Exercises for MedCof7 RTC Practical Training 2016 - Group A

Edmondo Di Giuseppe

November 16, 2016

Prelude 1

Create a directory anywhere named **R__practical__training**, then open RStudio and create a new project pointing at it. After that, create a new R file script named `R_course_Wed16Nov.R` and a new directory `/Data` in it (for those who don not use RStudio, a similar procedure can be followed except for the project creation phase). Finally, log in the Moodle platform and download the file `SeasFor_GCM_NCEP_datafile.xlsx` from **/Day two/R__lesson** directory of **Verification of Operational Seasonal Forecasts in the Mediterranean region** face-to-face course into `/R_practical_training/Data` directory. From now on, use `R_course_Wed16Nov.R` for copying and executing the chunks of this lesson one after the other.

Prelude 2

In order to be ready for the practical session **"Hands-on session using R Forecast Verification Package: a) unconditional biases and hits; b) scoring probabilistic forecasts, c) reliability and resolution; d) ROC diagrams"**, leaded by Dr Caio Augusto dos Santos Coelho please install and test the package verification

```
install.packages("verification")
library(verification)
```

Overview

R is an **interpreted language**. This means that code entered into the R console (or run as R script in batch mode) is executed by the interpreter, a program within the R system.

R code is composed of a series of **expressions** such as:

- **arithmetic expressions**
- **assignment statements**
- **conditional statements**

Arithmetic expressions

Chunk 1 (objects class and type of)

```
x = 12.5
x
```

```
## [1] 12.5
```

```r
class(x)
```

```
## [1] "numeric"
```

```r
typeof(x)
```

```
## [1] "double"
```

Chunk 2 (integer)

Integer vectors exist so that data can be passed to **C** or **Fortran** code which expects them:

```r
k = 1
k
```

```
## [1] 1
```

```r
is.integer(k)  #is.integer(x) does not test if x contains integer numbers!
```

```
## [1] FALSE
```

```r
is.double(k)
```

```
## [1] TRUE
```

```r
y = as.integer(k)
y
```

```
## [1] 1
```

```r
as.integer(3.14)
```

```
## [1] 3
```

```r
as.integer("5.27")
```

```
## [1] 5
```

```r
as.integer("Donald")
```

```
## Warning: si è prodotto un NA per coercizione
```

```
## [1] NA
```

Chunk 3 (complex numbers)

```r
z = 1 + 2i
z
```

```
## [1] 1+2i
```

```r
sqrt(-1)
```

```
## Warning in sqrt(-1): Si è prodotto un NaN
```

```
## [1] NaN
```

```r
sqrt(-1+0i)
```

```
## [1] 0+1i
```

```
sqrt(as.complex(-1))
```

```
## [1] 0+1i
```

Chunk 4 (operators)

```
x = 1; y = 2
z = x > y
z
```

```
## [1] FALSE
```

```
u = TRUE; v = FALSE
u & v
```

```
## [1] FALSE
```

```
u | v
```

```
## [1] TRUE
```

```
!u
```

```
## [1] FALSE
```

Chunk 5 (character)

```
s = "President Donald Trump"
nchar(s)
```

```
## [1] 22
```

```
x = as.character(3.14)
x
```

```
## [1] "3.14"
```

```
fname = "Mohamed"; lname ="Salah"
paste(fname, lname)
```

```
## [1] "Mohamed Salah"
```

```
sprintf("%s has %d dimar", "Samir", 100)
```

```
## [1] "Samir has 100 dimar"
```

```
substr("Amidou is from Mauritania.", start=11, stop=25)
```

```
## [1] "from Mauritania"
```

```
sub("little", "big", "Kleanthis has a little house in Cyprus.")
```

```
## [1] "Kleanthis has a big house in Cyprus."
```

Chunk 6 (factor)

```
a = factor("A")
class(a)
```

```
## [1] "factor"
```

```
x = factor(1)
y = factor(2)
x + y
```

```
## Warning in Ops.factor(x, y): '+' not meaningful for factors
```

```
## [1] NA
```

```
z <- c(x,y)
class(z)
```

```
## [1] "integer"
```

Assignment

Chunk 1 (set object names)

You can set names to objects taking into accounts what follows:

1. R is **case sensitive**: `Alpha` and `alpha` are two different objects.
2. object names cannot contain symbols like `! + - #`;
3. `.` and `_` are allowed, also a name starting with a dot;
4. object names can contain a number but cannot start with a number;

Chunk 2 (list objects)

List either the whole objects or a group of them stored in the Environment:

```
ls()
```

```
##  [1] "a"     "fname" "k"     "lname" "s"     "u"     "v"     "x"    
##  [9] "y"     "z"
```

```
ls(pattern = "n")      # all objects starting with letter "n"
```

```
## [1] "fname" "lname"
```

Chunk 3 (populate objects)

You can use `<-` or `=` to populate objects. The most common method to build a vector is the `c()` function.

```
X.num = c(0, 2, 5, 6.2, -4, 4)
X.num
```

```
## [1]  0.0  2.0  5.0  6.2 -4.0  4.0
```

```
str(X.num)        # print the class of the object and first elements
```

```
##  num [1:6] 0 2 5 6.2 -4 4
```

```
X.logic = X.num >= 5
X.logic
```

```
## [1] FALSE FALSE  TRUE  TRUE FALSE FALSE
```

```r
str(X.logic)
```

```
##  logi [1:6] FALSE FALSE TRUE TRUE FALSE FALSE
```

```r
X.str = c("Ibrahim","Branko","Awatif","Mirjana")
str(X.str)
```

```
##  chr [1:4] "Ibrahim" "Branko" "Awatif" "Mirjana"
```

```r
X.mixed <- c(X.num,X.str,TRUE)
str(X.mixed)
```

```
##  chr [1:11] "0" "2" "5" "6.2" "-4" "4" "Ibrahim" ...
```

Chunk 4 (sequences)

You can also create a vector using sequences.

```r
X.ahead = 1:10
X.ahead
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10
```

```r
X.seq = seq(1,10,by=0.5)
X.seq
```

```
##  [1]  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5
## [15]  8.0  8.5  9.0  9.5 10.0
```

```r
X.seq2 = seq(1,10,length.out = 20)
X.seq2
```

```
##  [1]  1.000000  1.473684  1.947368  2.421053  2.894737  3.368421  3.842105
##  [8]  4.315789  4.789474  5.263158  5.736842  6.210526  6.684211  7.157895
## [15]  7.631579  8.105263  8.578947  9.052632  9.526316 10.000000
```

```r
(X.rep = rep(NA,length = 10))     # round brackets for printing the object directly
```

```
##  [1] NA NA NA NA NA NA NA NA NA NA
```

```r
(X.rep2 = rep(c("RTC2016","WMO"),each = 10))
```

```
##  [1] "RTC2016" "RTC2016" "RTC2016" "RTC2016" "RTC2016" "RTC2016" "RTC2016"
##  [8] "RTC2016" "RTC2016" "RTC2016" "WMO"     "WMO"     "WMO"     "WMO"
## [15] "WMO"     "WMO"     "WMO"     "WMO"     "WMO"     "WMO"
```

```r
(X.rep3 = rep(paste("RTC2016","WMO"),each = 10))  # paste() instead of c()
```

```
##  [1] "RTC2016 WMO" "RTC2016 WMO" "RTC2016 WMO" "RTC2016 WMO" "RTC2016 WMO"
##  [6] "RTC2016 WMO" "RTC2016 WMO" "RTC2016 WMO" "RTC2016 WMO" "RTC2016 WMO"
```

```r
(X.rep4 = rep(c("RTC2016","WMO", "MEDCOF7"),times = 10))
```

```
##  [1] "RTC2016" "WMO"     "MEDCOF7" "RTC2016" "WMO"     "MEDCOF7" "RTC2016"
##  [8] "WMO"     "MEDCOF7" "RTC2016" "WMO"     "MEDCOF7" "RTC2016" "WMO"
## [15] "MEDCOF7" "RTC2016" "WMO"     "MEDCOF7" "RTC2016" "WMO"     "MEDCOF7"
## [22] "RTC2016" "WMO"     "MEDCOF7" "RTC2016" "WMO"     "MEDCOF7" "RTC2016"
## [29] "WMO"     "MEDCOF7"
```

Chunk 5 (subset objects)

```
X.seq[-1]
```

```
##  [1]  1.5  2.0  2.5  3.0  3.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0
## [15]  8.5  9.0  9.5 10.0
```

```
X.seq[-c(1,5,6)]
```

```
##  [1]  1.5  2.0  2.5  4.0  4.5  5.0  5.5  6.0  6.5  7.0  7.5  8.0  8.5  9.0
## [15]  9.5 10.0
```

```
(X.odd <-rep(c(TRUE,FALSE),times=5))
```

```
##  [1]  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE
```

```
X=1:10
X[X.odd]
```

```
## [1] 1 3 5 7 9
```

```
X[X.odd==FALSE]     # even elements of X
```

```
## [1]  2  4  6  8 10
```

```
Y=subset(X,X != 5)   # subset() is alternative to [,]
```

Conditional statements

Chunk 1 (if statement)

```
z = 1 + 2i
z
```

```
## [1] 1+2i
```

```
if (is.complex(z)==TRUE){print("z is complex")}
```

```
## [1] "z is complex"
```

Chunk 2 (if-else statement)

```
if (is.integer(z) == TRUE){
    print("z is complex")
}else{
    paste("z is",typeof(z))   # paste() combine objects and strings
  }
```

```
## [1] "z is complex"
```

Chunk 3 (if-else nested statement)

Exercise 1

Install and load `{latticeExtra}` package that contains a variety of datasets. For a complete list, use > library(help = "latticeExtra")

and print the `EastAuClimate` one, which contains Climate of the East Coast of Australia.

If `EastAuClimate` is a data.frame object, then extract `SummerMaxTemp` from it. If there are more than 10 data, then make a scatter plot.

Solution:

```
library(latticeExtra)
```

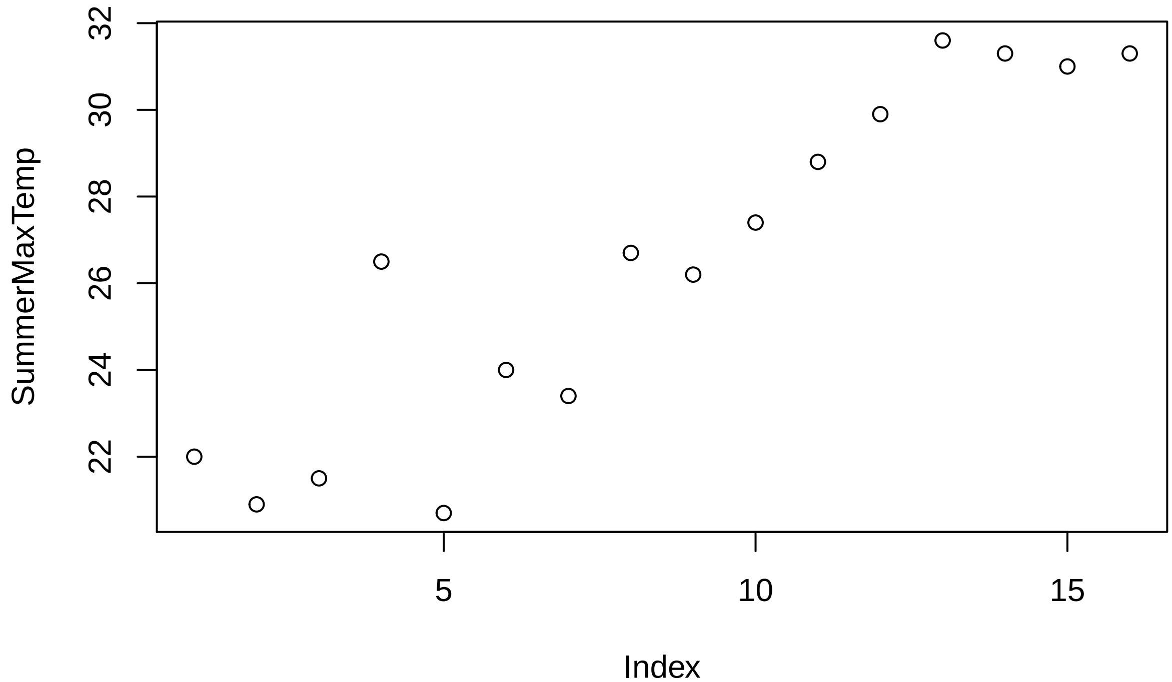
```
## Loading required package: lattice

## Loading required package: RColorBrewer
```

```
library(help = "latticeExtra")
str(EastAuClimate)
```

```
## 'data.frame':    16 obs. of  15 variables:
##  $ SummerMaxTemp: num  22 20.9 21.5 26.5 20.7 24 23.4 26.7 26.2 27.4 ...
##  $ SummerMinTemp: num  12.7 14.4 14.3 15.8 15.3 14.8 16.4 19.3 19.2 20.6 ...
##  $ WinterMaxTemp: num  12.2 13 12.9 13.9 12.3 14.7 16.1 17 18.2 19.4 ...
##  $ WinterMinTemp: num  4.7 7 7.7 6.8 8.8 5.8 6.6 7.4 7.9 11.7 ...
##  $ SummerRain   : num  28.1 30.8 32 32.3 33.8 ...
##  $ WinterRain   : num  44.1 78.4 107.3 46.8 125.2 ...
##  $ MeanAnnRain  : num  576 757 952 654 1109 ...
##  $ RainDays     : num  90.8 103.2 141 99.2 137.1 ...
##  $ ClearDays    : num  41.1 46 25.3 48.9 22.3 ...
##  $ CloudyDays   : num  177 140 221 178 218 ...
##  $ ID           : int  94029 92045 90015 86071 85096 84083 69022 66037 60026 58009 ...
##  $ Latitude     : num  -42.9 -41 -38.9 -37.8 -39.1 ...
##  $ Longitude    : num  147 148 144 145 146 ...
##  $ Elevation    : int  51 82 82 31 95 43 25 6 20 95 ...
##  $ State        : Factor w/ 4 levels "NSW","QLD","TAS",..: 3 3 4 4 4 4 1 1 1 1 ...
```

```
if(class(EastAuClimate)=="data.frame"){
  SummerMaxTemp<-EastAuClimate$SummerMaxTemp
  if(length(SummerMaxTemp)>10){plot(SummerMaxTemp)}
}
```



Data Import from Excel

We import NCEP seasonal forecast data at cell grid X(degree_east) 16.875W, Y(degree_north) 43.2542N from the file `Data/SeasFor_GCM_NCEP_datafile.xlsx`.

Clipboard

The **easiest way** to import data from Excel is as follows:

1. open Excel sheet, select the entire data table (or part of it) and copy on **clipboard** (CTRL+C);
2. open R console and type

Notice that you need to specify some arguments of the `read.table()` function accordingly with the characteristics of the file to be imported:

- `sep` is the field separator character;
- `dec` the character used in the file for decimal points;
- `header` a logical value indicating whether the file contains the names of the variables as its first line;
- ...

Read data directly from file

Obviously, the "clipboard" method is unefficient when you need to read multiple files. However, there are several ways to read Excel files. Here, we use the function `read.xls()` in `{gdata}` package

Exercise 2

1. Install `{gdata}`, `{Hmisc}`, `{maps}` and `{ggplot2}` packages

2. Import and print data from file `Data/SeasFor_GCM_NCEP_datafile.xlsx`

```
library(gdata)
```

```
## gdata: read.xls support for 'XLS' (Excel 97-2004) files ENABLED.

##

## gdata: read.xls support for 'XLSX' (Excel 2007+) files ENABLED.

##
## Attaching package: 'gdata'

## The following object is masked from 'package:stats':
##
##     nobs

## The following object is masked from 'package:utils':
##
##     object.size

## The following object is masked from 'package:base':
##
##     startsWith
```

```
Ncep <- read.xls("Data/SeasFor_GCM_Ncep_datafile.xlsx")

str(Ncep)
```

```
## 'data.frame':    497 obs. of  3 variables:
##  $ L                 : num  0.5 1.5 2.5 3.5 4.5 5.5 6.5 0.5 1.5 2.5 ...
##  $ Forecast.Started  : Factor w/ 71 levels "1998-07-01T00:00",..: 1 1 1 1 1 1 1 2 2 2 ...
##  $ precipitation.rate: num  0.82 0.511 1.343 2.074 3.142 ...
```

```
summary(Ncep)
```

```
##        L                 Forecast.Started precipitation.rate
##  Min.   :0.5   1998-07-01T00:00:  7     Min.   :0.511
##  1st Qu.:1.5   1998-08-01T00:00:  7     1st Qu.:1.238
##  Median :3.5   1998-09-01T00:00:  7     Median :2.074
##  Mean   :3.5   1998-10-01T00:00:  7     Mean   :2.077
##  3rd Qu.:5.5   1998-11-01T00:00:  7     3rd Qu.:2.826
##  Max.   :6.5   1998-12-01T00:00:  7     Max.   :4.487
##                (Other)         :455     NA's   :50
```

3. Transform data of `$Forecast.Started` column in date format and extract the number of days from each forecasted month

```
Ncep$Forecast.Started<-as.Date(Ncep$Forecast.Started,format="%Y-%m-%dT%H:%M")
head(Ncep)
```

```
##     L Forecast.Started precipitation.rate
## 1 0.5       1998-07-01          0.8201456
## 2 1.5       1998-07-01          0.5109688
## 3 2.5       1998-07-01          1.3429490
## 4 3.5       1998-07-01          2.0742040
## 5 4.5       1998-07-01          3.1423840
## 6 5.5       1998-07-01          3.3625960
```

```
library(Hmisc)   # for monthDays()
```

```
## Loading required package: survival
## Loading required package: Formula
## Loading required package: ggplot2
##
## Attaching package: 'ggplot2'
## The following object is masked from 'package:latticeExtra':
##
##     layer
##
## Attaching package: 'Hmisc'
## The following object is masked from 'package:gdata':
##
##     combine
## The following objects are masked from 'package:base':
##
##     format.pval, round.POSIXt, trunc.POSIXt, units
```

```
Forecast.Days0<-monthDays(Ncep$Forecast.Started)
```

4. Calculate the cumulated precipitation starting from data in `$precipitation.rate` column (precipitation rate unity measure is *mm/day*)

```
Ncep$Forecast.Days<-Forecast.Days0*Ncep$L
Ncep$Forecast.Date<-Ncep$Forecast.Started+ Ncep$Forecast.Days
Ncep$ConvertDays<-monthDays(Ncep$Forecast.Date)

Ncep$Mon.Cum <- round(Ncep$precipitation.rate * Ncep$ConvertDays,1)
range(Ncep$Mon.Cum,na.rm=T)     # a quick check
```

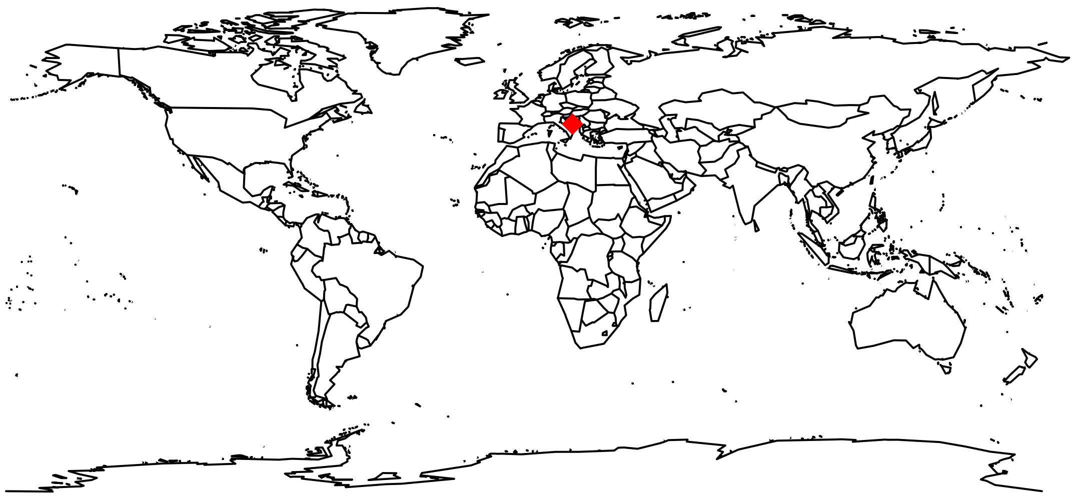
```
## [1]  15.8 139.1
```

5. Locate the cell over the world map

```
require(maps)
```

```
## Loading required package: maps
```

```
map("world")
points(16.875, 43.2542, col="red", pch=18,cex=1.5)
```



6. Group and plot by outlooks (to do step by step with teacher)

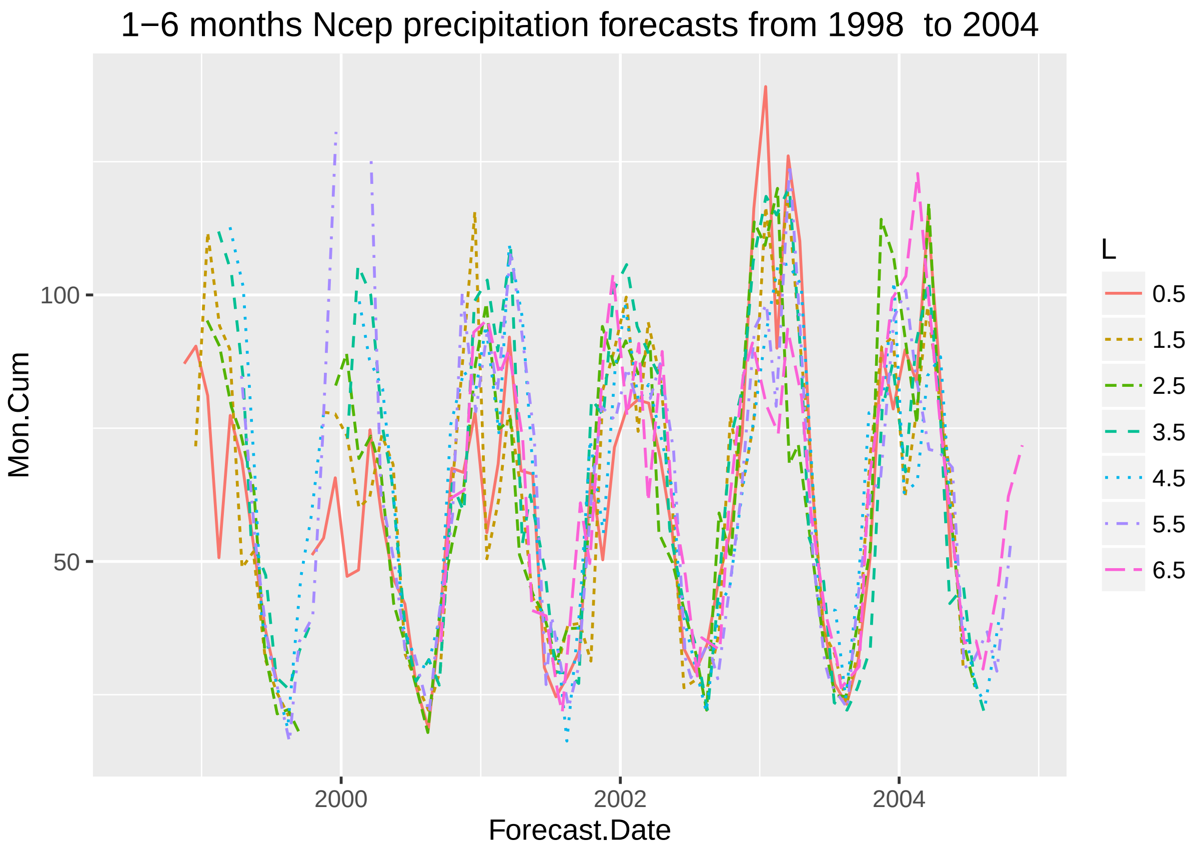
```
library(ggplot2)

# get years  from date and select the first and the last:
Ybegin<-as.numeric(getYear(Ncep$Forecast.Date)[1])
Yend<-as.numeric(getYear(Ncep$Forecast.Date)[nrow(Ncep)] )

#transform Ncep data frame:
Ncep2<-Ncep
Ncep2$L<-as.factor(Ncep$L)    # L as factor

#Plot
p<-ggplot(Ncep2, aes(x=Forecast.Date, y=Mon.Cum,  colour=L,linetype=L)) + geom_line()
p+ ggtitle(paste("1-6 months Ncep precipitation forecasts from",Ybegin," to", Yend))
```

```
## Warning: Removed 20 rows containing missing values (geom_path).
```




```
ggsave(paste(getwd(),"/Plots/Ncep_forecasts.pdf",sep=""),
       width = 19, height = 14, units = "cm")
```

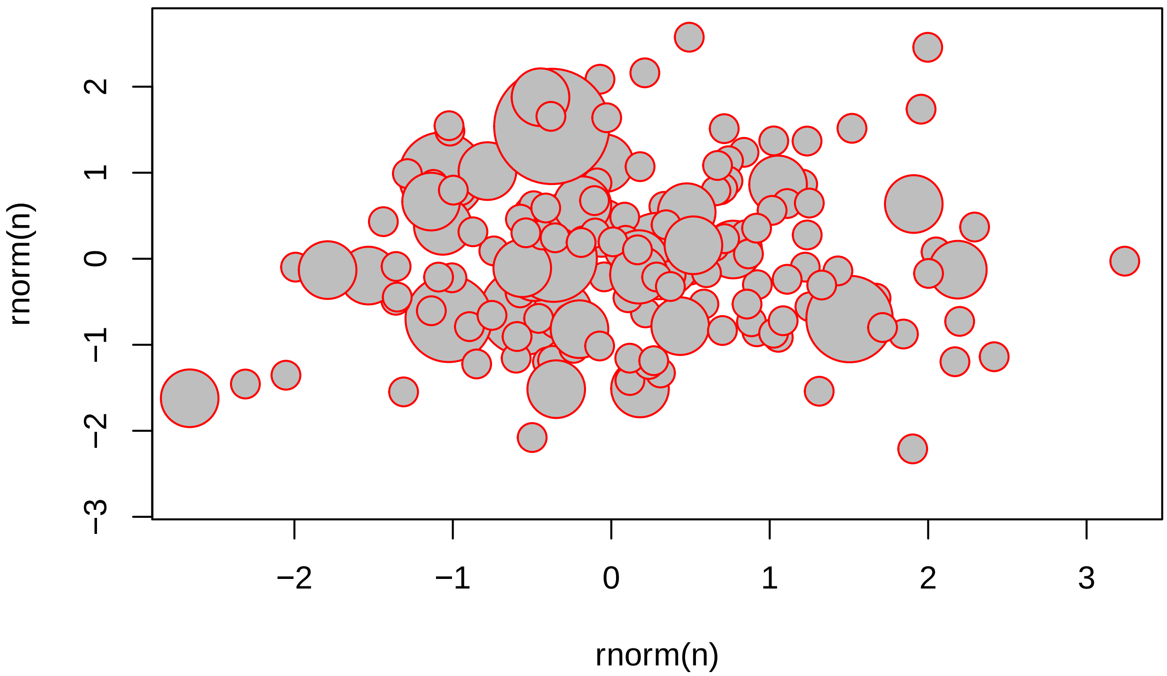
```
## Warning: Removed 20 rows containing missing values (geom_path).
```

Visualizing

Exercise 3

Make a bubble plot (i.e. a scatter plot that represents its points as circles) with 500 data extracted from `rnorm()` function, where the bubble size is determined by a poisson process with $\lambda = 0.4$.

```
n = 500
set.seed(123)      # fix the seed for number random generation
#par(mar = c(4, 4, 0.1, 0.1))
plot(rnorm(n), rnorm(n), pch = 21, cex = 2 * rpois(n,lambda = 0.4), col = "red", bg = "gray")
```



Recommended book

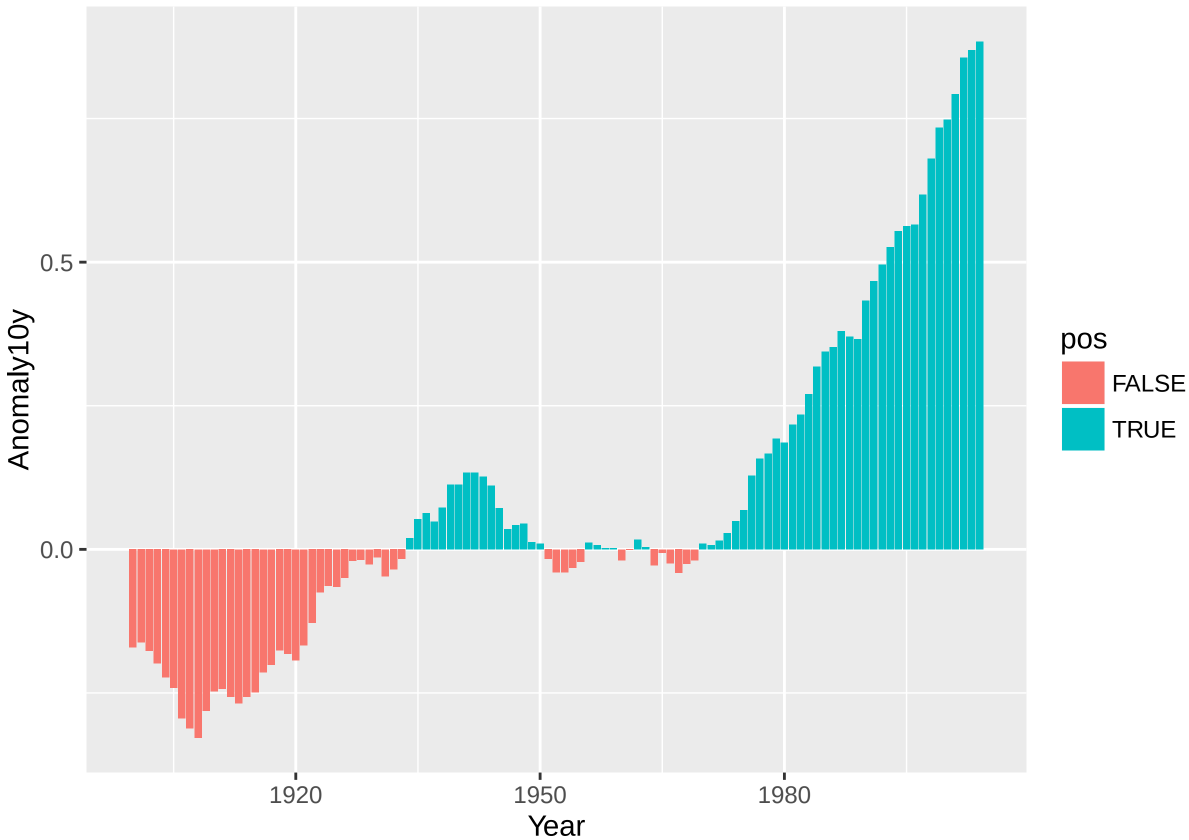
1. install package {gcookbook} and load it, then have a look at climate object, which contains **Global climate temperature anomaly data from 1800 to 2011**;
2. follow the example below to plot data (this example is taken from [Cookbook for R] (http://www.cookbook-r.com)

```
library(gcookbook)
library(help="gcookbook")
str(climate)
```

```
## 'data.frame':    499 obs. of  6 variables:
##  $ Source    : chr  "Berkeley" "Berkeley" "Berkeley" "Berkeley" ...
##  $ Year      : num  1800 1801 1802 1803 1804 ...
##  $ Anomaly1y : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Anomaly5y : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ Anomaly10y: num  -0.435 -0.453 -0.46 -0.493 -0.536 -0.541 -0.59 -0.695 -0.763 -0.818 ...
##  $ Unc10y    : num  0.505 0.493 0.486 0.489 0.483 0.475 0.468 0.461 0.453 0.451 ...
```

```
csub <- subset(climate, Source=="Berkeley" & Year >= 1900)
csub$pos <- csub$Anomaly10y >= 0

ggplot(csub, aes(x=Year, y=Anomaly10y, fill=pos)) +
      geom_bar(stat="identity", position="identity")
```


0.5
0.0
Anomaly10y
1920
1950
1980
Year
pos
FALSE
TRUE