precipitation anomaly data, $m = 15$, and $n = 10$. So, Fig. 5.1 is a 15 times 10 matrix.

Matrix $A = [a_{ij}]$ is equal to matrix $B = [b_{ij}]$ if and only if $a_{ij} = b_{ij}$ for all the elements. That is, A is identical to B. We can understand this by considering the two identical photos. A black/white photo is a matrix of pixel brightness values. Two photos are identical only when the corresponding brightness values of the two photos are the same. So, when considering that a matrix is equal to another, we may regard the equality as two same-size photos, maps, or climate charts being identical to one another.

The matrix addition is simply the addition of the corresponding elements:

$$A + B = [a_{ij} + b_{ij}]. \tag{5.4}$$

Of course, two matrices can be added to each only when they have the same size $m \times n$. If the two matrices represent dimensional data, such as precipitation, then their units must be the same for corresponding elements to add. However, each element in a matrix can represent a different climate parameter. For example, a climate data matrix for San Diego for 24 hours may have its first row representing temperature, the second precipitation, the third atmospheric pressure, the fourth relative humidity, etc, while the first columns represent time from 1:00 (i.e., 1:00AM) to 24:00 (i.e., 12:00 AM).

Matrix subtraction is defined in a similar way:

$$A - B = [a_{ij} - b_{ij}]. \tag{5.5}$$

Example 1. Matrix subtraction:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} - \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix} = \begin{bmatrix} 0 & -1 \\ -2 & -1 \end{bmatrix} \tag{5.6}$$

The R code for the above matrix subtraction can be written as follows:

```
matrix(c(1,1,1,-1), nrow=2) - matrix(c(1,3,2,0), nrow=2)
#     [,1] [,2]
#[1,]    0   -1
#[2,]   -2   -1
```

One can use matrix subtraction to quantify the differences of climate between two space-time domains when the climate data for each domain are in a matrix form.

5.2.2 Matrix multiplication

5.2.2.1 A row vector times a column vector

The single column n-row matrix is often called an n-dimensional vector, or an n-dimensional column vector. Similarly, one can define an n-dimensional row vector as a single row n-column matrix.

A row vector of n elements times a column vector of the same number of elements is equal to a scalar, which is the sum of the products of each pair of corresponding elements:

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = a_1b_1 + a_2b_2 \cdots + a_nb_n \tag{5.7}$$

This is also called the dot product of two vectors $\mathbf{a} = (a_1, a_2, \cdots, a_n)$ and $\mathbf{b} = (b_1, b_2, \cdots, b_n)$, denoted by

$$\mathbf{a} \cdot \mathbf{b} = a_1b_1 + a_2b_2 \cdots + a_nb_n. \tag{5.8}$$

In 2- or 3-dimensional space, the dot product of two vectors has simple geometric interpretation:

$$\mathbf{a} \cdot \mathbf{b} = ||\mathbf{a}||\ ||\mathbf{b}|| \cos\gamma, \tag{5.9}$$

where $||\mathbf{a}||$ stands for the length of a vector $\mathbf{a}$, and γ is the angle between $\mathbf{a}$ and $\mathbf{b}$. The proof of the equivalence of the above two expressions (5.7) and (5.9) is given in the appendix.

When $\mathbf{a}$ is force and $\mathbf{b}$ is the displacement of a subject moved by the force, then $\mathbf{a} \cdot \mathbf{b}$ is the work done by the force to the subject.

Example 2. Dot product of two vectors in a 2-dimensional space:

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = 1 \tag{5.10}$$

The same result can be computed from the geometric meaning formula (5.9). The length of the first vector is $\sqrt{2}$ since it is the diagonal vector of a unit square in the first quadrant, and the length of the second vector is 1 since it is the unit square's side on the x-axis. The angle between the two vectors is 45°. Thus, the dot product of the two vectors are $\sqrt{2} \times 1 \times \cos(45°) = 1$.

5.2.2.2 A scalar times a matrix

A scalar c times a matrix $A = [a_{ij}]$ is formed by multiplying the scalar into every element of the matrix.

$$c \times A = [c \times a_{ij}]. \tag{5.11}$$

The product is again a matrix of the same size. A physically meaningful product requires that the dimension of the scalar and the dimensions of the matrix elements are compatible. For example, if the matrix is the price data of many products, and if the scalar is the number of sales of each product, then the scalar times the matrix gives the revenue matrix of all the products.

Example 3. A scalar 3 times a 2-by-2 matrix

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

is

$$3 \times A = 3 \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 3 & 3 \\ 3 & -3 \end{bmatrix} \tag{5.12}$$

5.2.2.3 A m-by-n matrix times an n-by-k matrix

The multiplication of two matrices is defined as a set of dot products between row vectors of the first matrix and the column vectors of the second matrix. Because of the requirement to form dot products, the column number of the first matrix must be the same as the row number of the second matrix. $A_{m\times n}B_{n\times k}$ is defined as a new matrix $C_{m\times k} = [c_{ij}]$, where the element c_{ij} is the dot product of the ith row vector of $A_{m\times n}$ and jth column vector of $B_{n\times k}$. Namely,

$$A_{m\times n}B_{n\times k} = \left[\sum_{l=1}^{n} a_{il}b_{lj}\right]_{m\times k}. \tag{5.13}$$

Thus, $A_{m\times n}B_{n\times k}$ is equal to a matrix of $m \times k$ dot products, and is tedious to compute.

Example 4. Matrix multiplications:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} = \begin{bmatrix} 3 & 7 \\ -1 & -1 \end{bmatrix}, \tag{5.14}$$

and

$$\begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 4 & -2 \\ 6 & -2 \end{bmatrix}. \tag{5.15}$$

The first element of the right hand side matrix is 3, which is the product of the first row vector of A: $(1, 1)$, and the first column vector of B: $(1, 2)$. Their dot product is

$$(1, 1) \cdot (1, 2) = 1 \times 1 + 1 \times 2 = 3. \tag{5.16}$$

In the same way, one can verify every element of the above multiplication results.

An R code for the above products is below

```
matrix(c(1,1,1,-1), nrow=2) %*% matrix(c(1,2,3,4), nrow=2)
#      [,1] [,2]
#[1,]    3    7
#[2,]   -1   -1
matrix(c(1,2,3,4), nrow=2) %*% matrix(c(1,1,1,-1), nrow=2)
#       [,1] [,2]
#[1,]    4   -2
#[2,]    6   -2
```

In the above example, the second matrix multiplication (5.15) involves the same matrices as the first one (5.14), but in a different order: If eq. (5.14) is denoted by AB, then eq. (5.15) is BA. Clearly, the results are different. In general,

$$AB \neq BA \tag{5.17}$$

for a matrix multiplication. Thus, matrix multiplication does not have the commutative property which the multiplication of two scalars x and y does have: $xy = yx$.

5.2.2.4 Transpose matrix, and diagonal, identity and zero matrices

Although a space-time climate data matrix often has its rows to mark spatial locations and columns to mark time, the row and column roles may need to exchange for some applications, which uses rows to mark time and columns to mark spatial locations. This operation of exchanging rows and columns is called matrix transpose. A transposed matrix of A is denoted by A^t. The columns of A^t are the rows of A.

Example 5.

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}, A^t = \begin{bmatrix} 1 & 3 \\ 2 & 4 \end{bmatrix} \tag{5.18}$$

It is obvious that

$$(A+B)^t = A^t + B^t. \tag{5.19}$$

However, a true but less obvious formula is the transpose of a matrix multiplication:

$$(AB)^t = B^t A^t. \tag{5.20}$$

When a matrix whose only non-zero elements are on the diagonal elements, this matrix is call a diagonal matrix:

$$D = \begin{bmatrix} d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n \end{bmatrix}, \tag{5.21}$$

An identity matrix is a special type of diagonal matrix, whose diagonal elements are all one and whose off-diagonal elements are all zero, and is denoted by I:

$$I = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}, \tag{5.22}$$

which plays a role in matrix operations similar to the role of the value 1.0 in the familiar real number system.

A zero matrix is a matrix whose elements are all zero. If two matrices are the same, then their difference is a zero matrix.

5.2.2.5 Matrix division and inverse

The division of a scalar y by another non-zero scalar x can be written as y times the inverse of x:

$$y/x = y \times x^{-1}. \tag{5.23}$$

Thus, the division problem becomes a multiplication problem when the inverse is found. The inverse of x is defined as $x^{-1} \times x = 1$.

Matrix division is defined in the same way:

$$A/B = A \times B^{-1}, \tag{5.24}$$

where B^{-1} is the inverse matrix of B defined as

$$B^{-1}B = I \ , \tag{5.25}$$

where I is the identity matrix.

The R command to find the inverse is
`solve(B)`

Example 6.

```
solve(matrix(c(1,1,1,-1), nrow=2))
#     [,1] [,2]
#[1,]  0.5  0.5
#[2,]  0.5 -0.5
```

That is

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}^{-1} = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \tag{5.26}$$

One can verify this by hand, or by the following R code

```
matrix(c(0.5,0.5,0.5,-0.5), nrow=2) %*% matrix(c(1,1,1,-1), nrow=2)
#      [,1] [,2]
#[1,]    1    0
#[2,]    0    1
```

Finding the inverse matrix of a matrix B "by hand" is usually a very difficult and involves a long procedure for a large matrix, say, a 4×4 matrix. Modern climate models can involve multiple $n \times n$ matrices with n from several hundred to several million, or even billion. In this book, we use R to find inverse matrices and do not attempt to explain how to find the inverse of a matrix by hand. Mastering the material in this book will not require you to have this skill. A typical linear algebra textbook will devote a large portion of its material to finding an inverse of a matrix. A commonly used scheme is called echelon reduction through row operations. For detailed information, see the excellent text *Introduction to Linear Algebra* by Gilbert Strang.

5.3 A set of linear equations

A somewhat different matrix example is the coefficient matrix of a system of linear equations. Solving a system of linear equations is very common in science and engineering. Finding numerical solutions for a climate model based on partial differential equations will usually involve solving large systems of linear equations.

As an introduction to the coefficient matrix of linear equations, let us look at a simple elementary school mathematics problem: The sum of the ages of two brothers is 20 years and the difference of the ages is 4 years. What are the ages of the two brothers? One can easily guess that the older brother is 12 years old, and the younger one is 8.

If we form a set of equations, which would be

$$\begin{aligned} x_1 + x_2 &= 20 \\ x_1 - x_2 &= 4 \end{aligned} \tag{5.27}$$

when x_1 and x_2 stand for the brothers' ages.

The matrix form of these equations would be

$$Ax = b \tag{5.28}$$

which involves three matrices:

$$A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}, b = \begin{bmatrix} 20 \\ 4 \end{bmatrix}. \tag{5.29}$$

Here, Ax means a matrix multiplication: $A_{2\times 2}x_{2\times 1}$.

The matrix notation of a system of two linear equations can be extended to systems of many linear equations, hundreds or millions of equations in climate modeling and climate data analysis. Typical linear algebra textbooks introduce matrices in this way by describing linear equations in a matrix form. However, this approach may be less intuitive for climate science, which emphasizes data. Thus, our book uses data to introduce matrices as shown at the beginning of this chapter.

Although one can easily guess that the solution to the above simple matrix equation (5.28) is $x_1 = 12$ and $x_2 = 8$, a more general method for computing the solution may be using the R code shown below:

```
solve(matrix(c(1,1,1,-1),nrow=2),c(20,4))
#[1] 12  8  #This is the result x1=12, and x2=8.
```

In this R command, `matrix(c(1,1,1,-1),nrow=2)` yields the A matrix, and `c(20,4)` the b vector.

This type of R program can solve more complicated systems of linear equations, such as a system with 1,000 unknowns rather than two unknowns, as in this example.

The solution may be represented as

$$x = A^{-1}b, \tag{5.30}$$

where A^{-1} is the inverse matrix of A and was found earlier in eq. (5.26). One can verify that

$$A^{-1}b = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & -0.5 \end{bmatrix} \begin{bmatrix} 20 \\ 4 \end{bmatrix} = \begin{bmatrix} 12 \\ 8 \end{bmatrix} \tag{5.31}$$

is indeed the solution of the system of two linear equations.

5.4 Eigenvalues and eigenvectors of a square space matrix

"Eigenvalue" is a partial translation of the German word "eigenwert," meaning "self-value" or intrinsic value." In German, "eigen" can mean "self" or"own", as in "one's own," and"wert" means"value."

A square matrix is a matrix for which the number of rows is equal to the number of columns. The matrix thus has the shape of a square. Similarly, other matrices may be called rectangular matrices, or tall matrices.

In Climate science, one often considers the covariance or correlation among N stations or N grid points. The covariance matrix is thus a square matrix. The ij-th element is equal to the covariance of the data of a climate parameter at the i-th station with those at the j-th station $(i, j = 1, 2, \cdots, N)$. If Y is the time length, say Y years, of the anomaly data $A_{N\times Y}$, then the covariance matrix is

$$C = \frac{1}{Y} AA'. \tag{5.32}$$

Example 1. This example's data matrix `A` has $N = 2$ and $Y = 3$. The data's covariance matrix is `covm`.

```
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
```

The covariance matrix times a vector `u` yields a new vector in a different direction.

```
u=c(1,-1)
v=covm%*%u
v
#     [,1]
#[1,]    4
#[2,]    0
#u and v are in different directions.
```

In general, when we consider a given square matrix C and a given vector u, the product Cu is usually not in the same direction as u, as shown in the above example. However, there always a "self-vector" vector w for each covariance matrix C such that Cw is in the same direction as w, i.e., Cw and w are parallel expressed as

$$Cw = \lambda w, \tag{5.33}$$

where λ is a scalar which has the property that it simply scales w so that the above equation holds. This scalar λ is called an eigenvalue (i.e., a "self-value", "own-value", or "characteristic value" of the matrix C), and w is called an eigenvector.

R can calculate the eigenvalues and eigenvectors of a covariance matrix `covm` with a command `eigen(covm)`. The output is in an R data frame, which has two storages: `ew$values` for eigenvalues and `ew$vector` for eigenvectors, as shown below.

```
eigen(covm)
eigen(covm)$values
#[1] 4.7748518 0.5584816
eigen(covm)$vectors
#[,1]       [,2]
#[1,] -0.9870875  0.1601822
#[2,] -0.1601822 -0.9870875
#Verify the eigenvectors and eigenvalues
covm%*%ew$vectors[,1]/ew$values[1]
#[,1]
#[1,] -0.9870875
#[2,] -0.1601822
#This is the first eigenvector
```

A 2×2 covariance matrix has two eigenvalues, and two eigenvectors (λ_1, w_1) and (λ_2, w_2), which are shown below from the R computation above for our example covariance matrix C:

$$\lambda_1 = 4.7748518, \quad \mathbf{w_1} = \begin{bmatrix} -0.9870875 \\ -0.1601822 \end{bmatrix}, \tag{5.34}$$

$$\lambda_2 = 0.5584816, \quad \mathbf{w_2} = \begin{bmatrix} 0.1601822 \\ -0.9870875 \end{bmatrix}. \tag{5.35}$$

The eigenvectors are frequently called modes, or empirical orthogonal functions (EOFs) in climate science. The term EOF was coined by Edward Lorenz in his 1956 paper on statistical weather prediction. The first few eigenvectors of a large climate covariance matrix of climate data often represent some typical patterns of climate variability. Examples which are of great importance in climate science include the El Niño Southern Oscillation (ENSO), North American Oscillation (NAO), and Pacic Decadal Oscillation (PDO).

It is often the case that the components of the first mode, i.e., the elements of the first eigenvector, have the same sign, either all positive or all negative. The components of the second mode will then have half negative and half positive signs. Exceptions can occur, however.

Each eigenvalue is equal to the variance of the data projection on the corresponding eigenvector, and is thus positive. The sum of all the eigenvalues of such an $N \times N$ matrix represents the total variance of the climate system observed at these N stations. The first eigenvalue λ_1 is the largest, corresponding to the largest spatial variability of the climate field under study. The eigenvalues sizes follow the order $\lambda_1 \geq \lambda_2 \geq \cdots$.

However, in carrying out an analysis of climate data, one can often nd the important patterns as eigenvectors more directly from the anomaly data matrix A without computing the covariance matrix C explicitly. This is known as the singular value decomposition (SVD) approach, which we discuss next. It separates the space-time anomaly data into a space part, a time part, and what we may think of as

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	$D54$

a variation in energy part. This mathematical method of space-time decomposition is universally applicable to any data that we sample in space and time, and it can often help to develop physical insight and scientific understanding of the phenomena and their properties as contained in the observational data. Efcient computing methods of SVD have been extensively researched and developed since the 1960s. Gene H. Golub (1932-2007) was a leading gure in this effort and is remembered as "Professor SVD" by his Stanford colleagues and the world mathematics community.

5.5 An SVD representation model for space-time data

We encounter space-time data every day, a simple example being the air temperature at different locations at different times. If you take a plane to travel from San Diego to New York, you may experience the temperature at San Diego in the morning when you depart and that at New York in the evening after your arrival. Such data have many important applications. We may need to examine the precipitation conditions around the world at different days in order to monitor agricultural yields. A cellphone company may need to monitor its market share and the temporal variations of that quantity in different countries. A physician may need to monitor a patients symptoms in different areas of the body at different times. The observed data in all these examples can form a space-time data matrix with the row position corresponding to the spatial location and the column position corresponding to time, as in Table 5.1.

Graphically, the space-time data may typically be plotted as a time series at each given spatial position, or as a spatial map at each given time. Although these straight-forward graphical representations can sometimes provide very useful information as input for signal detection, the signals are often buried in the data and may need to be detected by different linear combinations in space and time.

Sometimes the data matrices are extremely large, with millions of data points in either space or time. Then the question arises as to how can we extract the essential information in such a big data matrix? Can we somehow manage to represent the data in a more simple and yet more useful way? A very useful approach to such a task involves a space-time separation. Singular value decomposition (SVD) is a method 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^t, i.e., the data matrix A is decomposed into

$$A_{N\times Y} = U_{N\times m} D_{m\times m} (V^t)_{m\times Y}. \tag{5.36}$$

where N is the spatial dimension, Y is the temporal length, $m = min(N, Y)$, and V^t the transpose of V. The columns of U are a set of orthonormal vectors known as spatial eigenvectors, i.e.

$$\mathbf{u}_l \cdot \mathbf{u}_k = \delta_{lk}, \tag{5.37}$$

where $\mathbf{u}_l$ are column vectors of U, and δ_{lk} is the Kronecker delta equal to zero when $k \neq l$ and one when $k = l$. The columns of V are also a set of orthonormal vectors known as temporal eigenvectors.

Usually, the elements of the U and V matrices are unitless (i.e., dimensionless), and the unit of the D elements is the same as the elements of the data matrix. For example, if A is a space-time precipitation data matrix with a unit [mm/day], then the dimension of the D elements is also [mm/day].

Example 1. SVD for the 2×3 data matrix A in Section 5.4.

```
#Develop a 2-by-3 space-time data matrix for SVD
A=matrix(c(1,-1,2,0,3,1),nrow=2)
A
#     [,1] [,2] [,3]
#[1,]    1    2    3
#[2,]   -1    0    1
#Perform SVD calculation
msvd=svd(A)
msvd
msvd$d
#[1] 3.784779 1.294390
msvd$u
#          [,1]       [,2]
#[1,] -0.9870875 -0.1601822
#[2,] -0.1601822  0.9870875
msvd$v
#           [,1]       [,2]
#[1,] -0.2184817 -0.8863403
#[2,] -0.5216090 -0.2475023
#[3,] -0.8247362  0.3913356
```

```
#One can verify that A=UDV', where V' is transpose of V.
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:

$$C = \frac{1}{Y} AA^t, \tag{5.38}$$

where Y is the number of columns of A and is equal to the length of time being considered.

```
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 eigenvectors of a covariance matrix are the same as the SVD eigenvectors of the anomaly data matrix. The eigenvalues of the covariance matrix and the SVD have following relationship

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

Therefore, the EOFs from a given space-time dataset can be calculated directly by using an SVD command and do not need the step of calculating the covariance matrix. With efficient SVD algorithms, this shortcut can save significant amount of time for an EOF analysis, also known as the principal component analysis (PCA) in the statistics community, compared to the traditional covariance matrix approach. So, the result is extremely helpful for the EOF analysis, which is an indispensable modern tool of climate data analysis. The result is formally described as a theorem whose proof is also provided as follows.

Theorem 5.1 *The eigenvectors bfu_k of the covariance matrix $C = (1/Y)AA^t$*

$$\mathbf{C}\mathbf{u}_k = \lambda_i \mathbf{u}_k, \quad (k = 1, 2, \cdots, N), \tag{5.39}$$

are the same as the SVD spatial modes of $A = UDV^t$. The eigenvalues λ_k of $C_{N\times N}$ and the SVD eigenvalues d_k of $A_{N\times Y}$ have the following relationship

$$\lambda_k = d_k^2/Y \quad (k = 1, 2, \cdots, Y), \quad when Y \leq N, \tag{5.40}$$

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).

Proof The SVD of the space-time data matrix is

$$A = UDV^t. \tag{5.41}$$

The data matrix A's corresponding covariance matrix is thus

$$\begin{aligned} C &= \frac{1}{Y}AA^t \\ &= \frac{1}{Y}UDV^t(UDV^t)^t \\ &= \frac{1}{Y}UDV^t(VDU^t) \\ &= \frac{1}{Y}UD(V^tV)DU^t \\ &= \frac{1}{Y}UDIDU^t \\ &= \frac{1}{Y}UD^2U^t. \end{aligned} \tag{5.42}$$

In the above, we have used $V^tV = I_Y$ is a $Y \times Y$ identity matrix according to the SVD definition. The identity matrix's dimension is Y because if $Y \leq N$. Otherwise, $V^tV = I_N$.

The expression $C = \frac{1}{Y}UD^2U^t$ is the EOF expansion of the covariance matrix and means that a covariance matrix consists of EOFs and its associated variance, or "energy."

The covariance matrix's eigenvalue problem is

$$CU = \frac{1}{Y}UD^2U^tU = \frac{1}{Y}UD^2 = U\Lambda, \tag{5.43}$$

where

$$\Lambda = \frac{1}{Y}D^2 \tag{5.44}$$

is the diagonal eigenvalue matrix with its elements as

$$\lambda_k = d_k^2/Y \quad (k = 1, 2, \cdots, Y). \tag{5.45}$$

In eq. (5.43), we used $U^tU = I_Y$ based on the SVD definition. Equation (5.43) becomes an eigenvalues problem because Λ is a diagonal matrix: $C\mathbf{u}_k = \lambda_k\mathbf{u}_k$ ($k = 1, 2, \cdots, Y$), where $\mathbf{u}_k$ is the kth column vector of the space matrix U. Thus, eq. (5.43) implies the first part of the theorem: The space eigenvectors U from the SVD of the space-time data matrix A are the same as the eigenvectors of the corresponding covariance matrix C.

Equation (5.45) is exactly the second part of the theorem. The proof is thus complete.

In the above proof, we implicitly assumed that the columns of data matrix A are independent when $Y \leq N$. Independence of a set of vectors means that no one can expressed in terms of the others. So no vectors can be replaced. Real climate data always satisfy this independence assumption.

5.6 SVD analysis of Southern Oscillation Index

This section is an application example of the SVD method for constructing an optimally weighted Southern Oscillation Index (WSOI).

SOI is an index that monitors ENSO and is the standardized measure of an important atmospheric pressure gradient. Specifically, it is the atmospheric sea level pressure (SLP) at Tahiti (18°S, 149°W) minus that at Darwin (12°S, 131°E). It thus measures the SLP difference between the eastern tropical Pacic and the western tropical Pacic. During a “normal” (non-El Nio) year, Darwins anomaly pressure is lower than that of Tahiti, and it is this pressure difference which maintains the easterly trade winds and is dynamically consistent with the existence of the western Pacic warm pool, an area of higher ocean surface temperatures compared to the eastern tropical Pacific. When the wind direction reverses, the pressure anomalies have the opposite sign, leading to westerlies in the region where trade winds are normally found, and to the accumulation of anomalously warm surface water in the central or eastern tropical Pacic. This latter ocean temperature pattern is the characteristic signal of El Niño.

The SLP data of these two stations between January 1951 and December 2015 can be downloaded from the NOAA Climate Prediction Center (CPC)’s website `http://www.cpc.ncep.noaa.gov/data/indices/` We will first examine SOI from the standardized Tahiti and Darwin SLP data, and then use the same data but the SVD approach to develop a new southern oscillation index: WSOI, which can more accurately quantify the El Niño signal.

5.6.1 Standardized SLP data and SOI

Figure 5.2 shows the the standardized Tahiti and Darwin SLP data, SOI, cumulative SOI index (CSOI), and Atlantic Multi-decadal Oscillation (AMO) index. The AMO index is defined as the standardized average sea surface temperature (SST) over the North Atlantic region (80°W-0°E, 0°N-60°N). The AMO data from January 1951 to December 2015 can be downloaded from the NOAA Earth System Research Laboratory website
`https://www.esrl.noaa.gov/psd/data/correlation/amon.us.data`

It appears that the SCOI and AMO index follow a similar nonlinear trend. Both CSOI and AMO decrease from 1950s to a bottom in the 1970s, then increase to around year 1998 followed by about a decade plateau, and then start to decrease around 2006. CSOI has a much smaller variance than the AMO index and has a more clear signal.

The first three panels of Fig. 5.2 can be generated from the following R code.

```
#SOI and the Standardized SLP data at Darwin and Tahiti
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",
     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)
```

The following R code can generate the fourth panel of Fig. 5.2.

```
#CSOI and AMO time series comparison
#Cumulative SOI
plot.new()
par(mar=c(4,4,4,4))
cnegsoi<--cumsum(ptamonv-pdamonv)
plot(xtime, cnegsoi,type="l",xlab="Year",ylab="Negative CSOI index",
```

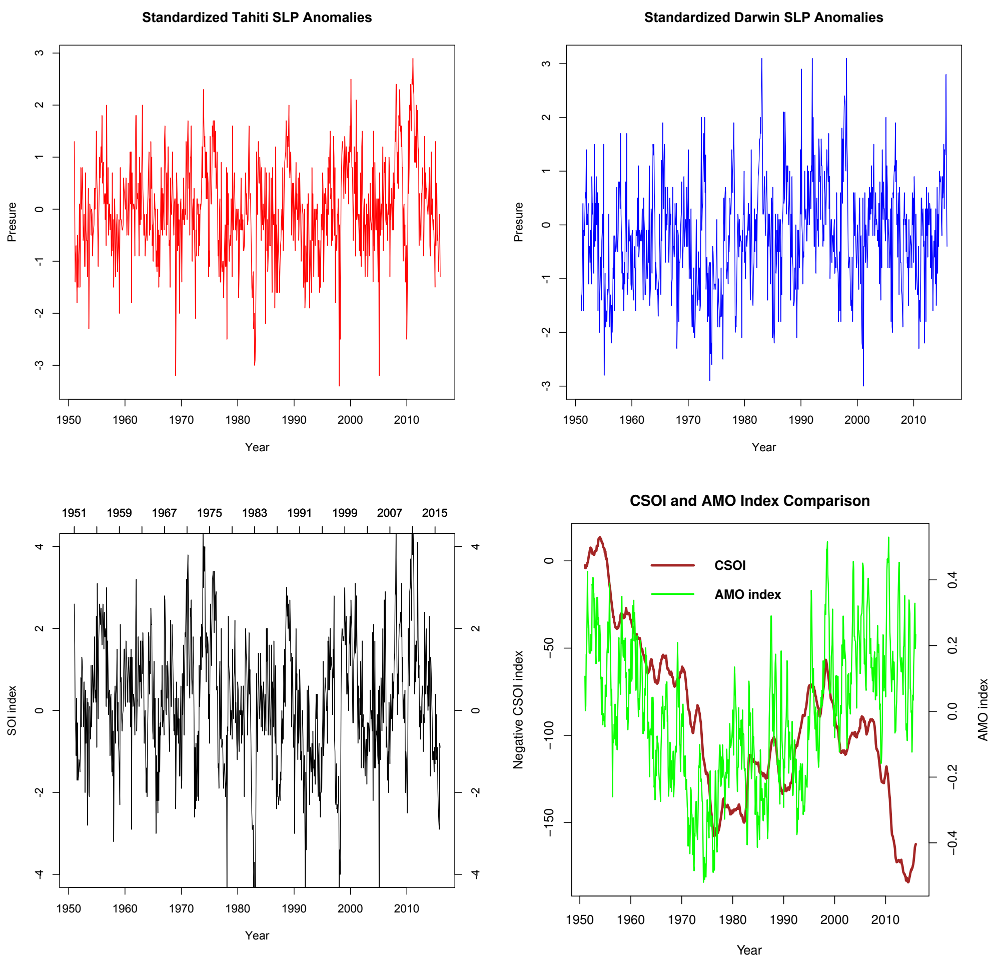


Fig. 5.2 Standardized sea level pressure anomalies of Tahiti (upper left panel), that of Darwin (upper right). SOI time series (lower left), and the cumulative of the negative SOI time series (lower right) and AMO time series.

```
     main="CSOI and AMO Index Comparison", mpg=c(2,2,4,0),
     col="red",xlim=range(xtime), ylim=range(cnegsoi), lwd=1.5)
legend(1960,15, col=c("red"),lty=1,lwd=2.0,
       legend=c("CSOI"),bty="n",text.font=2,cex=1.0)
#AMO data and plot
amodat=read.table("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2017/R Code/Ch5/
amots=as.vector(t(amodat[,2:13]))
par(new=TRUE)
plot(xtime, amots,type="l",col="blue",lwd=1.5,axes=FALSE,xlab="",ylab="")
legend(1960,0.45, col=c("blue"),lty=1,lwd=2.0,
       legend=c("AMO index"),bty="n",text.font=2,cex=1.0)
#Suppress the axes and assign the y-axis to side 4
axis(4)
mtext("AMO index",side=4,line=3)
```

5.6.2 Weighted SOI computed by the SVD method

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 column is the Darwin SLP. Because normally the spatial position is indicated by row and the time is indicated 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 $C = (1/t)AA^t$, where $A_{n\times t}$ is the original data matrix of n spatial dimension and t temporal dimension. The SVD decomposition of the matrix A becomes

$$A_{780\times 2} = U_{2\times 2}D_{2\times 2}V_{2\times 780}. \tag{5.46}$$

Our EOF matrix U is

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

The two column vectors of U are the covariance matrix' eigenvectors, i.e., the empirical orthogonal functions (EOF) in the atmospheric science literature as described earlier. The EOFs represent spatial patterns. 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 could be the weighted SOI:

$$WSOI1 = -0.6147 P_{Tahiti} + 0.7888 P_{Darwin} \tag{5.47}$$

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 that is not true in this example. Here the second mode's energy level is $d_2 = 22.25$, which is equal to 71% of the first energy level 31.35.

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

$$WSOI2 = 0.7888 P_{Tahiti} + 0.6147 P_{Darwin} \tag{5.48}$$

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

$$U^tA = DV^t, \tag{5.49}$$

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

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

$$\mathbf{v}_1 = WSOI1/d_1 \tag{5.50}$$

$$\mathbf{v}_2 = WSOI2/d_2 \tag{5.51}$$

The two PCs are orthonormal vectors, meaning the dot product of two different PC vectors is zero, and that of the two same PC vectors is one. The two EOFs are also orthonormal vectors. 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

$$d_k PC_k = WSOI_k \quad (k = 1, 2). \tag{5.52}$$

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
#CWSOI and smoothed AMO
#Comparison among CWSOI1, CSOI, and the Smoothed AMO Index
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))
xtime<-seq(1951, 2015, length=780)
#Do the same for Darwin data
Pda<-read.table("PSTANDdarwin.txt", header=F)
pdamon<-Pda[,seq(2,13)]
pdamonv<-c(t(pdamon))
ptada <-cbind(ptamonv,pdamonv)
ptada1<-t(ptada)
svdptd<-svd(ptada1)
U=svdptd$u
D=diag(svdptd$d)
V=svdptd$v
wsoi1=D[1,1]*t(V)[1,]
cwsoi1=cumsum(wsoi1)
plot.new()
```

```
par(mar=c(4,4,3,4))
plot(xtime, cwsoi1,type="l",xlab="Year",ylab="Weighted SOI 1",
     col="blue",lwd=3, ylim=c(-180,20),
     main="Comparison between CWSOI1 and the Smoothed AMO Index")
#axis (3, at=seq(1951,2015,4), labels=seq(1951,2015,4))
legend(1970,20, col=c("blue"),lty=1,lwd=3.0,
       legend=c("CWSOI1"),bty="n",text.font=2,cex=1.0)
#Superimpose CSOI time series on this CWSOI1
cnegsoi<--cumsum(ptamonv-pdamonv)
lines(xtime, cnegsoi,type="l",col="brown", lwd=3.0)
legend(1970,2, col=c("brown"),lty=1,lwd=3.0,
       legend=c("CSOI"),bty="n",text.font=2,cex=1.0)
#24-month ahead moving average of the monthly AMO index
amodat=read.table("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2017/R Code/Ch5/A
amots=as.vector(t(amodat[,2:13]))
#install.packages("TTR")
library("TTR")
amomv=SMA(amots,n=24, fill=NA)
#Average of the previous n points
par(new=TRUE)
xtime=seq(1951,2015,len=780)
plot(xtime, amomv[37:816],type="l",col="green",lwd=1.5,axes=FALSE,xlab="",ylab="")
legend(1970,0.165, col=c("green"),lty=1,lwd=2.0,
       legend=c("AMO index"),bty="n",text.font=2,cex=1.0)
#Suppress the axes and assign the y-axis to side 4
axis(4)
mtext("AMO index",side=4,line=3)

#Plot cumulative WSOI2: CWSOI2
wsoi2=D[2,2]*t(V)[2,]
cwsoi2=cumsum(wsoi2)
plot.new()
#par(mar=c(4,4,3,4))
plot(xtime, cwsoi2,type="l",xlab="Year",ylab="Weighted SOI 2",
     col="red",lwd=3,
     main="CWSOI2 Index: The Cumulative PC2")
```

When the cumulative WSOI1 decreases, so does the SH surface air temperature from 1951 to 1980. When the cumulative WSOI1 increases, so does the temperature from the 1980s to the peak 1998. Later, the cumulative WSOI1 decreases to a plateau from 1998 to 2002, then remains on plateau until 2007, then decreases again. This also agrees with nonlinear trend of the SH surface air temperature anomalies before 1998.

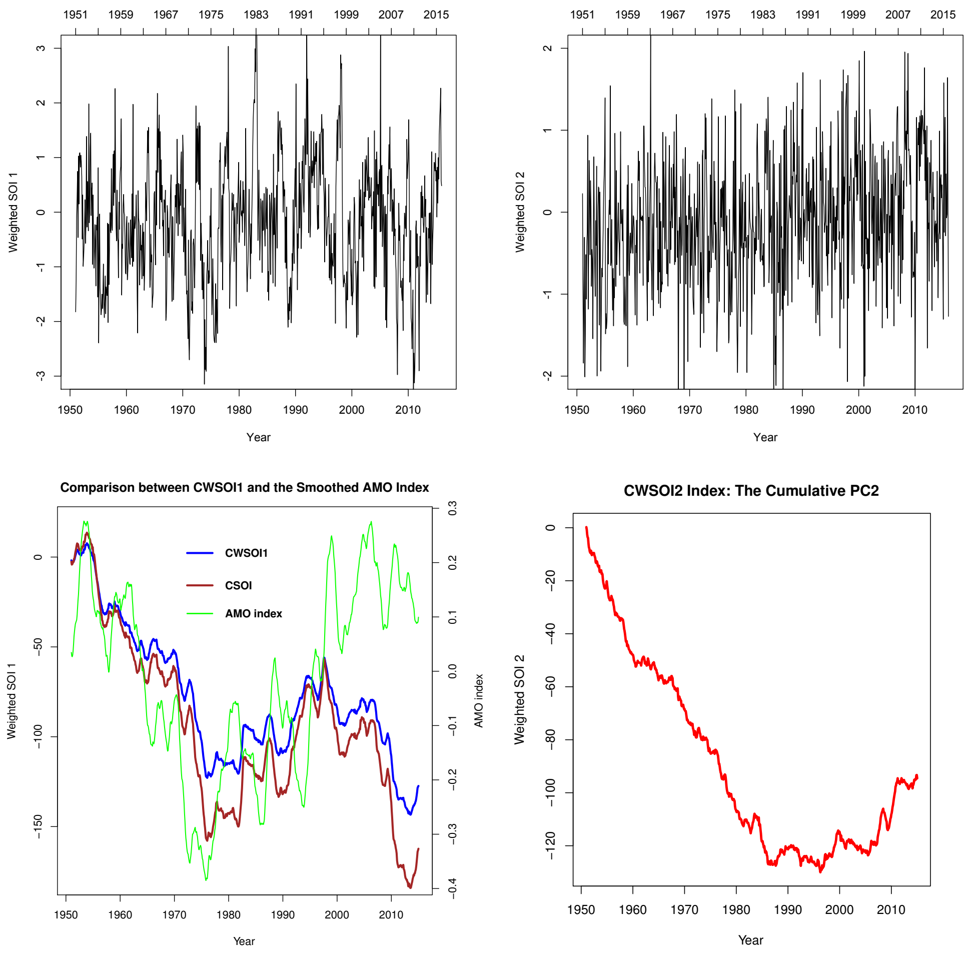


Fig. 5.3 Weighted SOI1 (upper left panel), weighted SOI2 (upper right), cumulative WSOI1 (lower left), and cumulative WSOI2 (lower right).

The CWSOI2 decreases from 1951 to the 1980s, remains at a flat valley until its further increase from around 2007. This increase coincides with the persistent global surface air temperature increase in the last decade.

Therefore, SVD results may lead to physical interpretations and may help provide physical insight and is thus a valuable and convenient tool to use.

5.6.3 Visualization of the ENSO mode computed from the SVD method

We present a visualization of EOFs from the above SVD results using `ggplot`.

The space-time data matrix `ptada` of the SLP at Tahiti and Darwin from January

1951 to 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 represents the El Niño mode. Note that the eigenvectors are determined except for a positive or negative sign. Because Tahiti has a positive SST anomaly during an El Niño, we 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 $(-0.79, 0.61) \cdot (0.61, 0.79) = 0$. They 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()
```

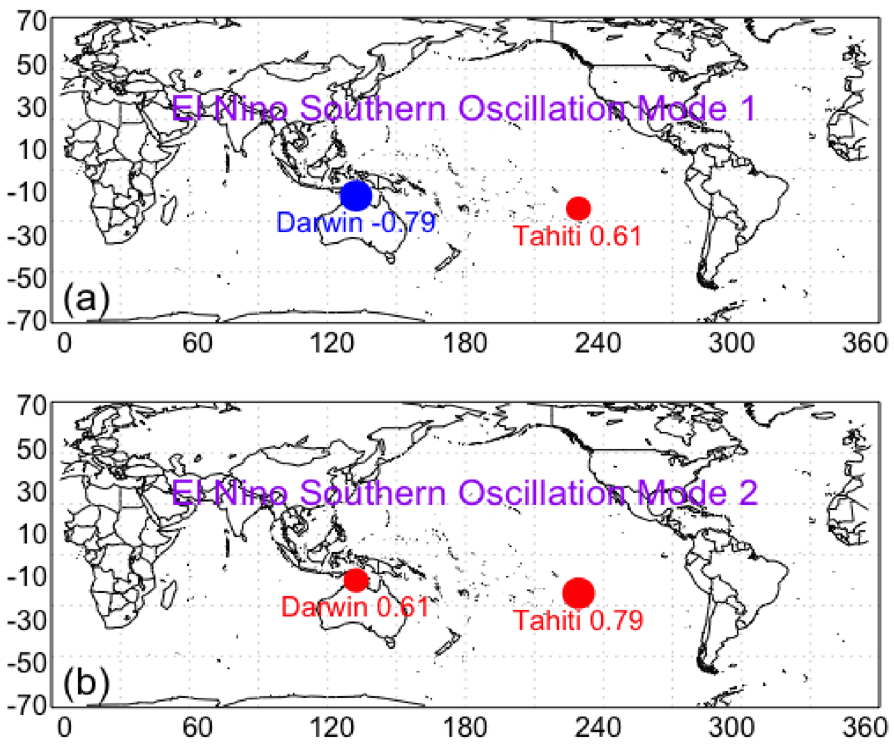


Fig. 5.4 The two orthogonal ENSO modes from the Tahiti and Darwin standardized SLP data. The relative data sizes are proportional to the component values of each eigenvector in the U matrix. Red color indicates positive values, and blue indicates the negative values.

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")
```

More advanced visualization of EOFs and PCs by R will be described in Chapters 13 and 14 of this book for more complex data, such as those from a climate model output or remote sensing.

5.7 Mass balance for chemical equations in marine chemistry

This section is another application of linear algebra.

Marine chemistry involves various kinds of chemical reaction equations which require the conservation of mass during the reactions. Here we use photosynthesis as an example to illustrate a way to determine the numbers of molecules on each side of an equation depicting a chemical reaction.

In the process of photosynthesis, plants convert the solar radiant energy carried by photons, plus carbon dioxide (CO_2) and water (H_2O), into glucose ($C_6H_{12}O_6$) and oxygen (O_2). The chemical equation for this conversion could be written schematically as

$$CO_2 + H_2O \longrightarrow C_6H_{12}O_6 + O_2 \tag{5.53}$$

THowever, the conservation of mass requires that the atomic weights on both sides of the equation be equal. The photons have no mass. Thus, the chemical equation as written above is incorrect. The correct equation should specify precisely how many CO_2 molecules react with how many H_2O molecules to generate how many $C_6H_{12}O_6$ and O_2 molecules. Suppose that these coefficients are x_1, x_2, x_3, x_4. We then have

$$x_1CO_2 + x_2H_2O \longrightarrow x_3C_6H_{12}O_6 + x_4O_2 \tag{5.54}$$

Making the number of atoms of carbon on the left and right sides of the equation equal yields

$$x_1 = 6x_3 \tag{5.55}$$

because water and oxygen contain no carbon. Doing the same for hydrogen atoms leads to

$$2x_2 = 12x_3. \tag{5.56}$$

Similarly, the balance of oxygen atoms is

$$2x_1 + x_2 = 6x_3 + 2x_4. \tag{5.57}$$

We thus have three equations in four variables. Thus, these equations have infinitely many solutions. We can set any variable fixed, and express the other three using this fixed variable. Since the largest molecule is glucose, we set its coefficient x_3 fixed. Then we have

$$x_1 = 6x_3, \quad x_2 = 6x_3, \quad x_4 = 6x_3. \tag{5.58}$$

Thus, the chemical equation is

$$6x_3CO_2 + 6x_3H_2O \longrightarrow x_3C_6H_{12}O_6 + 6x_3O_2. \tag{5.59}$$

If we want to produce one glucose molecule, i.e., $x_3 = 1$, then we need 6 carbon dioxide and 6 water molecules:

$$6CO_2 + 6H_2O \longrightarrow C_6H_{12}O_6 + 6O_2. \tag{5.60}$$

Similarly, one can write chemical equations for many common reactions, such as iron oxidation

$$3Fe + 4H_2O \longrightarrow 4H_2 + Fe_3O_4, \tag{5.61}$$

and the redox reaction in a human body which consumes glucose and converts the glucose into energy, water and carbon dioxide:

$$C_6H_{12}O_6 + 6O_2 \longrightarrow 6CO_2 + 6H_2O. \tag{5.62}$$

5.8 Multivariate linear regression using matrix notations

This section is an application of matrix algebra in statistical data analysis, particularly on a linear regression for more than one variable. 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 suggest that there may be a physical mechanism to explain how the January SOI inuences the U.S. January temperature.

Here, we use $y = ax + b$ as the representation of a deterministic fitting function. No random variables are considered. Most statistics books would consider random linear models, which distinguishes random variables and their deterministic estimator by data. Our simple linear mathematical model formulation here is equivalent to the deterministic estimators.

The U.S. January temperature can be influenced by multiple factors in addition to the SOI. These factors may include the North Atlantic sea surface temperature (SST), the North Pacic SST, Arctic pressure conditions, etc. Then, the linear model becomes

$$y = b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n. \tag{5.63}$$

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

In the following we will present a few R examples of multivariate regression and its applications. The mathematical theory behind the R code for a multivariate regression is mostly the matrix operations, which can be found from any standard textbooks with the materials of multivariate regression.

Example 1. This example shows a two-variable regression.

$$y = b_0 + b_1 x_1 + b_2 x_2. \tag{5.64}$$

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 by means of 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 2. This example will show that four arbitrarily specified points cannot all be 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 3. 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, also called explainable variables, and which is the dependent variable, also called response variable.

```
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

$$Z = 0.37 + 0.12W - 0.53X - 0.04Y. \tag{5.65}$$

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 result is to be expected, because the data are randomly generated and thus should not have a trend.

In practical applications, a user can simply convert 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
```

References

[1] Golub, G.H. and Reinsch, C., 1970: Singular value decomposition and least squares solutions. Numerische mathematik, 14(5), pp.403-420.
[2] Larson, R., 2013: Elementary linear algebra, 7th edition. Brooks/Cole Cengage Learning, Boston, 390pp.
[3] Strang, G., 1993: Introduction to Linear Algebra. Wellesley, MA: Wellesley-Cambridge Press, 400pp.

Exercises

5.1 The following are 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 $A = UDV^t$ 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. For extra credit: Describe the spatial and temporal modes, and their corresponding variances or energies.

5.2 Use R and the updated Darwin and Tahiti standardized SLP data to reproduce the EOFs and PCs and to plot the EOF pattern maps and PC time series.

5.3 Do the same procedures in the previous problem but for original non-standardized data. Comment on the difference of the results from the previous problem.

5.4 (a) Download the monthly precipitation data at five different stations over the United States from the USHCN website:

`http://cdiac.ornl.gov/epubs/ndp/ushcn/ushcn_map_interface.html`

(b) Use R to organize the January data from 1951 to 2010 into the space-time format.
(c) Compute the climatology of each station as the 1971-2000 mean.
(d) Compute the space-time anomaly data matrix A as the original space-time data matrix minus the climatology.
(e) Use R to make the SVD decomposition of the space-time anomaly data matrix $A = UDV^t$.
(f) Write the U and D matrices.

5.5 In the previous problem, use R and the formula UDV^t to reconstruct the original data matrix A. This is a verification of the SVD decomposition, and is also called EOF-PC reconstruction.

5.6 Use R to plot the maps of the first three EOF modes, similar to the two El Niño mode maps shown in Fig. 5.4. Try to explain the climate meaning of the EOF maps.

5.7 Use R to plot the first three PC time series. Try to explain the climate meaning of the time series.

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

$$C = X \cdot X^t / Y \tag{5.66}$$

This is an $N \times N$ matrix. (a) Choose a Y matrix from the USHCN annual total precipitation data at three California stations from north to south [Berkeley, CA (040693); Santa Barbara, CA (047902); Cuyamaca, CA (042239)] and five years from 2001 to 2005. and calculate a covariance matrix for $N = 3$ and $Y = 5$.
(b) Use R to find the inverse matrix of the covariance matrix C.

(c) Use R to find the eigenvalues and eigenvectors of C.
(d) Use R to make SVD decomposition of the data matrix $X = UDV^t$. Explicitly write out the three matrices U, D and V.
(e) Use R to explore the relationship between the eigenvalues of C and the matrix D.
(f) Compare the eigenvectors of C and the matrix U.
(g) Plot the PC time series and describe their behavior.

5.9 The burning of methane (CH_4) with oxygen (O_2) produces water (H_2O)and carbon dioxide (CO_2). Balance the chemical reaction equation.
5.10 The burning of propane (C_3H_8) with oxygen (O_2) produces water (H_2O)and carbon dioxide (CO_2). Balance the chemical reaction equation.
5.11 The burning of gasoline (C_8H_{18}) with oxygen (O_2) produces water (H_2O)and carbon dioxide (CO_2). Balance the chemical reaction equation.

6 Basic Statistical Methods for Climate Data Analysis

The word “statistics” comes from the Latin “status” meaning “state.” We use the term “statistics” to mean a suite of scientific methods for analyzing data and for drawing credible conclusions from the data. Statistical methods are routinely used for analyzing and drawing conclusions from climate data, such as for calculating the climate “normal” of precipitation at a weather station and for quantifying the reliability of the calculation. Statistical methods are often used for demonstrating that global warming is occurring, based on a significant upward trend of surface air temperature (SAT) anomalies, and on establishing a given significance level for this trend. Statistical methods are also used for inferring a significant shift from a lower state of North Pacific sea level pressure (SLP) to a higher state or from a lower temperature regime to a higher one. A list of questions such as those just cited can be infinitely long. The purpose of this chapter is to provide basic concepts and a kind of “user manual” covering the most commonly used statistical methods in climate data analysis, so that users can arrive at credible conclusions based on the data, together with a given error probability.

R codes will be supplied for examples in this chapter. Users can easily apply these codes, and the formulas given in this book, for their data analysis needs without any need for an extensive background knowledge of calculus, and without a deep understanding of statistics. To interpret the statistical results in a meaningful way, however, knowledge of the domain of climate science will be very useful, when using statistical concepts and the results of calculations to establish conclusions from specific climate datasets.

The statistical methods in this chapter have been chosen in order to focus on making credible inferences about the climate state, with a given error probability, based on the analysis of climate data, so that observational data can lead to objective and reliable conclusions. We will first describe a list of statistical indices, such as the mean, variance and quantiles, for climate data. We will then take up the topics of probability distributions and statistical inferences.

6.1 Statistical indices from the global temperature data from 1880 to 2015

The following link provides data for the global average annual mean surface air temperature anomalies from 1880 to 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 to 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 20th century mean, i.e., the 1900-1999 climatology period. The global average of the 20th century mean is 12.7 °C. The 2015 anomaly was 0.65 °C. Thus, the 2015's global average annual mean temperature is 13.4°C.

Because we have just quoted numbers that purport to be observations of annual mean global mean surface temperatures, this may be a good place to mention an important caveat. The caveat is that observational estimates of the global mean surface temperature are less accurate than similar estimates of year-to-year changes. This is one of several reasons why global mean surface temperature data are almost always plotted as anomalies (such as differences between the observed temperature and a long-term average temperature) rather than as the temperatures themselves. It is also important to realize that the characteristic spatial correlation length scale for surface temperature anomalies is much larger (hundreds of kilometers) than the spatial correlation length scale for surface temperatures. The use of anomalies is also a way of reducing or eliminating individual station biases that are invariant with time. A simple example of such biases is that due to station location, which usually does not change with time. It is easy to understand, for instance, that a station located in a valley in the middle of a mountainous region might report surface temperatures that are higher than an accurate mean surface temperature for the entire region, but the anomalies at the station might be more accurately reflect the characteristics of the anomalies for the region. For a clear and concise summary of these important issues, with references, see

```
http://www.realclimate.org/index.php/archives/2017/08/
observations-reanalyses-and-the-elusive-absolute-global-mean-temperature/
```

```
 [1] -0.367918 -0.317154 -0.317069 -0.393357 -0.457649 -0.468707
 [7] -0.451778 -0.498811 -0.403252 -0.353712 -0.577277 -0.504825
[13] -0.556487 -0.568014 -0.526737 -0.475364 -0.340468 -0.367002
[19] -0.505967 -0.368630 -0.315155 -0.387099 -0.494861 -0.585158
[25] -0.663492 -0.535226 -0.457892 -0.617208 -0.684107 -0.672176
[31] -0.624129 -0.675199 -0.570521 -0.558340 -0.379505 -0.308313
```

```
 [37] -0.531023 -0.551480 -0.444860 -0.444257 -0.451256 -0.388185
 [43] -0.469536 -0.455500 -0.489551 -0.385962 -0.305391 -0.393436
 [49] -0.416556 -0.538602 -0.339823 -0.316963 -0.360309 -0.486954
 [55] -0.347795 -0.383147 -0.356958 -0.262097 -0.272009 -0.257514
 [61] -0.152032 -0.050356 -0.095295 -0.088983  0.044418 -0.073264
 [67] -0.251405 -0.297744 -0.296136 -0.303984 -0.405346 -0.255647
 [73] -0.218081 -0.146923 -0.358796 -0.377482 -0.441748 -0.194232
 [79] -0.133076 -0.184608 -0.222896 -0.165795 -0.154384 -0.137509
 [85] -0.393492 -0.322453 -0.267491 -0.257946 -0.274517 -0.151345
 [91] -0.207025 -0.322901 -0.216440 -0.080250 -0.316583 -0.241672
 [97] -0.323398 -0.046098 -0.131010 -0.016080  0.021495  0.057638
[103] -0.061422  0.099061 -0.093873 -0.109097 -0.015374  0.125450
[109]  0.129184  0.050926  0.186128  0.159565  0.010836  0.038629
[115]  0.092131  0.211006  0.074193  0.269107  0.384935  0.194762
[121]  0.177381  0.296912  0.351874  0.363650  0.329436  0.408409
[127]  0.362960  0.360386  0.291370  0.385638  0.453061  0.325297
[133]  0.370861  0.416356  0.491245  0.650217
```

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

```
setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2017/
Book-ClimMath-Cambridge-PT1-2017-07-21/Data")
dat1 <- read.table("aravg.ann.land_ocean.90S.90N.v4.0.0.2015.txt")
dim(dat1)
tmean15=dat1[,2] #Take only the second column of this data matrix
head(tmean15) #The first five values
#[1] -0.367918 -0.317154 -0.317069 -0.393357 -0.457649 -0.468707
mean(tmean15)
#[1] -0.2034367
sd(tmean15)
#[1] 0.3038567
var(tmean15)
#[1] 0.09232888
library(e1071)
#This R library is needed to compute the following parameters
skewness(tmean15)
#[1] 0.7141481
kurtosis(tmean15)
#[1] -0.3712142
median(tmean15)
#[1] -0.29694
quantile(tmean15,probs= c(0.05,0.25, 0.75, 0.95))
#        5%        25%        75%        95%
```

```
#-0.5792472 -0.4228540 -0.0159035  0.3743795
```

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
#-13.208662     0.006678
# 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 0.6678 oC per century",
 col="red",cex=1.2)
```

The above statistical indices were computed using the following mathematical formulas, described by $x = \{x_1, x_2, \cdots, x_n\}$ as the sampling data for a time series:

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

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

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

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

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

The significance of these indices is as follows. The mean gives the average of samples. The variance and standard deviation measure the spread of samples. They are large when the samples have a broad spread. Skewness is a dimensionless quantity. It measures the asymmetry of samples. Zero skewness signifies a symmetric distribution. For example, the skewness of a normal distribution is zero. Negative skewness denotes a skew to the left, meaning that the long distribution tail is on the left side of the distribution. Positive skewness has a long tail on the right side.

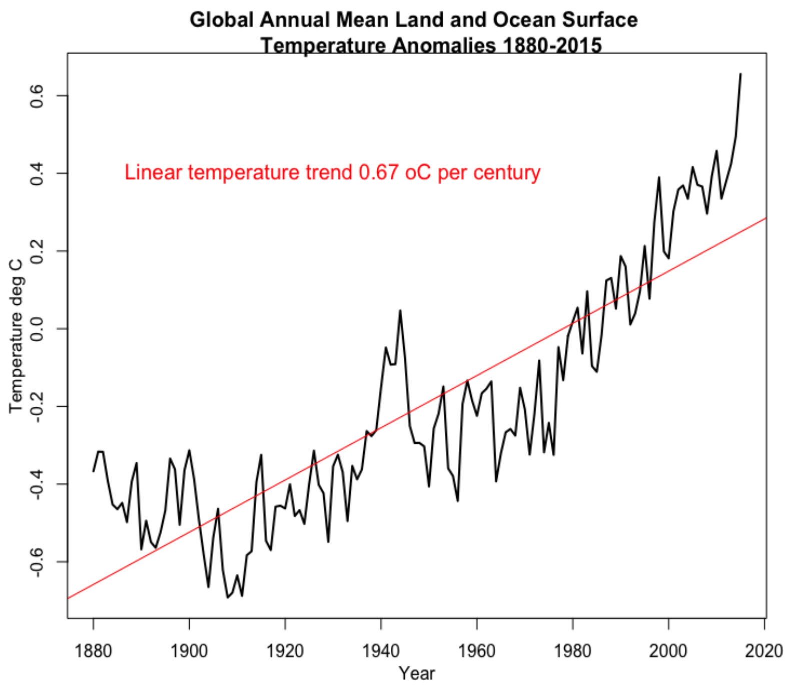


Fig. 6.1 Time series of the global average annual mean temperature with respect to 1900-1999 climatology: 12.7 °C.

Kurtosis is also dimensionless and measures the peakedness of a distribution. The kurtosis of a normal distribution is zero. Positive kurtosis means a high peak at the mean, thus a slim and tall shape for the distribution. This is referred to as leptokurtic. “Lepto” is Greek in origin and means thin or fine. Negative kurtosis means a low peak at the mean, thus a fat and short shape for the distribution, referred to as platykurtic. “Platy” is also Greek in origin and means flat or broad. “Kurtic” and “kurtosis” are Greek in origin and mean peakedness.

For the 136 years of global average annual mean temperature data given above, the skewness is 0.71, meaning skew to the right with a long tail on the right, thus with more extreme high temperatures than low temperatures, as shown in the histogram in Fig. 6.2. The kurtosis is -0.37, meaning the distribution is flatter than a normal distribution, also shown in the histogram.

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

Quantiles are defined in the same way by sorting. For example, 25-percentile (also called 25th percentile) is a sample such that 25% of sample values are less than this sample value. By definition, 75-percentile is thus larger than 40-percentile.

Obviously, 100-percentile is the largest sample, and 0-percentile is the smallest sample. Often, a box plot is used to show the typical quantiles. See Fig. 6.3 for the box plot of the 136 years of global average annual mean temperature data.

The 50-percentile (or 50th percentile) is called the median. If the distribution is symmetric, then the median is equal to mean. Otherwise these two quantities are not equal. If the skew is to the right, then the mean is on the right of the median: the mean is greater than the median. If the skew is to the left, then the mean is on the left of the median: the mean is less than the median. Our 136 years of temperature data are right skewed and have mean equal to -0.2034°C, greater than their median equal to -0.2969°C.

6.2 Commonly used statistical plots

We will use the 136 years of temperature data and R to illustrate some commonly used statistical figures, namely the histogram, boxplot, 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 estimate.

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

6.2.2 Box plot

Figure 6.3 is the box plot of the 136 years of global average annual mean temperature data, and can be made from the following R command
`b=boxplot(tmean15, ylab="Temperature anomalies")`

The rectangular box's mid line indicates the level of the median, which is -0.30°C. The rectangular box's lower boundary is the first quartile, i.e., 25-percentile. The

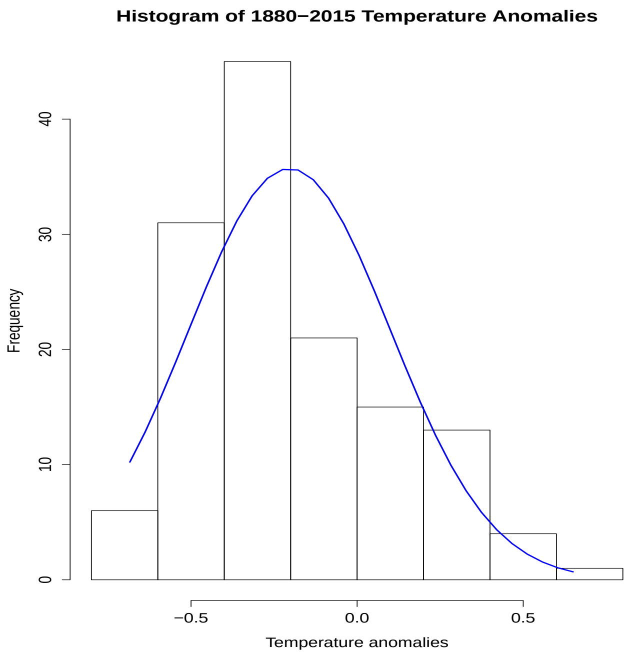


Fig. 6.2 Histogram of the global average annual mean temperature anomalies from 1880-2015.

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 the 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 would then be lower than the lower extreme. Thus, the lower whisker takes the value of the lower extreme, which is -0.68 °C. The points outside of the two whiskers are considered outliers. Our dataset has one outlier, which is 0.65 °C. This is the hottest year in the dataset. It was year 2015.

Sometimes, one may need to plot multiple box plots on the same figure, which can be done by R. One can look at an example in the R-project document
`http://www.inside-r.org/r-doc/graphics/boxplot`

6.2.3 Scatter plot

The scatter plot is convenient for displaying whether two datasets are correlated with one another. 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 code can produce the scatter plot shown in Fig. 6.4.

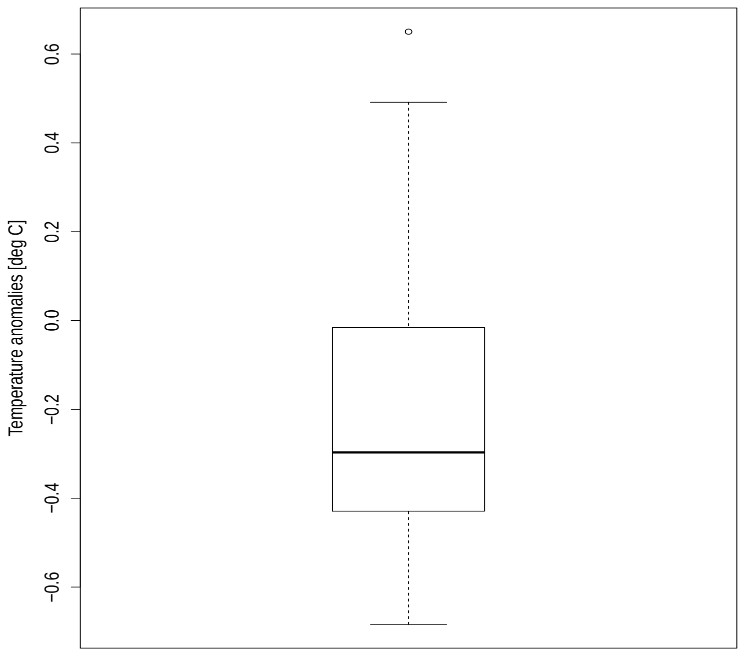


Fig. 6.3 Box plot of the global average annual mean temperature anomalies from 1880-2015.

```
#Use setwd("working directory") to work in the right directory
rm(list=ls())
setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2016/chap4data-refs")
par(mgp=c(1.5,0.5,0))
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,xlim=c(-4,4), ylim=c(-8,8),
     main="January SOI and the U.S. Temperature",
     xlab="SOI [dimensionless]",
     ylab="US Temperature deg F",
     pch=19, cex.lab=1.3)
# Plot the scatter plot
soiust=lm(ustj ~ soij) #Linear regression
```

```
abline(soiust, col="red", lwd=3) #Linear regression line
```

The correlation between the two datasets is 0. Thus, the slope of the red trend line is also zero.

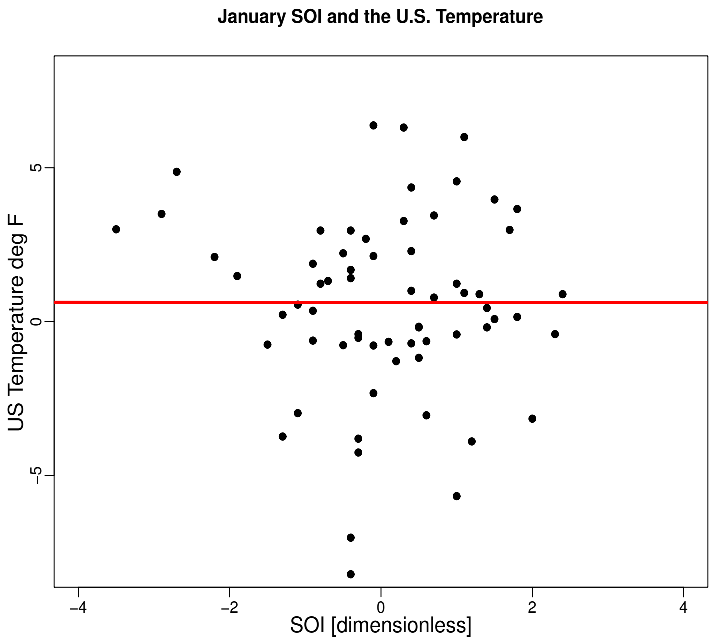


Fig. 6.4 Scatter plot of the January U.S. temperature vs. the January SOI from 1951-2016.

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

The R commands to retain the data without the above five El Niño 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 predictive skill for the January temperatures of the contiguous United States, for the non-El Niño years. This correlation is stronger for specific regions of the U.S. The physical reason for this result has to do with the fact that the temperature field over the U.S. is inhomogeneous, and in different regions, it is related to the tropical ocean dynamics in different ways. This gives us a hint as to how to find the predictive skill for a specific objective field:

to create a scatter plot using the objective field, which is being predicted, and the field used for making the prediction. The objective field is called the predicant or predictand, and the field used to make the prediction is called the predictor. A very useful predictive skill would be that the predictor leads the predicant by a certain time, say one month. Then the scatter plot will be made from the pairs between predictor and predicant data with one-month lead. The absolute value of the correlation can then be used as a measure of the predictive skill. Since the 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 (CCA). Therefore, before a prediction is made, it is a good idea to examine the predictive 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 predictions or for weakly nonlinear relationships. Nature can sometimes be very nonlinear, which require more sophisticated assessments of predictive skill, such as neural networks and time-frequency analysis. The CCA and other advanced statistical prediction methods are beyond the scope of this book.

6.2.4 QQ-plot

Figure 6.5 is called a QQ-plot. It shows a considerable degree of scattering of the QQ-plot points away from the red diagonal line, which is called the standard normal line. We may intuitively conclude that the global average annual temperature anomalies from 1880 to 2015 are not exactly distributed according to a normal (or Gaussian) distribution. However, we may also conclude that the distribution of these temperatures is not very far away from the normal distribution either, because the points on the QQ-pot are not very far away from the red diagonal line.

The function of a QQ-plot is to compare the distribution of a given set of data with a specific reference distribution, such as a standard normal distribution with zero mean and standard deviation equal to one, denoted by $N(0, 1)$. A QQ-plot lines up the percentiles of data on the vertical axis and the same number of percentiles of the specific distribution on the horizontal axis. The pairs of the percentiles $(x_i, y_i), i = 1, 2, \cdots, n$ determine the points on the QQ-plot. A QQ-line is plotted as if the vertical axis values are also the percentiles of the given specific distribution. Thus, the QQ-line should be a diagonal line when the vertical scale and the horizontal scale are the same.

To check if our global average annual mean temperature data are normally distributed, we first standardize (or normalize) the data by subtracting the data mean and dividing by the data's standard deviation, and then we plot the QQ-plot, which is shown in Fig. 6.5. The figure was plotted using the following R code:

```
#qq-plot for standardized anomalies
tstand = (tmean15-mean(tmean15))/sd(tmean15)
qqnorm(tstand, ylab="Global Temperature Anomalies [deg C]",
```

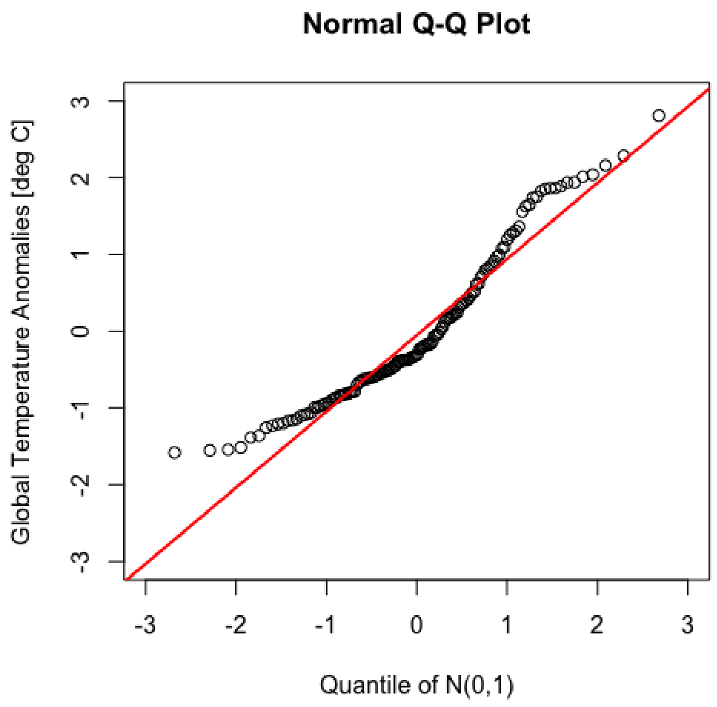


Fig. 6.5 QQ-plot of the standardized global average annual mean temperature anomalies vs. standard normal distribution.

```
        xlab="Quantile of N(0,1)", xlim=c(-3,3),ylim=c(-3,3))
qqline(tstand, col = "red", lwd=2)
```

6.3 Probability distributions

This section describes a few basic probabilistic distributions in addition to the “bell-shaped” normal or Gaussian distribution we often have in mind.

6.3.1 What is a probability distribution?

A probability distribution is chance of occurrence of an event at a certain value or an interval of values. For example, if the daily weather at a location is classified as being in one of two categories: clear weather days, defined as from 0 to 3/10 average sky cover by clouds, and cloudy weather days, defined as from 4/10 to 10/10 average sky cover, then the resulting probability distribution is the probability value of clear and cloudy days. Table 6.1 shows the probabilities of clear weather for three United States cities based on historical data: Seattle 0.16, San Diego, 0.58, and Las Vegas 0.58. The probability distribution table obviously reflects the very different climates of the three cities. Seattle is a Pacific Northwest U.S. city characterized by weather that is often cloudy or rainy, particularly in the winter. San Diego is a Pacific Southwest U.S. city where it rarely rains, but where cloud cover may be relatively large in May and June, the so-called "May Gray and June Gloom." Las Vegas is a U.S. Southwest inland desert city, which has often experiences a clear sky during the daytime.

Table 6.1 Probability distribution of weather

Location	Clear Sky	Cloudy Sky
Las Vegas	0.58	0.42
San Diego	0.40	0.60
Seattle	0.16	0.84

Data source: NOAA Desert Research Institute, July 2017
`https://wrcc.dri.edu/htmlfiles/westcomp.clr.html`

The data of Table 6.1 can also be displayed by the bar chart in Fig. 6.6. This figure visually displays the different cloudiness climates of the three cities. Thus, either the table or the figure demonstrates that a probability distribution can be a good description of important properties of a random variable, such as cloud cover. Here, a random variable means a variable that can take on a value in a random way, such as weather conditions (sunny, rainy, snowy, cloudy, stormy, windy, etc). Almost anything we deal with in our daily lives is a random variable, that is to say, a variable which has a random nature, in contrast to a deterministic variable. We describe a random variable by probability and explore what is the probability of the variable having a certain value or a certain interval of values. This description is the probability distribution.

Figure 6.6 can be generated by the following R code.

```
plot.new()
layout(matrix(c(1,2,3), 1, 3, byrow = TRUE),
       widths=c(3,3,3), heights=c(1,1,1))
lasvegas=c(0.58,0.42)
sandiego=c(0.4,0.6)
seattle=c(0.16,0.84)
names(lasvegas)=c("Clear","Cloudy")
names(sandiego)=c("Clear","Cloudy")
```

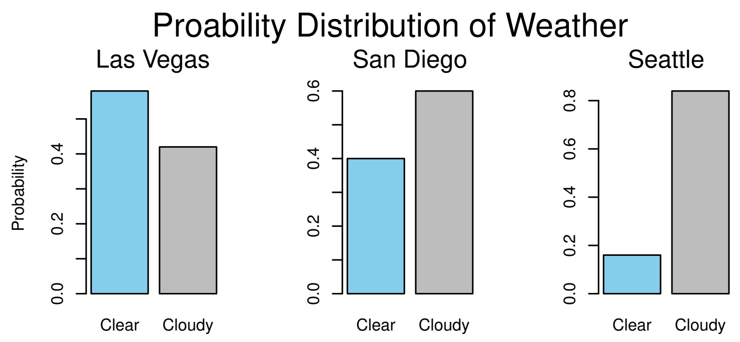


Fig. 6.6 Probability distributions of different climate conditions according to cloudiness for three cities in the United States.

```
names(seattle)=c("Clear","Cloudy")
barplot(lasvegas,col=c("skyblue","gray"),ylab="Probability")
mtext("Las Vegas", side=3,line=1)
barplot(sandiego,col=c("skyblue","gray"))
mtext("San Diego", side=3,line=1)
barplot(seattle,col=c("skyblue","gray"))
mtext("Seattle", side=3,line=1)
mtext("Proability Distribution of Weather",
      cex=1.3,side = 3, line = -1.5, outer = TRUE)
```

A probability distribution can be expressed not only by a table as shown above, but also by bar chart, a curve, or a function $y = f(x)$. Bart charts are used for the random variables which can take on discrete values, such as clear sky or cloudy sky, or intervals of continuous values, such as the temperature in the intervals $(0-5, 6-10, 11-15, 16-20, 21-25, 25-30, 31-35)$°C for San Diego. A smooth curve or a function $y = f(x)$ is often used to describe a continuous distribution, of which a random variable can take on any real value in a given range, such as San Diego temperature in the range of $(-50, 50)$°C. In the case of a continuous curve, the curve's vertical coordinate value $f(x)$ is not probability, but the value times an interval length. Thus, $f(x)\Delta x$ is the probability for the random variable to be in the interval $(x, x + \Delta x)$. In this sense, the curve resembles density in the case of mass calculation. We therefore call the curve the probability density function (pdf). The domain of the pdf $f(x)$ is the entire range of all the possible values of the random variable x. Thus, the probability for x to have a value somewhere in the entire range is one, i.e., the sum of $f(x)\Delta x$ for the entire range is one. Following the method of calculus, when Δx approaches zero and is denoted by dx, the probability one can be expressed as an integral of the pdf $f(x)$:

$$\int_D f(x)dx = 1, \tag{6.6}$$

where D is the domain of the pdf, the entire range of the possible x values, e.g., $D = (-50, 50)$°C in the case of temperature for the U.S. This formula is called the probability normalization condition, as shown in Fig. 6.7.

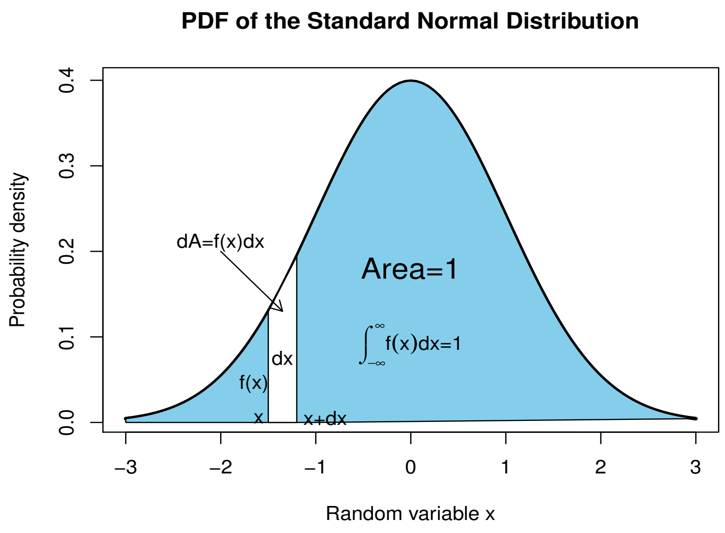


Fig. 6.7 Normalization condition of a probability distribution function.

Figure 6.7 can be generated by the following R code

```
# Create data for the area to shade
cord.x <- c(-3,seq(-3,3,0.01),-1)
cord.y <- c(0,dnorm(seq(-3,3,0.01)),0)
# Make a curve
curve(dnorm(x,0,1), xlim=c(-3,3), lwd=3,
      main='PDF of the Standard Normal Distribution',
      xlab="Random variable x",
      ylab='Probability density')
# Add the shaded area using many lines
polygon(cord.x,cord.y,col='skyblue')
polygon(c(-1.5,-1.5, -1.2, -1.2),c(0, dnorm(-1.5),
         dnorm(-1.2), 0.0),col='white')
text(0,0.18, "Area=1", cex=1.5)
text(-1.65,0.045,"f(x)")
text(-1.35,0.075,"dx")
text(-1.6,0.005,"x")
text(-0.9,0.005,"x+dx")
arrows(-2,0.2,-1.35,0.13, length=0.1)
text(-2,0.21,"dA=f(x)dx")
text(0,0.09,expression(paste(integral(f(x)*dx,- infinity,infinity),"=1")))
```

Of course, the normalization condition for a discrete random value, such as clear and cloudy skies, is a summation, rather than the above integral. Consider the San Diego case in Table 6.1. The normalization condition is $0.40 + 0.60 = 1.0$.

6.3.2 Normal distribution

Figure 6.8 shows five different normal distributions, each of which is a bell-shaped curve with the highest density when the random variable x takes the mean value, and approaches zero as x goes to infinity. The figure can be generated by the following R code.

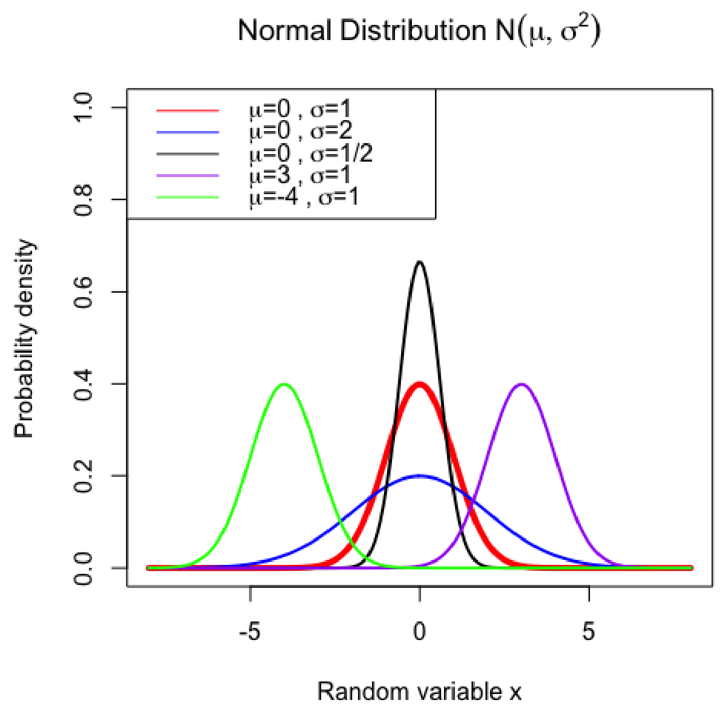


Fig. 6.8 Probability density function for five normal distributions.

```
#Normal distribution plot
```

```
x <- seq(-8, 8, length=200)
plot(x,dnorm(x, mean=0, sd=1), type="l", lwd=4, col="red",
     ylim = c(0,1),
     xlab="Random variable x",
     ylab ="Probability D=density",
     main=expression(Normal~Distribution ~ N(mu,sigma^2)))
lines(x,dnorm(x, mean=0, sd=2), type="l", lwd=2, col="blue")
lines(x,dnorm(x, mean=0, sd=0.6), type="l", lwd=2, col="black")
lines(x,dnorm(x, mean=3, sd=1), type="l", lwd=2, col="purple")
lines(x,dnorm(x, mean=-4, sd=1), type="l", lwd=2, col="green")
#ex.cs1 <- expression(plain(sin) * phi,  paste("cos", phi))
ex.cs1 <- expression(paste(mu, "=0",~",",~ sigma, "=1"),
                     paste(mu, "=0",~",",~ sigma, "=2"),
                     paste(mu, "=0",~",",~ sigma, "=1/2"),
                     paste(mu, "=3",~",",~ sigma, "=1"),
                     paste(mu, "=-4",~",",~ sigma, "=1"))
legend("topleft",legend = ex.cs1, lty=1,
       col=c('red','blue','black','purple','green'), cex=1, bty=n)
```

The bell-shaped normal distribution curve can be expressed by a mathematical formula

$$f(x|\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right), \tag{6.7}$$

where μ is the mean and σ is the standard deviation of the normal distribution, and σ^2 is called the variance. The mean is the value one would expect to occur with the highest probability, and is called the expected value. The standard deviation measures how much the actual values deviate away from the mean. The pdf's peak is at the mean. The pdf is flatter for a large standard deviation, and more peaked for a smaller standard deviation. Figure 6.8 clearly shows these properties. Notice how the bell-shaped curve changes due to different values of μ and σ. The mean reflects the mean state of the random variable and hence determines the position of the bell-shaped curve; and the standard deviation reflects the diversity of the random variable and determines the shape of the curve.

Here, x, μ, and σ have the same unit, and, of course, the same dimension.

The probability, or the area, under the entire bell-shaped curve is one. The probability in the interval $(\mu - 1.96\sigma, \mu + 1.96\sigma)$ is 0.95, and that in $(\mu - \sigma, \mu + \sigma)$ is 0.68. These are commonly used properties of a normal distribution. Sometimes we regard 1.96 approximately as 2, and $(\mu - 1.96\sigma, \mu + 1.96\sigma)$ as two standard deviations away from the mean. A corresponding mathematical expression is

$$\int_{\mu-2\sigma}^{\mu+2\sigma} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) dx = 0.95. \tag{6.8}$$

One can use an R code to verify this formula:

```
mu=0
sig=1
intg <- function(x){(1/(sig*sqrt(2*pi)))*exp(-(x-mu)^2/(2*sig^2))}
integrate(intg,-2,2)
#0.9544997 with absolute error < 1.8e-11
#Or using the R built-in function dnorm to get the same result
integrate(dnorm,-2,2)
#0.9544997 with absolute error < 1.8e-11
integrate(dnorm,-1.96,1.96)
#0.9500042 with absolute error < 1e-11
```

6.3.3 Student's t-distribution

Figure 6.9 shows Student's t-distribution , or simply the t-distribution. It is used when estimating the mean of a normally distributed variable with a small number of data points and an unknown standard deviation. William Gosset (1876-1937) published the t-distribution under the pseudonym"Student" while working at the Guinness Brewery in Dublin, Ireland. Gosset worked as a brewer, because Guinness hired scientists who could apply their skills to brewing. In 1904, Gosset wrote a report called The Application of the Law of Error to the work of the Brewery. In his report, Gosset advocated using statistical methods in the brewing industry. Gosset published under a pseudonym, because the brewery did not allow its scientists to publish their research using their real names, perhaps because the information contained in the research might give a competitive advantage to the brewery. Gosset corresponded with leading statisticians of the time, however, and gained their respect because of his research.

If $x_1, x_2, \cdots, x_n$ are normally distributed data with a given mean μ, an unknown standard deviation, and a small sample n, say, $n < 30$, then

$$t = \frac{\bar{x} - \mu}{S/\sqrt{n}} \tag{6.9}$$

follows a t-distribution with $n - 1$ degrees of freedom (df), where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{6.10}$$

is the estimated sample mean, and

$$S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 \tag{6.11}$$

is the estimated sample variance.

The random variable t is essentially a measure of the deviation of the sample mean from the given mean value normalized by the estimated standard deviation scaled down by *sqrtn*. The pdf of the random variable t can be plotted by the following R code

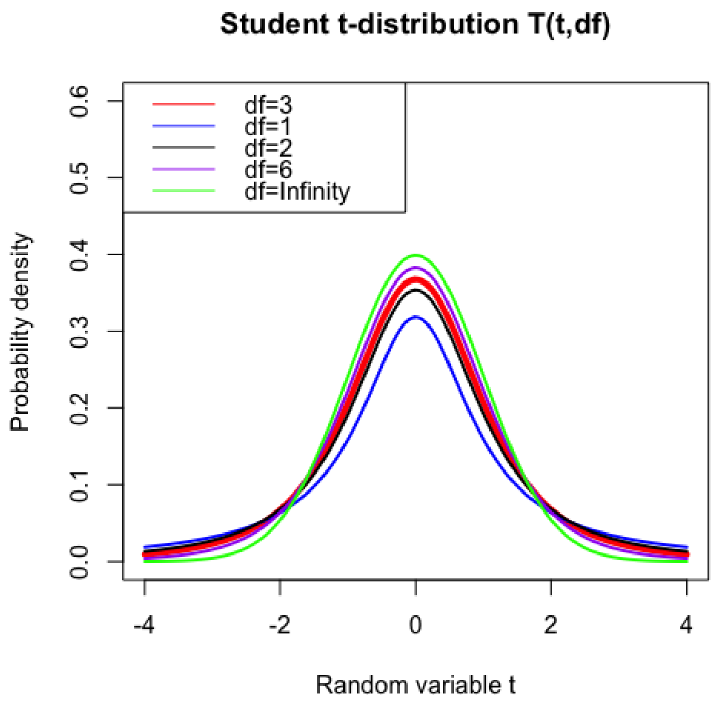


Fig. 6.9 Probability density function for five t-distributions with different degrees of freedom.

```
 #Plot t-distribution by R
x <- seq(-4, 4, length=200)
plot(x,dt(x, df=3), type="l", lwd=4, col="red",
     ylim = c(0,0.6),
     xlab="Random variable t",
     ylab ="Probability density",
     main="Student t-distribution T(t,df)")
lines(x,dt(x, df=1), type="l", lwd=2, col="blue")
lines(x,dt(x, df=2), type="l", lwd=2, col="black")
lines(x,dt(x, df=6), type="l", lwd=2, col="purple")
lines(x,dt(x, df=Inf), type="l", lwd=2, col="green")
#ex.cs1 <- expression(plain(sin) * phi,  paste("cos", phi))
ex.cs1 <- c("df=3", "df=1","df=2","df=6","df=Infinity")
legend("topleft",legend = ex.cs1, lty=1,
       col=c('red','blue','black','purple','green'), cex=1, bty=n)
```

When the df, the number of degrees of freedom ($df = n - 1$) is infinity, the t-distribution is exactly the same as the standard normal distribution $N(0, 1)$. Even when $df = 6$, the t-distribution is already very close to the standard normal distribution. Thus, t-distribution is meaningfully different from the standard normal distribution only when the sample size is small, say, n=5 (i.e., df=4).

The exact mathematical expression of the pdf for the t-distribution is quite complicated and uses a Gamma function, which is a special function beyond the scope of this book.

6.4 Estimate and its error

6.4.1 Probability of a sample inside a confidence interval

If the data $(x_1, x_2, \cdots, x_n)$ are normally distributed with the same mean μ and standard deviation σ, then the sample mean, i.e., the mean of the data

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{6.12}$$

is normally distributed with mean equal to μ and standard deviation equal to $\sigma/\sqrt{n}$.

Given the sample size n, mean μ, and standard deviation σ for a set of normal data, what is the interval $[a, b]$ such that the 95% of the sample means will occur within the interval $[a, b]$? Intuitively, the sample mean should be close to the true mean μ most of the times. However, because the sample data are random, the sample means are also random and may be very far away from the true mean. For the example of the global temperature, we might assume that the "true" mean is 14°C and the "true" standard deviation is 0.3 °C. Here, "true" is an assumption, however, since no one knows the truth. The sample means are close to 14 most of the time, but climate variations may lead to a sample mean being equal to 16°C or 12°C, thus far away from the "true" mean 14°C. We can use the interval $[a, b]$ to quantify the probability of the sample mean being inside this interval. We wish to say that with 95% probability, the sample mean is inside this interval $[a, b]$. This leads to the following confidence interval formula.

For a normally distributed population $(x_1, x_2, \cdots, x_n)$ with the same mean μ and standard deviation σ, the confidence interval at the 95% confidence level is

$$(\mu - 1.96\sigma/\sqrt{n}, \mu + 1.96\sigma/\sqrt{n}). \tag{6.13}$$

Namely, with 95% probability, the sample mean $\bar{x}$ is

$$\mu - 1.96\sigma/\sqrt{n} < \bar{x} < \mu + 1.96\sigma/\sqrt{n}. \tag{6.14}$$

Usually, $a = \mu - 1.96\sigma/\sqrt{n}$ is called the lower limit of the confidence interval, and $b = \mu + 1.96\sigma/\sqrt{n}$ the upper limit.

One can easily simulate this confidence interval formula by the following R code.

```
 #Confidence interval simulation
mu=14 #true mean
sig=0.3 #true sd
n=50 #sample size
d=1.96*sig/sqrt(n)
lowerlim=mu-d
upperlim=mu+d
ksim=10000 #number of simulations
k=0 #k is the simulation counter
for (i in 1:ksim)
{
xbar=mean(rnorm(n, mean=mu, sd=sig))
if (xbar >= lowerlim & xbar <= upperlim)
  k=k+1
}
print(c(k,ksim))
#[1]  9496 10000

#plot the histogram
hist(xbar,breaks=51,xlab="Temperature [deg C]",
     main="Histogram of Simulated
     Sample Mean Temperatures",xaxt="n",
     ylim=c(0,600))
axis(1,at =c(13.92, 14.0, 14.08))
text(14,550,"95% Confidence Interval (13.92,14.08)",cex=1.2)
```

This simulation shows that 9,496 of the 10,000 simulations have the sample means inside the confidence interval. The probability is thus 0.9496, or approximately 0.95. Figure 6.10 displays the histogram of the simulation results. It shows that 9,496 sample means from among 10,000 are in the confidence interval $(13.92, 14.08)$. Only 504 sample means are outside the interval with 254 in $(-\infty, 13.92)$ and 250 in $(14.08, \infty)$. Thus, the confidence level is the probability of the sample mean falling into the confidence interval. Intuitively, when the confidence interval is small, the confidence level is low since there is a smaller chance for the sample mean to fall into a smaller interval.

6.4.2 Mean of a large sample size: Approximately normal distribution

6.4.2.1 Confidence interval of the sample mean

The purpose of computing the sample mean is to use it as an estimate for the real true mean that we do not know in practice. This estimation is more accurate

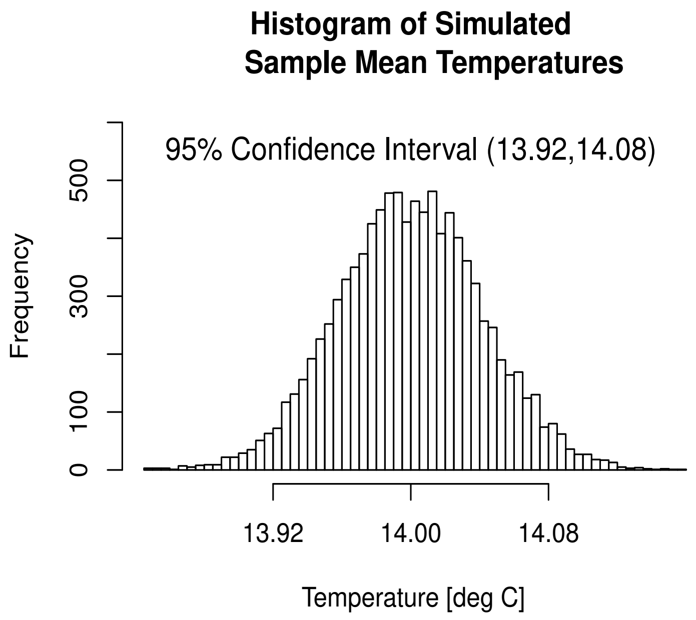


Fig. 6.10 Histogram of 10,000 simulated sample mean temperature based on the assumption of normal distribution with the "true" mean equal 14°C and "true" standard deviation 0.3 °C. Approximately, 95% of the sample means are within the confidence interval (13.92, 14.08), 2.5% in $(14.08, \infty)$, and 2.5% in $(-\infty, 13.92)$.

when the confidence interval is small. The extreme case is that the confidence interval has zero length, which means that with 95% chance, the sample mean is exactly equal to the true mean. The chance to be wrong is only 5%. To be more accurate, our intuition suggests that we need to have a small standard deviation, and have a large sample. The above confidence interval formula (6.13) quantifies this intuition $(\mu - 1.96\sigma/\sqrt{n}, \mu + 1.96\sigma/\sqrt{n})$. A small σ and a large n enable us to have a small confidence interval, and hence an accurate estimation of the mean. Thus, to obtain an accurate result in a survey, one should use a large sample. This subsection shows a method to find out how large a sample should be, for the case when the confidence probability is given. We also want to deal with the practical situation where the true mean and standard deviation are almost never known. Furthermore, it is usually not known whether the random variable is in fact normally distributed. These two problems can be solved by a very important theoretical result of mathematical statistics, called the central limit theorem (CLT), which says that when the sample size n is sufficiently large, the sample mean $\bar{x} = \sum_{i=1}^{n} x_i/n$ is approximately normally distributed, regardless of the distributions of $x_i (i = 1, 2, \cdots, n)$. The approximation becomes better when n becomes larger.

Some textbooks suggest that $n = 30$ is good enough to be considered a "large" sample; others use $n = 50$. In climate science, we often use $n = 30$.

When the number of samples is large in this sense, the normal distribution assumption for the sample mean is taken care of. We then compute the sample mean and sample standard deviation by the following formulas

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \tag{6.15}$$

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i, -\bar{x})^2}. \tag{6.16}$$

The standard error of the sample mean is defined as

$$SE(\bar{x}) = \frac{S}{\sqrt{n}}. \tag{6.17}$$

This gives the size of the "error bar" $\bar{x} \pm SE$ when approximating the true mean using the sample mean.

The error margin at a 95% confidence level is

$$EM = 1.96\frac{S}{\sqrt{n}}, \tag{6.18}$$

where 1.96 comes from the 95% probability in $(\mu - 1.96\sigma, \mu + 1.96\sigma)$ for a normal distribution. When the confidence level α is raised from 0.95 to a larger value, the number 1.96 will be increased to a larger number accordingly.

The confidence interval for a true mean μ is then defined as

$$(\bar{x} - EM, \bar{x} + EM) \text{ or } (\bar{x} - 1.96\frac{S}{\sqrt{n}}, \bar{x} + 1.96\frac{S}{\sqrt{n}}). \tag{6.19}$$

This means that the given samples imply that the probability for the true mean to be inside the confidence interval $(\bar{x} - EM, \bar{x} + EM)$ is 0.95, or α in general. Similarly, the probability for the true mean to be inside the error bar $\bar{x} \pm SE$ is 0.68. See Fig. 6.11 for the confidence intervals at 95% and 68% confidence levels.

When the sample size n goes to infinity, the error margin EM goes to zero, and accordingly, the sample mean is equal to the true mean. This is correct with 95% probability, and wrong with 5% probability.

One can also understand the sample confidence interval for a new variable

$$z = \frac{\bar{x} - \mu}{S/n}, \tag{6.20}$$

which is a normally distributed variable with mean equal to zero and standard deviation equal to one, i.e., it has standard normal distribution. The variable $y = -z$ also satisfies the standard normal distribution. So, the probability of $-1 < z < 1$ is 0.68, and $-1.96 < z < 1.96$ is 0.95. The set $-1.96 < z < 1.96$ is equivalent to $\bar{x} - 1.96S/\sqrt{n} < \mu < \bar{x} + 1.96S/\sqrt{n}$. Thus, the probability of the true mean in the confidence interval of the sample mean $\bar{x} - 1.96S/\sqrt{n} < \mu < \bar{x} + 1.96S/\sqrt{n}$ is 1.96. This explanation is visually displayed in Fig. 6.11.

In addition,the formulation $\bar{x} = \mu + zS/\sqrt{n}$ corresponds to a standard statistics problem for an instrument with observational errors:

$$y = x \pm \epsilon, \tag{6.21}$$

where ϵ stands for errors, x is the true but never-known value to be observed, and y is the observational data. Thus, data are equal to the truth plus errors. The expected value of the error is zero and the standard deviation of the error is $S/\sqrt{n}$, also called standard error.

The confidence level 95% comes into the equation when we require that the observed value must lie in the interval $(\mu - EM, \mu + EM)$ with a probability equal to 0.95. This corresponds to the requirement that the standard normal random variable z is found in the interval (z_-, z_+) with a probability equal to 0.95, which implies that $z_- = -1.96$ and $z_+ = 1.96$. Thus, the confidence interval of the sample mean at the 95% confidence level is

$$(\bar{x} - 1.96S/\sqrt{n}, \bar{x} + 1.96S/\sqrt{n}), \tag{6.22}$$

or

$$(\bar{x} - z_{\alpha/2}S/\sqrt{n}, \bar{x} + z_{\alpha/2}S/\sqrt{n}), \tag{6.23}$$

where $z_{\alpha/2} = z_{0.05/2} = 1.96$. So, $1 - \alpha = 0.95$ is used to represent the probability inside the confidence interval, while $\alpha = 0.05$ is the "tail probability" outside of the confidence interval. Outside of the confidence interval means occurring on either the left side or the right side of the distribution. Each side represents $\alpha/2 = 0.025$ tail probability. The red area of Fig. 6.11 indicates the tail probability.

Figure 6.11 can be plotted by the following R code.

```
#Figure of confidence intervals and tail probability
rm(list=ls())
par(mgp=c(1.4,0.5,0))
curve(dnorm(x,0,1), xlim=c(-3,3), lwd=3,
      main='Confidence Intervals
      and Confidence Levels',
      xlab="True mean as a random variable",
      ylab='Probability density',xaxt="n",
      cex.lab=1.3)
polygon(c(-1.96, seq(-1.96,1.96,len=100), 1.96),
        c(0,dnorm(seq(-1.96,1.96,len=100)),0),col='skyblue')
polygon(c(-1.0,seq(-1.0, 1, length=100), 1),
        c(0, dnorm(seq(-1.0, 1, length=100)), 0.0),col='white')
polygon(c(-3.0,seq(-3.0, -1.96, length=100), -1.96),
        c(0, dnorm(seq(-3.0, -1.96, length=100)), 0.0),col='red')
polygon(c(1.96,seq(1.96, 3.0, length=100), 3.0),
```

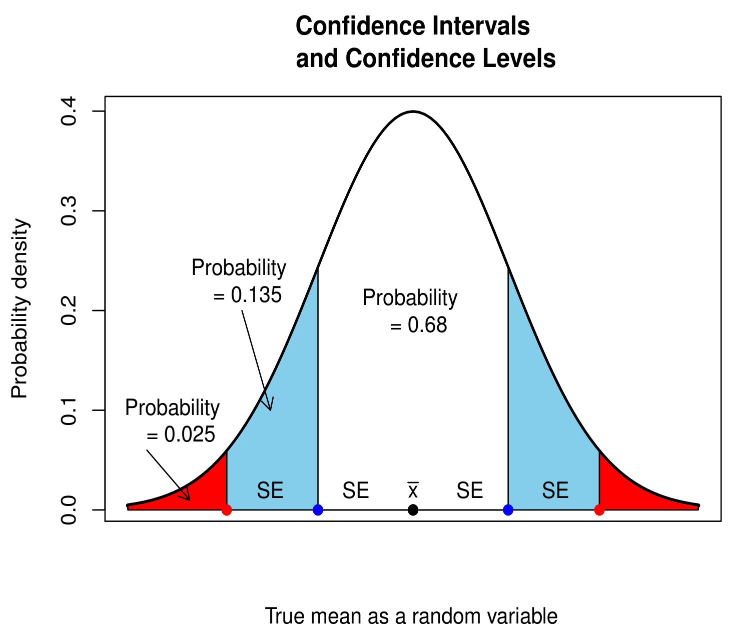


Fig. 6.11 Schematic illustration of confidence intervals and confidence levels of a sample mean for a large sample size. The confidence interval at 95% confidence level is between the two red points, and that at 68% is between the two blue points. SE stands for the standard error, and 1.96 SE is approximately regarded as 2 SE in this figure.

```
        c(0, dnorm(seq(1.96, 3.0, length=100)), 0.0),col='red')
points(c(-1,1), c(0,0), pch=19, col="blue")
points(0,0, pch=19)
points(c(-1.96,1.96),c(0,0),pch=19, col="red")
text(0,0.02, expression(bar(x)), cex=1.0)
text(-1.50,0.02, "SE", cex=1.0)
text(-0.60,0.02, "SE", cex=1.0)
text(1.50,0.02, "SE", cex=1.0)
text(0.60,0.02, "SE", cex=1.0)
text(0,0.2, "Probability
     = 0.68")
arrows(-2.8,0.06,-2.35,0.01, length=0.1)
text(-2.5,0.09, "Probability")
```

In practice, we often regard 1.96 as 2.0, and the 2σ-error bar as the 95% confidence interval.

Example 1. Estimate (a) the mean of the 1880-2015 global average annual mean temperatures of the Earth, and (b) the confidence interval of the sample mean at the 95% confidence level.

The answer is that the mean is -0.2034°C and the confidence interval is $(-0.2545, -0.1524)$°C. These values may be obtained by the following R code.

```
#Estimate the mean and error bar for a large sample
#Confidence interval for NOAAGlobalTemp 1880-2015
setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2017/Book-ClimMath-Cambridge-
dat1 <- read.table("aravg.ann.land_ocean.90S.90N.v4.0.0.2015.txt")
dim(dat1)
tmean15=dat1[,2]
MeanEst=mean(tmean15)
sd1 =sd(tmean15)
StandErr=sd1/sqrt(length(tmean15))
ErrorMar = 1.96*StandErr
MeanEst
#[1] -0.2034367
print(c(MeanEst-ErrorMar, MeanEst+ErrorMar))
#[1] -0.2545055 -0.1523680
```

6.4.2.2 Estimate the required sample size

The standard error $SE = \sigma/\sqrt{n}$ measures the accuracy of using a sample mean as an estimate of the true mean when the standard deviation of the population is given as σ. A practical problem is to determine the sample size when the accuracy level SE is given. The formula is then

$$n = \left(\frac{\sigma}{SE}\right)^2. \tag{6.24}$$

Example 2. The standard deviation of the global average annual mean temperature is given to be 0.3°C. The standard error is required to be less or equal to 0.05°C. Find the minimal sample size required.

The solution is $(0.3/0.05)^2 = 36$. The sample size must be greater than or equal to 36.

6.4.2.3 Statistical inference for $\bar{x}$ using a z-score

Figure 4.1 seems to suggest that the average of the global average annual mean temperature anomalies from 1880 to 1939 is significantly below zero. We wish to know whether we can statistically justify that this inference is true, with the probability of being wrong less than or equal to 0.025, or 2.5%. This probability is called the significance level. Figure 6.12 shows the significance level as the tail probability in $(-\infty, z_{0.025})$.

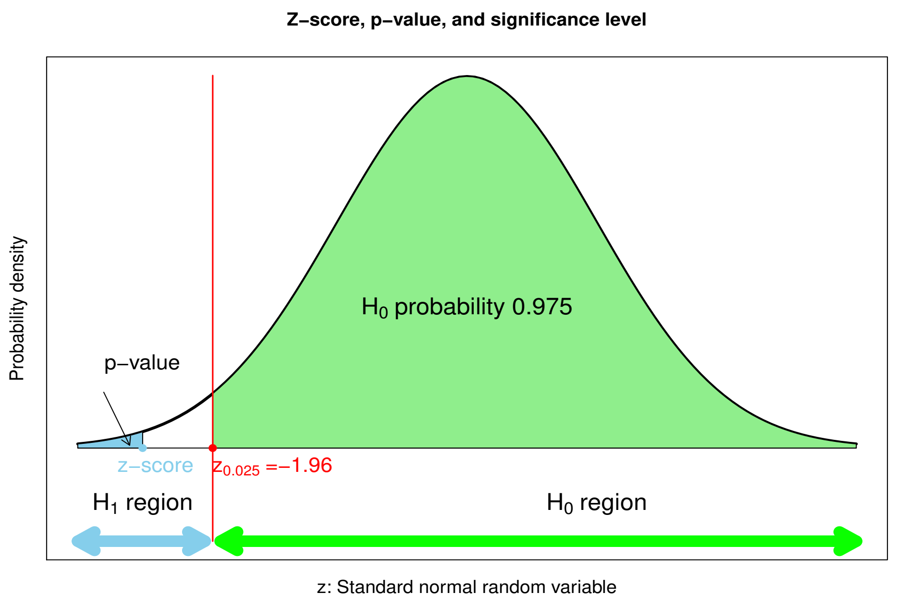


Fig. 6.12 The standard normal distribution chart for statistical inference: z-score, p-value for $\bar{x} < \mu$, and significance level 2.5%. The value $z_{0.025} = -1.96$ is called the critical z-score for this hypothesis test.

Figure 6.12 can be generated by the following R code.

```
rm(list=ls())
par(mgp=c(1.4,0.5,0))
curve(dnorm(x,0,1), xlim=c(-3,3), lwd=3,
main='Z-score, p-value, and significance level',
xlab="z: standard normal random variable",
ylab='Probability density',xaxt="n",
cex.lab=1.2, ylim=c(-0.02,0.5))
lines(c(-3,3),c(0,0))
lines(c(-1.96,-1.96),c(0, dnorm(-1.96)),col='red')
polygon(c(-3.0,seq(-3.0, -2.5, length=100), -2.5),
        c(0, dnorm(seq(-3.0, -2.5, length=100)), 0.0),col='skyblue')
points(-1.96,0, pch=19, col="red")
points(-2.5,0,pch=19, col="skyblue")
text(-1.8,-0.02, expression(z[0.025]), cex=1.3)
text(-2.40,-0.02, "z-score", cex=1.1)
arrows(-2.8,0.06,-2.6,0.003, length=0.1)
text(-2.5,0.09, "p-value", cex=1.3)
```

To make the justification, we compute a parameter

$$z = \frac{\bar{x} - \mu}{S/\sqrt{n}}, \tag{6.25}$$

where $\bar{x}$ is the sample mean, S is the sample standard deviation, and n is the sample size. This z value is called the z-statistic, or simply the z-score, which follows the standard normal distribution, because the sample size $n = 60$ is large. From the z-score, we can determine the probability of the random variable z being in a certain interval, such as $(-\infty, z_s)$. This significance level 2.5% corresponds to $z_s = -1.96$ according to Fig. 6.11. Thus, the z-score can quantify how significantly is z different from zero, which is equivalent to the sample mean being significantly different from the assumed or given value. The associated probability, e.g., the probability in $(-\infty, z)$, is called the p-value that measures the chance of a wrong inference. We want this p-value to be small in order to be able to claim significance. The typical significance levels used in practice are 5%, 2.5%, and 1%. Choosing which level to use depends on the nature of the problem. For drought conditions, one may use 5%, while for flood control and dam design, one may choose 1%. A statistical inference is significant when the p-value is less than the given significance level.

For our problem of 60 years of data from 1880-1939, the sample size is $n = 60$. The sample mean can be computed by an R command `xbar=mean(tmean15[1:60])`, and the sample standard deviation can be computed by `S=sd(tmean15[1:60])`. The results are $\bar{x} = -0.4500$ and $S = 0.1109$.

When $\mu = 0$, the z-score computed using formula (6.25) is -31.43. The probability in the interval $(-\infty, z)$ is tiny, namely 4.4×10^{-217}, which can be regarded as zero. We can thus conclude that the sample mean from 1880-1939 is significantly less than zero at a p-value equal to 4.4×10^{-217}, which means tat our conclusion is correct at a significance level of 2.5%.

A formal statistical terminology for the above inference is called hypothesis test, which tests a null hypothesis

$$H_0 : \bar{x} \geq 0, \quad \text{(Null hypothesis: the mean is not smaller than zero)} \tag{6.26}$$

and an alternative hypothesis

$$H_1 : \bar{x} < 0, \quad \text{(Alternative hypothesis: the mean is smaller than zero).} \tag{6.27}$$

Our question of the average temperature from 1880-1939 is to reject the null hypothesis and confirm the alternative hypothesis. The method is to examine where the z-score point is on a standard normal distribution chart and what is the corresponding p-value. Thus, the statistical inference becomes a problem of z-score and p-value using the standard normal distribution chart (See Fig. 6.12). Our z-score is -31.43 in the H_1 region, and our p-value is 4.4×10^{-217}, much less than 0.025. We thus accept the alternative hypothesis, i.e., we reject the null hypothesis with a tiny p-value 4.4×10^{-217}. We conclude that the 1880-1939 mean temperature is significantly less than zero.

One can similarly formulate a hypothesis test for a warming period from 1981-2015 and ask whether the average temperature during this period is significantly greater than zero. The two hypotheses are

$$H_0 : \bar{x} \leq 0, \quad \text{(Null hypothesis: the mean is not greater than zero)} \tag{6.28}$$

and an alternative hypothesis

$$H_1 : \bar{x} > 0, \quad \text{(Alternative hypothesis: the mean is greater than zero).} \tag{6.29}$$

One can follow the same procedure to compute the z-score, see whether it in the H_0 region or H_1 region, and compute the p-value. Finally an inference can be made based on the z-score and the p-value.

6.4.3 Mean of a small sample size: t-test

6.4.3.1 $H_1 : \bar{T} > 0$ test for the 2006-2016 global average annual temperature

The hypothesis test in the above subsection is based on the standard normal distribution for the cases of a large sample size, say, at least 30. When the sample size is small, the sample mean satisfies a t-distribution, not a normal distribution.

Thus, when the sample size n is small, say less than 10, and the variance is to be estimated, then we should use a t-distribution, because

$$t = \frac{\bar{x} - \mu}{S/\sqrt{n}} \tag{6.30}$$

follows a t-distribution of the degrees-of-freedom (df or dof) equal to $n-1$. Figure 6.9 shows that the t-distribution is flatter than the corresponding normal distribution (of the same sample mean and sample variance) and has fatter tails. When the dof increases to infinity, the t-distribution approaches the normal distribution of the sample mean and sample variance.

The hypothesis test procedure is the same as before, except the standard normal distribution is now replaced by the t-distribution with dof equal to $n - 1$.

Example 3. Test whether the global average annual mean temperature from 2006-2015 is significantly greater than the 1961-1990 climatology, i.e., whether the sample mean is greater than zero.

The two hypotheses are

$$H_0 : \bar{T} \leq 0, \quad \text{(Null hypothesis: The 2006-2015 mean is not greater than zero)} \tag{6.31}$$

and

$$H_1 : \bar{T} > 0, \quad \text{(Alternative hypothesis: The mean is greater than zero).} \tag{6.32}$$

One can follow the same procedure as in the last section to compute the t-score, see whether it in the H_0 region or H_1 region, and to compute the p-value. We have 10 years of data from 2006-2015, which is a small sample with a sample size $n = 10$.

The following R code computes the sample mean 0.4107, standard deviation 0.1023, t-core 12.6931, p-value 2.383058×10^{-7}, and the critical t-value: $t_{0.975} = 2.2622$. The t-score is in the alternative hypothesis region with a very small p-value. Therefore, we conclude that the average temperature from 2006-2015 is significantly greater than zero.

```
 #Hypothesis test for NOAAGlobalTemp 2006-2015
setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2017/Book-ClimMath-Cambridge-
dat1 <- read.table("aravg.ann.land_ocean.90S.90N.v4.0.0.2015.txt")
tm0615=dat1[127:136,2]
MeanEst=mean(tm0615)
MeanEst
#[1] 0.4107391
sd1 =sd(tm0615)
sd1
#[1] 0.1023293
n=10
t_score=(MeanEst -0)/(sd1/sqrt(n))
t_score
#[1] 12.69306
1-pt(t_score, df=n-1)
#[1] 2.383058e-07 #p-value
qt(1-0.025, df=n-1)
#[1] 2.262157 #critical t-score
```

For the standard normal distribution, $z_{0.975} = 1.96 < t_{0.975} = 2.2622$, because the t-distribution is flatter than the corresponding normal distribution and has fatter tails. Thus, the critical t-scores are larger.

Clearly, one should use the t-test to make the inference when the sample size is very very small, say, n = 7. However, it is unclear whether one should use the t-test or the z-test if the sample size is, say, 27. The recommendation is to always use the t-test if you are not sure whether the z-test is applicable, because t-test has been mathematically proven to be accurate, while the z-test is an approximation. Since the t-distribution approaches the normal distribution when dof approaches infinity, the t-test will yield the same result as the z-test when the z-test is applicable.

6.4.3.2 Compare temperatures of two short periods

A common question in climate science is whether the temperature in one decade is significantly greater than the temperature in another. The task is thus to compare the temperatures of two decades.

The general problem is whether the sample mean of the data $\{T_{11}, T_{12}, \cdots, T_{1n_1}\}$ and the sample mean of another set of data $\{T_{21}, T_{22}, \cdots, T_{2n_2}\}$ are significantly different from each other. The t-statistic for this problem can be computed using the following formula:

$$t = \frac{\bar{T}_2 - \bar{T}_1}{S_{pooled}\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}, \tag{6.33}$$

where $\bar{T}_1$ and $\bar{T}_2$ are the two sample means

$$\bar{T}_1 = \frac{T_{11} + T_{12} + \cdots + T_{1n_1}}{n_1}, \tag{6.34}$$

$$\bar{T}_2 = \frac{T_{21} + T_{22} + \cdots + T_{2n_2}}{n_2}, \tag{6.35}$$

S_{pooled} is the pooled sample standard deviation

$$S_{pooled} = \sqrt{\frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}}, \tag{6.36}$$

and S_1 and S_2 are the two sample standard deviations

$$S_1 = \sqrt{\frac{(T_{11} - \bar{S}_1)^2 + (T_{12} - \bar{S}_1)^2 + \cdots + (T_{1n_1} - \bar{S}_1)^2}{n_1 - 1}}, \tag{6.37}$$

$$S_2 = \sqrt{\frac{(T_{21} - \bar{S}_2)^2 + (T_{22} - \bar{S}_2)^2 + \cdots + (T_{2n_1} - \bar{S}_2)^2}{n_2 - 1}}. \tag{6.38}$$

This t-statistic follows a t-distribution of dof equal to $n_1 + n_2 - 2$.

Example 4. Investigate whether the global average annual mean temperature in the decade of 1991-2000 is significantly different from the previous decade.

The two statistical hypotheses are

$$H_0 : \bar{T}_1 = \bar{T}_2 \quad \text{(Null hypothesis: The temperatures of the two decades are the same)} \tag{6.39}$$

and

$$H_1 : \bar{T}_1 \neq \bar{T}_2 \quad \text{(Alternative hypothesis: The two decades are different).} \tag{6.40}$$

This is a two-sided test. The alternative region is a union of both sides $(-\infty, t_{0.025})$ and $(t_{0.975}, \infty)$ if the significance level is set to be 5%. We will compute the t-score using formula (6.33). The result is below:

a. The t-score is 2.5784,

b. The H_0 region is $(-2.1009, 2.1009)$,

c. The p-value is 0.009470,

d. The mean temperature anomalies in 1981-1990 is 0.036862°C, and
e. The mean of the temperature anomalies in 1991-2000 is 0.161255°C.

The t-score is outside the H_0 region. Thus, the H_0 is rejected. The 1991-2000 mean temperature anomaly 0.161255°C is significantly different from the 1981-1990 mean 0.036862°C with a p-value equal to 1%. The temperature difference of the two decades is $0.124392 = 0.161255 - 0.036862$°C which is significantly different from zero.

The above results were obtained by the following R code.

```
 #Hypothesis test for global temp for 1981-1990 and 1991-2000
setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2017/Book-ClimMath-Cambridge-
dat1 <- read.table("aravg.ann.land_ocean.90S.90N.v4.0.0.2015.txt")
tm8190=dat1[102:111,2]
tm9100=dat1[112:121,2]
barT1=mean(tm8190)
barT2=mean(tm9100)
S1sd=sd(tm8190)
S2sd=sd(tm9100)
n1=n2=10
Spool=sqrt(((n1 - 1)*S1sd^2 + (n2 - 1)*S2sd^2)/(n1 + n2 -2))
t = (barT2 - barT1)/(Spool*sqrt(1/n1 + 1/n2))
tlow = qt(0.025, df= n1 + n2 -2)
tup = qt(0.975, df= n1 + n2 -2)
paste("t-score=", round(t,digits=5),
      "tlow=", round(tlow,digits=5),
      "tup=", round(tup,digits=5))
#[1] "t-score= 2.57836 tlow= -2.10092 tup= 2.10092"
pvalue = 1-pt(t,  df= n1 + n2 -2)
paste( "p-value=", pvalue)
#[1] "p-value= 0.00947040009284539"
paste("1981-90 temp=", barT1, "1991-00 temp=",barT2)
#[1] "1981-90 temp= 0.0368621 1991-00 temp= 0.1612545"
barT2 - barT1
#[1] 0.1243924
```

The above is a two-sided test to determine if a sample mean is different from zero. However, the time series of the global temperature in Fig. 6.1 had already indicated that the 1991-2000 decade is warmer than 1981-1990. If we take this as a given prior knowledge, then we should use the one-sided test with the following two hypotheses

$$H_0 : \bar{T}_1 > \bar{T}_2 \quad \text{(Null hypothesis: The temperatures of the two decades are the same)} \tag{6.41}$$

and

$$H_1 : \bar{T}_1 \leq \bar{T}_2 \quad \text{(Alternative hypothesis: The two decades are different).} \tag{6.42}$$

The t-score is the same as the above, but the critical t-score is now $t_{0.95} = 1.734$. Again, the t-score 2.57836 is in the H_1 region.

6.5 Statistical inference of a linear trend

When studying climate change, one often makes a linear regression and ask if a linear trend is significantly positive, negative, and different from zero. For example, is the linear trend of the global average annual mean temperature from 1880-2015 shown in Fig. 6.1 significantly greater than zero? This is again a t-test problem. The estimated trend $\hat{b}$ from a linear regression follows a t-distribution.

With the given data pairs $\{(x_i, y_i), i = 1, 2, \cdots, n\}$ and their regression line discussed in Chapter 3

$$\hat{y} = \hat{b}_0 + \hat{b}_1 x, \tag{6.43}$$

the t-score for the trend $\hat{b}_1$ is defined by the following formula

$$t = \frac{\hat{b}_1}{S_n \sqrt{S_{xx}}}, \qquad \text{dof = n=2.} \tag{6.44}$$

Here,

$$S_n = \sqrt{\frac{SSE}{n-2}} \tag{6.45}$$

with the sum of squared errors SSE defined as

$$SSE = \sum_{i=1}^{n} \left[y_i - (\bar{b}_0 + \bar{b}_1 x_i)\right]^2, \tag{6.46}$$

and

$$S_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2 \tag{6.47}$$

with the sample mean of x-data $\bar{x}$ defined as

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}. \tag{6.48}$$

The dof of this t-score is $n - 2$. With this dof and a specified significance level, one can then find the critical t-values and determine whether the t-score is in the H_0 region or H_1 region.

We wish to use the t-inference procedure to check if the 1880-2015 temperature anomalies trend is significantly greater than zero. The statistical hypotheses are

$$H_0 : \bar{b}_1 < 0 \quad \text{(Null hypothesis: The trend is not greater than zero)} \tag{6.49}$$

and

$$H_1 : \bar{b}_1 \geq 0 \quad \text{(Alternative hypothesis: The trend is greater than zero).} \tag{6.50}$$

This is a one-sided test. The critical t-score is now $t_{0.95} = 1.734$. The summary of the linear regression command gives the needed statistical values:

a. The trend is $\hat{b}_1 = 0.667791°$C per century,
b. The t-score for $\hat{b}_1$ is 20.05,
c. The p-value is 1×10^{-16}, and
d. The critical t value is 1.6563 by an R command `qt(0.95, 134)`.

Clearly, the t-score 20.05 is in the H_1 region. We conclude that the trend is significantly greater than zero with a p-value equal to 1×10^{-16}.

The above results were computed by the following R code.

```
 setwd("/Users/sshen/Desktop/MyDocs/teach/SIOC290-ClimateMath2017/Book-ClimMath-Cambridge
dat1 <- read.table("aravg.ann.land_ocean.90S.90N.v4.0.0.2015.txt")
tm=dat1[,2]
x = 1880:2015
summary(lm(tm ~ x))
#Coefficients:
#  Estimate Std. Error t value Pr(>|t|)
#(Intercept) -1.321e+01  6.489e-01  -20.36   <2e-16 ***
#  x            6.678e-03  3.331e-04   20.05   <2e-16 ***
```

Sometimes one may need to check if the trend is greater than a specified value β_1. Then, the t-score is defined by the following formula

$$t = \frac{\hat{b}_1 - \beta_1}{S_n\sqrt{S_{xx}}}, \qquad \text{dof} = \text{n-2}. \tag{6.51}$$

In this case, the t-score must be computed from the formulas, not from the summary of a linear regression by R.

6.6 Free online statistics tutorials

This statistics chapter has presented a very brief course in statistics, but it provides a sufficient statistics basics and R codes for doing simple statistical analysis of climate data. This chapter also provides the foundation for expanding a reader's statistics knowledge and skills by studying more comprehensive or advanced materials on climate statistics. A few free statistics tutorials available online are introduced below.

The manuscript by David Stephenson of the University of Reading, the United Kingdom, provides the basics of statistics with climate data as examples:

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

This online manuscript is appropriate for readers who have virtually no statistics background.

Eric Gilleland of NCAR authored a slide for using R to do climate statistics, particular the analysis of extreme values:

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

This set of lecture notes provides many R codes for analyzing climate data, such as risk estimation. The material is very useful for climate data users, and does not require much mathematical background.

The "Statistical methods for the analysis of simulated and observed climate data applied in projects and institutions dealing with climate change impact and adaptation" by the Climate Service Center, Hamburg, Germany, is particularly useful for weather and climate data.

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

This online report provides a "user's manual" for a large number of statistical methods used for climate data analysis with real climate data examples. The material is an excellent references fro users of the statistics for climate data.

References

[1] Climate Service Center, Germany, 2013: Statistical methods for the analysis of simulated and observed climate data. Report 13, Version 2.0,

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

[2] Gilleland, E., 2009: Statistical software for weather and climate: The R programming language.

`http://www.maths.lth.se/seamocs/meetings/Malta_Posters_and_Talks/MaltaShortCourseSlide`

[3] Stephenson, D.B., 2005: Data analysis methods in weather and climate research. Lecture notes, 98pp:

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

Exercises

6.1 Assume that the average bank balance of U.S. residents is $\$5,000$. Assume that the bank balances are normally distributed. A group of 25 samples was taken. The sample data have a mean equal to $\$5,000$ and standard deviation of $\$1,000$. Find the confidence interval of this group of samples at 95% confidence level.

6.2 The two most commonly used datasets of global ocean and land average annual mean surface air temperature (SAT) anomalies are those credited to the research groups led by Dr. James E. Hansen of NASA (relative to 1951-1980 climatology period) and Professor Phil Jones, of the University of East Anglia (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 for each period of 15 years, starting at 1880.
(b) Use the t-distribution to find the confidence interval of each 15-year period SAT average at the 95% confidence level using the t-distribution. You can use either Hansen's data or Jones' data. Figure SPM.1(a) of IPCC 2013 (AR4) is a helpful reference.
(c) Find the hottest and the coldest 15-year periods from 1880-2014, which is divided into nine disjoint 15-year periods. Use the t-distribution to check whether the temperature difference in the hottest 15-year period minus that in the coldest 15-year period is significantly greater than zero. Do this problem for either Hansen's data or Jones data.
(d) Discuss the differences between the Hansen and Jones datasets.

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

$$t^* = \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}, \qquad (6.52)$$

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., df) 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 a statistical sense with a confidence interval). The alternative hypothesis is that the difference is significantly different from zero. Now you can choose to use a one-sided test when the difference is positive. Use a significance level of 5% or 1%, or another level of at your own choosing.
(a) Choose two 15-year periods which have very different average anomalies. Use the higher one minus the lower one. Use the t-test method for a one-sided

test to check if the difference is significantly greater than zero. Do this for the global average annual mean temperature data from either Hansen's dataset or Jones dataset.

(b) Choose two 15-year periods which have very similar average anomalies. Use the higher one minus the lower one. Use the t-test method for a two-sided test to check if the difference is not significantly different from zero. Do this for the global average annual mean temperature data from either Hansen's dataset or Jones dataset.

Author index

Subject index