CHAPTER 17

ANALYSIS OF VARIANCE

Table 17.1 default

`Rcode.exercise.Chapter18.R`	main program
`[year]11_tref_NCEP-CFSv2.nc`	NetCDF data files
`land_cover.nc`	NetCDF file for the land-sea mask
`interesting.points.R`	auxiliary file: R function for identifying large values in a field
`index.climate.v2.R`	auxiliary file: R function for defining N. American land area
`eof.latlon.R`	auxiliary file: calculate EOFs
`plot_latlon_v4.R`	auxiliary file: plot spatial maps
`gev.R`	auxiliary file for solving generalized eigenvalue problem

In this homework set you will use Analysis of Variance (ANOVA) to quantify the predictability of seasonal forecasts from the CFSv2 Retrospective Hindcasts. More precisely, you will determine whether April-mean temperature over North America is predictable by CFSv2 based on November initial conditions.

The above table shows the programs and data sets you need to download to do this homework. Download these files and run `Rcode.exercise.Chapter18.R`. As usual, modify `dir.data` and `dir.Rlib` to correspond to your data and library directories, respectively. `Rcode.exercise.Chapter18.R` should run to completion without any error or warning messages (please contact me if this is not true). After completion, the following variables are available to you:

```
## VARBL: NAME OF THE VARIABLE (E.G., TEMPERATURE)
## ICMON: INITIAL CONDITION MONTH (E.G., 'NOV')
## VFMON: VERIFICATION MONTH (E.G., 'JAN')
## EPIC: SELECTED ENSEMBLE MEMBERS (E.G., 1,2,...,24)
## NENS: TOTAL NUMBER OF ENSEMBLE MEMBERS (LENGTH(EPIC))
## NYRS: NUMBER OF YEARS
## IYST: FIRST YEAR OF THE DATA SET
## LON[NLON]: LONGITUDE VALUES OF THE DOMAIN
## LAT[NLAT]: LATITUDE VALUES OF THE DOMAIN
## HCST.DATA[NLON,NLAT,NENS,NYRS]: HINDCAST DATA
## VIEW[NLON,NLAT]: N. AMERICAN MASK
```

Note that the data array has the format [space, ensemble, year], which is not the same as in the notes (the notes assume [ensemble, year, space]). There is a good reason why the notes use a different format than the usual format for data, but that is a long story.

In dealing with multivariate data in R, your programs will run much faster if instead of using `for` loops, you make use of the commands `rowMeans`, `rowSums`, `colMeans`, and `colSums`. In order to make use of these commands, you will need to "reshape" your arrays using the `dim` command. Any array can be reshaped into another array with different dimensions, as long as the product of the dimensions are the same. The rule is that R fills up columns first, then rows. For instance, consider the following set of commands:

```
> x = rnorm(20)
> print(x)
 [1] -0.08253025 -0.66934158 -2.55017522  2.19427432 -0.56563243  1.17370178  1.43692367
 [8] -0.63444609 -1.17081896 -2.23559009 -1.83286874  0.76325786 -0.20322325 -0.59558207
[15] -2.17997749 -0.99505771  0.23259272  1.14159830 -1.63726907 -0.70107359
> dim(x) = c(4,5)
> print(x)
            [,1]       [,2]       [,3]       [,4]       [,5]
[1,] -0.08253025 -0.5656324 -1.1708190 -0.2032233  0.2325927
[2,] -0.66934158  1.1737018 -2.2355901 -0.5955821  1.1415983
[3,] -2.55017522  1.4369237 -1.8328687 -2.1799775 -1.6372691
[4,]  2.19427432 -0.6344461  0.7632579 -0.9950577 -0.7010736
```

In the above example, a random vector of length 20 is printed. Then, the vector is reshaped into a 4×5 array and printed. You can see that R takes the sequence of numbers in the vector and fills the 4×5 array, filling *columns first*, then the rows.

To illustrate the above technique, we will calculate a few quantities that are needed in ANOVA. One quantity that is needed is the grand mean at each grid point. This can be calculated as follows:

```
## COMPUTE GRAND MEAN
dim(hcst.data) = c(nlon*nlat,nens*nyrs)
gmean          = rowMeans(hcst.data)
```

The resulting variable `gmean` is a vector of length `nlon*nlat` giving the mean at each grid point. Another quantity that is needed is the ensemble mean, which can be calculated as follows:

```
## COMPUTE ENSEMBLE MEAN
dim(hcst.data) = c(nlon*nlat,nens,nyrs)
emean          = array(NA,dim=c(nlon*nlat,nyrs))
for ( ny in 1:nyrs) emean[,ny] = rowMeans(hcst.data[,,ny])
```

The first line reshapes the data array. The second line creates an array of the appropriate dimensions, but fills the array with `NA`. It is good practice to always initialize arrays with `NA`, so that if you accidentally do not fill up the entire array you will get an `NA` whenever any algebraic calculation is performed with those elements. The resulting `NA` is then useful for debugging. The last line calculates the ensemble mean for each year (i.e., the "condition" is "year"). The unbiased estimate of the variance of conditional means is therefore

```
## COMPUTE SIGNAL VARIANCE
var.sig = rowSums((emean - gmean)^2) / (nyrs-1)
```

Note that `emean` is a 2-dimensional array while `gmean` is a vector. R will automatically repeat `gmean` until it fills up an array of the same size as `emean`, so that the subtraction can be performed. After the subtraction, each term in the resulting array is squared, then the sum of the squares is computed. Dividing by the degrees of freedom yields the unbiased estimate of the signal variance.

A key quantity in ANOVA is the critical value of F for testing significance. This is very simple to do in R using the command `qf`. For the above parameters, the $\alpha 100\%$ significance threshold is

```
f.crit = qf(alpha,nyrs-1,nyrs*(nens-1),lower.tail=FALSE)
```

Assignment

Exercise 17.1. Write a function called `f.anova.array` that performs ANOVA for each grid point in an array of data. The output should give the F-value at each grid point and the corresponding significance level. The header of this function should be the following:

```
f.anova.array = function(x,nspace,nens,ncon,alpha=0.05) {
### COMPUTES THE F-STATISTIC IN ANOVA FOR EACH ELEMENT IN NSPACE
# INPUT:
#    X: [NSPACE,NENS,NCON] ARRAY OF DATA
#    NSPACE: NUMBER OF SPATIAL ELEMENTS FOR INDIVIDUALLY COMPUTING F
#    NENS: NUMBER OF ENSEMBLE MEMBERS
#    NCON: NUMBER OF CONDITIONS
#    ALPHA: SIGNIFICANCE LEVEL
# OUTPUT: LIST WITH THE FOLLOWING VARIABLES
#    F: [NSPACE] VECTOR CONTAINING THE F-VALUES
#    F.CRIT: THE CRITICAL F-VALUE FOR F AT THE ALPHA*100% SIGNIFICANCE LEVEL
```

Turn in a copy of your program.

□

Exercise 17.2. Use your function to calculate the F-statistic for CFSv2 hindcasts of January 2m temperature. Mask out insignificant values, like so:

```
f.list = f.anova.array(hcst.data,nlon*nlat,nens,nyrs)

### MASK OUT INSIGNIFICANT F'S
fval = f.list$f
fval[fval < f.list$f.crit] = NA
```

Make a plot of the F values and turn it in.

Also, print out the result of `summary(fval)`, which should be 2 lines giving the minimum, 1st quantile, median, etc.

□

Exercise 17.3. Use `interesting.points` to identify grid points with large Fvalues. For instance, this can be done as follows:

```
ilist  = interesting.points(fval,max,3,nlon,nlat,lon,lat,20,20)
```

The output of `ilist` should be self-explanatory (but see me if it doesn't make sense to you). Show a box-whisker plot of the data using the command

```
npic = 1 # (or 2 or 3)
y      = hcst.data[ilist$x.pic[npic],ilist$y.pic[npic],,]
boxplot(y,names=year,col='grey')
```

In the title of each figure, state the corresponding value of F and the longitude and latitude of the point. Explain how the resulting figure is consistent with the value of F. □

Exercise 17.4. Explore what happens to the F values as you reduce the number of ensemble members or change the lead time. These parameters can be changed by changing `epic` and `lead`, respectively. For instance, the default settings are

```
lead     = 6
epic     = 9:24
```

`lead = 6` tells the code to pick the 6th lead time. Since the data is monthly, starting on November, this corresponds to predicting April (November is the "first lead"). `epic = 9:24` tells the code to pick ensemble members 9-24. The ensemble members are stored in reverse order, as indicated in the following table from Saha et al. (2014).

```
 8 Oct at 0000, 0600, 1200, and 1800 UTC
13 Oct at 0000, 0600, 1200, and 1800 UTC
18 Oct at 0000, 0600, 1200, and 1800 UTC
23 Oct at 0000, 0600, 1200, and 1800 UTC
28 Oct at 0000, 0600, 1200, and 1800 UTC
 2 Nov at 0000, 0600, 1200, and 1800 UTC
 7 Nov at 0000, 0600, 1200, and 1800 UTC
```

So, to choose an 8-member ensemble, you would set `epic = 17:24`. Try a few choices, plot the results, and explain why the results make sense. For instance, what do you think happens to predictability when you decrease the ensemble size? Or decrease the lead time? □