EXERCISES FOR STATISTICAL METHODS FOR CLIMATE SCIENTISTS

Timothy DelSole and Michael K. Tippett

CONTENTS

Foreward

This document contains exercises and solutions to accompany *Statistical Methods for Climate Scientists* by Timothy DelSole and Michael K. Tippett. The usefulness of these solutions would be greatly diminished if they were posted on an insecure website. Therefore, we hope that instructors are mindful to not post these solutions on an insecure website. We will greatly appreciate users of this document notifying us about errors or ambiguities in these exercises and solutions.

CHAPTER 1

EXERCISES: BASIC CONCEPTS IN PROBABILITY AND STATISTICS

Table 1.1 Sample Outcome of 100 pairs of coin tosses

	H_2	T_2
H_1	30	21
T_1	22	27

Exercise 1.1 (Empirical Probabilities)**.** The *relative frequency* of an event is the fraction of times that the event occurs. Suppose a coin is tossed 200 times. The outcome of 100 consecutive pairs of tosses is summarized in the contingency table 1.1, where H_1 and T_1 indicate the number of heads and tails on the first toss, respectively, and H_2 and T_2 indicate the number of heads and tails on the second toss. For instance, the "21" listed in the top right entry of the table means that 21 out of 100 pairs was "heads-tails." The empirical probability of "heads-tails" is therefore 21/100.

a) What is the empirical probability of heads on the first toss of a pair, denoted $P(H_1)$?

b) What is the empirical probability of a heads in the second toss of a pair, denoted $P(H_2)$?

c) What is the empirical probability of the joint event H_1 and T_2, denoted $P(H_1, T_2)$?

d) What is the empirical probability of T_2 *given* H_1, denoted $P(T_2|H_1)$? Compute this by examining only those cases in which H_1 is true.

e) Verify that the empirical conditional probability computed in (d) is the ratio of the joint probability over the marginal: $P(T_2|H_1) = P(T_2, H_1)/P(H_1)$.

(Incidentally, this exercise suggests why unconditional probabilities are called marginal probabilities, namely because they can be calculated by summing values in a table along rows and columns and writing the results along the margins of the table.) □

Exercise 1.2. If X and Y are independent, then show that $p(x, y) = p(x)p(y)$ implies $E[XY] = E[X]E[Y]$. □

Exercise 1.3. Let X and Y be independent random variables with respective means μ_X and μ_Y, and respective variances σ_X^2 and σ_Y^2. In terms of the population means and variances:

(a) Compute $E[XY]$?

(b) Compute $\mathrm{cov}[X, Y]$?

(c) Compute $\mathrm{var}[X - Y]$?

(d) Compute $\mathrm{var}[XY]$?

□

Exercise 1.4 (Properties of Variance)**.** Let k be a constant and X be a random variable with expectation μ_X and variance σ_X^2. Using the definition of variance and the properties of expectation, prove the following:

(a) $\mathrm{var}[k] = 0$

(b) $\mathrm{var}[kX] = k^2 \,\mathrm{var}[X]$?

(c) $\mathrm{var}[k + X] = \mathrm{var}[X]$?

□

Exercise 1.5. Using the properties of expectation, show that

$$\mathrm{cov}[X, Y] = E[XY] - E[X]E[Y]. \tag{1.1}$$

□

Exercise 1.6. If X is a random variable and k is a constant (i.e., independent of X), then show that $E[(X - k)^2]$ is minimized when $k = E[X]$. (Hint: there are at least two ways to prove this: one way is to use calculus, the other way is to add and subtract an expectation. Can you find both proofs?) Be sure to prove that the solution is actually a *minimum*. □

Exercise 1.7 (Bounds on the Correlation Coefficient)**.** Prove that the correlation coefficient between any two random variables X and Y is always between -1 and 1. Explain when the correlation is exactly equal to one, and when the correlation is exactly equal to -1, in terms of the relation between X and Y. Hint: use the fact that

$$E[(t(X - E[X]) + (Y - E[Y]))^2] = t^2 \,\mathrm{var}[X] + 2t \,\mathrm{cov}[X, Y] + \mathrm{var}[Y] \geq 0, \tag{1.2}$$

for any t. Note that this equation for t is a parabola. What is the minimum value? □

Exercise 1.8 (Distribution of a Difference in Means)**.** Suppose $X_1, X_2, \ldots, X_N$ are independent random random variables drawn from a Gaussian distribution with mean μ_X and variance σ^2. Similarly, let $Y_1, Y_2, \ldots, Y_N$ be identically distributed random variables drawn from a Gaussian distribution with different mean μ_Y but common variance σ^2. Also, assume that the X's and Y's are independent.

- What are the distributions of the sample means $\hat{\mu}_X$ and $\hat{\mu}_Y$? (Do not just say "normal." Tell me all the parameters– e.g., the mean and variance.)
- What is the distribution of the difference in sample means $\hat{\mu}_X - \hat{\mu}_Y$?
- What is the distribution of the sample variances $\hat{\sigma}_X^2$ and $\hat{\sigma}_Y^2$? (Tell me all parameters.)
- What is the distribution of the sum of sample variances $\hat{\sigma}_X^2 + \hat{\sigma}_Y^2$?

□

Exercise 1.9 (Expectation of a Roll of a Die)**.** If X is the outcome of the roll of a fair die, what is $E[X]$? What is $\mathrm{var}[X]$? □

Exercise 1.10 (Expectations on Dice)**.** Let X be the *sum* of the outcome of the roll of three fair dice. What is $E[X]$? What is $\mathrm{var}[X]$? Assume the outcome of any individual die is independent of that of the others. □

Exercise 1.11 (Prove the Sample Variance is Unbiased)**.** Let $X_1, \ldots, X_N$ be independent random samples drawn from a population with expectation μ and variance σ^2. Show that the expectation of the sample variance $\hat{\sigma}_X^2$ is

$$E[\hat{\sigma}_X^2] = \sigma^2 \tag{1.3}$$

For this reason, $\hat{\sigma}_X^2$ is called an *unbiased* estimate of σ_X^2. (This question is hard) □

Exercise 1.12. Suppose you are predicting whether Y will be above or below normal. If "normal" is defined as the median, then, by definition, Y will be above normal 50% of the time (like a coin flip). Suppose, however, you are predicting Y given knowledge of another variable X. If X and Y are independent, then knowledge of X tells you nothing new about Y; i.e., the probability that Y is above normal is still 50%. However, if X and Y have a non-zero correlation, then knowledge of one tells you something new about the other. Suppose that the correlation between X and Y is 0.5. An example of such a sample is shown in fig. 1.3c. If X is above normal, what is the probability that Y is above normal? To answer this question, use example 1.7 to generate random numbers with a population correlation of $\rho = 0.5$, and then from these numbers compute the probability that $Y > 0$ given that $X > 0$. You should generate enough random samples to obtain *robust* estimates of the probability (i.e., ensure your answer does not change if you re-run your code with different random numbers).

□

Exercise 1.13 (Random Walk in Two Dimensions)**.** A particle takes random steps in the xy-plane. The step increment in the x-direction is determined by drawing independent random numbers from a normal distribution with zero mean and variance σ^2. Positive

values correspond to steps in the positive x-direction and negative values correspond to steps in the negative x-direction. The step increment in the y-direction is determined in the analogous way. Find the equation for the radius of the circle such that there is a 95% probability that the particle lies within the circle after 10 steps. Evaluate this equation for the case $\sigma = 2$. □

Exercise 1.14. In exercise 1.6, you proved that $k = \mathbb{E}[X]$ minimized $\mathbb{E}[(X - k)^2]$. This says that the best prediction of X (in a mean square sense) is the mean. However, the population mean is unknown. Accordingly, consider a prediction of X based on the sample mean $\hat{\mu}_X$, derived from $X_1, \ldots, X_N$. It turns out that the sample mean is *not* the best prediction of X', when X' is independent of the sample used to construct $\hat{\mu}_X$. Introduce a scaling factor α, and show that the α that minimizes

$$\mathbb{E}[(X' - \alpha\hat{\mu})^2],$$

is

$$\alpha = \frac{1}{\left(\frac{\sigma_X^2}{N\mu_X^2}\right) + 1},$$

where μ_X and σ_X^2 are the population means and variances of X. Show that $\alpha < 1$, which explains why this predictor is sometimes called a *shrinkage* estimator. □

CHAPTER 2

EXERCISES FOR STATISTICAL INFERENCE

Exercise 2.1. A widely used measure of the state of the Pacific Ocean is the Pacific Decadal Oscillation (PDO) index. This index is a certain linear combination of sea surface temperatures in the Pacific Ocean poleward of 20N. More details and background information can be found at http://jisao.washington.edu/pdo/, but this information is not necessary in order to complete this homework assignment. The January-March mean value of this index is plotted in fig. 2.1. A glance at the figure reveals that the PDO index was predominantly negative during the period 1950-1976 and predominantly positive 1977-2017. In this homework set, you will address the following questions about the PDO:

1. Are the samples independent?

2. Has the mean PDO changed in recent decades?

To address the above questions, you need to download the data file `PDO.latest.txt` *from the class website*. The R tutorial discussed how to download this data set, but the data file from the PDO webpage changed slightly, so I have updated the data set and included R code for reading it called `pdo.student.R`. Both files can be downloaded from the class website[1] (click "data" and "repository of R programs"). This code should run without generating any errors, so please contact me if you have trouble running this code.

[1] http://cola.gmu.edu/delsole/clim762/webpage/clim762_frontpage.html

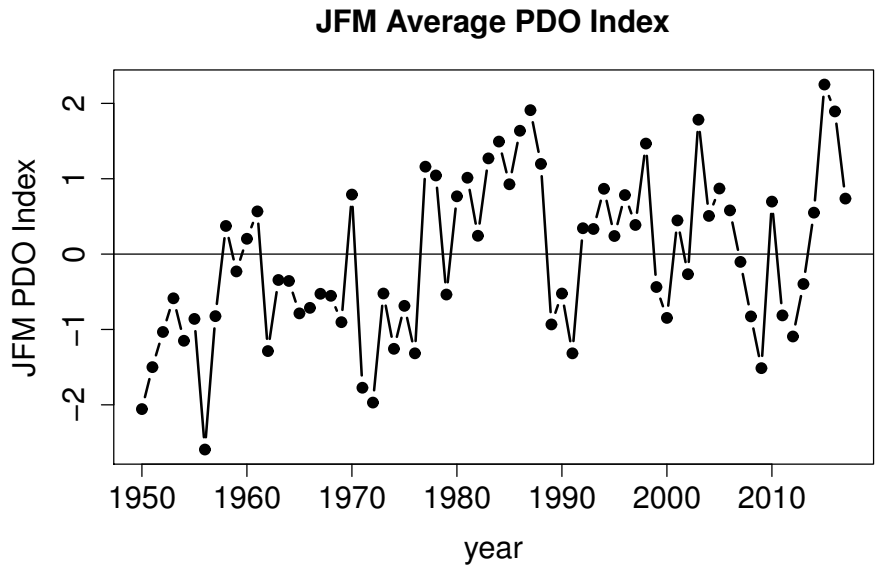


Figure 2.1 The January-March mean Pacific Decadal Oscillation (PDO) index. Positive values are indicated by thick bars. Statistics of the January-March mean PDO Index are given in the table.

For this homework set, you may use the built-in functions `mean` and `var`, but do not use the built-in functions `cov` and `cor`; rather, you should compute these quantities explicitly using the `sum` command (as shown in the tutorial). The purpose of requiring you to calculate these directly is to ensure you understand how to do these calculations yourself. You can, however, use these functions to check your results. You may also use these built-in functions after this homework set.

You will write a *function* that performs a hypothesis test. You should read the appropriate sections of the tutorial to become familiar with functions. In particular, the tutorial illustrates a complete example for one particular hypothesis test (i.e., equality-of-variance test), which in turn can serve as a template for the other hypothesis tests in this homework.

In each problem, you should explicitly state your null hypothesis, test statistic, the observed value of the statistic, rejection region, and final decision about the hypothesis. You should also state all the assumptions that were made in performing the hypothesis test.

(a) **Are the Samples Independent?** Recall that one assumption in both the F-test and t-test is that the samples are independent. Climate time series have the property that the degree of dependence decays with time separation. For instance, 1-day forecasts are more accurate than 5-day forecasts. One approach to checking the independence assumption is by testing if the correlation *between two consecutive time steps* vanishes. If no correlation can be detected between consecutive time steps, then it is unlikely it can be detected for larger time separations. Write an all-purpose function called `cor.equal.test` that tests the

hypothesis that the correlation between two given data sets vanishes. Turn in a copy of your code. The call to this function should start as follows:

```
cor.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS VANISHING CORRELATION BETWEEN
## TWO BI-VARIATE, NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   RHO: SAMPLE CORRELATION BETWEEN DATA1 AND DATA2
#   RHO.CRIT: 100*ALPHA% CRITICAL VALUE FOR THE CORRELATION
#   PVAL: P-VALUE OF THE STATISTIC RHO
```

(b) Using this function, test the hypothesis that during 1950-1977 the correlation between consecutive years is zero. This means that whatever you use for the time series in `data1`, `data2` is the *same time series shifted by one year*. This also means you cannot input the entire time series, but instead the length of the time series is short by one year. Do the same test for 1978-2017. What is your answer to these questions? Turn in a copy of the output of your function, which should give numerical values for all quantities like $\hat{\rho}$, ρ_{crit}, p-value.

Incidentally, an R function called cor.test also performs this test. Check that your function agrees with `cor.test`.

(c) Repeat the above test, but this time for the whole period 1950-2017. You should find that $\rho = 0$ is rejected more strongly than for the two separate periods individually. Explain why. Hint: draw a scatter diagram for the three separate cases.

(d) Answer the other questions posed at the beginning (i.e., define your test statistic, state the value of the statistic, specify the rejection region, etc.).

(e) **Has the mean PDO changed in recent decades?** To address this question, we test the hypothesis of no difference in population mean between the periods 1950-1977 and 1978-2017? Write a function called `mean.equal.test` that performs the hypothesis test. Run your function on the PDO data. Turn in a copy of your code and its output. The function should begin as follows:

```
mean.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS EQUALITY OF MEANS OF TWO IID
## NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N1]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N2]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   DIFF.MEAN: DIFFERENCE IN MEANS (MEAN1 - MEAN2)
#   DIFF.MEAN.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE DIFFERENCE IN MEANS
#   PVAL: P-VALUE OF THE T-STATISTIC
#   T: T-STATISTIC FOR DIFFERENCE IN MEANS
#   T.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE T STATISTIC
#   MEAN1: ESTIMATE OF THE MEAN OF DATA1
#   MEAN2: ESTIMATE OF THE MEAN OF DATA2
#   SPOOL: POOLED ESTIMATE OF THE STANDARD DEVIATION
```

(f) An output quantity not discussed in class (or the notes) is `diff.mean.crit`: this is the threshold value of the absolute difference in means for deciding whether to reject the null hypothesis. Write an equation that gives this threshold value. State the numerical value for this problem.

(g) State your test statistic, the value of the statistic, and specify the rejection region. What is your decision? Be sure to explicitly state all assumptions made in your hypothesis test.

(h) Explain whether the $\rho = 0$ test and equality-of-variance tests, which examined above and in the R tutorial, are relevant to testing equality of means.

□

CHAPTER 3

EXERCISES FOR CONFIDENCE INTERVALS

Exercise 3.1. Suppose $X_1, \ldots, X_N$ are independent and identically distributed as $\mathcal{N}(10, 3)$. Derive a formula for the 95% confidence interval for the mean. (The answer is given in (3.17), but I want to see you derive it.) Write an R code that generates $N = 20$ samples $\mathcal{N}(10, 3)$, computes the confidence interval, and repeats this many times (1000s). Compute the fraction of intervals that include the true mean $\mu = 10$. Verify that this fraction is close to 95%. □

Exercise 3.2. The tutorial contains the function `var.equal.test` for testing equality of variance. Augment this function to also calculate a confidence interval for the ratio of variances. The preamble of the revised function should like the following:

```
var.equal.test = function(data1,data2,alpha=0.05) {
### THIS FUNCTION TESTS EQUALITY OF VARIANCE OF TWO
### IID NORMALLY DISTRIBUTED RANDOM VARIABLES
###
# INPUT:
#   DATA1: [N1]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N2]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   F.MAX: RATIO OF VARIANCE, CONSTRUCTED TO BE GREATER THAN 1
#   F.CRIT: THE UPPER CRITICAL THRESHOLD OF SIGNIFICANCE
#   PVAL: P-VALUE OF THE F.MAX RATIO
#   VAR1: UNBIASED ESTIMATE OF THE VARIANCE OF DATA1
#   VAR2: UNBIASED ESTIMATE OF THE VARIANCE OF DATA2
#   RATIO: VAR1/VAR2
#   RATIO.LOWER: LOWER LIMIT OF ALPHA% CONFIDENCE INTERVAL FOR VARIANCE RATIO
#   RATIO.UPPER: UPPER LIMIT OF ALPHA% CONFIDENCE INTERVAL FOR VARIANCE RATIO
```

The above is identical to the function in the tutorial, but with three additional lines at the end. Be sure the resulting confidence interval pertains to `var1/var2` and not its reciprocal. Run this function on exactly the same PDO data you used in exercise 2.1 to test if the variance before and after 1977 differ. Report the 95% confidence interval. Comment on whether the result is consistent with the hypothesis test you performed in exercise 2.1. □

Exercise 3.3. In the last exercise, you wrote the function `cor.equal.test` to test the correlation coefficient. Modify this function to also calculate a confidence interval for the correlation coefficient. The preamble of this function should look like the following:

```
cor.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS VANISHING CORRELATION BETWEEN
## TWO BI-VARIATE, NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   RHO: SAMPLE CORRELATION BETWEEN DATA1 AND DATA2
#   RHO.CRIT: 100*ALPHA% CRITICAL VALUE FOR THE CORRELATION
#   PVAL: P-VALUE OF THE STATISTIC RHO
#   RHO.LIMITS: [2] (1-ALPHA)*100% CONFIDENCE LIMITS OF THE CORRELATION
```

Run this function on the same PDO data to test if the lag-1 correlation is significant before 1977, after 1977, and the full period. Report the 95% confidence intervals. Comment on whether the result is consistent with the hypothesis tests you performed in exercise 2.1. □

Exercise 3.4. In the last exercise you wrote the function `mean.equal.test` to test equality of means. Modify this function to also compute a confidence interval for the difference in mean. The preamble of this function should look like the following:

```
mean.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS EQUALITY OF MEANS OF TWO IID
## NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N1]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N2]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   DIFF.MEAN: DIFFERENCE IN MEANS (MEAN1 - MEAN2)
#   DIFF.MEAN.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE DIFFERENCE IN MEANS
#   PVAL: P-VALUE OF THE T-STATISTIC
#   T: T-STATISTIC FOR DIFFERENCE IN MEANS
#   T.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE T STATISTIC
#   MEAN1: ESTIMATE OF THE MEAN OF DATA1
#   MEAN2: ESTIMATE OF THE MEAN OF DATA2
#   SPOOL: POOLED ESTIMATE OF THE STANDARD DEVIATION
#   DIFF.MEAN.LIMITS: [2] (1-ALPHA)100% CONFIDENCE
#       LIMITS FOR THE DIFFERENCE IN MEANS
```

Apply this function to the same PDO data to test if the mean PDO index changed before and after 1977. State the 95% confidence limits. Comment on whether the result is consistent with the hypothesis tests you performed in the last exercise. □

Exercise 3.5 (Bootstrap Confidence Interval for a Correlation Coefficient)**.** Write an R function to compute a confidence interval for a correlation coefficient using the bootstrap method. The preamble of this function should look like the following:

```
cor.equal.boot = function(x,y,alpha=0.05,nboot=100000) {
## ESTIMATES (1-ALPHA)100% CONFIDENCE INTERVAL
## FOR THE CORRELATION COEFFICIENT USING BOOTSTRAP METHODS
## INPUT:
##  X[NSAMP]: FIRST DATA SET
##  Y[NSAMP]: SECOND DATA SET
##  ALPHA: EQUIVALENT SIGNIFICANCE LEVEL (DEFAULT = 5%)
##  NBOOT: NUMBER OF BOOTSTRAP SAMPLES (DEFAULT = 100000)
## OUTPUT:
##  CONFIDENCE LIMITS[2]
```

To test your code, generate $N = 20$ random samples of (X,Y) from a population with correlation $\rho = 0.5$ (do you remember how to do this? The answer is in homework 1.). Show both the bootstrap interval and the interval computed from `cor.test`. Discuss.

The following are some coding hints. R has a function called `sample` that will randomly sample a vector with replacement. To compute a confidence interval for a correlation coefficient, you must sample the two data sets X and Y in such a way as to preserve their pairing. This can be done by sampling the *indices* rather than the values themselves. For instance, if the sample size is $N = 10$, then the indices $1, \ldots, 10$ can be sampled using the following commands:

```
> nsamp = 10
> npic  = sample(nsamp,nsamp,replace=TRUE)
> npic
 [1] 4 6 6 4 6 1 1 2 2 8
```

The bootstrap sample and its correlation is then

```
x = rnorm(nsamp); y = rnorm(nsamp)
xy.cor.obs  = cor(x,y)
xy.cor.boot = cor(x[npic],y[npic])
```

After you generate lots of bootstrap samples, the $(1-\alpha)100\%$ confidence limits can be obtained using the command

```
confidence.limits = quantile(xy.cor.boot,probs=c(alpha/2,1-alpha/2))
```

Because the bootstrap requires 1000s of iterations, you want a code that is fast. Unfortunately, R is slow whenever `for` loops are involved. Can you write a function that performs the bootstrap without `for` loops? If you cannot figure out how to do this, then just use loops– I much prefer that you turn in a slow code that is correct, rather than a fast code that is incorrect! Hint: use `rowMeans` or `colMeans`. Advice: set `nboot` to a small number (e.g., 100 or 1000) while you debug your code, so it will run fast, and then when you believe the code is working, increase `nboot` to 100,000 as requested in the above preamble. □

CHAPTER 4

EXERCISES FOR STATISTICAL TESTS BASED ON RANKS

Exercise 4.1. Write a function to perform the Wilcoxon Rank-Sum test for a difference in medians. The preamble of this function should be as follows:

```
diff.mean.nonparametric = function(data1,data2,alpha=0.05) {
### PERFORMS MANN-WHITNEY TEST FOR A DIFFERENCE IN MEDIANS
### INPUT:
###   DATA1[N1]: SAMPLE 1
###   DATA2[N2]: SAMPLE 2
###   ALPHA: SIGNIFICANCE LEVEL (DEFAULT = 5%)
### OUTPUT: LIST$
###   Z: Z-VALUE FOR THE NORMAL APPROXIMATION TO MANN-WHITNEY TEST
###   Z.CRIT: CRITICAL VALUE FOR ALPHA*100% SIGNIFICANCE
###   PVAL: P-VALUE OF THE TEST
```

The following R functions are useful for non-parametric testing: `rank()`, `sort()`, `order()`. You can read the documentation for these functions and play with them by generating the random numbers seeing what happens:

```
> x = rnorm(5)
> x
[1]  0.5726722 -0.6338714  0.3260635 -1.6519467  0.9540662
> sort(x)
[1] -1.6519467 -0.6338714  0.3260635  0.5726722  0.9540662
> order(x)
[1] 4 2 3 1 5
> x[order(x)]
[1] -1.6519467 -0.6338714  0.3260635  0.5726722  0.9540662
> rank(x)
[1] 4 2 3 1 5
```

Rank is a very useful function for this homework set because it automatically deals with ties correctly.

```
> x[1] = x[2]
> x
[1] -0.6338714 -0.6338714  0.3260635 -1.6519467  0.9540662
> rank(x)
[1] 2.5 2.5 4.0 1.0 5.0
```

□

Exercise 4.2. Run this function on exactly the same PDO data you used in the previous 2 exercise sets to test if the median before and after 1977 differ. Clearly state the z-value, critical z value, and the p-value of the test. Clearly state whether you reject the null hypothesis or not, and whether this conclusion is consistent with your conclusions on previous homeworks. You can check your answer using `wilcox.test(pdo1,pdo2)` □

Exercise 4.3. In exercise 2.1, you wrote the function `cor.equal.test` to test the correlation coefficient. Write a new function that performs a correlation test based on Spearman's rank correlation. The preamble of this function should look like the following:

```
cor.test.nonparametric = function(data1,data2,alpha=0.05) {
### PERFORMS RANK CORRELATION TEST
### INPUT:
###   DATA1[N1]: SAMPLE 1
###   DATA2[N2]: SAMPLE 2
###   ALPHA: SIGNIFICANCE LEVEL (DEFAULT = 5%)
### OUTPUT: LIST$
###   RHO.SPEARMAN = SPEARMAN'S RANK CORRELATION
###   ZVAL: Z-VALUE FOR THE NORMAL APPROXIMATION TO MANN-WHITNEY TEST
###   Z.CRIT: CRITICAL VALUE FOR ALPHA*100% SIGNIFICANCE
###   PVAL: P-VALUE OF THE TEST
```

□

Exercise 4.4. Run this function on the same PDO data you used in previous exercises to test if the lag-1 correlation is significant before 1977, after 1977, and the full period. State whether the procedure rejects the null hypothesis or not. Comment on whether the result

is consistent with the hypothesis tests you performed in exercise 2.1. You can check your answer to some extent using

```
cor.test(pdo1[l0],pdo1[l0+1],method='spearman',continuity=FALSE,exact=FALSE)
```

However, the p-value will not be exactly the same (but they are close). The reason they are not the same is because R computes p-values using

$$t = \frac{\hat{\rho}_{Sp}\sqrt{N-2}}{\sqrt{1-\hat{\rho}_{Sp}^2}}, \tag{4.1}$$

and then it refers to a t-distribution with $N-2$ degrees of freedom.

□

Exercise 4.5. Write a function called `diff.dispersion` for testing a difference in dispersion. The preamble of this function should be as follows:

```
diff.dispersion = function(data1,data2,alpha=0.05) {
## PERFORMS WILCOXON SQUARED-RANK TEST
##  ON ABSOLUTE DEVIATIONS |X - MEDIAN(X)|
### INPUT:
###   DATA1[N1]: SAMPLE 1
###   DATA2[N2]: SAMPLE 2
###   ALPHA: SIGNIFICANCE LEVEL (DEFAULT = 5%)
### OUTPUT: LIST$
###   ZVAL: Z-VALUE FOR THE NORMAL APPROXIMATION TO MANN-WHITNEY TEST
###   Z.CRIT: CRITICAL VALUE FOR ALPHA*100% SIGNIFICANCE
###   PVAL: P-VALUE OF THE TEST
```

Run your function on exactly the same PDO data you used in previous exercises to test if the variance before and after 1977 differ. Clearly state the z-value, critical z value, and the p-value of the test. Clearly state whether you reject the null hypothesis or not, and whether this conclusion is consistent with your conclusions on previous homeworks. (Hint: most of this function is similar to your difference in medians test.)

□

CHAPTER 5

EXERCISES FOR STOCHASTIC PROCESSES

Exercise 5.1. Write an R function to compute the autocorrelation function of a time series. The preamble of the function should be as follows:

```
acf.brute = function(x,lag.max=NULL) {
## COMPUTES THE AUTOCORRELATION FUNCTION OF A TIME SERIES
## INPUT:
##   X: [NTOT]-LENGTH NUMERICAL VECTOR OF THE TIME SERIES
##   LAG.MAX: MAXUMIM LAG TO COMPUTE
##       (DEFAULT = MAX(1,FLOOR(10*log10(LENGTH(X)))))
## OUTPUT:
##   X.ACF: [1:(LAG.MAX+1)] AUTOCORRELATION FUNCTION OF X
##       FOR LAGS 0 TO LAG.MAX
```

Test your function on the time series {`set.seed(1); x = rnorm(20)`} and state the correlation values for all `lag.max` lags. Show that these values equal those obtained from the built-in R function `acf(x)`, by printing them out using the commands

```
acf.R    = acf(x)
as.numeric(acf.R$acf)
```

□

Exercise 5.2. Write a function to generate a time series from the AR1 model

$$x_{t+1} = \phi_1 x_t + w_t + k \tag{5.1}$$

where $w_t \sim GWN(0, 1)$. The beginning of this function should be as follows:

```
ar1.ts = function(phi1,ntot,stdv=1,constant=0,iseed=1) {
## GENERATE A REALIZATION FROM THE AR1 MODEL
## X(T+1) = PHI1 * X(T) + W(T) + CONSTANT
## WHERE W(T) IS GAUSSIAN WHITE NOISE WITH MEAN 0 AND ST. DEV. 'STDV'
## INITIAL CONDITION IS RANDOMLY DRAWN FROM STATIONARY DISTRIBUTION
## INPUT:
#    PHI1: AUTOREGRESSIVE PARAMETER
#    NTOT: LENGTH OF THE DESIRED TIME SERIES
#    STDV: STANDARD DEVIATION OF THE NOISE (DEFAULT = 1)
#    CONSTANT: CONSTANT TERM IN THE AR1 MODEL (DEFAULT = 0)
#    ISEED: SEED FOR THE RANDOM NUMBER GENERATOR (DEFAULT = 1)
#  OUTPUT:
#    X: [NTOT]- RANDOM TIME SERIES FROM AR(1) PROCESS
```

The initial condition should be drawn from the *stationary distribution* of the AR process. What is this distribution? State this distribution for general k, ϕ_1, σ_W^2. Use this equation in your function. □

Exercise 5.3. Use the above functions to generate a time series of length 50, and corresponding sample autocorrelation functions, for $\phi_1 = 0, 0.5, 0.9$. Make plots of both quantities. In addition, on the plot for the autocorrelation, superimpose the population autocorrelation function, and the approximate 95% confidence interval for zero correlation. Comment on whether the results make sense, or whether they are 'surprising.'

□

Exercise 5.4. Use the above function to generate a time series of length 50 for $\phi_1 = 0$. Split this time series into two halves, each of length 25. Use the function `mean.equal.test` to calculate the t-statistic for testing equality of means. Repeat this 1000 times, thereby generating 1000 t-values. Plot a histogram of these t-values. Superimpose on this histogram the expected distribution of the t-statistic. This can be done using the `curve` command, as follows:

```
hist(tval,col="grey",freq=FALSE)
curve(dt(x,dof),add=TRUE)
```

(In the above example, `x` does not need to be pre-defined. See the manual on R graphics or the help pages for `curve`.) You should find that the histogram is reasonably consistent with the exact t distribution. Repeat this for $\phi_1 = 0.5, 0.9$. Are the histograms still consistent with the exact t distribution? Explain the result you find. If someone concludes that there is significant difference in mean based on a standard t-test (i.e., using iid assumption), how might this conclusion be affected if autocorrelation were taken into account? Does the true significance level increase or decrease when autocorrelation is taken into account? □

Exercise 5.5 (Mean and Variance of an AR(p) Process)**.** Consider the AR(p) process

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + W_t + k, \tag{5.2}$$

where W_t is white noise with zero mean and variance σ^2, and *k* is a constant. Show that the expectation of the stationary AR(p) process is

$$\mathbb{E}[X_t] = \frac{k}{1 - \phi_1 - \phi_2 - \cdots - \phi_p}. \tag{5.3}$$

Show that the variance of the stationary process is

$$\text{var}[X_t] = \frac{\text{var}[W_t]}{1 - \rho_1\phi_1 - \rho_2\phi_2 - \cdots - \rho_p\phi_p}. \tag{5.4}$$

(Hint: multiply both sides of (5.2) by $X_t - \mathbb{E}[X_t]$ and take the expectation.) □

Exercise 5.6 (Autocorrelation of a running mean)**.** Consider the running mean

$$Y_t = \frac{1}{2K+1} \sum_{k=-K}^{K} X_{t+k}. \tag{5.5}$$

Assume $X_t \sim GWN(0, \sigma^2)$. Compute the mean and variance of Y_t. Show that the autocorrelation function of Y_t is

$$\text{cor}[Y_{t+\tau}, Y_t] = \begin{cases} 1 - \frac{|\tau|}{2K+1} & \text{for } |\tau| \leq 2K+1 \\ 0 & \text{for} |\tau| > 2K+1 \end{cases}. \tag{5.6}$$

Note that even though the process X_t is uncorrelated, running averages of X_t are correlated for lags less than twice the window size K. Explain why this makes sense. □

CHAPTER 6

EXERCISES ON THE POWER SPECTRUM

In this homework you will determine power spectra from models and data. To do this, you will need to use the Fast Fourier Transform, which is performed by the R function `fft()`. The output of `fft` is somewhat technical, so I give you the following program to convert the output into a periodogram:

```
periodogram = function(x) {
## COMPUTES THE PERIODOGRAM OF A VECTOR X USING THE FFT
## ASSUMES THE LENGTH OF X IS EVEN AND HIGHLY COMPOSITE
##   (E.G., EXPRESSIBLE AS POWERS OF 2,3,5)
# INPUT:
#    X[NTOT]: DATA VECTOR
# OUTPUT:
#    PGRAM[NTOT/2]: VECTOR OF PERIODOGRAM AMPLITUDES AT HARMONIC FREQUENCIES
#      1/N, 2/N, . . ., 1/2 CYCLES PER TIME STEP (A TOTAL NTOT/2 FREQUENCIES)
# COMMENT: (BIASED) VARIANCE = 2 * SUM(PGRAM[1:(NTOT/2-1)]) + PGRAM[NTOT/2]

ntot  = length(x)
if ( ntot %% 2 != 0 ) stop('length of x must be even')
x.fft = fft(x)
pgram = Re(x.fft * Conj(x.fft))[1:(ntot/2)+1] / ntot
pgram
}
```

You should verify that this function works by setting x to be specific sum of sines and cosines and ensuring that the output gives the appropriate amplitudes, when appropriately scaled. The following example shows a successful test of this function:

```
> set.seed(4)
> ntot = 32
> t    = 1:ntot - 1
> x    = 1 * cos(2*pi*t*5/ntot) - 3 * sin(2*pi*t*13/ntot)
       + 8 * cos(2*pi*t*(ntot/2)/ntot)
> x.pgram = periodogram(x)
> x.amplt = sqrt(4*x.pgram/ntot)   # convert pgram to fourier amplitude
# special conversion for last harmonic
> x.amplt[ntot/2] = sqrt(x.pgram[ntot/2]/ntot)
> x.amplt
 [1] 2.519803e-15 2.181560e-15 7.616681e-16 1.930989e-15 1.000000e+00
 [6] 1.260945e-15 7.422817e-16 3.202373e-15 2.989367e-15 2.373832e-15
[11] 3.143748e-15 1.223717e-15 3.000000e+00 3.084970e-16 1.271623e-15
[16] 8.000000e+00
```

The output shows that the absolute amplitude of the 5th, 13th, and 16th harmonics are 1, 3, 8, as they should, while the other amplitudes are effectively zero, as expected. Another check is to verify the formula

$$\sum_{t=1}^{N} (x_t - \overline{x})^2 = 2 * \sum_{j=1}^{\frac{N}{2}-1} \hat{P}(j/N) + \hat{P}(N/2), \tag{6.1}$$

which holds for even N. The following lines perform this check:

```
> set.seed(1)
> x       = rnorm(ntot)
> ssx     = sum((x-mean(x))^2)
>
> x.pgram = periodogram(x)
> ssx.fft = 2*sum(x.pgram[1:(ntot/2-1)]) + x.pgram[ntot/2]
> print(all.equal(ssx,ssx.fft))
[1] TRUE
```

Exercise 6.1. Write an R function to compute the spectral average

$$\hat{p}_M(\omega_j) = \frac{1}{2M+1} \sum_{m=-M}^{M} \hat{p}(\omega_j). \tag{6.2}$$

The preamble of this R function should be as follows:

```
spectrum.smooth = function(pgram,window,alpha=0.1) {
## COMPUTES SPECTRAL RUNNING MEAN AS:
##   SPEC_AVE = SMOOTHED PERIODOGRAM FROM M = -WINDOW TO M = WINDOW
## INPUT:
##    PGRAM[NTOT/2]: PERIODOGRAM VALUES FROM PERIODOGRAM.R
##    WINDOW: HALF-WIDTH OF AVERAGING WINDOW
##    ALPHA: THE (1-ALPHA)100% LEVEL FOR THE CONFIDENCE LIMITS
## OUTPUT: LIST
##    PGRAM.AVE[NTOT/2]: AVERAGED PERIODOGRAM
##    FREQ[NTOT/2]: FREQUENCIES (CYCLES PER TIME STEP)
##    BW[2]: BANDWIDTH C(-WINDOW/NTOT,WINDOW/NTOT)
##    CI.LIMITS[2]: PROPORTIONALITY CONSTANT FOR LOWER AND UPPER CONFIDENCE LIMITS
##       (DOF/CHI-SQUARE[ALPHA/2, DOF], DOF/CHI-SQUARE[1-ALPHA/2,DOF])
##    PGRAM[NTOT/2]: REPEATS PERIODOGRAM (FOR LATER PLOTTING)
```

The spectral average near the extreme low and high frequencies, where the smoothing cannot be performed consistently, should be set to `NA`. □

Exercise 6.2. Generate a time series of length 64 from a standardized Gaussian white noise process, perform a spectral average using $M = 5$, and write out the results, as follows:

```
> set.seed(1)
> x.sd        = 1
> window      = 5
> ntot        = 64
> freq        = 1:(ntot/2) / ntot
> x           = rnorm(ntot,sd=x.sd)
> x.pgram     = periodogram(x)
> x.pgram.smooth = spectrum.smooth(x.pgram,window)
> cbind(freq,x.pgram,x.pgram.smooth$pgram.ave)
```

Plot the results using the R function `spectrum.plot.R`, provided on the class website. Based on this result, would you say that the estimated power spectrum is consistent with a white noise process? Explain why or why not. □

Exercise 6.3. What is the theoretical (i.e., population) power spectrum for the Gaussian white noise process? Superimpose this theoretical power spectrum on your spectrum plot. Is the estimated power spectrum consistent with the theoretical spectrum? □

Exercise 6.4. Generate a time series of length 64 from an AR(1) process with parameter $\phi_1 = 0.8$, as follows:

```
ntot        = 64
phi1        = 0.8
x           = ar1.ts(phi1,ntot,iseed=1)
```

Perform a spectral average of this time series using $M = 5$, and write out the results, as follows:

```
> window      = 5
> x.pgram     = periodogram(x)
> x.pgram.smooth = spectrum.smooth(x.pgram,window)
> cbind(freq,x.pgram,x.pgram.smooth$pgram.ave)
```

Make a plot of the power spectrum, smoothed periodogram, bandwidth, and 90% confidence interval using `spectrum.plot.R`. Is the estimated spectrum more consistent with white noise or an AR(1) process? Explain why or why not. □

Exercise 6.5. What is the theoretical power spectrum of the above AR(1) process? Superimpose this theoretical spectrum on your spectrum plot. Is your estimated power spectrum consistent with the theoretical spectrum? □

Exercise 6.6. Estimate the power spectrum of the NINO3.4 index. The NINO3.4 index can be extracted using the following R program:

```
fname       = paste(dir.enso,'nina34.data',sep='')
line1       = scan(fname,nlines=1,quiet=TRUE)
nino34.yr   = line1[1]:line1[2]
nino34.nyrs = length(nino34.yr)
cnames      = c('YEAR',month.abb[])
data2       = read.table(fname,skip=1,nrows=nino34.nyrs,
   na.strings=-99.99,col.names=cnames)

## CREATE VECTOR WITH NINO34 TIME SERIES AND TIME AXIS
nino34.ts   = as.numeric(t(as.matrix(data2[,1:12+1])))
nino34.time = seq(from=data2[1,1],length.out=length(nino34.ts),by=1/12)
```

The resulting time series will have `NA`'s which need to be stripped out before performing spectral analysis. Trim the length of the index to be highly composite (i.e., a product of powers of 2, 3, 5). State the final length that you use. What are the units of "frequency"? Are there any significant peaks? Do these peaks make sense (do you expect certain peaks)? Could an AR-process fit this spectrum? ENSO is often said to be characterized by oscillations around 4-7 years. Does your estimated power spectrum support this characterization? Explore different bandwidths, choose a single one that you believe is most revealing, and show that one. □

CHAPTER 7

EXERCISES FOR VECTORS, MATRICES, AND GEOMETRY

Exercise 7.1 (Linear combination of random vectors)**.** Let $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_K$ be independent M-dimensional random vectors. However, the vectors are not drawn from the same population. Instead, the populations have individual means $\mathbb{E}[\mathbf{x}_k] = \boldsymbol{\mu}_k$ and covariance matrices $\mathrm{cov}[\mathbf{x}_k] = \boldsymbol{\Sigma}_k$. Because the vectors are independent,

$$\mathrm{cov}[\mathbf{x}_k, \mathbf{x}_{k'}^T] = \begin{cases} \mathbf{0} & k \neq k' \\ \boldsymbol{\Sigma}_k & k = k' \end{cases}. \tag{7.1}$$

Note that there are K distinct covariance matrices $\boldsymbol{\Sigma}_1, \boldsymbol{\Sigma}_2, \ldots, \boldsymbol{\Sigma}_K$. Suppose $\mathbf{y} = c_1\mathbf{x}_1 + c_2\mathbf{x}_2 + \cdots + c_N\mathbf{x}_N$, where the c's are known scalars. What is the mean and covariance matrix of $\mathbf{y}$? □

Exercise 7.2 (Matrix combinations of random vectors)**.** Let $\mathbf{x} \sim \mathrm{N}_M(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Suppose $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{b}$, where $\mathbf{A}$ is a constant matrix and $\mathbf{b}$ is a constant vector. What is the complete distribution of $\mathbf{y}$? □

Exercise 7.3 (Properties of the Trace and Determinant)**.** Let $\mathbf{A}$ be a square matrix, but not necessarily symmetric. Square matrices still have eigenvectors and eigenvalues that satisfy

$$\mathbf{A}\mathbf{s} = \lambda\mathbf{s}. \tag{7.2}$$

As a result, the matrix is close enough for practical purposes to having an eigen decomposition of the form

$$\mathbf{A} = \mathbf{S}\boldsymbol{\Lambda}\mathbf{S}^{-1}, \tag{7.3}$$

in which case A is said to diagonalizable and the properties below are easy to prove. If $\mathbf{A}$ is symmetric, then we know that there exists a decomposition of the above form where $\mathbf{S}$ is an orthogonal matrix and $\mathbf{\Lambda}$ has real diagonal elements, but these special properties are not needed in the following problems.

(a) Show that $\mathrm{tr}\,\mathbf{A} = \mathrm{tr}\,\mathbf{A}^T$ for any matrix $\mathbf{A}$.

(b) Show that $\mathrm{tr}\{\mathbf{AB}\} = \mathrm{tr}\{\mathbf{A}^T\mathbf{B}^T\}$ for any two matrices $\mathbf{A}$ and $\mathbf{B}$ such that $\mathrm{tr}\{\mathbf{AB}\}$ is defined.

(c) Show that $\mathrm{tr}\,\mathbf{A} = \sum \lambda_i$, where the λ_i's are the eigenvalues of $\mathbf{A}$, and the sum is over all eigenvalues.

(d) What is $\mathrm{tr}\{\mathbf{A}^k\}$ in terms of the eigenvalues of $\mathbf{A}$?

(e) Show that $\mathrm{tr}[\mathbf{A}^{-1}] = \sum_{m=1}^{M} \lambda_m^{-1}$, where the λ_m's are the eigenvalues of $\mathbf{A}$.

(f) Show that $\mathrm{tr}\{\mathbf{AA}^T\} = \sum_i \sum_j a_{ij}^2$, where a_{ij} is the (i, j) element of $\mathbf{A}$.

(g) Show that $|\mathbf{A}| = \lambda_1\lambda_2 \ldots \lambda_p$, where $\mathbf{A}$ is a $p \times p$ matrix and the λ_i's are the eigenvalues of $\mathbf{A}$.

(h) If $\mathbf{A}$ is symmetric and positive definite, show that $\mathrm{tr}\,\mathbf{A} > 0$ and $|\mathbf{A}| > 0$.

(i) Show that the trace of a positive semi-definite matrix is non-negative. Show that the determinant of a positive semi-definite matrix is non-negative.

□

Exercise 7.4 (Idempotent matrices)**.** An *idempotent* matrix is a matrix $\mathbf{A}$ such that $\mathbf{A}^2 = \mathbf{A}$. Show that the eigenvalues of an idempotent matrix can have only two values, namely 0 and 1. Show that the number of eigenvalues equal to 1 is given by $\mathrm{tr}[\mathbf{A}]$. □

Exercise 7.5 (Properties of Orthogonal Matrices)**.**

(a) Show that orthogonal transformations preserve the Euclidean length of a vector. That is, if $\boldsymbol{\xi} = \mathbf{Ux}$, where $\mathbf{U}$ is an orthogonal matrix, then show that $\boldsymbol{\xi}^T\boldsymbol{\xi} = \mathbf{x}^T\mathbf{x}$.

(b) Show that orthogonal transformations preserve the cosine-angle between two vectors. That is, show that the cosine-angle between $\mathbf{x}$ and $\mathbf{y}$, and the cosine-angle between $\boldsymbol{\xi} = \mathbf{Ux}$ and $\boldsymbol{\eta} = \mathbf{Uy}$, are equal. The cosine-angle between two vectors $\mathbf{x}$ and $\mathbf{y}$ is

$$\cos\theta = \frac{\mathbf{x}^T\mathbf{y}}{\sqrt{(\mathbf{x}^T\mathbf{x})(\mathbf{y}^T\mathbf{y})}}. \tag{7.4}$$

(c) Show that the determinant of an orthogonal matrix is $+1$ or -1.

(d) Show that $|\mathbf{UAU}^T| = |\mathbf{A}|$, where $\mathbf{U}$ is an orthogonal matrix and $\mathbf{A}$ is any matrix.

(e) Show that $\mathrm{tr}\{\mathbf{UAU}^T\} = \mathrm{tr}\{\mathbf{A}\}$, where $\mathbf{U}$ is orthogonal and $\mathbf{A}$ is a square matrix.

(f) If $\mathbf{U}_1$ and $\mathbf{U}_2$ are orthogonal matrices, show that $\mathbf{U}_1\mathbf{U}_2$ is also orthogonal. Hint: remember that orthogonal matrices satisfy *two* properties.

□

Exercise 7.6. Suppose $\mathbf{A}$ is a constant matrix and $\mathbf{y}$ is a random vector with mean $\boldsymbol{\mu}_Y$ and covariance matrix $\boldsymbol{\Sigma}_Y$. Show that the expected value of the quadratic form $\mathbf{y}^T\mathbf{A}\mathbf{y}$ is $\mathrm{tr}[\mathbf{A}\boldsymbol{\Sigma}_Y] + \boldsymbol{\mu}^T\mathbf{A}\boldsymbol{\mu}$. □

Exercise 7.7. Show that the inverse of a transpose equals the transpose of the inverse:

$$\left(\mathbf{A}^T\right)^{-1} = \left(\mathbf{A}^{-1}\right)^T. \tag{7.5}$$

Hint: start with $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}$. □

Exercise 7.8 (Frobenius Norm)**.** The "length" of a vector is defined as the square root of the sum square vector elements. A natural generalization of "length" to a matrix $\mathbf{X}$ is

$$\|\mathbf{X}\|_F = \sqrt{\sum_i \sum_j x_{ij}^2}. \tag{7.6}$$

This measure is called the *Frobenius Norm*. It is clear that the Frobenius norm vanishes if and only if $\mathbf{X} = \mathbf{0}$, consistent with a measure of "length."

(a) Show that the Frobenius Norm can be written equivalently as

$$\|\mathbf{X}\|_F^2 = \mathrm{tr}\left[\mathbf{X}\mathbf{X}^T\right]. \tag{7.7}$$

(b) Suppose $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices with dimensions such that the product $\mathbf{U}\mathbf{A}\mathbf{V}$ is defined. Show that

$$\|\mathbf{U}\mathbf{A}\mathbf{V}\|_F = \|\mathbf{A}\|_F. \tag{7.8}$$

□

Exercise 7.9. Consider a $M \times M$ matrix $\mathbf{A}$ with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_M$. Show that the eigenvalues of $a\mathbf{I} + b\mathbf{A}$ are $a + b\lambda_m$, $m = 1, 2, \ldots, M$. □

Exercise 7.10. If $\mathbf{A}$ is positive definite, show that *each* diagonal element of $\mathbf{A}$ must be positive. □

Exercise 7.11 (Distribution of a random quadratic form)**.** Suppose $\mathbf{x} \sim \mathrm{N}_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, where $\boldsymbol{\Sigma}$ is positive definite. In this problem you will determine the distribution of the quadratic form

$$Q = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}). \tag{7.9}$$

(a) Let the eigenvector decomposition of the covariance matrix be

$$\boldsymbol{\Sigma} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T, \tag{7.10}$$

where $\mathbf{U}$ is an orthogonal matrix, and $\boldsymbol{\Lambda}$ is the diagonal eigenvalue matrix. What is the eigenvector decomposition of $\boldsymbol{\Sigma}^{-1}$?

(b) Using the results from part (a), construct a transformation matrix $\mathbf{A}$ such that

$$\boldsymbol{\zeta} = \mathbf{A}(\mathbf{x} - \boldsymbol{\mu}) \quad \text{and} \quad \boldsymbol{\zeta}^T\boldsymbol{\zeta} = Q. \tag{7.11}$$

What is $\mathbf{A}$?

(c) What is the distribution of $\boldsymbol{\zeta}$?

(d) A theorem in statistics states that if two random variables are uncorrelated *and* distributed as a multivariate Gaussian, then they are independent. Use this theorem to show that ζ_i and ζ_j are independent if $i \neq j$.

(e) What is the distribution of Q?

□

CHAPTER 8

EXERCISES FOR LINEAR REGRESSION: LEAST SQUARES ESTIMATION

In this homework assignment you will write an R function that estimates the regression parameter $\boldsymbol{\beta}$ in the model

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{8.1}$$

where $\mathbf{y}$ is an N-dimensional vector of predictands, and $\mathbf{X}$ is an $N \times M$ dimensional matrix of predictors. The core of this function is to compute an estimate of $\boldsymbol{\beta}$ based on solution of the normal equations

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}. \tag{8.2}$$

In practice, most packages use the SVD or QR algorithm to estimate $\boldsymbol{\beta}$. You will not be asked to use these algorithms because we are interested in focusing on *statistics*, not *numerics*. Having said that, you should be aware that you are solving the least squares problem inefficiently and perhaps inaccurately.

To calculate (8.2), we need to calculate $\mathbf{X}^T\mathbf{X}$ and $\mathbf{X}^T\mathbf{y}$. In R, the transpose operation is `t()`, and matrix multiplication is `%*%`. Thus, these terms are obtained by the commands

```
1 xtx      = t(x) %*% x
2 xty      = t(x) %*% y
```

The inverse of a general matrix can be calculated using the `solve` command. However, the matrix we want to invert is symmetric, and it is faster and more accurate to invert

a symmetric matrix using the *Cholesky decomposition*. R can invert a matrix using the Cholesky decomposition as follows:

```
xtx.inv = chol2inv(chol(xtx))
```

Verify that `xtx %*% xtx.inv` equals the identity matrix (to within roundoff error).

In addition to estimating the parameters $\boldsymbol{\beta}$, you also should estimate the sum square error SSE and coefficient of determination R^2. The sum square error is

$$\text{SSE} = \left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right)^T \left(\mathbf{y} - \mathbf{X}\hat{\boldsymbol{\beta}}\right). \tag{8.3}$$

In the notes, R^2 was estimated from *centered* variables. It is not necessary to center the variables *as long as you include the intercept as a predictor*. Therefore, your R function should *automatically insert* the intercept as a predictor. That is, you will give the function all the predictors except the intercept, and the function will insert the intercept among the predictors. You can generate a vector of 1s using the command `rep`, and then join this vector with the predictor matrix using the command `cbind`.

In general, R^2 can be calculated as

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}}, \tag{8.4}$$

where

$$\text{SST} = \sum_n \left(y_n - \hat{\mu}\right)^2. \tag{8.5}$$

Exercise 8.1. Write an R function that sets up the normal equations for solving (8.1), solves the normal equations to obtain the least squares estimates, and computes R^2. The function call based on the normal equations should be `regress.normal(y,x)`, where `y` and `x` are the appropriate vector and matrix of the regression model (8.1). The preamble of this function should be the following:

```
regress.normal = function(y,x,include.intercept=TRUE) {
#####################################################################
## DETERMINES THE LEAST SQUARES ESTIMATE OF B IN THE EQUATION Y = XB + E
## BASED ON THE NORMAL EQUATIONS
## INPUT:
##   Y[NTOT]:      N-DIMENSIONAL VECTOR OF PREDICTANDS
##   X[NTOT,MTOT]: N X M DIMENSIONAL MATRIX OF PREDICTORS
##   INCLUDE.INTERCEPT: INCLUDE THE INTERCEPT? (DEFAULT=TRUE)
## OUTPUT:
##   BHAT: M-DIMENSIONAL VECTOR OF ESTIMATES OF B
##   R2:  R-SQUARED
##   SSE: SUM SQUARE ERROR OF THE LEAST SQUARES PREDICTION
##   DOF: DEGREES OF FREEDOM OF THE SSE.
##   RES.SE: STANDARD ERROR OF THE RESIDUALS
#####################################################################
```

□

Exercise 8.2. Apply your function to the following random numbers:

```
> set.seed(1)
> ntot  = 20
> y     = rnorm(ntot); pred1 = rnorm(ntot); pred2 = rnorm(ntot)
> x     = cbind(pred1,pred2)
```

After running your function, print out $\hat{\beta}$, R^2, SSE, and dof produced by your function.

You should check your function by comparing with the R function `lm()`. To do this, apply the `lm` function as follows:

```
1  > xy.lm     = lm(y~x)
2  > summary(xy.lm)
3
4  Call:
5  lm(formula = y ~ x)
6
7  Residuals:
8      Min      1Q  Median      3Q     Max
9  -2.0455 -0.6782  0.2175  0.5049  1.4532
10
11 Coefficients:
12             Estimate Std. Error t value Pr(>|t|)
13 (Intercept)   0.1494     0.2072   0.721    0.481
14 xpred1       -0.1516     0.2504  -0.605    0.553
15 xpred2        0.2894     0.2695   1.074    0.298
16
17 Residual standard error: 0.9119 on 17 degrees of freedom
18 Multiple R-squared: 0.1078,        Adjusted R-squared: 0.002868
19 F-statistic: 1.027 on 2 and 17 DF,  p-value: 0.3791
```

In line 1, the function `lm` is called using the formula notation. Then, in line 2, the summary function is used to extract basic information about the linear model. For the purpose of the present homework, we are interested in only three parts of this summary: the coefficients, R^2, and residual standard error. The value of the coefficients are listed under `Coefficients: Estimate`, and are 0.1494, -0.1516, and 0.2894, corresponding to the three predictors: intercept, `pred1`, and `pred2`. These should match the values computed from `regress.normal` above. The R^2 is in the second to last line: 0.1078, and should agree with that calculated from `regress.normal`. Finally, the residual standard error derived from `regress.normal` should agree with the result from `lm` (line 17). □

Exercise 8.3. Apply your regression function to estimate the growth rate of atmospheric CO_2 concentration over the past half-century or so. State the growth rate in units of ppm/yr. The CO_2 concentration data can be downloaded as `co2_mm_mlo.txt` from the class website (which in turn was downloaded from http://www.esrl.noaa.gov/gmd/ccgg/trends/). This data set can be read into R as follows:

```
iyst      = 1960
iynd      = 2017

########################################
######## GET CO2 DATA
########################################
fdata = '/Users/delsole/data/indices/co2_mm_mlo.txt'
nskip = 72
col.names = c('year','month','date','average','interp','trend','#days')
co2.table = read.table(fdata,skip=nskip,col.names=col.names,na.strings=-99.99)

year.get = co2.table[,'year'] >= iyst & co2.table[,'year'] <= iynd
year.say = co2.table[year.get,'date']
month    = co2.table[year.get,'month']
co2      = co2.table[year.get,'average']
plot(year.say,co2,type="l",col="black",xlab='year',ylab='Parts Per Million')
```

You should change `fdata` to correspond to the data file on your computer. The resulting plot should reproduce the figure in the notes.

Unfortunately, there exists missing data. Therefore, *inside your R function*, you will have to strip out this missing data before applying the least squares method. I recommend including the following inside your R function:

```
ntot = length(y)
if ( length(x) %% ntot != 0) stop('x not dimensioned correctly')
mtot = length(x)/ntot
if ( ntot <= mtot ) stop('regression problem is not over-determined')

### STRIP MISSING DATA
dim(x)     = c(ntot,mtot)
is.missing = is.na(y) | is.na(rowSums(x))
x.good     = x[!is.missing,]
y.good     = y[!is.missing ]
nsamp      = sum(!is.missing)
```

Note that the correct sample size after missing data has been stripped is `nsamp`. This is important when you augment the predictor matrix $\mathbf{X}$ by a column of ones.

After running your function, print out $\hat{\beta}$, R^2, SSE, and dof produced by your function. You can check your calculations against the built-in R function `lm` (you will need to use the `na.action=na.omit` option to deal with missing data). □

Exercise 8.4. Compute the residuals of the regression equation. Make a plot of them, and *state the first 50 values* as a vector (e.g., `as.numeric(residuals[1:50])`. □

Exercise 8.5. Use your regression function to estimate the annual cycle of the CO_2 concentration. The annual cycle should be defined as the first two Fourier harmonics of the annual cycle (i.e., sin/cos function with periods of 12 months and 6 months). The predictor matrix for *just these harmonics* can be constructed in R as follows:

```
year = iyst + (1:ntot - 0.5)/12
year.shift = year - 1960
t = seq(year.shift)/12
nharm = 2
x = NULL
for ( n in 1:nharm) x = cbind(x,cos(2*pi*t*n),sin(2*pi*t*n))
colnames(x)  = c(paste(rep(c('cos','sin'),nharm),
  rep(1:nharm,each=2),sep=""))
```

State the five coefficients of this fit (i.e, the intercept, the 2 coefficients for the sin function, 2 coefficients for the cosine function). *Print out the resulting* coefficients, R^2, dof, and SSE. □

Exercise 8.6. Plot the residuals after the annual cycle has been removed. Superimpose a plot of the actual CO_2 data for comparison. To do this, you will need to add a constant term to the residuals to make them fit on the same figure; state what constant should you use and why. □

CHAPTER 9

EXERCISES FOR LINEAR REGRESSION: INFERENCE

In this homework, you will investigate whether Guam sea level has been rising over the last few decades. To do this, you need to download the following files from the class website:

file name	purpose
540.rlrdata	Guam sea level Data file
detrend.nino34.ascii.txt	SST data file
enso.index.v2.R	R code for reading SST
hw.LinRegInference.student.R	R code for reading sea level

Sea level data was downloaded from https://www.psmsl.org/data/obtaining/stations/540.php. The SST data was downloaded from https://www.cpc.ncep.noaa.gov/products/analysis_monitoring/ensostuff/detrend.nino34.ascii.txt. The first few lines of the sea level data file look like this:

```
1  1948.0417;  6977; 0;000
2  1948.1250;  6992; 0;000
3  1948.2083;  7023; 0;000
4  1948.2917;  7114; 0;000
```

The first column is the year-month is the decimal form year + (month-0.5)/12.0. The second column is the mean sea level value for the month, in mm, and the remaining numbers pertain to missing data. If there are no values to average, then the value for the month is set to -99999. However, you do not need to know these details. The R code reads the data file and extracts the data needed to do the homework. At the end of the code, you will see the following lines explaining the variables:

```
#### AT THIS POINT, THE FOLLOWING VARIABLES ARE DEFINED:
#### SEASON: CHARACTER DESCRIBING THE SEASON
####     (E.G., 'OND' FOR 'OCT-NOV-DEC')
#### IYST: START YEAR
#### IYND: END YEAR
#### NYRS: TOTAL NUMBER OF YEARS
#### Y[NYRS]: GUAM SEA LEVEL (mm) AVERAGED OVER THE SEASON
#### X.YEAR[NYRS]: YEAR
#### X.SST[NYRS]: NINO3.4 INDEX FOR THE SAME YEAR/SEASON
```

Note: the numbers derived in this homework will not match those in the notes because the data set here has been updated based on new observations (also, the units in the notes were incorrect).

Exercise 9.1. Make a plot of the tide gauge data for Guam during JFM, 1973-2006. Be sure that the axis labels are accurate (with units) and the title explains what is shown in the figure, including the station and the calendar period. Comment on whether the figure suggests that the tide gauge measurements have been increasing in the last few decades. □

Exercise 9.2. Use `regress.normal` to determine the trend of sea level. The trend is obtained by fitting sea level to the model

$$\text{sea level} = \beta_1 + \beta_2 \text{ year} + \epsilon_Y. \tag{9.1}$$

Print out the output of `regress.normal`. Report the trend in units of cm per year. □

Exercise 9.3. Augment `regress.normal` so that it calculates the t-statistic and associated p-value for each regression coefficient. Call these values `tval` and `pval` in the output list. Recall that if the model is defined as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}, \tag{9.2}$$

where $\mathbf{X}$ is $N \times M$, then the least squares estimate of $\boldsymbol{\beta}$ is

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}, \tag{9.3}$$

and the t-statistic for testing $\beta_j = 0$ is

$$t = \frac{\hat{\beta}_j}{s_e\sqrt{d_j}}, \tag{9.4}$$

where $s_e = \sqrt{SSE/(N-M)}$ and d_j is the j'th diagonal element of $\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$. The t-statistic has a t-distribution with $N - M$ degrees of freedom, which allows you to calculate the p-value. You are encouraged to check your calculations using `lm`.

□

Exercise 9.4. Is the trend significant at the 5% level? To answer this, test the hypothesis $\beta_2 = 0$ in model (9.1). Print out the output of `regress.normal`. Check to ensure that the t-value you calculate agrees with the t-value obtained from the R function `lm()`.

□

Exercise 9.5. Augment `regress.normal` to calculate the $(1 - \alpha)100\%$ confidence interval of each regression coefficient. Call the interval `bhat.ci` in the output list (it should be dimensioned `[MTOT,2]`). The confidence interval can be calculated from

$$\hat{\beta}_j \pm t_{\alpha/2,N-M} s_e \sqrt{d_j}. \tag{9.5}$$

Give the 95% confidence interval for the slope and intercept. Is this confidence interval consistent with your answer to the previous question? You are encouraged to check your calculations using `confint`. □

Exercise 9.6. Test whether sea level has a significant trend *after the effect of ENSO has been taken into account*. This test is performed by fitting the model

$$\text{sea level} = \beta_1 + \beta_2 * \text{year} + \beta_3 * \text{NINO3.4} + \epsilon \tag{9.6}$$

and testing the hypothesis $\beta_2 = 0$. Print out the output of `regress.normal`. Use this output to form a conclusion about whether sea level has a statistically significant trend. □

CHAPTER 10

EXERCISES FOR MODEL SELECTION

In this problem set you will estimate the annual cycle of the NINO3.4 index and apply model selection criteria to determine the number of terms in the Fourier series that should be fitted. The NINO3.4 index can be downloaded from

```
1 http://www.esrl.noaa.gov/psd/data/correlation/nina34.data
```

An R-code for reading this data set is `ModSel.student.R`, which can be downloaded from the class website. You will need to modify the variables `dir.rlib` and `dir.enso` to point to the directories containing your R-functions and ENSO time series, respectively. The variable `nharm` specifies the maximum number of Fourier harmonics to be considered. When you run this code, it should produce a figure showing the *raw* NINO3.4 time series (without annual cycle removed).

The exercises below will ask you to write a function to evaluate a variety of model selection criteria. Each exercise will focus on a single criterion, but in the end you should submit a single function that calculates all these criteria. This function should have the following name and preamble:

```
modsel.loo.simple = function(y,x,kmax=dim(x)[2]+1) {
#### EVALUATES VARIOUS MODEL SELECTION CRITERIA FOR THE MODEL
#### Y = X B + E
#### NOTE: THE CONSTANT (INTERCEPT) SHOULD NOT BE INCLUDED IN X;
####   IT IS INSERTED BY THIS PROGRAM
#
# INPUT:
#   Y[N]: PREDICTAND, WHERE N = NUMBER OF INDEPENDENT SAMPLES
#   X[N,K]: PREDICTOR MATRIX, K = NUMBER OF PREDICTORS
#      (NOT INCLUDING CONSTANT TERM)
#   KMAX: MAXIMUM NUMBER OF PREDICTORS (INCLUDING THE CONSTANT)
#      (DEFAULT: ALL PREDICTORS)
# OUTPUT:
#   SE.SQR[KMAX]: UNBIASED ERROR VARIANCE OF EACH MODEL 1, 2, ..., KMAX
#   NSE.SQR[KMAX]: NORMALIZED UNBIASED ERROR VARIANCE OF EACH MODEL
#   CVMSE[KMAX]: LOO CROSS-VALIDATED MEAN SQUARE ERROR FOR EACH MODEL
#   CVSTD[KMAX]: STANDARD ERROR OF LOO CROSS-VALIDATED SQUARED ERRORS
#   AIC[KMAX],AICC[KMAX],BIC[KMAX]: INFORMATION CRITERIA FOR EACH MODEL
#   IC.STD[KMAX]: STANDARD ERROR OF THE INFORMATION CRITERIA
```

Exercise 10.1. We want to fit the regression model

$$y_n = \beta_0 + \sum_{h=1}^{H} \left(c_h \cos\left(2\pi n h/12\right) + s_h \sin\left(2\pi n h/12\right) \right) \tag{10.1}$$

where y_n is the NINO3.4 index and $n = 1, 2, \ldots, N$. Show that this model can be written in the form

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}. \tag{10.2}$$

What is $\mathbf{X}$? What is $\boldsymbol{\beta}$? Hint: in R, the predictor matrix containing the Fourier harmonics can be generated as

```
x.pred = NULL
for ( nh in 1:nharm) x.pred =
  cbind(x.pred,cos(2*pi*nh*nino34.time),sin(2*pi*nh*nino34.time))
```

□

Exercise 10.2. Write a function that computes the unbiased error variance of the regression model (10.1) as a function of the number of harmonics H. The unbiased error variance is

$$s_e^2 = \frac{SSE}{N - M}. \tag{10.3}$$

The input to this function is the time series $\mathbf{y}$ and the predictor matrix $\mathbf{X}$ *without the constant term*. Internally, you can insert the constant function as follows:

```
ntot    = length(y)
x.const = cbind(rep(1,ntot),x); # ADD COLUMN OF ONES IN PREDICTOR MATRIX
```

If k is the number of columns of `x.const`, then the following commands will fit the regression model and produce fitted values and residuals.

```
for ( k in 1:kmax) {
        x.pred      = x.const[,1:k]
        xy.lm       = lm(y~x.pred-1)
        yhat        = fitted(xy.lm)
        yresid      = residuals(xy.lm)
}
```

Add to this code by computing s_e^2 and the normalized error variance

$$\frac{s_e^2}{\hat{\sigma}_Y^2}, \tag{10.4}$$

where $\hat{\sigma}_Y^2$ is the sample variance of $\mathbf{y}$. Make a plot of the normalized error variance as a function of the number of predictors. Where does the minimum occur? How many Fourier harmonics does this correspond to? How much variance is explained by the annual cycle? How much variance is unexplained by the annual cycle?

□

Exercise 10.3. Write a function that computes the leave-one-out cross validated mean square error. Hint: use "negative" arguments in R arrays to leave data out. For instance:

```
yhat = numeric(ntot)
for ( n in 1:ntot) {
        y.pred  = y[-n]
        x.pred  = x.const[-n,1:k]
        yhat[n] = sum(coefficients(lm(y.pred~x.pred-1)) * x.const[n,1:k])
}
```

State the values of the leave-one-out cross validated mean square error (after normalizing by the variance of y). □

Exercise 10.4. The above approach to leave-one-out cross validation is computationally intensive. A faster numerical method is explained in exercise 11.2. You do not have to prove exercise 11.2, but you will use the result. Write a function that computes the leave-one-out cross validated mean square error based on the numerical approach discussed in exercise 11.2. Verify that the result is identical to what you found in the previous exercise. Discuss how much faster the new function is relative to the old.

□

Exercise 10.5. Compute the standard error of the cross validated mean square error. Indicate the standard error as error bars on the figure produced in the previous problem. Error bars can be plotted using the `arrows` command as follows:

```
modsel.list = modsel.loo.simple(nino34.ts,x.pred)
x0 = 0:(2*nharm)
y0 = modsel.list$cvmse - modsel.list$cvstd
y1 = modsel.list$cvmse + modsel.list$cvstd
yrange = range(y0,y1)
plot(x0,modsel.list$cvmse,type='b',pch=19,xlab='number of Fourier terms',
  ylab='unbiased error variance',ylim=yrange)
arrows(x0,y0,x0,y1,length=0.1,angle=90,code=3,lwd=2)
```

What model does the "one-standard-deviation rule" select?

□

Exercise 10.6. Write a function to evaluate AIC, BIC, and AICC. Plot these values as a function of the number of predictors in the annual cycle model. State the numerical values of these quantities. What model does these criteria select?

□

Exercise 10.7. Write a function that computes the standard error of the information criteria. Show the standard error in the plot generated in the previous exercise. What model does the "one-standard-error" rule select? □

Exercise 10.8. Based on all of the above results, you should have found that either 1 or 2 harmonics should be used to fit the annual cycle of NINO3.4. Let us decide to use 2 harmonics (so this means you have 5 predictors: 1 intercept term, cos/sine for the first harmonic, and cos/sine for the second harmonic). Now subtract this annual cycle from the NINO3.4 index and plot the residual. This is called the NINO3.4 *anomaly*. You should recognize the 1982, 1998, and 2015 El Niño events.

□

Exercise 10.9 (Apparently Different Information Criteria)**.** Different definitions of AIC, AICC, BIC appear in the literature. However, these definitions differ by either a factor of N or an additive constant, neither of which affect the *location* of the minimum value. For example, some papers define AIC_c as

$$AIC_c = N \log\left(\frac{SSE}{N}\right) + \frac{2KN}{N-K-1}, \tag{10.5}$$

where K is the total number of "parameters" in the regression model *including the error variance* σ_ϵ^2; that is, $K = M + 1$ in terms of the model defined in (**??**). Show that this definition differs from the AIC_c defined (10.28) by a constant and therefore the two expressions (10.28) and (10.5) are equivalent model selection criteria. □

Exercise 10.10 (Shortcut for Leave-One-Out Cross Validation)**.** Show that the leave-one-out cross validated error of the regression model (**??**) can be written as

$$\epsilon_k^{LOO} = y_k - \mathbf{x}_k \boldsymbol{\beta}_k = \frac{y_k - \mathbf{x}_k \hat{\boldsymbol{\beta}}}{1 - \alpha_k}, \tag{10.6}$$

where $\hat{\boldsymbol{\beta}}$ is the least-squares estimate of $\boldsymbol{\beta}$ using all data and α_k is the scalar

$$\alpha_k = \mathbf{x}_k \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{x}_k^T. \tag{10.7}$$

Interestingly, the cross validated error $y_n - \mathbf{x}_n \boldsymbol{\beta}_n$ is merely an *inflated* version of the residual of the least squares model derived from the entire data set $y_n - \mathbf{x}_n \hat{\boldsymbol{\beta}}$. Equation (10.6) also shows that the leave-one-out cross validated mean square error can be determined from the traditional least squares solution, without re-fitting the model N separate times. Although the method requires computing the inverse of $\mathbf{X}^T\mathbf{X}$ once, this inverse is already available since it is required to compute $\hat{\boldsymbol{\beta}}$. Hint: note that

$$\boldsymbol{\beta}_k = \left(\mathbf{X}^T\mathbf{X} - \mathbf{x}_k^T\mathbf{x}_k\right)^{-1}\left(\mathbf{X}^T\mathbf{y} - \mathbf{x}_k^T y_k\right), \tag{10.8}$$

and apply the Sherman-Morrison-Woodbury formula.

□

CHAPTER 11

PITFALLS OF STATISTICAL INFERENCE

Exercise 11.1 (Selecting Predictors Using Screening)**.** A forecaster wants to predict hurricanes for the season. To do this, the forecaster evaluates the correlation between hurricane counts and 50 variables around the globe that may be relevant. The correlations are computed using 30 years of data. Based on this analysis, the forecaster selects the predictor with the maximum correlation, which has a correlation of 0.6 with hurricane counts, and uses that predictor to make a forecast. Is the predictor statistically significant at the 5% level? For your null hypothesis, you may assume the 50 variables are independent and normally distributed. Does your conclusion change if the variables are not independent? □

Exercise 11.2 (Selecting the Best Forecasts)**.** A forecaster has nine forecasts and would like to combine them to make a single, superior forecast. The forecaster decides to select the four forecasts that are most correlated with observations from a 20-year historical record, and then average those forecasts to make a prediction. When the four best forecasts are averaged, the correlation between the mean 4-member forecast and observations is 0.55. For reference, the 5% significance level for a 20-year forecast is about 0.45. Since the actual correlation exceeds the threshold, the forecaster claims that the 4-member mean forecast has a statistically significant correlation. What is "wrong" with this argument? What is the experimentwise error rate for this procedure? Plot a histogram of the correlation skill under the null hypothesis of independent variables.

Hint: generate random numbers to simulate the above scenario and perform the same steps as the forecaster. Repeat this procedure numerous times to estimate an empirical distribution of the correlation under the null hypothesis of no-skill (all variables are in-

dependent). You can use the `cor` function to determine the correlation between 1 variable and 9 others (e.g., if `obs` is 20 elements long, and `frcs` is a 20×9 matrix, then `as.numeric(cor(frcs,obs))` gives 9 correlations). Also, the `order` command allows you to identify the forecasts with the four largest correlations.

□

Exercise 11.3 (Discovery of a new Predictor)**.** An investigator believes that a particular index, called *DOOM*, is useful for predicting JFM temperature. The 50-year correlation between JFM temperature and *DOOM* is 0.2 in December and 0.1 in November, but 0.3 in October. Since the 5% significance threshold for a correlation coefficient based on 50 independent samples is about 0.28, the investigator promptly publishes a paper proclaiming that the October value of *DOOM* is a good predictor of JFM temperature. Has the significance test been applied properly? If not, why not? Is the predictor significant when screening is taken into account? If not, how large would the correlation need to be to ensure a true 5% significance? □

CHAPTER 12

EXERCISES FOR PRINCIPAL COMPONENT ANALYSIS

In this problem set you will calculate the principal components of December mean sea surface temperature (SST). To do this, you will need to download some R code, data sets, and install appropriate libraries. This homework set will walk you through these steps.

First, you need prepare R to make plots of spatial fields. The libraries for making spatial plots are not automatically included in R, so you must install them. The libraries you need are the following:

- `maps`
- `fields`
- `rworldmap`
- `ncdf4`
- `chron`

On the MAC, it is very easy to install libraries:

1. Click "Packages & Data."
2. Click "Package Installer."

3. Click "Get list" (if a list is not already there).
4. Click the appropriate library (see above list).

Next, you need to download the data. The data file is `sst.mnmean.v4.nc`.

Finally, you should download two R programs

- `plot_latlon_v4.R` for plotting spatial fields
- `sst.pca.student.R`, which reads this data and plots the SST for a single year.

As usual, before running `sst.pca.student.R`, you should modify the directory names `dir.data` and `dir.lib` to correspond to the directories where you save data and R functions, respectively. If you have successfully followed all of the above directions, then `sst.pca.student.R` will run with no errors or warnings and will generate a plot showing SST in December 1983.

To calculate a principal component, you must first calculate the climatological mean, and then subtract it out from all fields. Remember that R is faster for vector calculations than for `for` loops. Therefore, we want to calculate the climatological mean using vector calculations. A trick for doing this is to *re-shape* the data array so that the command `rowMeans` can be used. When converting from vector to matrix or vice versa, R assumes the elements are stored *column-wise*. You can get the idea from the following R session:

```
> dum = letters[1:12]
> print(dum)
 [1] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j" "k" "l"
> dim(dum) = c(3,4)
> print(dum)
     [,1] [,2] [,3] [,4]
[1,] "a"  "d"  "g"  "j"
[2,] "b"  "e"  "h"  "k"
[3,] "c"  "f"  "i"  "l"
```

Taking advantage of this convention, the monthly anomalies can be calculated efficiently as follows:

```
dim(sst) = c(nlon*nlat,nyrs)
sst.clim = rowMeans(sst)
sst.anom = sst.data - sst.clim
```

In addition to subtracting out the mean, you also need to create weights that take into account the fact that the area of a grid point is proportional to the cosine of latitude. A concise way of generating the appropriate vector of weights is the following.

```
######## CREATE AREA WEIGHTING VECTOR #############
weight  = rep(sqrt(cos(lat*pi/180)),each=nlon)

#### CONSTRUCT WEIGHTED ANOMALY DATA
sst.anom.weight = sst.anom  * weight
```

Make sure you understand what both commands do and why!

Note that the array `sst.anom.weight` is shaped as `[space, time]`, whereas the notes assumes the transpose. There is a good reason why I define the data array the way that I did, but let's not get into that. There are two ways to deal with this inconsistency: take the transpose of the data array before performing PCA, or perform SVD on the data array but switch the labels $\mathbf{U}$ and $\mathbf{V}$. The second approach is taken in this homework.

Another pesky issue is that SST is defined only over the ocean– the grid points over land have no SST value. The undefined points need to be "stripped away" before computing the EOFs. The basic idea is to construct a new (smaller) data matrix in which each column contains *only* the defined ocean points, apply PCA to the resulting data matrix, then "fill in" the undefined land points at the end of the calculation. Since this is a programming trick and not statistics, I'll just give you the answer for a generic data set `data.array`:

```
# IDENTIFY MISSING OR 0-WEIGHTED DATA
dim(data.array) = c(nlon*nlat,ntot)
lbad = is.na(rowSums(data.array)) | weight == 0

# COMPRESS DATA BY ELIMINATING MISSING GRID POINTS
data.array = data.array[!lbad,]
ndef = sum(!lbad)

# COMPUTE SVD
mmin = min(ndef,ntot)
data.svd = svd(data.array,nu=mmin,nv=mmin)

# FILL IN MISSING POINTS IN EOF
eof  = array(NA,dim=c(nlon*nlat,mmin))
eof [!lbad,] = data.svd$u
```

Exercise 12.1. Note that the data array `sst` is shaped as `[space, time]`, whereas the lecture notes assumed the data array is `[time, space]`. This means that if you take the SVD of `sst`, the identification of $\mathbf{U}$ and $\mathbf{V}$ differs from that of the notes. Starting from the SVD of the weighted data

$$\underset{M\times M}{\mathbf{W}}\quad\underset{M\times N}{\mathbf{Y}'} \;=\; \underset{M\times N}{\dot{\mathbf{U}}}\quad\underset{N\times N}{\dot{\mathbf{S}}}\quad\underset{N\times N,}{\dot{\mathbf{V}}^T} \tag{12.1}$$

where $\mathbf{W}$ is a diagonal matrix whose diagonal elements equal the square root of the cosine of latitude, state the equations for the scaled EOFs $\mathbf{E}$ and scaled PCs $\mathbf{F}$ such that

$$\mathbf{Y}' = \mathbf{E}\mathbf{F}^T. \tag{12.2}$$

Show mathematically that the scaled PCs have unit variance. □

Exercise 12.2. Based on the above hints, write a function called `eof.latlon.simple` that performs principal component analysis on a data array formatted as `[space, time]`. The preamble of this function should be the following:

```
eof.latlon.simple <- function(lon, lat, data.array, neof=30) {

# COMPUTE PRINCIPAL COMPONENTS OF A DATA ARRAY.

# INPUT:
# LON[NLON]: VECTOR SPECIFYING LONGITUDES
# LAT[NLAT]: VECTOR SPECIFYING LATITUDES
# DATA.ARRAY[NLON,NLAT,NTOT] OR DATA.ARRAY[NLON*NLAT,NTOT]:
#   THE DATA ARRAY IN [SPACE1,SPACE2,TIME] OR [SPACE,TIME] FORMAT
# NEOF = NUMBER OF SPATIAL EOFS (WITH MASK) TO BE OUTPUTTED.  DEFAULT = 30

# OUTPUT LIST:
# $EOF[NLON*NLAT,NEOF]: THE FIRST NEOF SCALED EOFS
# $PC[NTOT,NEOF]: THE PCS, NORMALIZED TO UNIT VARIANCE
# $FEXPVAR: FRACTION OF EXPLAINED VARIANCE FOR EACH EOF (ALL OF THEM).
# $FEXPVAR.CI: CONFIDENCE INTERVALS FOR FRACTION OF EXPLAINED VARIANCE.
# $EOFI[NLON*NLAT,NEOF]: PSEUDO INVERSE OF EOF (I.E.,T(EOFI) %*% EOF = I)
# $NEOF: MINIMUM OF (NEOF IN ARGUMENT LIST, RANK OF DATA.ARRAY)
```

This function should subtract out the time mean, apply an area weighting, allow for undefined points, and produce a list containing the quantities indicated above. □

Exercise 12.3. Apply your function to compute the principal components of December SST. Show a plot of the spatial structure and time series of the leading component. Also, show a plot of the explained variances of the leading components along with confidence intervals. These plots should be similar to those displayed in the notes. □

Exercise 12.4. Consider the linear combination

$$r = a_1 X_1 + a_2 X_2 + \cdots + a_M X_M = \mathbf{a}^T \mathbf{x}, \tag{12.3}$$

where $a_1, \ldots, a_M$ are constants and $X_1, \ldots, X_M$ are random variables. A *standardized linear combination* is one in which the coefficients $\mathbf{a}$ are constrained such that $\mathbf{a}^T\mathbf{a} = 1$.

(a) Show that $\mathbf{b} = \mathbf{a}/\sqrt{\mathbf{a}^T\mathbf{a}}$ satisfies $\mathbf{b}^T\mathbf{b} = 1$ for any choice of $\mathbf{a} \neq \mathbf{0}$. It follows from this that

$$\mathbf{b}^T\mathbf{x} = \frac{\mathbf{a}^T\mathbf{x}}{\sqrt{\mathbf{a}^T\mathbf{a}}} \tag{12.4}$$

is a standardized linear combination for *any* choice of $\mathbf{a} \neq \mathbf{0}$.

(b) Write an expression for the variance of r in terms of the covariance matrix of $\mathbf{x}$, $\mathbf{\Sigma}$.

(c) Consider the new variable

$$\tilde{\mathbf{a}} = \mathbf{U}^T\mathbf{a}, \tag{12.5}$$

where $\mathbf{U}$ is the orthogonal eigenvector matrix of $\mathbf{\Sigma}$ (i.e, $\mathbf{\Sigma} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$, where $\mathbf{U}^T\mathbf{U} = \mathbf{I}$). Write the variance of r in terms of $\tilde{\mathbf{a}}$. Something "special" happens under this transformation. What is it? Write out a *simple* expression for the variance of r without using vector notation (explicitly write out sums and products).

(d) Show that maximum variance of r is λ_1 (i.e., the largest eigenvalue of $\mathbf{\Sigma}$). What vector $\tilde{\mathbf{a}}$ gives this maximum? What vector $\mathbf{a}$ gives this maximum?

(e) Suppose the covariance matrix is estimated from the sample covariance matrix of the *centered* data matrix $\mathbf{X}$:

$$\hat{\mathbf{\Sigma}} = \frac{1}{N-1}\mathbf{X}^T\mathbf{X}. \tag{12.6}$$

How are the eigenvalues and eigenvectors of $\hat{\mathbf{\Sigma}}$ related to the singular values and singular vectors of $\mathbf{X}$?

(f) Use these results to give an alternative description of the leading principal component. That is, instead of describing the leading component as something that best approximates the data, describe it in terms of standard linear combinations.

□

CHAPTER 13

COMPUTATIONAL EXERCISES ON FIELD SIGNIFICANCE

In this homework set, you will perform a series of field significance tests. The first step of this homework is to obtain the following files from the class website. It is good practice to put all of your data in a single directory (say "data"), and all your auxiliary programs into a single directory that can be referred to in future R programs.

`fieldsig.student.R`	main program: reads data and computes EOFs
`cca.data.clim763.DJF.RData`	data (will be used in future homeworks)
`eof.latlon.R`	auxiliary file: calculate EOFs
`plot_latlon_v4.R`	auxiliary file: plot spatial maps
`gev.R`	auxiliary file: function to solve eigenvalue problem
`index.climate.v2.R`	auxiliary file: function to compute NINO3.4 index

Your starting point is the R program `fieldsig.student.R`, which does the following things things:

- It reads the data file `cca.data.clim763.DJF.RData` and loads them in the arrays `tem.data` and `sst.data`. Inside `fieldsig.student.R`, you need to modify the variable `dir.data` to correspond to the directory to which you downloaded the data.

- It reads the auxiliary programs (e.g., `eof.latlon.R` and `plot_latlon_v4.R`). You will need to modify `dir.Rlib` to correspond to the directory to which you downloaded the auxiliary programs.
- It computes the nino3.4 index from `sst.data`.
- It also computes the EOFs of U.S. temperature using `eof.latlon`, which creates the list variable `tem.eof`. The EOFs are `tem.eof$eof[space,neof]`, and the pc time series are `tem.eof$pc[time,neof]`.
- It plots the leading EOF and PC of North American Temperature and SST. The resulting figure should look like fig. 13.1.

Note: temperature over the ocean is set to `NA`. Also, temperature is read as an array with dimension `nlon, nlat, time`, but eventually is "reshaped" into an array with dimension `nlon*nlat, nyrs`. If you do not understand what this means, please see me!

The purpose of this homework is to test the hypothesis that tropical Pacific SSTs influence North American temperature in DJF. You will begin by looking at the correlation map between temperature and NINO3.4. This is very convenient to plot in R. First, we compute the correlation between NINO3.4 and every grid point of temperature using the command

```
cor.map = cor(t(tem.data),nino34)
```

The R-function `cor` assumes data matrices are in the form TIME x SPACE, so we must take the transpose of `tem.data` (which is an array with dimensions SPACE x TIME) to use `cor`. To plot this, we then use the command

```
plot_latlon_v4(lon,lat,cor.map).
```

To "maskout" insignificant correlations, we use the commands

```
cor.crit = 2/sqrt(nyrs)
cor.map[abs(cor.map)<cor.crit] = NA
plot_latlon_v4(lon,lat,cor.map)
```

The resulting plot should look like fig. 13.1 in the notes.

Assignment

Exercise 13.1. Write an R code that applies the Livezey-Chen field significance test. The preamble of the function should be the following:

```
livezey.chen = function(lon,lat,xdata,ydata,ntrials=1000,alpha=0.05) {
### PERFORMS THE LIVEZEY-CHEN FIELD SIGNIFICANCE TEST
# INPUT:
#   LON[NLON]: LONGITUDE OF THE FIELD DATA
#   LAT[NLAT]: LATITUDE OF THE FIELD DATA
#   XDATA[NLON,NLAT,NTOT]: DATA ARRAY FOR THE FIELD
#   YDATA[NTOT]: REFERENCE TIME SERIES
#   NTRIALS: NUMBER OF MONTE CARLO TRIALS (DEFAULT = 10000)
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT:
#   HISTOGRAM OF AREAS WITH SIGNIFICANT CORRELATIONS
#   LIST:
#     $CRIT.NULL: SIGNIFICANCE THRESHOLD FROM LIVEZEY-CHEN TEST
#     $SIG.AREA.OBS: PERCENTAGE AREA WITH SIGNIFICANT CORRELATIONS
```

Apply the Livezey-Chen field significance test to test whether NINO3.4 is related to DJF temperature over North America. Make a histogram of the Monte Carlo results.

Be sure to express the answer in terms of percentage area with significant correlations. Suggestion: create another field that gives the fractional area at each grid point, then sum over those points. For instance:

```
area.frac    = rep(cos(lat*pi/180),each=nlon)
lgood        = !is.na(xdata[,1])
area.frac[!lgood] = NA
area.frac    = area.frac/sum(area.frac,na.rm=TRUE)

cor.map      = as.numeric(cor(t(xdata),ydata))
sig.area.obs = sum(area.frac[abs(cor.map) > 2/sqrt(ntot)],na.rm=TRUE)*100
```

You should obtain a plot similar to fig. 13.4 of the lecture notes. Make a conclusion regarding whether the field is significant. □

Exercise 13.2. Write an R code that applies the permutation field significance test. The preamble of the function should be the following:

```
fieldsig.permutation = function(lon,lat,xdata,ydata,ntrials=1000,alpha=0.05) {
### PERFORMS PERMUTATION FIELD SIGNIFICANCE TEST
# INPUT:
#   LON[NLON]: LONGITUDE OF THE FIELD DATA
#   LAT[NLAT]: LATITUDE OF THE FIELD DATA
#   XDATA[NLON,NLAT,NTOT]: DATA ARRAY FOR THE FIELD
#   YDATA[NTOT]: REFERENCE TIME SERIES
#   NTRIALS: NUMBER OF TRIALS (DEFAULT = 10000)
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT:
#   HISTOGRAM OF AREAS WITH SIGNIFICANT CORRELATIONS
#   LIST:
#     $CRIT.NULL: SIGNIFICANCE THRESHOLD FROM LIVEZEY-CHEN TEST
#     $SIG.AREA.OBS: PERCENTAGE AREA WITH SIGNIFICANT CORRELATIONS
```

The `sample` function is very useful for generating a permutation. For instance, `sample(N)` generates a permutation of the integers 1-N. To ensure there are no accidental matches, the following loop may be helpful

```
temp  = TRUE
while(temp) {
        nperm = sample(nyrs)
        temp  = any(nperm == 1:nyrs)
}
```

Answer the same questions as the previous exercise. How does this result compare to that from the Livezey-Chen test? □

Exercise 13.3. Write a function that evaluates Mutual Information Criterion (MIC) for the field significance problem. MIC is a function of R-square, which can be computed from R functions as follows

```
r.square = summary(lm(y~x))$r.square
```

Show a plot of MIC versus the number of EOFs of U.S. temperature and identify the minimum. The figure should be similar to fig. 13.5 in the notes. □

Exercise 13.4. Write a code to evaluate the significance level for MIC. Include this in your function for MIC. Plot the significance level. □

Exercise 13.5. Write an R code that computes the False Discovery Rate. The preamble of this function should be the following:

```
fdr.fieldsig = function(y,x,fdr=0.1) {
## APPLIES FALSE DISCOVERY RATE TO A CORRELATION MAP
## INPUT:
#     Y[NTOT]: REFERENCE TIME SERIES
#     X[NTOT,MTOT]: FIELD TIME SERIES
#     FDR: FALSE DISCOVERY RATE
## OUTPUT:
#     LREJECT[MTOT]: LOGICAL VECTOR INDICATING REJECTIONS
```

Make a plot showing where the FDR implies rejection of the null hypothesis. The plot should be similar to fig. 12.7 of the notes. □

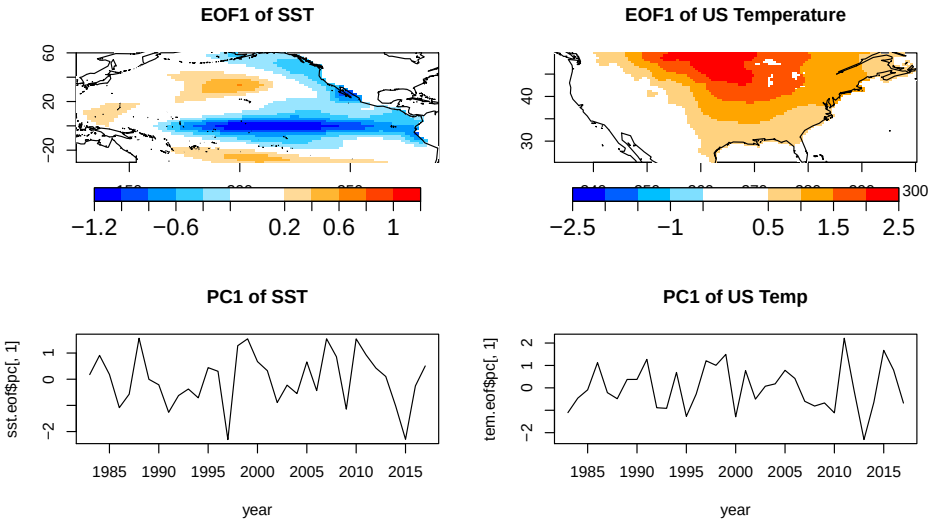


Figure 13.1 Leading EOF of DJF North American temperature during 1982-2017 (top) and correlation between DJF NINO34 index and DJF temperature during 1982-2017 (bottom).

CHAPTER 14

EXERCISES FOR MULTIVARIATE LINEAR REGRESSION

Exercise 14.1. It is natural to wonder whether a "better" estimate of $\mathbf{B}$ can be obtained by weighting the SSE toward certain variables. To explore this possibility, suppose w_s is the weight for the s'th spatial location. Then, the SSE in (14.20) can be modified as

$$SSE = \sum_{s=1}^{S} \sum_{n=1}^{N} w_s \left(Y_{ns} - \sum_{m=1}^{M} X_{nm} B_{ms} \right)^2 . \tag{14.1}$$

More generally, one can specify a positive-definite weight matrix $\mathbf{W}$ and define

$$SSE = \sum_{s'=1}^{S} \sum_{s=1}^{S} \sum_{n=1}^{N} W_{ss'} \left(Y_{ns} - \sum_{m=1}^{M} X_{nm} B_{ms} \right) \left(Y_{ns'} - \sum_{m=1}^{M} X_{nm} B_{ms'} \right), \tag{14.2}$$

which can be written equivalently as

$$SSE_W = \mathrm{tr} \left[(\mathbf{Y} - \mathbf{XB}) \, \mathbf{W} \, (\mathbf{Y} - \mathbf{XB})^T \right]. \tag{14.3}$$

Show that the matrix $\mathbf{B}$ that minimizes SSE_W is independent of $\mathbf{W}$ and equal to the least squares estimate (14.25). This means that there is nothing to gain by weighting different spatial locations differently. Explain why this is so. □

Computational Exercise for Multivariate Linear Regression

In this homework, you will derive a multivariate linear prediction model for Sea Surface Temperature.

Getting the Data The data set will be the ERSSTv5,[2] which is on the COLA servers.[3] However, you might prefer to download the data set directly on your laptop (I would recommend this). To download this data using R, grab the following files on the class website:

```
download.ersstv5.R
download_ftp_file.R
create_directory.R
```

Then type `source('download.ersstv5.R')`. The code will take a few minutes to run and will show the file names being downloaded. It also will generate a warning message about creating directories, which can be ignored.

Reading the Data To make this assignment easier, I have written an R code called `mlinreg.student.R` that does the following things:

1. read the NetCDF files for ERSSTv5
2. load the data into the array `sst.full[space,month,year]`
3. regress out the mean and linear trend from each grid point
4. define the Pacific Ocean domain to be between 30°S and 60°N
5. compute the EOFs of monthly SST in the prescribed domain

This code also requires the following files from the class website:

```
eof.latlon.R
mask.define.R
```

It is good practice to put all of your data in a single directory (say "data"), and all your auxiliary programs into a single directory that can be referred to in future R programs. You should change `dir.Rlib` to point to the directory of your R functions, and `dir.ersst` to point to the directory of your ERSSTv5 data.

After running `mlinreg.student.R`, you should see a plot of the leading EOF and PC. If the code crashes or gives an error message, please see me. Otherwise, you are ready! After the code runs to completion, you should have the following data arrays available.

```
# 1) THE GRIDDED DATA SET IS IN SST.FULL[SPACE,MONTH,YEAR]
# 2) EOFS ARE IN EOF.LIST
#   (E.G., EOF.LIST$EOF[SPACE,NEOF], EOF.LIST$PC[TIME,NEOF])
# 3) LAT/LON IS LON.SST, LAT.SST
# 4) MON.INIT IS THE CALENDAR MONTH OF THE INITAL CONDITION
# 5) MON.TARG IS THE CALENDAR MONTH OF THE TARGET
```

[2] https://www.ncdc.noaa.gov/data-access/marineocean-data/extended-reconstructed-sea-surface-temperature-ersst-v5

[3] /shared/obs/sst/ncdc/ersst-v5

Temperature over the ocean is set to NA. Also, temperature is read as an array with dimension nlon, nlat, time, but eventually is "reshaped" into an array with dimension nlon*nlat, 12, nyrs. If you do not understand what this means, please see me!

Exercise 14.2. Write a R function to compute MIC given two data arrays. The function also should compute the 5% significance threshold for MIC. The preamble should be as follows

```
mic.gaussian = function(x,y,equal.dim=TRUE,alpha=0.05) {
### THIS FUNCTION COMPUTES CORRECTED MUTUAL INFORMATION CRITERION (MIC)
###     FOR Y = XB + E
### INPUT:
##    X[NSAMP,XDIM]: X DATA ARRAY, OFTEN FORMATTED AS [TIME,EOF]
##    Y[NSAMP,YDIM]: Y DATA ARRAY, OFTEN FORMATTED AS [TIME,EOF]
##    EQUAL.DIM: LOGICAL INDICATING WHETHER
##      (TRUE) EQUAL NUMBER OF X'S, Y'S CHOSEN: MIC[MIN(XDIM,YDIM)];
##      (FALSE) MIC FOR ALL TRUNCATIONS ARE COMPUTED: MIC[XDIM,YDIM]
### OUTPUT: LIST (DIMENSIONS DEPEND ON EQUAL.DIM)
##    $MIC: MIC VALUES
##    $PENALTY: THE PENALTY TERM IN MIC
##    $CRIT: SIGNIFICANCE THRESHOLD OF MIC
```

□

Exercise 14.3. Consider the case of predicting December SST based on June SST. Calculate the MIC for EOF truncations at least for 1-25. The MIC values should be the following:

```
> micc
 [1]   -0.93434963   -1.23487398   -1.61938679   -1.67394748
 [5]   -1.84103698   -1.50387241   -1.19900423   -0.69182271
 [9]    0.05407005    0.33422799    1.48966123    2.50855073
[13]    4.09619446    5.50819524    7.95854904   11.03580560
[17]   14.07453324   18.15159799   23.34799814   30.02343607
[21]   38.16355309   48.57419513   62.29384427   79.60195564
[25]  101.97693930  131.36865590  172.06833168  227.00353490
[29]  312.44543534  452.13622542  710.10017386 1323.74697742
[33] 4435.93164492            NA
```

Plot the MIC values and indicate the minimum, which should be 5 EOFs. Next, do the same thing, except for predicting December SST based on September initial condition. Print the values of MIC (as above). Decide how many EOFs should be used and state your answer. □

Exercise 14.4. Use multivariate regression to make a prediction of December SSTs based on June SSTs using 5 EOFs. Submit your code for doing this. Make sure your results are consistent with the notes. Then, make a prediction for December SSTs using September SSTs and 5 EOFs. Submit plots of your predictions. □

CHAPTER 15

EXERCISES FOR CANONICAL CORRELATION ANALYSIS

`cca.student.R`	main program
`cca.data.clim763.DJF.RData`	SST and N. American temperature for DJF
`eof.latlon.R`	auxiliary file: calculate EOFs
`plot_latlon_v4.R`	auxiliary file: plot spatial maps

In this homework you will write an R function that performs Canonical Correlation Analysis. You will need to download data and a few R programs. These files are summarized in the above table. The R code `cca.student.R` reads the data file `cca.data.clim763.DJF.RData` and computes the EOFs of SST and U.S. temperature, and plots out the leading EOF and PC time series. In a previous homework you investigated the relation between ENSO and U.S. temperature using the concept of field significance. However, that analysis assumed you knew the NINO3.4 index. Here, you will *derive* an index for predicting U.S. temperature based on SST.

The following exercises break up CCA into discrete steps. However, in the end, you should submit a single function that performs all the calculations. The preamble of this function should be the following:

```
cca.pca = function(x.pca,y.pca,tx=NULL,ty=NULL) {
#########################################################
## PERFORMS CANONICAL CORRELATION ANALYSIS ON X AND Y.
## X AND Y ARE ASSUMED TO BE IN THE FOLLOWING FORMS:
## X = FX %*% EX^T   ### AND ### Y = FY %*% EY^T
## WHERE FX AND FY HAVE COVARIANCE MATRICES = I
## FOR EXAMPLE: FROM PRINCIPAL COMPONENT ANALYSIS
## IF TX OR TY = NULL, THEN BOTH (TX,TY) SELECTED USING MIC
## INPUT:
#    X.PCA: LIST OUTPUT FROM EOF.LATLON[X-DATA]
#    Y.PCA: LIST OUTPUT FROM EOF.LATLON[Y-DATA]
#    TX: TRUNCATION FOR X (TX <= MX); IF NULL, TX IS SELECTED
#    TY: TRUNCATION FOR Y (TY <= MY); IF NULL, TY IS SELECTED
## OUTPUT LIST:
#    MIC[MX,MY]: MUTUAL INFORMATION CRITERION
#    NMIN[1,2]: VALUES OF TX,TY THAT MINIMIZES MIC
#    CAN.COR[MIN(TX,TY)]: CANONICAL CORRELATIONS
#    RX[NTOT,MIN(TX,TY)]: CANONICAL VARIATES FOR X
#    RY[NTOT,MIN(TX,TY)]: CANONICAL VARIATES FOR Y
#    PX[SX  ,MIN(TX,TY)]: CANONICAL LOADING VECTORS FOR X
#    PY[SY  ,MIN(TX,TY)]: CANONICAL LOADING VECTORS FOR Y
#    QX.TILDE[TX,MIN(TX,TY)]: WEIGHTING VECTORS FOR X-FEATURES
#    QY.TILDE[TY,MIN(TX,TY)]: WEIGHTING VECTORS FOR Y-FEATURES
#    TX, TY: SELECTED VALUES OF TX AND TY
#########################################################
```

Following the notes, the data sets are assumed to have been decomposed into the form

$$\underset{N \times S_X}{\mathbf{X}} = \underset{N \times M_X}{\mathbf{F}_X} \quad \underset{M_X \times S_X,}{\mathbf{E}_X^T} \tag{15.1}$$

and

$$\underset{N \times S_Y}{\mathbf{Y}} = \underset{N \times M_Y}{\mathbf{F}_Y} \quad \underset{M_Y \times S_Y,}{\mathbf{E}_Y^T} \tag{15.2}$$

where

$$\frac{1}{N-1}\mathbf{F}_X^T\mathbf{F}_X = \mathbf{I} \quad \text{and} \quad \frac{1}{N-1}\mathbf{F}_Y^T\mathbf{F}_Y = \mathbf{I}. \tag{15.3}$$

This decomposition is accomplished in `cca.student.R` using principal component analysis. You will be performing CCA on the time series matrices $\mathbf{F}_X$ and $\mathbf{F}_Y$.

Exercise 15.1. In a previous homework you wrote a function that computed Mutual Information Criterion (MIC). Augment that function to compute MIC for all truncations T_X and T_Y, up to some maximum number (what *is* the maximum number?). As a reminder, MIC is defined as

$$MIC = \sum_i \log(1-\hat{\rho}_i^2) + \mathbb{P}, \tag{15.4}$$

where $\hat{\rho}_i$ is the i'th sample canonical correlation and

$$\mathbb{P} = (N+1)\left(\frac{T_X+T_Y}{N-T_X-T_Y-2} - \frac{T_X}{N-T_X-2} - \frac{T_Y}{N-T_Y-2}\right). \tag{15.5}$$

Plot MIC for a range of values and identify the value of T_X and T_Y that minimizes MIC. Print out the values for the first 5 rows and columns (i.e., print `mic[1:5,1:5]`). These results should match the example in the notes. Where is the minimum value, and what is it? The location of the minimum can be found using the following R commands:

```
nmin    = which(mic == min(micc,na.rm=TRUE),arr.ind=TRUE)
tx      = nmin[1]
ty      = nmin[2]
```

□

Exercise 15.2. Compute all canonical correlations for the optimum choice of T_X and T_Y. State the values. These values should be consistent with those in the notes. □

Exercise 15.3. Write a function that computes canonical variates. Compute the leading canonical variates between SST and U.S. temperature using the same truncation parameters as above. Make a plot that shows the leading canonical variate for the two data sets. Verify that the sample covariance matrices of the canonical variates equals the identity matrix. The sample covariance matrix of `rx` can be obtained using the R command `cov(rx)`. □

Exercise 15.4. Write a function that computes canonical loading vectors. Compute the leading canonical loadings between SST and U.S. temperature using the same truncation parameters as above. Make a plot that shows the leading canonical loadings for the two data sets. □

Exercise 15.5. Write a function to compute the fraction of variance explained by the canonical component. This is somewhat tricky because (1) the area weighting needs to be included, (2) missing data should be skipped, and (3) the total variance needs to be computed to compute the fraction. All of this information is available from the output of `eof.latlon`. To help you out, here is the way to compute it for the EOFs of SST:

```
var.x.tot = sum(sst.eof$sval^2)/(nyrs-1)
exp.var.x = rep(NA,dim=tx)
for ( n in 1:tx) exp.var.x[n] = sum(px[!sst.eof$lbad,n]^2 *
   sst.eof$weight[!sst.eof$lbad]^2)/var.x.tot
```

Verify that the sum of the fractional variances equals the sum of the fractional variances of the first T_X EOFs `sum(sst.eof$fexpvar[1:tx])`. You might have to compute *all* of the singular vectors using the command `svd(cov.mat,nu=tx,nv=ty)`. State the explained variances of the canonical components for SST and for U.S. Temperature. These variances should match those in the notes. □

Exercise 15.6. Write a *separate* code that computes the 5% significance levels of the canonical correlations based on 5000 trials of Monte Carlo experiments. State the critical values for each canonical correlation based on the optimum choice of T_X and T_Y. Use these results to decide whether the canonical correlations computed from data are significant. Clearly state your conclusion. (Coding advice: first use only 100 trials until you have debugged your code, and then change to 5000 trials when the code is working.) □

Exercise 15.7. Repeat the above steps, except this time *do not detrend the data*. To prevent detrending, set `npoly =0` at the top of the code. How do the results differ? Explain why this makes sense. □

Exercise 15.8. Show mathematically that the error covariance matrix for the model (15.80) is

$$\tilde{\mathbf{\Sigma}}_{\epsilon Y} = \mathbf{I} - \hat{\mathbf{S}}_{\rho}^{2}. \tag{15.6}$$

This result shows that the error covariance matrix is diagonal, and that the diagonal elements equal $1 - \hat{\rho}_k^2$, which can be interpreted as the variance explained by the canonical variate. □

CHAPTER 16

EXERCISES: COVARIANCE DISCRIMINANT ANALYSIS

Exercise 16.1. Prove (16.14). Hint, substitute (16.12) into the outer product and expanding gives

$$(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_Y)(\mathbf{x}_t - \hat{\boldsymbol{\mu}}_Y)^T = \mathbf{A} + \mathbf{B} + \mathbf{C}, \tag{16.1}$$

where

$$\mathbf{A} = (\mathbf{u}_t^* - \hat{\boldsymbol{\mu}}_Y)(\mathbf{u}_t^* - \hat{\boldsymbol{\mu}}_Y)^T \tag{16.2}$$

$$\mathbf{B} = \mathbf{f}_t \mathbf{f}_t^T \tag{16.3}$$

$$\mathbf{C} = (\mathbf{u}_t^* - \hat{\boldsymbol{\mu}}_Y)\mathbf{f}_t^T + \mathbf{f}_t(\mathbf{u}_t^* - \hat{\boldsymbol{\mu}}_Y)^T. \tag{16.4}$$

Show that $\mathbb{E}[\mathbf{C}] = \mathbf{0}$. What is $\mathbb{E}[\mathbf{A}]$? □

Exercise 16.2. Suppose $d(\sigma_X^2, \sigma_Y^2)$ is some measure of the difference between the variances σ_X^2, σ_Y^2. Suppose further that the function $d(,)$ is invariant to an invertible linear transformation of X and Y. Prove that $d(,)$ can depend only on the ratio of variances

$$d(\sigma_X^2, \sigma_Y^2) = f\left(\frac{\sigma_X^2}{\sigma_Y^2}\right) \tag{16.5}$$

□

Exercise 16.3 (Discriminant analysis via SVD)**.** Given two data matrices $\mathbf{X}$ and $\mathbf{Y}$, solve discriminant analysis based on the singular value decomposition (SVD) (i.e., without solv-

ing an eigenvalue problem). Show how the discriminant ratios, variates, and loading vectors can be derived from the results of the SVD. Hint: compute the SVD of $\mathbf{Y}$ to construct a whitening transformation, then compute the SVD of the whitened data matrix $\mathbf{X}$. □

Exercise 16.4 (Loading Patterns)**.** Show that the matrix $\mathbf{P}$ that minimizes

$$\gamma_Y = E\left[\|\mathbf{y} - E[\mathbf{y}] - \mathbf{P}\mathbf{r}_Y\|_W^2\right], \tag{16.6}$$

is

$$\mathbf{P} = \mathrm{cov}[\mathbf{y}, \mathbf{r}_Y]\left(\mathrm{cov}[\mathbf{r}_Y, \mathbf{r}_Y]\right)^{-1}, \tag{16.7}$$

thereby proving (16.64). Also, show that the matrix $\mathbf{P}$ that minimizes

$$\dot{\gamma}_Y = \|\dot{\mathbf{Y}} - \mathbf{R}_Y\mathbf{P}^T\|_W^2. \tag{16.8}$$

is

$$\dot{\mathbf{P}} = \dot{\mathbf{Y}}^T\mathbf{R}_Y\left(\mathbf{R}_Y^T\mathbf{R}_Y\right)^{-1}, \tag{16.9}$$

thereby proving (16.94). Note that $\dot{\mathbf{P}}$ and $\mathbf{P}$ are merely the sample and population versions of each other.

□

Exercise 16.5. Prove that if $\mathbf{\Sigma}_X \neq \mathbf{\Sigma}_Y$, then there exists a $\mathbf{q}$ such that $\lambda \neq 1$, where

$$\lambda = \frac{\mathbf{q}^T\mathbf{\Sigma}_X\mathbf{q}}{\mathbf{q}^T\mathbf{\Sigma}_Y\mathbf{q}}. \tag{16.10}$$

□

Exercise 16.6. Define the sum total variance of $\dot{\mathbf{Y}}$ as

$$\|\dot{\mathbf{Y}}\|_W^2 = \frac{1}{N_Y - 1}\,\mathrm{tr}\left[\dot{\mathbf{Y}}\mathbf{W}\dot{\mathbf{Y}}^T\right], \tag{16.11}$$

where $\mathbf{W}$ is a positive definite matrix defining how different points are weighted. If $\dot{\mathbf{Y}} = \mathbf{R}_Y\dot{\mathbf{P}}_Y^T$, use the properties of CDA to show explicitly that the sum total variance can be written as

$$\|\dot{\mathbf{Y}}\|_W^2 = \sum_{k=1}^{T}\mathbf{p}_k^T\mathbf{W}\mathbf{p}_k. \tag{16.12}$$

□

Numerical Exercises

`discr.student.R`	program for reading the data sets
tas_Amon_CCSM4_historical_r1i1p1_185001-200512.nc	20C, 1st member
tas_Amon_CCSM4_historical_r2i1p1_185001-200512.nc	20C, 2nd member
tas_Amon_CCSM4_piControl_r1i1p1_080001-130012.nc	PI simulation

In this homework you will write an R function to perform covariance discriminant analysis. You will need to download data and a few R programs. These files are summarized in the above table. The R code `discr.student.R` reads data files for the 20C and PI simulations, combines them, and computes EOFs. The following exercises break up the discriminant analysis into discrete steps, but you should submit a single function that performs all the calculations. The preamble of this function should be the following:

```
cda.eof = function(xdata,ydata,eof.list) {
### PERFORMS COVARIANCE DISCRIMINANT ANALYSIS ON X AND Y
### INPUT:
###   XDATA[NX,MDIM]
###   YDATA[NY,MDIM]
###   EOF.LIST: LIST FROM EOF CALCULATION
### OUTPUT:
###   MIC[NEOF]: MIC AS A FUNCTION OF NUMBER OF PCS
###   NMIN: LOCATION OF MINIMUM MIC
###   DISCR.RATIO[NEOF]: DISCRIMINANT RATIOS VS. NUMBER OF PCS
###   RX[NX,NMIN]: VARIATE TIME SERIES FOR X
###   RY[NY,NMIN]: VARIATE TIME SERIES FOR Y
###   PMAT[SPACE,NMIN]: LOADING VECTOR
```

In the following calculations, you should include *both* ensemble members from 20C. A trick for doing this is to reshape the array so that the time series looks twice as long:

```
### RESHAPE PC.20C[TIME,NENS,NEOF] TO PC.20C[TIME*NENS,NEOF]
dim(pc.20c) = c(tdim.20c*nens,neof)
```

Then, when you want individual ensemble members, you can reshape the array back to [time,ensemble,eof]

Exercise 16.7. Write a function that evaluates Mutual Information Criterion (MIC) for comparing covariance matrices. MIC is defined as

$$\text{MIC} = \frac{1}{N_T} \log \left(\frac{|\overline{\mathbf{\Sigma}}_X|^{N_X} |\overline{\mathbf{\Sigma}}_Y|^{N_Y}}{|\overline{\mathbf{\Sigma}}_T|^{N_T}} \right) + \mathbb{P}, \tag{16.13}$$

where

$$\mathbb{P} = \frac{T}{N_T} \left(\frac{N_X(N_X+1)}{N_X - P - 2} + \frac{N_Y(N_Y+1)}{N_Y - P - 2} - \frac{N_T(N_T+1)}{N_T - P - 2} \right),$$

and

$$\hat{\mathbf{\Sigma}}_T = \frac{N_X \overline{\mathbf{\Sigma}}_X + N_Y \overline{\mathbf{\Sigma}}_Y}{N_T} \quad \text{and} \quad N_T = N_X + N_Y. \tag{16.14}$$

The function should evaluate MIC over all possible EOF truncations T. Plot MIC for a range of values and identify the value of T that minimizes MIC. Print out the first 5 values of MIC. These results should match the example in the notes. □

Exercise 16.8. Compute the discriminant ratios for the optimum choice of T. State the values. These values should be consistent with those in the notes □

Exercise 16.9. Write a function that computes discriminant variates. Compute the variates for the optimum choice of T and plot them. Verify that the sample covariance matrix of the PI variates equals the identity matrix. Verify that the sample covariance matrix of 20C variates is diagonal, with diagonal elements equal to the discriminant ratios. □

Exercise 16.10. Write a function that computes the loading vectors. Plot the leading loading vector. □

Exercise 16.11. Write a *separate* code that computes the 5% significance levels of the discriminant ratios based on 5000 trials of Monte Carlo experiments. State the 95% percentile for all discriminant ratios for the optimum choice of T. Are your discriminant ratios significant or not? □

Exercise 16.12. What is the 5% significance level of the *univariate* F-test for equality of variances for sample sizes $N_X = 51$, $N_Y = 51$? Using your Monte Carlo code, show a plot of the 5% and 95% percentiles of *the leading discriminant ratio* as a function of the truncation parameter T. In this exercise, let $N_X = 51$, $N_Y = 51$, and the number of trials = 1000. Also, let the maximum dimension be 30. What happens to the ratios as T increases? Explain why this happens. The 95% percentile from the Monte Carlo experiments should be close to the univariate F-test for $T = 1$. □

CHAPTER 17

ANALYSIS OF VARIANCE

Table 17.1 default

`prca.student.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 `prca.student.R`. As usual, modify `dir.data` and `dir.Rlib` to correspond to your data and library directories, respectively. `prca.student.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? □

CHAPTER 18

PREDICTABLE COMPONENT ANALYSIS

In this homework set you will apply Predictable Component Analysis (PrCA) to quantify the predictability of seasonal forecasts over North America from the CFSv2 Retrospective Hindcasts. The data set and codes for reading this data set are identical to those you used in the last homework for ANOVA.

To apply predictable component analysis, you must first compute the principal components of the data set. This can be done by pooling all ensemble members and years together as if it were one long time series. This will be done automatically by `eof.latlon`. Since the data set is dimensioned as

```
1 hcst.data[nlon,nlat,nens,nyrs]
```

On output, the PCs will have dimension `[nens*nyrs,neof]`. When desired, the PCs can be re-shaped into an array of dimension `[nens,nyrs,neof]`.

In this homework, you will eventually write an R function called `prca` that performs predictable component analysis on the PCs. The homework assignment below will direct you to write this code in "pieces." In the end, you should combine all these pieces together to produce a single function that performs all the calculations. The header for this function should be:

```
prca.eof = function(data.eof,nens,ncon,alpha=0.01) {
## PERFORMAS PREDICTABLE COMPONENT ANALYSIS (OR EQUIVALENTLY MANOVA)
## ON PC-TIME SERIES OF AN ENSEMBLE FORECAST DATA SET
## INPUT:
##    DATA.EOF: LIST OUTPUT FROM EOF.LATLON
##    NENS: NUMBER OF ENSEMBLE MEMBERS
##    NCON: NUMBER OF CONDITIONS
##    ALPHA: SIGNIFICANCE LEVEL (DEFAULT = 1%)
## OUTPUT: LIST$
#    MIC: VALUES OF MUTUAL INFORMATION CRITERION
#    MIC.CRIT: SIGNIFICANCE VALUES OF MIC
#    F.MAX[NTRUN]: MAXIMIZED F-RATIOS
#    R[NENS,NCON,NTRUN]: PREDICTABLE VARIATES
#    P[NSPACE,NTRUN]: PREDICTABLE LOADING VECTORS
#    FEXPVAR[NMIN]: FRACTION OF VARIANCE EXPLAINED
#    RSQR.ADJ[NMIN]: ADJUSTED R-SQUARE
#    SNR.ADJ[NMIN]: ADJUSTED SIGNAL-TO-NOISE RATIO
#    NENS, NCON, NMIN, ALPHA: OTHER PARAMETERS
```

Exercise 18.1. Write an R program called `prca.null.fixdim` that estimates the 1% significance level of each eigenvalue from predictable component analysis using `ntrun` feature vectors. Run this program using `ntrun = 6` and state the significance values. The header of this program should be

```
prca.null.fixdim = function(ndim,ncon,nens,alpha=0.01,ntrial=10000) {
## ESTIMATES SIGNIFICANCE LEVEL OF THE ALL EIGENVALUES FROM PRCA/MANOVA
## FOR FIXED TRUNCATION NDIM
## BY MONTE CARLO METHODS
## INPUT:
#    NDIM: DIMENSION OF THE RANDOM VECTOR
#    NCOM: NUMBER OF CONDITIONS (E.G., NUMBER OF YEARS)
#    NENS: NUMBER OF ENSEMBLE MEMBERS (E.G., NUMBER OF REPITITIONS)
#    ALPHA: DESIRED SIGNIFICANCE LEVELS (CAN BE MORE THAN ONE)
#    NTRIALS: NUMBER OF MONTE CARLO TRIALS
## OUTPUT:
#    EVAL.CRIT[2,NDIM]: (alpha,1-alpha) SIGNIFICANCE THRESHOLDS FOR EACH EIGENVALUE
```

It is a good idea to start out with `ntrial = 100` while you are writing and debugging your code, and then change to `ntrial = 10000` at the last step.

□

Exercise 18.2. Write an R function to compute the MIC as a function of the number of PCs. The expression for MIC that is appropriate for PrCA is:

$$\text{MIC} = \log\left|\overline{\mathbf{\Sigma}}_N\right| - \log\left|\overline{\mathbf{\Sigma}}_T\right| + \frac{2CT + T(T+1)}{CE - C - T - 1} - \frac{2T + T(T+1)}{CE - 1 - T - 1}, \qquad (18.1)$$

where T is the number of PCs. State the values of MIC that you obtain, and state the value of T at which the minimum occurs. □

Exercise 18.3. Write an R function to compute the maximized F-ratios from PrCA, and the associated adjusted signal-to-noise ratios and R-square. This function will need to

compute the signal and noise covariance matrices from the PCs and then solve a generalized eigenvalue problem. State the maximized F-ratios for $T = 6$. Also state the adjusted signal-to-noise ratios and R-square. □

Exercise 18.4. Write an R function that computes the predictable variates. Remember to normalize the weight vector $\mathbf{q}$ so that the total variance equals unity. Run this function for $T = 6$. Make a plot of the variates for the two most predictable components. If the variate is dimensioned as `r[nens,nyrs,ntrun]`, then the variates for the most predictable component can be plotted using the command

```
boxplot(r[,,1],names=year,col='grey')
```

Label the plot as in the lecture notes, so that all information required to interpret the plot is contained in the plot. □

Exercise 18.5. Verify that the predictable variates are orthogonal. More generally, verify that the sample covariance matrix of the predictable components equals the identity matrix. □

Exercise 18.6. Write a function that calculates the loading vectors. Plot them. □

CHAPTER 19

DATA ASSIMILATION

In this homework set you will apply the Kalman Filter to assimilate observations using a vector autoregressive model. A vector autoregressive model allows probabilities of forecasted variables to be solved exactly, so it illustrates the Kalman Filter in an "ideal" situation. However, this ideal situation is very unrealistic, so there is potential for being mislead.

A vector autoregressive model is of the form

$$\underset{M\times 1}{\mathbf{y}_t} = \underset{M\times M}{\mathbf{A}}\ \underset{M\times 1}{\mathbf{y}_{t-1}} + \underset{M\times 1\text{'}}{\mathbf{w}_t} \tag{19.1}$$

where $\mathbf{A}$ is a matrix called the *dynamical operator* and $\mathbf{w}_t$ is Gaussian white noise with zero mean and covariance matrix $\mathbf{Q}$. After specifying the matrices $\mathbf{A}$ and $\mathbf{Q}$ and an initial condition $\mathbf{y}_0$, the vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots$ can be solved for a particular realization of the noise $\mathbf{w}_t$.

In this homework set, you will consider the following experimental situation. First, you will create a vector time series $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{100}$ from (19.1) for a particular choice of $\mathbf{A}$, $\mathbf{Q}$, and $\mathbf{y}_0$ and for a particular realization of $\mathbf{w}_t$. The resulting vector time series will be the "truth." To mimic reality, we will pretend that we do not know the truth. Instead, you will create "observations" using the model

$$\underset{K\times 1}{\mathbf{o}_t} = \underset{K\times M}{\mathbf{H}}\ \underset{M\times 1}{\mathbf{y}_t} + \underset{K\times 1\text{'}}{\mathbf{r}_t} \tag{19.2}$$

for a particular choice of $\mathbf{H}$ and covariance matrix $\mathbf{R}$ of $\mathbf{r}_t$. The observations $\mathbf{o}_t$ are known and our goal is to estimate the truth $\mathbf{y}_t$ from these observations (pretending that we don't know the truth). You will use the Kalman Filter to assimilate these observations to estimate the true state. Then you will compare this estimate with the true state to see how well the Kalman Filter works.

Download the R program `kf_student.R` from the class website. This program generates a time series from (19.1) for the choice

$$\mathbf{A} = \begin{pmatrix} 0.9 & 0.5 & 0.3 \\ 0.0 & 0.5 & 2.0 \\ 0.0 & 0.0 & 0.4 \end{pmatrix}, \quad \mathbf{Q} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad \mathbf{y}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \tag{19.3}$$

and generates "observations" using the choice

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{R} = 49. \tag{19.4}$$

Note that this choice implies that only one variable (i.e., the first element of $\mathbf{y}_t$) is observed with noise. The program assumes that $\mathbf{Q}$ is diagonal, so that the elements of the noise vector $\mathbf{w}_t$ are independent and have variances equal to the diagonal elements of $\mathbf{Q}$. This special form allows the noise vector to be specified very efficiently numerically as

```
w = rnorm(M,sd=sqrt(diag(Q)))
```

Similarly, $\mathbf{R}$ is assumed to be diagonal and hence the observations generated by (19.2) can be specified efficiently in a similar way. This is done in the R code that you will download.

After the R code runs (hopefully without error messages!), you should have the following quantities available:

```
# NDIM:  THE DIMENSION OF THE VECTOR AUTOREGRESSIVE MODEL (VAR); DEFAULT = 3
# NTOT:  THE TOTAL LENGTH OF THE TIME SERIES; DEFAULT = 100
# NOBS:  NUMBER OF OBSERVATIONS PER TIME STEP; DEFAULT = 1
# X:     [NDIM, NTOT] MATRIX SPECIFYING THE "TRUTH" AT EACH TIME STEP
# OBS:   [NOBS, NTOT] MATRIX SPECIFYING THE "OBSERVATIONS" AT EACH TIME STEP
# DYNOP: [NDIM, NDIM] MATRIX SPECIFYING THE DYNAMICAL OPERATOR OF THE VAR
# HOP:   [NOBS, NDIM] MATRIX SPECIFYING THE INTERPOLATION OPERATOR
# Q.COV: [NDIM, NDIM] MATRIX SPECIFYING THE NOISE COVARIANCE MATRIX OF THE VAR
# R.COV: [NOBS, NOBS] MATRIX SPECIFYING THE ERROR COVARIANCE MATRIX OF THE OBS
```

Exercise 19.1. Plot the first element of the true time series. Also plot the corresponding observations. The result should look something like the curve and dots in fig. 19.1. □

Exercise 19.2. Write a function to evaluate the Kalman Filter equations

$$\boldsymbol{\mu}_t^A = \boldsymbol{\mu}_t^B + \boldsymbol{\Sigma}_t^B \mathbf{H}_t^T \left(\mathbf{H}_t \boldsymbol{\Sigma}_t^B \mathbf{H}_t^T + \mathbf{R}_t \right)^{-1} \left(\mathbf{o}_t - \mathbf{H}_t \boldsymbol{\mu}_t^B \right) \tag{19.5}$$

$$\boldsymbol{\Sigma}_t^A = \boldsymbol{\Sigma}_t^B - \boldsymbol{\Sigma}_t^B \mathbf{H}_t^T \left(\mathbf{H}_t \boldsymbol{\Sigma}_t^B \mathbf{H}_t^T + \mathbf{R}_t \right)^{-1} \mathbf{H}_t \boldsymbol{\Sigma}_t^B. \tag{19.6}$$

The function call should be

```
kalman.population = function(mub,sigmab,hop,r.cov,obs) {
##########
## EVALUATES THE KALMAN FILTER EQUATIONS FOR THE MEAN AND COVARIANCE MATRIX OF THE ANALYSIS
## BASED ON *POPULATION* COVARIANCE MATRICES AND MEANS
## INPUT:
#    MUB:    [NDIM] MEAN OF THE BACKGROUND DISTRIBUTION
#    SIGMAB: [NDIM,NDIM] BACKGROUND COVARIANCE MATRIX
#    HOP:    [NOBS,NDIM] INTERPOLATION OPERATOR
#    R.COV:  [NOBS,NOBS] COVARIANCE MATRIX OF THE OBSERVATIONAL ERROR
#    OBS:    [NOBS] THE OBSERVATIONS
## OUTPUT
#    MUA:    [NDIM] VECTOR OF THE ANALYSIS DISTRIBUTION
#    SIGMAA: [NDIM,NDIM] COVARIANCE MATRIX OF THE ANALYSIS DISTRIBUTION
```

As you can see from the equations, you will need to compute the inverse of a matrix. You should invert the matrix using the commands `chol2inv(chol(MATRIX))`, which takes advantage of the fact that the matrix being inverted is symmetric. □

Exercise 19.3. The background distribution is obtained from the *previous* analysis. In particular, the mean and covariance matrix of the background distribution are

$$\boldsymbol{\mu}_t^B = \mathbf{A}\boldsymbol{\mu}_{t-1}^A \quad \text{and} \quad \boldsymbol{\Sigma}_t^B = \mathbf{A}\boldsymbol{\Sigma}_{t-1}^A\mathbf{A}^T + \mathbf{Q}. \tag{19.7}$$

Assume the initial analysis $\boldsymbol{\mu}_0^A = \mathbf{0}$ and

$$\boldsymbol{\Sigma}_0^A = \begin{pmatrix} 400 & 0 & 0 \\ 0 & 400 & 0 \\ 0 & 0 & 400 \end{pmatrix}. \tag{19.8}$$

Combine these equations together with your Kalman Filter function to construct an analysis for one hundred time steps. Plot the mean analysis and its uncertainty for the first element of $\mathbf{y}_t$. The result should look something like the grey shading in fig. 19.1, which shows $(\boldsymbol{\mu}^A)_1 \pm \sqrt{(\boldsymbol{\Sigma}^A)_{11}}$. □

Exercise 19.4. Plot the standard error of the analysis for each element of $\mathbf{y}_t$ as a function of time (i.e., plot the square roots of the diagonal elements of $\boldsymbol{\Sigma}_t^A$). What happens to the errors after a long time? □

Exercise 19.5. In this problem only, set $\mathbf{R}_t = 1$ and show the analysis, observations (if available), and the truth for the first and second elements of $\mathbf{y}_t$. How do these plots differ from the previous case? Explain why this difference makes sense. □

Exercise 19.6. Suppose that, instead of observing $(\mathbf{y}_t)_1$, we can observe only the sum $z(t) = (\mathbf{y}_t)_1 + (\mathbf{y}_t)_2$. Furthermore, suppose the error of this observation is normally distributed with zero mean and variance 25. Explain how you would assimilate these observations (e.g., explain how you would change your R code to handle this case). Actually do this and show the standard error of the analysis and the truth for $(\mathbf{y}_t)_1$. □

Exercise 19.7 (Least Squares Derivation of the Kalman Filter Equations)**.** The Kalman Filter equations can be derived by the method of least squares. The essential problem is that we are given a set of observations $\mathbf{o}$, and based on this want to estimate the state $\mathbf{x}$. To do this, we assume a linear prediction model

$$\hat{\mathbf{y}} = \mathbf{A}\mathbf{o} + \mathbf{b}. \tag{19.9}$$

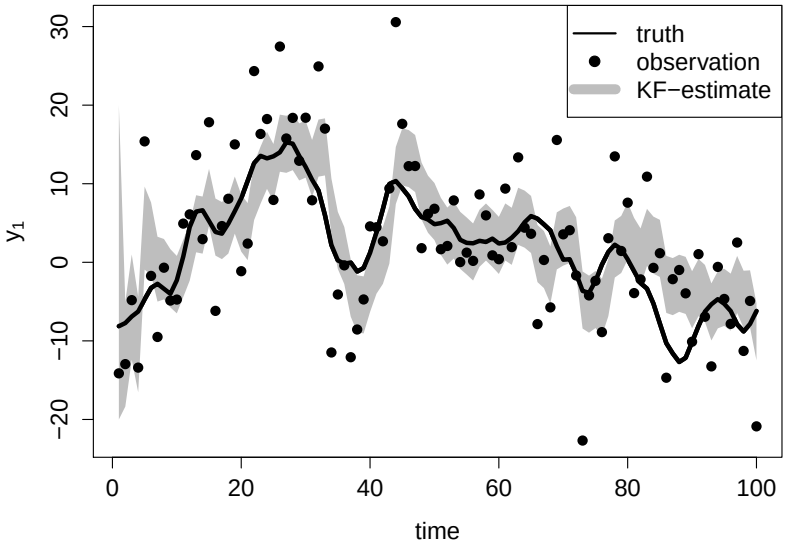


Figure 19.1 A particular realization of the first element $y_1(t)$ generated by the vector autoregressive model (19.1) (solid curve), corresponding observations (dots), and the analysis mean plus a standard deviation, as estimated state by the Kalman Filter (grey shading).

Show that the method of least squares gives

$$\mathbf{A} = \mathbf{\Sigma}_{YO}\mathbf{\Sigma}_O^{-1} \quad \text{and} \quad \mathbf{b} = \mathbb{E}[\mathbf{y}] - \mathbf{A}\mathbb{E}[\mathbf{o}], \tag{19.10}$$

where the expectation operator $E[\cdot]$ is an average over all possible realizations of the observations and background quantities. Under this definition, $E[\mathbf{y}] = \boldsymbol{\mu}_B$ and $\mathrm{cov}[\mathbf{y}] = \mathbf{\Sigma}_B$. The observations are assumed to be related to the state $\mathbf{y}$ through the model

$$\mathbf{o} = \mathbf{H}\mathbf{y} + \mathbf{r}, \tag{19.11}$$

where $\mathbf{H}$ is the interpolation operator and $\mathbf{r}$ is independent and normally distributed with zero mean and covariance matrix $\mathbf{\Sigma}_R$. Show that this equation implies

$$\mathbf{\Sigma}_{OY} = \mathbf{H}\mathbf{\Sigma}_Y = \mathbf{H}\mathbf{\Sigma}_B \tag{19.12}$$

and

$$\mathbf{\Sigma}_O = \mathbf{H}\mathbf{\Sigma}_Y\mathbf{H}^T + \mathbf{\Sigma}_R = \mathbf{H}\mathbf{\Sigma}_B\mathbf{H}^T + \mathbf{\Sigma}_R. \tag{19.13}$$

and

$$\mathbb{E}[\mathbf{o}] = \mathbf{H}\boldsymbol{\mu}_B \tag{19.14}$$

Substituting these equations into (19.10) gives

$$\mathbf{A} = \mathbf{\Sigma}_B\mathbf{H}^T\left(\mathbf{H}\mathbf{\Sigma}_B\mathbf{H}^T + \mathbf{\Sigma}_R\right)^{-1} \tag{19.15}$$

and

$$\mathbf{b} = \boldsymbol{\mu}_B - \boldsymbol{\Sigma}_B \mathbf{H}^T \left(\mathbf{H} \boldsymbol{\Sigma}_B \mathbf{H}^T + \boldsymbol{\Sigma}_R \right)^{-1} \mathbf{H} \mu_B. \tag{19.16}$$

It follows that the linear regression model (19.9) yields

$$\mathbf{y}_a = \boldsymbol{\mu}_B + \boldsymbol{\Sigma}_B \mathbf{H}^T \left(\mathbf{H} \boldsymbol{\Sigma}_B \mathbf{H}^T + \boldsymbol{\Sigma}_R \right)^{-1} \left(\mathbf{o} - \mathbf{H} \boldsymbol{\mu}_B \right), \tag{19.17}$$

which is precisely the Kalman Filter equation for the mean analysis.

Show that the error covariance of the regression model (19.9) is

$$\begin{aligned} \boldsymbol{\Sigma}_a &= \mathbb{E}\left[\left(\mathbf{y} - \mathbf{y}_a \right) \left(\mathbf{y} - \mathbf{y}_a \right)^T \right] \\ &= \mathbb{E}\left[\left(\mathbf{y} - \boldsymbol{\mu}_B - \mathbf{A}(\mathbf{o} - \mathbf{H}\boldsymbol{\mu}_B) \right) \left(\mathbf{y} - \boldsymbol{\mu}_B - \mathbf{A}(\mathbf{o} - \mathbf{H}\boldsymbol{\mu}_B) \right)^T \right] \\ &= \boldsymbol{\Sigma}_Y - \boldsymbol{\Sigma}_{YO} \mathbf{A}^T - \mathbf{A} \boldsymbol{\Sigma}_{OY} + \mathbf{A} \boldsymbol{\Sigma}_O \mathbf{A}^T \\ &= \boldsymbol{\Sigma}_B - \boldsymbol{\Sigma}_B \mathbf{H}^T \left(\mathbf{H} \boldsymbol{\Sigma}_B \mathbf{H}^T + \boldsymbol{\Sigma}_R \right)^{-1} \mathbf{H} \boldsymbol{\Sigma}_B, \end{aligned} \tag{19.18}$$

which is precisely the Kalman Filter equation for the analysis error covariance.

□

CHAPTER 20

ENSEMBLE SQUARE ROOT FILTERS

Exercise 20.1. Recall from sec. 20.5 that observations can be assimilated sequentially if their errors are independent. In this exercise, you will derive how this is done for the square root filter. In the case of one observation, (21.18) can be written as

$$\mathbf{D} = \mathbf{I} - \beta \mathbf{w}\mathbf{w}^T, \tag{20.1}$$

where

$$\mathbf{w} = \mathbf{X}^{BT}\mathbf{H}_t^T \quad \text{and} \quad \beta = \left(\mathbf{w}^T\mathbf{w} + \mathbf{R}_t\right)^{-1}. \tag{20.2}$$

Since only one observation is under consideration, β is a scalar. The matrix $\mathbf{D}$ differs from the identity matrix by a rank-1 matrix. It is natural to assume that the square root also differs from the identity by a term proportional to the same rank-1 matrix. Therefore, we seek a symmetric square root of $\mathbf{D}$ of the form

$$\mathbf{I} - \beta \mathbf{w}\mathbf{w}^T = \left(\mathbf{I} - \delta \mathbf{w}\mathbf{w}^T\right)\left(\mathbf{I} - \delta \mathbf{w}\mathbf{w}^T\right), \tag{20.3}$$

where δ is a parameter to be determined. Expanding the right side and simplifying yields

$$\mathbf{I} - \beta \mathbf{w}\mathbf{w}^T = \mathbf{I} - 2\delta \mathbf{w}\mathbf{w}^T + \delta^2 \mathbf{w}\mathbf{w}^T\mathbf{w}\mathbf{w}^T \tag{20.4}$$

$$0 = \left(\mathbf{w}^T\mathbf{w}\delta^2 - 2\delta + \beta\right)\mathbf{w}\mathbf{w}^T. \tag{20.5}$$

This equation is satisfied for arbitrary $\mathbf{w}$ only if the quadratic equation for δ vanishes, which requires

$$\delta = \frac{1 \pm \sqrt{1 - \mathbf{w}^T\mathbf{w}\beta}}{\mathbf{w}^T\mathbf{w}} = \frac{1 \pm \sqrt{\beta\mathbf{R}_t}}{\mathbf{w}^T\mathbf{w}} = \beta\left(1 \mp \sqrt{\frac{\mathbf{R}}{\mathbf{R} + \mathbf{w}^T\mathbf{w}}}\right)^{-1}. \tag{20.6}$$

This gives two solutions, but only the solution

$$\delta = \frac{1 - \sqrt{1 - \mathbf{w}^T\mathbf{w}\beta}}{\mathbf{w}^T\mathbf{w}} = \frac{1 - \sqrt{\beta\mathbf{R}_t}}{\mathbf{w}^T\mathbf{w}} = \beta\left(1 + \sqrt{\frac{\mathbf{R}}{\mathbf{R} + \mathbf{w}^T\mathbf{w}}}\right)^{-1}. \tag{20.7}$$

produces a positive semi-definite square root matrix. Show that this is true. □

Exercise 20.2. Prove the Schur Product theorem (theorem 21.1). Hint: show that the Schur product between a positive semi-definite matrix and a rank-1 matrix is positive semi-definite. Then, use the spectral factorization theorem to argue that any symmetric, positive semi-definite matrix can be represented by a sum of rank-1 matrices. □

CHAPTER 21

EXERCISES: EXTREME VALUE ANALYSIS

In this homework set, you will use the Generalized Extreme Value Distribution to quantify the return period for extreme cold temperatures in the mid-Atlantic U.S. The data set and R code to read this data are summarized in the table below and can be downloaded from the class website. You also will need to install the `extRemes` library. Also, inside `tele_index.R`, you need to change `indices.fname` to the full directory path of the data file `tele_index.nh`.

Table 21.1 default

GHNC_OBS_T_mid_atlantic_1950_2015.txt	data file for mid-Atlantic temperature
gev.student.R	R code to read the mid-Atlantic temperature
tele_index.R	R code to read climate indices
tele_index.nh	NOAA climate indices

Exercise 21.1. Run `gev.student.R` and ensure it generates a time series/box plot identical to fig. 19.1. Next, fit the cold extremes to a GEV using the commands

```
t.cold        = -t.cold # reverse sign to use extreme value functions
cold.x        = fevd(t.cold)
```

Submit a print out of the output when you type `cold.x`. This print out should be consistent with table 19.1.

Make a plot using the command `plot(cold.x)` and submit it. This plot is discussed in Gilleland and Katz (2016). Consult this document and describe what this plot shows.

What are the return levels for 2-year, 20-year, and 200-year events (you will need a special command in Gilleland and Katz (2016) to answer this question)? □

Exercise 21.2. Is the fitted distribution a Gumbel, Fréchet, or Weibull? Explain your answer. □

Exercise 21.3. Fit the data to a GEV, but this time use NAO as a covariate for the location parameter. The NAO is `index` in the R code. Is this covariate significant? Is the fitted distribution a Gumbel, Fréchet, or Weibull? Explain your answer. Generate a plot similar to the first exercise above and describe the new information that it conveys. □

Exercise 21.4. Suppose the probability of exceedance in any randomly selected year is p. If the exceedances in different years are independent, then what is the average number of years one will wait before observing the next exceedance? Hint: the average number of years one will wait before observing the next exceedance is computed as

$$\text{return period} = \sum_{y=1}^{\infty} y * (\text{probability that next exceedance occurs in year } y) \,. \qquad (21.1)$$

What is the probability that the exceedance occurs in year 1? year 2? etc?

□

CHAPTER 22

ANSWERS TO EXERCISES

Exercise 1.1, page 2

The frequency of H_1 is the sum over all joint frequencies involving H_1:

$$P(H_1) = P(H_1, H_2) + P(H_1, T_2) = (30 + 21)/100 = 51\%$$

Similarly for H_2

$$P(H_2) = P(H_1, H_2) + P(T_1, H_2) = (30 + 22)/100 = 52\%$$

The joint event $P(H_1, T_2)$ can be read directly from the table:

$$P(H_1, T_2) = 21/100 = 21\%.$$

The conditional frequency $P(T_2|H_1)$ is the number of counts of T_2 *out of all cases in which the first toss is heads*:

$$P(T_2|H_1) = 21/(30 + 21) = 21/51 \approx 41\%.$$

It is readily verified that

$$P(T_2|H_1) = P(T_2, H_1)/P(H_1) = 21/51 \approx 41\% \tag{22.1}$$

Exercise 1.2, page 2

Since X and Y are independent, $p(x, y) = p(x)p(y)$. Therefore:

$$\begin{aligned} E[XY] &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xyp(x,y)dxdy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} xyp(x)p(y)dxdy \\ &= \left(\int_{-\infty}^{\infty} xp(x)dx\right)\left(\int_{-\infty}^{\infty} yp(y)dy\right) \\ E[XY] &= E[X]E[Y]. \end{aligned} \tag{22.2}$$

Exercise 1.3, page 2

(a) Since X and Y are independent, $E[XY] = E[X]E[Y] = \mu_X\mu_Y$.

(b) $\mathrm{cov}[X, Y] = 0$, since X and Y are independent.

(c)

$$\begin{aligned} \mathrm{var}[X-Y] &= E[((X-Y)-(\mu_X-\mu_Y))^2] \\ &= E[(X-\mu_X)^2 + (Y-\mu_y)^2 - 2(X-\mu_X)(Y-\mu_Y)] \\ &= \sigma_X^2 + \sigma_Y^2. \end{aligned}$$

(d)

$$\begin{aligned} \mathrm{var}(XY) &= E[(XY - E[XY])^2] \\ &= E[(XY - \mu_X\mu_Y)^2] \\ &= E[X^2Y^2 + \mu_X^2\mu_Y^2 - 2XY\mu_X\mu_Y] \\ &= E[X^2]E[Y^2] + \mu_X^2\mu_Y^2 - 2\mu_X^2\mu_Y^2 \\ &= (\sigma_X^2 + \mu_X^2)(\sigma_Y^2 + \mu_Y^2) - \mu_X^2\mu_Y^2 \\ &= \sigma_X^2\sigma_Y^2 + \sigma_X^2\mu_Y^2 + \sigma_Y^2\mu_X^2. \end{aligned}$$

Exercise 1.4, page 2

$$\begin{aligned} \mathrm{var}[k] &= E[(k-E[k])^2] = E[(k-k)^2] = E[0] = 0. \\ \mathrm{var}[kX] &= E[(kX - E[kX])^2] = E[(kX - kE[X])^2] \\ &= E[k^2(X-E[X])^2] = k^2\,\mathrm{var}[X] \\ \mathrm{var}[k+X] &= E[((k+X)-E[k+X])^2] = E[(k+X-k-E[X])^2] \\ &= E[(X-E[X])^2] = \mathrm{var}[X] \end{aligned}$$

Exercise 1.5, page 2

$$\begin{aligned}\operatorname{cov}[X,Y] &= E\left[(X - E[X])\,(Y - E[Y])\right] \\ &= E\left[XY - XE[Y] - YE[X] + E[X]E[Y]\right] \\ &= E[XY] - E[X]E[Y] - E[X]E[Y] + E[X]E[Y] \\ &= E[XY] - E[X]E[Y]. \end{aligned} \tag{22.3}$$

Exercise 1.6, page 2

Calculus method: take the first and second derivatives with respect to k:

$$\begin{aligned}\frac{\partial E[(X-k)^2]}{\partial k} &= -2E[X-k] = 0 \\ \frac{\partial^2 E[(X-k)^2]}{\partial k^2} &= 2 \geq 0\end{aligned}$$

The first equation gives the solution $k = E[X]$. The second equation demonstrates that the second derivative is positive everywhere and hence the solution is a minimum.

Second method: add and subtract $E[X]$ and take expectations:

$$\begin{aligned}E[(X-k)^2] =& E[(X - E[X] + E[X] - k)^2] \\ =& E[(X - E[X])^2] + E[(E[X] - k)^2] + E[(X - E[X])\,(E[X] - k)]\end{aligned}$$

The last term vanishes because $E[X] - k$ is a constant and hence can be taken outside the expectation operator, leaving $E[X - E[X]] = E[X] - E[X] = 0$. It follows that

$$E[(X-k)^2] = E[(X - E[X])^2] + E[(E[X] - k)^2].$$

Both terms on the right hand side are positive, but only the last depends on k. Moreover, the last term vanishes if $k = E[X]$, so this choice minimizes the left hand side.

Exercise 1.7, page 2

From the hint,

$$E[(t(X - E[X]) + (Y - E[Y]))^2] = t^2 \operatorname{var}[X] + 2t \operatorname{cov}[X,Y] + \operatorname{var}[Y] \geq 0. \tag{22.4}$$

The minimum value is found by setting the derivative of t to zero and solving, which gives

$$t_{min} = -\operatorname{cov}[X,Y] / \operatorname{var}[X] \tag{22.5}$$

Substituting this value into the t-equation gives

$$\begin{aligned}
0 &\leq \left(\frac{-\operatorname{cov}[X,Y]}{\operatorname{var}[X]}\right)^2 \operatorname{var}[X] - 2\frac{\operatorname{cov}^2[X,Y]}{\operatorname{var}[X]} + \operatorname{var}[Y] \\
&\leq \frac{\operatorname{cov}^2[X,Y]}{\operatorname{var}[X]} - 2\frac{\operatorname{cov}^2[X,Y]}{\operatorname{var}[X]} + \operatorname{var}[Y] \\
&\leq \operatorname{var}[Y] - \frac{\operatorname{cov}^2[X,Y]}{\operatorname{var}[X]} \\
&\leq \operatorname{var}[Y]\left(1 - \frac{\operatorname{cov}^2[X,Y]}{\operatorname{var}[X]\operatorname{var}[Y]}\right) \\
&\leq \operatorname{var}[Y]\left(1-\rho^2\right)
\end{aligned}$$

Hence

$$\rho^2 = \frac{\operatorname{cov}^2[X,Y]}{\operatorname{var}[X]\operatorname{var}[Y]} \leq 1,$$

which proves that the correlation coefficient must be between -1 and 1.

Alternatively, one can reason geometrically: the equation is a parabola that is concave upward. For the inequality to be satisfied, there should either no roots or exactly one root (i.e., if there are two roots, then the parabola crosses the x-axis and hence must be negative). The condition for no root or exactly one root is that the discriminant is not positive. That is, for a parabola of the form $y = Ax^2 + Bx + C$, the discriminant $B^2 - 4AC \leq 0$. This condition gives:

$$4\operatorname{cov}^2[X,Y] - 4\operatorname{var}[X]\operatorname{var}[Y] \leq 0 \quad \rightarrow \quad \frac{\operatorname{cov}^2[X,Y]}{\operatorname{var}[X]\operatorname{var}[Y]} \leq 1. \tag{22.6}$$

If the squared correlation is *exactly* equal to 1, then the quadratic equation vanishes exactly. But the only way an expression of the form $E[Z^2]$ can equal zero is if Z itself is exactly zero, which implies

$$Y = -t(X - E[X]) + E[Y],$$

for some constant t; in other words, if X and Y are exactly linearly related.

Exercise 1.8, page 3

The answers to the questions are as follows:

$$\begin{aligned}
\hat{\mu}_X &\sim N(\mu_X, \sigma^2/N) \\
\hat{\mu}_Y &\sim N(\mu_Y, \sigma^2/N) \\
\hat{\mu}_X - \hat{\mu}_Y &\sim N(\mu_X - \mu_Y, 2\sigma^2/N) \\
(N-1)\hat{\sigma}_X^2/\sigma^2 &\sim \chi^2_{N-1} \\
(N-1)\hat{\sigma}_Y^2/\sigma^2 &\sim \chi^2_{N-1} \\
(N-1)(\hat{\sigma}_X^2 + \hat{\sigma}_Y^2)/\sigma^2 &\sim \chi_{2(N-1)}
\end{aligned}$$

Exercise 1.9, page 3

Since the die is fair, the probability of each outcome is equal, so $P(1) = \cdots = P(6) = 1/6$. Therefore, the expected value is

$$E[X] = 1\left(\frac{1}{6}\right) + 2\left(\frac{1}{6}\right) + \cdots + 6\left(\frac{1}{6}\right) = \frac{7}{2}. \tag{22.7}$$

Similarly, the expected squared value is

$$E[X^2] = 1^2\left(\frac{1}{6}\right) + 2^2\left(\frac{1}{6}\right) + \cdots + 6^2\left(\frac{1}{6}\right) = \frac{91}{6}. \tag{22.8}$$

Hence, the variance is

$$\mathrm{var}[X] = E[X^2] - (E[X])^2 = \frac{91}{6} - \left(\frac{7}{2}\right)^2 = \frac{35}{12}. \tag{22.9}$$

Exercise 1.10, page 3

Let X_i denote the outcome of the i'th die. Then $X = X_1 + X_2 + X_3$ and

$$E[X] = E[X_1] + E[X_2] + E[X_3] = 3\left(\frac{7}{2}\right) = \frac{21}{2} = 10.5 \tag{22.10}$$

where the answer from exercise 1.9 has been used. Similarly, since the dice rolls are independent, the variance of the sum is the sum of the variances:

$$\mathrm{var}[X] = \mathrm{var}[X_1] + \mathrm{var}[X_2] + \mathrm{var}[X_3] = 3\frac{35}{12} = \frac{35}{4} = 8.75. \tag{22.11}$$

where the answer from exercise 1.9 has been used again.

Exercise 1.11, page 3

$$
\begin{aligned}
E[\hat{\sigma}^2] &= E\left[\frac{1}{N-1}\sum_{n=1}^{N}\left(X_n-\hat{\mu}_X\right)^2\right] && \text{definition of } \hat{\sigma}^2\\
&= E\left[\frac{1}{N-1}\sum_{n=1}^{N}\left(X_n-\mu+\mu-\hat{\mu}_X\right)^2\right] && \text{insert } 0=\mu-\mu\\
&= E\left[\frac{1}{N-1}\sum_{n=1}^{N}\left(\left(X_n-\mu\right)^2+2\left(X_n-\mu\right)\left(\mu-\hat{\mu}_X\right)+\left(\hat{\mu}_X-\mu\right)^2\right)\right] && \text{algebra}\\
&= \frac{1}{N-1}\sum_{n=1}^{N}E\left[\left(X_n-\mu\right)^2+2\left(X_n-\mu\right)\left(\mu-\hat{\mu}_X\right)+\left(\hat{\mu}_X-\mu\right)^2\right] && \text{linearity of expectation}\\
&= \frac{1}{N-1}\sum_{n=1}^{N}\left(\operatorname{var}[X]+2E\left[\left(X_n-\mu\right)\left(\mu-\hat{\mu}_X\right)\right]+\frac{\operatorname{var}[X]}{N}\right) && (1.20) \text{ and } (1.45)\\
&= \frac{1}{N-1}\left(N\operatorname{var}[X]+2E\left[\sum_{n=1}^{N}\left(X_n-\mu\right)\left(\mu-\hat{\mu}_X\right)\right]+\operatorname{var}[X]\right) && \text{expectations are constant}\\
&= \frac{1}{N-1}\left(N\operatorname{var}[X]+2NE\left[\left(\hat{\mu}_X-\mu\right)\left(\mu-\hat{\mu}_X\right)\right]+\operatorname{var}[X]\right) && \text{def. of sample mean}\\
&= \frac{1}{N-1}\left(N\operatorname{var}[X]-2NE\left[\left(\hat{\mu}_X-\mu\right)^2\right]+\operatorname{var}[X]\right) && \text{algebra}\\
&= \frac{1}{N-1}\left(N\operatorname{var}[X]-2N\frac{\operatorname{var}[X]}{N}+\operatorname{var}[X]\right) && (1.45)\\
&= \frac{1}{N-1}\left(N-1\right)\operatorname{var}[X] && \text{algebra}\\
&= \operatorname{var}[X] && \text{algebra.}
\end{aligned}
$$

Exercise 1.12, page 3

The basic idea is to use a computer to generate samples (X, Y) from a population with $\operatorname{cor}[X, Y] = 0.5$ and then estimate the probability

$$P(Y > 0 \mid X > 0) = \frac{P(Y > 0 \text{ and } X > 0)}{P(X > 0)}. \tag{22.12}$$

A code that generates samples (X, Y) from a bivariate normal population with $\operatorname{cor}[X, Y] = \rho$ and then computes relative frequencies is the following:

```
ntrials = 1000000
rho     = 0.5
xvar    = rnorm(ntrials)
nvar    = rnorm(ntrials)
yvar    = rho * xvar + sqrt(1-rho^2)*nvar
x.ge.0  = xvar >= 0
y.ge.0  = yvar >= 0
prob.yge0.cond.xge0 = sum(x.ge.0 & y.ge.0) / sum(x.ge.0)
print(prob.yge0.cond.xge0)
```

After running this code, it is found that $P(Y > 0|X > 0) \approx 2/3$ when $\rho = 0.5$. A figure showing the probabilities for other values of ρ is shown in fig. 22.1.

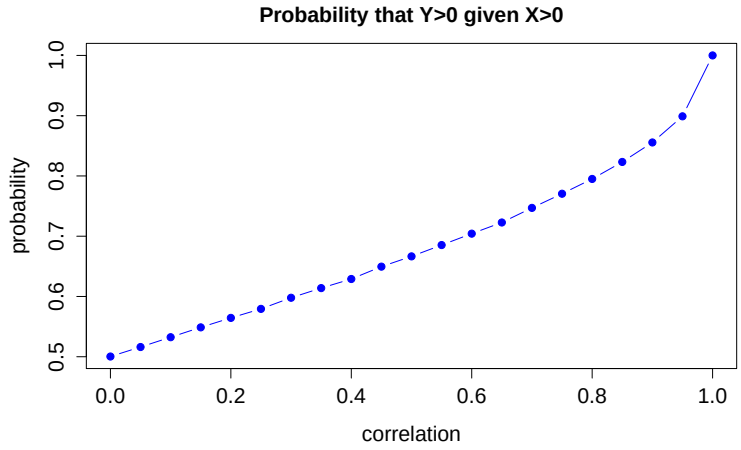


Figure 22.1 Probability that $Y > 0$ given that $X > 0$, as a function of the correlation $\text{cor}[X, Y]$ for a bivariate normal distribution with zero means

Exercise 1.13, page 3

The displacement in the x and y directions have the distributions

$$d_x = x_1 + x_2 + \cdots + x_N \sim N(0, N\sigma^2) \tag{22.13}$$

$$d_y = y_1 + y_2 + \cdots + y_N \sim N(0, N\sigma^2). \tag{22.14}$$

The total squared displacement is $r^2 = d_x^2 + d_y^2$. Since $d_x/\sigma/\sqrt{N}$ and $d_y/\sigma/\sqrt{N}$ are standardized Gaussians,

$$\frac{r^2}{N\sigma^2} = \left(\frac{d_x}{\sigma\sqrt{N}}\right)^2 + \left(\frac{d_y}{\sigma\sqrt{N}}\right)^2 \sim \chi_2^2. \tag{22.15}$$

The upper 5% percentile of a chi-squared distribution with 2 degrees of freedom is approximately 6. Therefore, the radius of the circle in which there is a 95% probability that the

particle is in the circle after 10 steps is

$$r_{95} \approx \sigma\sqrt{6N} = 2\sqrt{6 * 10} \approx 15.5. \tag{22.16}$$

Exercise 1.14, page 4

$$\begin{aligned}
\mathbb{E}[(X' - \alpha\hat{\mu})^2] &= \mathbb{E}[((X' - \mu_X) - (\alpha\hat{\mu}_X - \mu_X))^2] && \text{add } 0 = \mu_X - \mu_X \\
&= \mathbb{E}[(X' - \mu)^2] + \mathbb{E}[(\alpha\hat{\mu}_X - \mu_X)^2] && X' \text{ and } \hat{\mu}_X \text{ are independent} \\
&= \sigma_X^2 + \mathbb{E}[(\alpha\hat{\mu}_X - \alpha\mu_X + \alpha\mu_X - \mu_X)^2] && \text{definition of variance} \\
&= \sigma_X^2 + \mathbb{E}[(\alpha\hat{\mu}_X - \alpha\mu_X)^2] + \mathbb{E}[(\alpha\mu_X - \mu_X)^2] && \text{cross terms vanish} \\
&= \sigma_X^2 + \alpha^2\mathbb{E}[(\hat{\mu}_X - \mu_X)^2] + (\alpha - 1)^2\mu_X^2 && \text{basic identities} \\
&= \sigma_X^2 + \alpha^2\sigma_X^2/N + (\alpha - 1)^2\mu_X^2 && \mathrm{var}[\hat{\mu}_X] = \sigma^2/N
\end{aligned}$$

Differentiating with respect to α gives

$$\frac{d}{d\alpha}\mathbb{E}[(X' - \alpha\hat{\mu})^2] = 2\alpha\sigma_X^2/N + 2(\alpha - 1)\mu_X^2$$

Setting this to zero and solving for α gives the desired answer.

Exercise 2.1, page 8

(a) A function that tests for vanishing correlation is the following

```
cor.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS VANISHING CORRELATION BETWEEN
## TWO BI-VARIATE, NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   RHO: SAMPLE CORRELATION BETWEEN DATA1 AND DATA2
#   RHO.CRIT: 100*ALPHA% CRITICAL VALUE FOR THE CORRELATION
#   PVAL: P-VALUE OF THE STATISTIC RHO

if ( length(data1) != length(data2)) stop('data sets are not the same length')

ntot    = length(data1)

mean1 = sum(data1)/ntot
mean2 = sum(data2)/ntot

ssr1  = sum((data1-mean1)^2)
ssr2  = sum((data2-mean2)^2)
sscx  = sum((data1-mean1)*(data2-mean2))

rho.hat = sscx / sqrt(ssr1*ssr2)

t.crit  = qt(alpha/2,ntot-2,lower.tail=FALSE)
rho.crit = t.crit / sqrt( ntot - 2 + t.crit^2)

tval = rho.hat * sqrt(ntot-2) / sqrt(1-rho.hat^2)
pval = 2*pt(abs(tval),ntot-2,lower.tail=FALSE)

list(rho=rho.hat,rho.crit=rho.crit,pval=pval,tval=tval)
}
```

(b) Applying this function to lagged PDO indices can be done as follows:

```
>print('early period')
>npic       = 2:length(pdo1)
>pdo1.half1 = pdo1[npic-1]
>pdo1.half2 = pdo1[npic]
>print(cor.equal.test(pdo1.half1,pdo1.half2))$rho

[1] 0.05495659

$rho.crit
[1] 0.3808629

$pval
[1] 0.7854252

$tval
[1] 0.2751988
```

```
print('later period')
npic       = 2:length(pdo2)
pdo2.half1 = pdo2[npic-1]
pdo2.half2 = pdo2[npic]
print(cor.equal.test(pdo2.half1,pdo2.half2))

$rho
[1] 0.3836199

$rho.crit
[1] 0.3160319

$pval
[1] 0.01591744

$tval
[1] 2.526791
```

The year-to-year correlation for the first and second periods are 0.05 and 0.38. Only the second is large enough to reject the hypothesis of vanishing correlation at the 5% level, but only marginally (p-value is 2%).

(c) Performing the test on the whole period

```
print('full period')
npic       = 2:length(pdo)
pdo.half1  = pdo[npic-1]
pdo.half2  = pdo[npic]
print(cor.equal.test(pdo.half1,pdo.half2))

[1] "full period"
$rho
[1] 0.5081405

$rho.crit
[1] 0.1816386

$pval
[1] 4.964102e-09

$tval
[1] 6.326905

$rho.limits
[1] 0.3597967 0.6314279
```

For the whole 1950-2017 period, the correlation is 0.50, which is statistically significant at the 5% level.

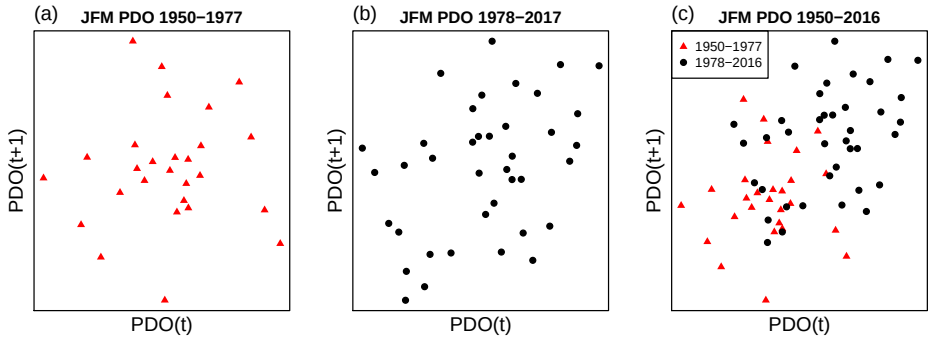


Figure 22.2 Scatter plots of the JFM PDO index with itself lagged by one year, for the periods 1950-1977 (a), 1978-2017 (b), and 1950-2017 (c).

Why do the above conclusions differ? Following the hint, we draw scatter diagrams for the three periods. The result is shown in fig. 22.2. There is only a weak positive relation for the two periods individually, shown in (a) and (b), but a much stronger linear relation in the last (c). Closer examination reveals that the values in the later period tend to be above those of the lower period. This "shift in mean" produces an apparent positive relation when the two data sets are pooled together. This apparent relation is not a local relation, in the sense that it exists no matter how the data is sampled (as assumed under stationarity), but instead it is a property of the data as a whole that arises when all the data are pooled together.

(d) The answer to the other questions are

null hypothesis: population correlation vanishes

test statistic:

$$\hat{\rho} = \frac{\sum_n (x_n - \hat{\mu}_X)(y_n - \hat{\mu}_Y)}{\sqrt{\left(\sum_n (x_n - \hat{\mu}_X)^2\right)\left(\sum_n (y_n - \hat{\mu}_X)^2\right)}} \tag{22.17}$$

value of the test statistic: 0.05 (1950-1977), 0.38 (1978-2017), 0.49 (1950-2017)

rejection region: $|\hat{\rho}| > \rho_c$ where ρ_c =0.38 (1950-1977), 0.32 (1978-2017), 0.24 (1950-2017)

final decision: can reject the null hypothesis $\rho = 0$ in the period 1950-1977, but not for 1978-2017 (marginally) nor for the whole period 1950-2017.

assumption: paired samples (X, Y) are iid from a Gaussian distribution.

(e) A function that tests for equality of means is the following:

```
mean.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS EQUALITY OF MEANS OF TWO IID
## NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N1]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N2]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   DIFF.MEAN: DIFFERENCE IN MEANS (MEAN1 - MEAN2)
#   DIFF.MEAN.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE DIFFERENCE IN MEANS
#   PVAL: P-VALUE OF THE T-STATISTIC
#   T: T-STATISTIC FOR DIFFERENCE IN MEANS
#   T.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE T STATISTIC
#   MEAN1: ESTIMATE OF THE MEAN OF DATA1
#   MEAN2: ESTIMATE OF THE MEAN OF DATA2
#   SPOOL: POOLED ESTIMATE OF THE STANDARD DEVIATION
#   DIFF.MEAN.LIMITS: [2] (1-ALPHA)100% CONFIDENCE LIMITS
#       FOR THE DIFFERENCE IN MEANS

n1    = length(data1)
n2    = length(data2)
dof   = n1 + n2 - 2

mean1 = sum(data1)/n1
mean2 = sum(data2)/n2
diff.mean = mean1 - mean2

ssr1  = sum((data1-mean1)^2)
ssr2  = sum((data2-mean2)^2)

spool = sqrt((ssr1 + ssr2)/dof)

tval   = diff.mean/spool/sqrt(1/n1 + 1/n2)
pval   = 2* pt(abs(tval),dof,lower.tail=FALSE)
t.crit = qt(alpha/2,dof,lower.tail=FALSE)

diff.mean.crit = t.crit * spool * sqrt(1/n1 + 1/n2)

list(diff.mean=diff.mean,diff.mean.crit=diff.mean.crit,pval=pval,
   tval=tval,t.crit=t.crit,mean1=mean1,mean2=mean2,spool=spool)
}
```

Applying this test to the PDO index gives:

```
> mean.equal.test(pdo1,pdo2)
> print(mean.equal.test(pdo1,pdo2))
$diff.mean
[1] -1.156262

$diff.mean.crit
[1] 0.4537684

$pval
[1] 3.22733e-06

$tval
[1] -5.08751

$t.crit
[1] 1.996564

$mean1
[1] -0.7405952

$mean2
[1] 0.4156667

$spool
[1] 0.9223707
```

(f) The critical value of t is computed in line 38 of the above code.

(g) Answers:

null hypothesis: population means are equal
test statistic: $(\hat{\mu}_X - \hat{\mu}_Y) / s_p / \sqrt{1/N_X + 1/N_Y}$
value of the test statistic: -5.09
rejection region: $|t| > 2.0$
final decision: the null hypothesis is rejected at the 5% level.
assumption: samples are iid from a Gaussian distribution.

(h) The $\rho = 0$ test is relevant to the equality-of-means test because the t-test assumes the samples are independent. The equality-of-variance test is relevant to to the equality-of-means test because the t-test assumes the variance are constant. Specifically, the fact that the year-to-year correlation in each sample is insignificant (or almost insignificant) implies that the independence assumption is reasonable for each sample separately (except possibly for the second period). The fact that the differences in variance are insignificant implies that the assumption of equal variances is reasonable.

Exercise 3.1, page 9

Since $X_1, \ldots, X_N \overset{iid}{\sim} \mathcal{N}(\mu, \sigma^2)$,

$$\hat{\mu} \sim \mathcal{N}(\mu, \sigma^2/N). \tag{22.18}$$

Hypothesis tests for the mean are based on

$$\frac{\text{observed} - \text{hypothesized}}{\text{standard error}} = \frac{\hat{\mu} - \mu}{\sigma/\sqrt{N}} \sim \mathcal{N}(0,1). \tag{22.19}$$

Replacing σ^2 by the sample estimate $\hat{\sigma}^2$ leads to the t-distribution:

$$t = \frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{N}} \sim t_{N-1}. \tag{22.20}$$

By definition of $t_{\alpha,N-1}$,

$$1 - \alpha = P\left(-t_{\alpha/2,N-1} \leq t < t_{\alpha/2,N-1}\right) \tag{22.21}$$

$$= P\left(-t_{\alpha/2,N-1} \leq \frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{N}} < t_{\alpha/2,N-1}\right) \tag{22.22}$$

$$= P\left(-t_{\alpha/2,N-1} \leq \frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{N}} \text{ and } \frac{\hat{\mu} - \mu}{\hat{\sigma}/\sqrt{N}} < t_{\alpha/2,N-1}\right) \tag{22.23}$$

$$\tag{22.24}$$

We want to solve for μ by itself:

$$1 - \alpha = P\left(\mu \leq \hat{\mu} + t_{\alpha/2,N-1}\hat{\sigma}/\sqrt{N} \text{ and } \hat{\mu} - t_{\alpha/2,N-1}\hat{\sigma}/\sqrt{N} < \mu\right) \tag{22.25}$$

$$= P\left(\hat{\mu} - t_{\alpha/2,N-1}\hat{\sigma}/\sqrt{N} < \mu \leq \hat{\mu} + t_{\alpha/2,N-1}\hat{\sigma}/\sqrt{N}\right). \tag{22.26}$$

An R code for computing the coverage of this confidence interval is

```
ntrials  = 100000
nsamp    = 20
mu       = 10
stdv     = 3
alpha    = 0.05

x        = array(rnorm(nsamp*ntrials,mean=mu,sd=stdv),dim=c(ntrials,nsamp))
x.mean   = rowMeans(x)
x.stdv   = sqrt(rowSums((x-x.mean)^2)/(nsamp-1)/nsamp)
tcrit    = qt(alpha/2,nsamp-1,lower.tail=FALSE)

mean.ci  = cbind(x.mean-tcrit*x.stdv,x.mean+tcrit*x.stdv)
coverage = sum(mu >= mean.ci[,1] & mu <= mean.ci[,2])/ntrials

print(paste('coverage=',coverage))
[1] "coverage= 0.94961"
```

The coverage is very close to 95%.

Exercise 3.2, page 10

An augmented `var.equal.test` that calculates a confidence interval is the following:

```
var.equal.test = function(data1,data2,alpha=0.05) {
### THIS FUNCTION TESTS EQUALITY OF VARIANCE OF TWO
### IID NORMALLY DISTRIBUTED RANDOM VARIABLES
###
# INPUT:
#   DATA1: [N1]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N2]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   F.MAX: RATIO OF VARIANCE, CONSTRUCTED TO BE GREATER THAN 1
#   F.CRIT: THE UPPER CRITICAL THRESHOLD OF SIGNIFICANCE
#   PVAL: P-VALUE OF THE F.MAX RATIO
#   VAR1: UNBIASED ESTIMATE OF THE VARIANCE OF DATA1
#   VAR2: UNBIASED ESTIMATE OF THE VARIANCE OF DATA2
#   RATIO: VAR1/VAR2
#   RATIO.LOWER: LOWER LIMIT OF ALPHA% CONFIDENCE INTERVAL FOR VARIANCE RATIO
#   RATIO.UPPER: UPPER LIMIT OF ALPHA% CONFIDENCE INTERVAL FOR VARIANCE RATIO

n1    = length(data1); n2    = length(data2)

mean1 = sum(data1)/n1; mean2 = sum(data2)/n2

var1  = sum((data1-mean1)^2)/(n1-1)
var2  = sum((data2-mean2)^2)/(n2-1)

if ( var1 > var2) {
  f.max  = var1/var2
  f.crit = qf(alpha/2,n1-1,n2-1,lower.tail=FALSE)
  pval   = 2*pf(f.max,n1-1,n2-1,lower.tail=FALSE)
} else {
  f.max  = var2/var1
  f.crit = qf(alpha/2,n2-1,n1-1,lower.tail=FALSE)
  pval   = 2*pf(f.max,n2-1,n1-1,lower.tail=FALSE)
}

f.upper = qf(alpha/2,n1-1,n2-1,lower.tail=FALSE)
f.lower = qf(alpha/2,n1-1,n2-1,lower.tail=TRUE)

ratio       = var1/var2
ratio.lower = ratio/f.upper
ratio.upper = ratio/f.lower

list(f.max=f.max,f.crit=f.crit,pval=pval,var1=var1,var2=var2,
  ratio=ratio,ratio.lower=ratio.lower,ratio.upper=ratio.upper)
}
```

Note that the critical F values `f.upper` and `f.lower` were recomputed after the `if` statement so that they correspond to the ratio `var1/var2` and not its reciprocal.

The output of this function for the PDO index is the following

```
> nbreak = which( year == 1977)
> pdo1 = pdo[1:nbreak]
> pdo2 = pdo[(nbreak+1):nyrs]
>
> ################################################
> ## TEST EQUALITY OF VARIANCE
> ################################################
> source(paste('var.equal.test.R',sep=""))
> print(var.equal.test(pdo1,pdo2))
$f.max
[1] 1.247894

$f.crit
[1] 2.074389

$pval
[1] 0.5522268

$var1
[1] 0.7420675

$var2
[1] 0.9260218

$ratio
[1] 0.80135

$ratio.lower
[1] 0.4043894

$ratio.upper
[1] 1.662312
```

Or if the data is entered in the opposite direction...

```
> var.equal.test(pdo2,pdo1)
$f.max
[1] 1.247894

$f.crit
[1] 2.074389

$pval
[1] 0.5522268

$var1
[1] 0.9260218

$var2
[1] 0.7420675

$ratio
[1] 1.247894

$ratio.lower
[1] 0.6015718

$ratio.upper
[1] 2.472864
```

Accordingly, the 95% confidence interval for the ratio of variances is (0.39, 1.63), or (0.61, 2.54), depending on how the two variances are ordered in the ratio. Note that this interval includes one, implying that the ratio is not significantly different from one at the 5% significance level, consistent with the p-value.

Exercise 3.3, page 10

A modified version of `cor.equal.test` that also computes a confidence interval is the following:

```
cor.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS VANISHING CORRELATION BETWEEN
## TWO BI-VARIATE, NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   RHO: SAMPLE CORRELATION BETWEEN DATA1 AND DATA2
#   RHO.CRIT: 100*ALPHA% CRITICAL VALUE FOR THE CORRELATION
#   PVAL: P-VALUE OF THE STATISTIC RHO
#   RHO.LIMITS: [2] (1-ALPHA/2)*100% CONFIDENCE LIMITS OF THE CORRELATION

if ( length(data1) != length(data2)) stop('data sets are not the same length')

ntot    = length(data1)

mean1 = sum(data1)/ntot
mean2 = sum(data2)/ntot

ssr1  = sum((data1-mean1)^2)
ssr2  = sum((data2-mean2)^2)
sscx  = sum((data1-mean1)*(data2-mean2))

rho.hat = sscx / sqrt(ssr1*ssr2)

t.crit  = qt(alpha/2,ntot-2,lower.tail=FALSE)
rho.crit = t.crit / sqrt( ntot - 2 + t.crit^2)

tval = rho.hat * sqrt(ntot-2) / sqrt(1-rho.hat^2)
pval = 2*pt(abs(tval),ntot-2,lower.tail=FALSE)

z       = 0.5 * log((1+rho.hat)/(1-rho.hat))
zc      = qnorm(alpha/2,mean=0,sd=1/sqrt(ntot-3),lower.tail=FALSE)
z.limits = c(z - zc, z+zc)

rho.limits = (exp(2*z.limits)-1)/(exp(2*z.limits)+1)

list(rho=rho.hat,rho.crit=rho.crit,pval=pval,tval=tval,rho.limits=rho.limits)
}
```

Applying this function to the PDO data gives the following

```
> ## test vanishing correlation for first half
> l0 = 1:(length(pdo1)-1)
> pdo1.l0 = pdo1[l0]; pdo1.l1 = pdo1[l0+1]
> print(cor.equal.test(pdo1.l0,pdo1.l1))
$rho
[1] 0.05495659

$rho.crit
[1] 0.3808629

$pval
[1] 0.7854252

$tval
[1] 0.2751988

$rho.limits
[1] -0.3319908  0.4260723
```

```
> ## test vanishing correlation for 2nd half
> l0 = 1:(length(pdo2)-1)
> pdo2.l0 = pdo2[l0]; pdo2.l1 = pdo2[l0+1]
> print(cor.equal.test(pdo2.l0,pdo2.l1))
$rho
[1] 0.3836199

$rho.crit
[1] 0.3160319

$pval
[1] 0.01591744

$tval
[1] 2.526791

$rho.limits
[1] 0.07748101 0.62365106
```

```
> ## test vanishing correlation for whole period
> l0 = 1:(length(pdo)-1)
> pdo.l0 = pdo[l0]; pdo.l1 = pdo[l0+1]
> print(cor.equal.test(pdo.l0,pdo.l1))
$rho
[1] 0.4915623

$rho.crit
[1] 0.2404471

$pval
[1] 2.398144e-05

$tval
[1] 4.550883

$rho.limits
[1] 0.2850066 0.6544904
```

The first confidence interval includes 0, implying that the correlation is not significantly difference from zero. The second confidence interval does not include 0, but only marginally. The third confidence interval does not include 0, hence the correlation is significantly different from zero. All three conclusions are consistent with the corresponding hypothesis tests.

Exercise 3.4, page 11

A modified version of `mean.equal.test` that also computes confidence limits for the difference in means is the following:

```
mean.equal.test = function(data1,data2,alpha=0.05) {
## THIS FUNCTION TESTS EQUALITY OF MEANS OF TWO IID
## NORMALLY DISTRIBUTED RANDOM VARIABLES
##
# INPUT:
#   DATA1: [N1]-DIMENSIONAL VECTOR OF DATA
#   DATA2: [N2]-DIMENSIONAL VECTOR OF DATA
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT LIST:
#   DIFF.MEAN: DIFFERENCE IN MEANS (MEAN1 - MEAN2)
#   DIFF.MEAN.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE DIFFERENCE IN MEANS
#   PVAL: P-VALUE OF THE T-STATISTIC
#   T: T-STATISTIC FOR DIFFERENCE IN MEANS
#   T.CRIT: 100*ALPHA% LEVEL CRITICAL VALUE OF THE T STATISTIC
#   MEAN1: ESTIMATE OF THE MEAN OF DATA1
#   MEAN2: ESTIMATE OF THE MEAN OF DATA2
#   SPOOL: POOLED ESTIMATE OF THE STANDARD DEVIATION
#   DIFF.MEAN.LIMITS: [2] (1-ALPHA)100% CONFIDENCE LIMITS
#     FOR THE DIFFERENCE IN MEANS

n1    = length(data1)
n2    = length(data2)
dof   = n1 + n2 - 2

mean1 = sum(data1)/n1
mean2 = sum(data2)/n2
diff.mean = mean1 - mean2

ssr1  = sum((data1-mean1)^2)
ssr2  = sum((data2-mean2)^2)

spool = sqrt((ssr1 + ssr2)/dof)

tval   = diff.mean/spool/sqrt(1/n1 + 1/n2)
pval   = 2* pt(abs(tval),dof,lower.tail=FALSE)
t.crit = qt(alpha/2,dof,lower.tail=FALSE)

diff.mean.crit = t.crit * spool * sqrt(1/n1 + 1/n2)

diff.mean.limits = diff.mean + c(-1,1)*diff.mean.crit

list(diff.mean=diff.mean,diff.mean.crit=diff.mean.crit,pval=pval,
   tval=tval,t.crit=t.crit,mean1=mean1,mean2=mean2,spool=spool,
   diff.mean.limits=diff.mean.limits)
}
```

Applying this to the PDO data gives

```
> print(mean.equal.test(pdo1,pdo2))
$diff.mean
[1] -1.156262

$diff.mean.crit
[1] 0.4537684

$pval
[1] 3.22733e-06

$tval
[1] -5.08751

$t.crit
[1] 1.996564

$mean1
[1] -0.7405952

$mean2
[1] 0.4156667

$spool
[1] 0.9223707

$diff.mean.limits
[1] -1.6100303 -0.7024935
```

The confidence limits do not include 0, so the difference in means is significantly different from zero. This is consistent with the hypothesis test.

Exercise 3.5, page 12

An R function for constructing a confidence interval for a correlation coefficient is the following:

```
cor.equal.boot = function(x,y,alpha=0.05,nboot=100000) {
## ESTIMATES (1-ALPHA)100% CONFIDENCE INTERVAL
## FOR THE CORRELATION COEFFICIENT USING BOOTSTRAP METHODS
## INPUT:
##  X[NSAMP]: FIRST DATA SET
##  Y[NSAMP]: SECOND DATA SET
##  ALPHA: EQUIVALENT SIGNIFICANCE LEVEL (DEFAULT = 5%)
##  NBOOT: NUMBER OF BOOTSTRAP SAMPLES (DEFAULT = 100000)
## OUTPUT:
##  CONFIDENCE LIMITS[2]
nsamp  = length(x)
if (length(y) != nsamp) stop('x and y must be same length')

npic   = sample(nsamp,nboot*nsamp,replace=TRUE)
x.boot = x[npic]
y.boot = y[npic]
dim(x.boot) = c(nboot,nsamp)
dim(y.boot) = c(nboot,nsamp)
x.boot = x.boot - rowMeans(x.boot)
y.boot = y.boot - rowMeans(y.boot)
cor.boot = rowSums(x.boot*y.boot)/sqrt(rowSums(x.boot^2)*rowSums(y.boot^2))
cor.boot.conf = quantile(cor.boot,probs=c(alpha/2,1-alpha/2))

cor.boot.conf
}
```

The result of running this function is:

```
nsamp   = 20
rho     = 0.5
nboot   = 100000
set.seed(310)
x         = rnorm(nsamp)
w         = rnorm(nsamp)
y         = rho * x + sqrt(1-rho^2)*w

rho.test1 = cor.equal.test(x,y)
rho.test2 = cor.test(x,y)

print(paste('Traditional Confidence interval:',
  paste(signif(rho.test1$rho.limits,4),collapse=', ')))
print(paste('Bootstrap confidence interval  :',
  paste(signif(cor.boot.conf,4),collapse=', ')))



[1] "Traditional Confidence interval: 0.3301, 0.8601"
[1] "Bootstrap confidence interval  : 0.328, 0.8761"
```

The bootstrap interval is nearly identical to the standard interval.

Exercise 4.1, page 14

An R function to perform the Wilcoxon-Mann-Whitney test is the following:

```
diff.mean.nonparametric = function(data1,data2,alpha=0.05) {
### PERFORMS MANN-WHITNEY TEST FOR A DIFFERENCE IN MEDIANS
### INPUT:
###   DATA1[N1]: SAMPLE 1
###   DATA2[N2]: SAMPLE 2
###   ALPHA: SIGNIFICANCE LEVEL (DEFAULT = 5%)
### OUTPUT: LIST$
###   ZVAL: Z-VALUE FOR THE NORMAL APPROXIMATION TO MANN-WHITNEY TEST
###   Z.CRIT: CRITICAL VALUE FOR ALPHA*100% SIGNIFICANCE
###   PVAL: P-VALUE OF THE TEST

n1           = length(data1)
n2           = length(data2)
rank.12      = rank(c(data1,data2))
rank.1       = rank.12[1:n1]
rank.2       = rank.12[1:n2+n1]
rank.sum.1   = sum(rank.1)
rank.sum.2   = sum(rank.2)

mu.rank.sum  = n1*(n1+n2+1)/2
var.rank.sum = n1*n2*(n1+n2+1)/12
if (rank.sum.1 > mu.rank.sum) continuity.cor = -0.5 else continuity.cor = 0.5
zval         = (rank.sum.1 - mu.rank.sum + continuity.cor)/sqrt(var.rank.sum)
z.crit       = qnorm(alpha/2,lower.tail=FALSE)
pval         = 2*pnorm(abs(zval),lower.tail=FALSE)

list(zval=zval,z.crit=z.crit,pval=pval,
  rank.sum.1=rank.sum.1,rank.sum.2=rank.sum.2)
}
```

Exercise 4.2, page 14

The result of applying this function to the PDO time series is:

```
> print(diff.mean.nonparametric(data1,data2))
$zval
[1] -4.243011

$z.crit
[1] 1.959964

$pval
[1] 2.205404e-05

$wval
[1] 219

$rank.sum.1
[1] 625

$rank.sum.2
[1] 1721
```

The z-value is significant, with a p-value of 0.022%, which is much less than 5%, hence the null hypothesis of equal medians should be rejected. This conclusion is consistent with the t-test from previous homeworks.

Exercise 4.3, page 14

An R function that computes Spearman's rank correlation and performs a hypothesis test is the following

```
cor.test.nonparametric = function(data1,data2,alpha=0.05) {
### PERFORMS RANK CORRELATION TEST
### INPUT:
###   DATA1[N1]: SAMPLE 1
###   DATA2[N2]: SAMPLE 2
###   ALPHA: SIGNIFICANCE LEVEL (DEFAULT = 5%)
### OUTPUT: LIST$
###   RHO.SPEARMAN = SPEARMAN'S RANK CORRELATION
###   ZVAL: Z-VALUE FOR THE NORMAL APPROXIMATION TO MANN-WHITNEY TEST
###   Z.CRIT: CRITICAL VALUE FOR ALPHA*100% SIGNIFICANCE
###   PVAL: P-VALUE OF THE TEST

if (length(data1) != length(data2)) stop('data sets not of equal length')
ntot   = length(data1)
r.mean = (ntot+1)/2
r.var  = ntot*(ntot^2-1)/12
rank.1 = rank(data1)
rank.2 = rank(data2)

rho.spearman = sum((rank.1-r.mean)*(rank.2-r.mean))/r.var

zval   = rho.spearman * sqrt(ntot - 1)
z.crit = qnorm(alpha/2,lower.tail=FALSE)
pval   = 2*pnorm(abs(zval),lower.tail=FALSE)

list(rho.spearman=rho.spearman,zval=zval,z.crit=z.crit,pval=pval)

}
```

Exercise 4.4, page 15

The result of testing correlations for the 3 data sets are as follows:

```
> ## test vanishing correlation for first half
> l0 = 1:(length(pdo1)-1)
> pdo1.l0 = pdo1[l0]; pdo1.l1 = pdo1[l0+1]
> print(cor.test.nonparametric(pdo1.l0,pdo1.l1))
$rho.spearman
[1] 0.05494505

$zval
[1] 0.2801659

$z.crit
[1] 1.959964

$pval
[1] 0.7793502
```

```
> ## test vanishing correlation for 2nd half
> l0 = 1:(length(pdo2)-1)
> print(cor.test.nonparametric(pdo2[l0],pdo2[l0+1]))
$rho.spearman
[1] 0.3487854

$zval
[1] 2.150058

$z.crit
[1] 1.959964

$pval
[1] 0.03155065
```

```
> ## test vanishing correlation for whole period
> l0 = 1:(length(pdo12)-1)
> print(cor.test.nonparametric(pdo12[l0],pdo12[l0+1]))
$rho.spearman
[1] 0.4570496

$zval
[1] 3.713089

$z.crit
[1] 1.959964

$pval
[1] 0.000204745
```

The null hypothesis is rejected in the first half, but not in the second half, or in the whole period. This result is consistent with the parametric correlation test.

Exercise 4.5, page 15

A function that tests difference in dispersions is

```
diff.dispersion = function(data1,data2,alpha=0.05) {
## PERFORMS WILCOXON SQUARED-RANK TEST ON ABSOLUTE DEVIATIONS |X - MEDIAN(X)|
### INPUT:
###   DATA1[N1]: SAMPLE 1
###   DATA2[N2]: SAMPLE 2
###   ALPHA: SIGNIFICANCE LEVEL (DEFAULT = 5%)
### OUTPUT: LIST$
###   ZVAL: Z-VALUE FOR THE NORMAL APPROXIMATION TO MANN-WHITNEY TEST
###   Z.CRIT: CRITICAL VALUE FOR ALPHA*100% SIGNIFICANCE
###   PVAL: P-VALUE OF THE TEST

n1              = length(data1)
n2              = length(data2)
ntot            = n1 + n2

median1         = median(data1)
median2         = median(data2)
absdev          = c(abs(data1-median1),abs(data2-median2))
rank.12         = rank(absdev)
rank.1          = rank.12[1:n1]
rank.2          = rank.12[1:n2+n1]
rank.sqr.sum.1  = sum(rank.1^2)
rank.sqr.sum.2  = sum(rank.2^2)

mu.rank.sum     = n1*(ntot+1)*(2*ntot+1)/6
var.rank.sum    = n1*(ntot-n1)/(ntot-1)/ntot*sum(((1:ntot)^2-mu.rank.sum/n1)^2)
zval            = (rank.sqr.sum.1 - mu.rank.sum)/sqrt(var.rank.sum)
z.crit          = qnorm(alpha/2,lower.tail=FALSE)
pval            = 2*pnorm(abs(zval),lower.tail=FALSE)

list(zval=zval,z.crit=z.crit,pval=pval,
rank.sqr.sum.1=rank.sqr.sum.1,rank.sqr.sum.2=rank.sqr.sum.2)

}
```

The result of running this function on the PDO data set is

```
> print(diff.dispersion(data1,data2))
$zval
[1] -0.9649911

$z.crit
[1] 1.959964

$pval
[1] 0.3345493

$mu.rank.sum
[1] 44114

$var.rank.sum
[1] 32644360

$rank.sqr.sum.1
[1] 38600.5

$rank.sqr.sum.2
[1] 68532
```

The null hypothesis is not rejected, and this is consistent with previous homeworks.

Exercise 5.1, page 17

An R function that calculates the sample autocorrelation function is the following:

```
acf.brute = function(x,lag.max=NULL) {
## COMPUTES THE AUTOCORRELATION FUNCTION OF A TIME SERIES
## INPUT:
##   X: [NTOT]-LENGTH NUMERICAL VECTOR OF THE TIME SERIES
##   LAG.MAX: MAXUMIM LAG TO COMPUTE
##      (DEFAULT = MAX(1,FLOOR(10*log10(LENGTH(X)))))
## OUTPUT:
##   X.ACF: [1:(LAG.MAX+1)] AUTOCORRELATION FUNCTION OF X
##      FOR LAGS 0 TO LAG.MAX

ntot   = length(x)

if ( is.null(lag.max)) lag.max = max(1,floor(10*log10(ntot)))
x.anom = x - mean(x)
x.ssa  = sum(x.anom^2)

x.acf  = numeric(lag.max+1)
for ( lag in 0:lag.max) {
npic  = 1:(ntot-lag)
x.acf[lag+1] = sum(x.anom[npic]*x.anom[npic+lag])/x.ssa
}

x.acf
}
```

Result of applying this function to `x = rnorm(20)` is:

```
> set.seed(1)
> x = rnorm(20)
> acf.mine = acf.brute(x)
> acf.R    = acf(x)
> print(acf.mine)
 [1]  1.00000000 -0.12228595 -0.18521142 -0.04940401  0.14719994
 [6] -0.28278013 -0.25514538  0.21242725  0.09722174 -0.11993122
[11] -0.18075942  0.28550651 -0.06259741  0.09444605
> print(as.numeric(acf.R$acf))
 [1]  1.00000000 -0.12228595 -0.18521142 -0.04940401  0.14719994
 [6] -0.28278013 -0.25514538  0.21242725  0.09722174 -0.11993122
[11] -0.18075942  0.28550651 -0.06259741  0.09444605
```

Exercise 5.2, page 18

The stationary distribution of the AR1 process is normal with mean $k/(1-\phi_1)$ and variance $\sigma_W^2/(1-\phi^2)$. An R function that generates a time series from an AR1 model is the following

```
ar1.ts = function(phi1,ntot,stdv=1,constant=0,iseed=1) {
## GENERATE A REALIZATION FROM THE AR1 MODEL
## X(T+1) = PHI1 * X(T) + W(T) + CONSTANT
## WHERE W(T) IS GAUSSIAN WHITE NOISE WITH MEAN 0 AND ST. DEV. 'STDV'
## INITIAL CONDITION IS RANDOMLY DRAWN FROM STATIONARY DISTRIBUTION
## INPUT:
#    PHI1: AUTOREGRESSIVE PARAMETER
#    NTOT: LENGTH OF THE DESIRED TIME SERIES
#    STDV: STANDARD DEVIATION OF THE NOISE (DEFAULT = 1)
#    CONSTANT: CONSTANT TERM IN THE AR1 MODEL (DEFAULT = 0)
#    ISEED: SEED FOR THE RANDOM NUMBER GENERATOR (DEFAULT = 1)
#  OUTPUT:
#    X: [NTOT]- RANDOM TIME SERIES FROM AR(1) PROCESS

set.seed(iseed)

x      = numeric(ntot)
x[1]   = rnorm(1,mean=constant/(1-phi),sd=stdv/sqrt(1-phi^2))
w      = rnorm(ntot,sd=stdv)
for ( n in 2:ntot) x[n] = phi1 * x[n-1] + w[n] + constant
x }
```

Exercise 5.3, page 18

Time series and corresponding autocorrelation functions can be generated as follows:

```
pdf('ts.acf.pdf')
par(mfrow=c(3,2),mar=c(5,5,3,2),cex.axis=1.5,cex.lab=1.5,cex.main=1.5)
ntot = 50
for ( phi1 in c(0,0.5,0.9)) {
  x    = ar1.ts(phi1,ntot,iseed=2)
  plot(x,type="l",col='blue',xlab='time',lwd=2,
    main=paste('Time Series for phi1=',signif(phi1,2)))
  acf.mine = acf.brute(x)
  lag.max  = length(acf.mine)-1
  yrange   = c(-0.4,1)
  plot(0:lag.max,acf.mine,xlab="lag",ylab="ACF",type="l",ylim=yrange,lwd=2,
     main=paste('Autocorrelation for phi1=',signif(phi1,2)))
  acf.exact = phi1^(1:ntot-1)
  par(new=TRUE)
  plot(1:ntot-1,acf.exact,type="l",lty="solid",col="red",xlim=c(0,lag.max),
    ylim=yrange,xlab="",ylab="",axes=FALSE,lwd=2)
  abline(h=2/sqrt(ntot),col="blue",lty="dashed",lwd=2)
  abline(h=-2/sqrt(ntot),col="blue",lty="dashed",lwd=2)
  legend('topright',legend=c('sample','population'),col=c('black','red'),lwd=2)
}
dev.off()
```

A sample plot is shown in fig. 22.3.

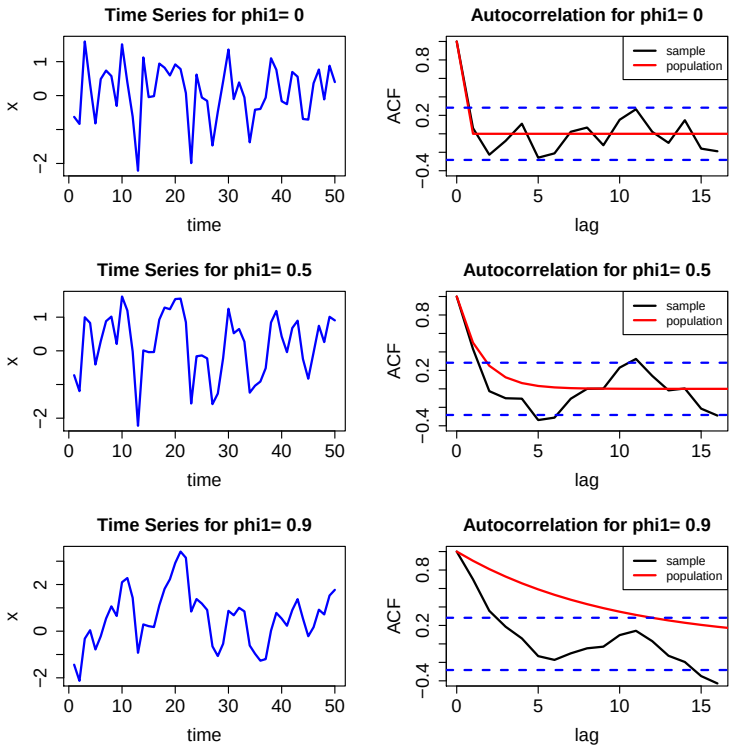


Figure 22.3 Sample time series and corresponding autocorrelation function from an AR1 model.

As $\phi_1 \to 1$, the time series get "smoother" and the ACF decays more slowly. The sample ACF for $\phi_1 = 0.9$ is "surprising" because it looks very different from the true ACF: the sample ACF oscillates, changes sign, and exceeds the 95% significance level. However, these results are not inconsistent with theory, because we expect an average of 0.75 autocorrelations out of 15 to exceed the 95% level. Since the true ACF is relatively high for all lags less than 15, any high correlation due to sampling variability will exhibit a smooth oscillation. The ACF is especially misleading because the trough and peak occur at multiples of 7, suggesting the existence of a oscillation with a period of 14, when in fact the process is strictly AR(1).

Exercise 5.4, page 18

An R code that generates time series, splits it into 2, and performs a t-test, is the following:

```
source('/Users/delsole/R/delsole_tools/mean.equal.test.R')
ntrials = 10000
nx      = 25
ny      = 25
tval    = numeric(ntrials)
yrange  = c(0,0.4)
for ( phi1 in c(0,0.5,0.9)) {
  for ( nt in 1:ntrials) {
    z = ar1.ts(phi1,nx+ny,iseed=nt)
    x = z[1:nx]
    y = z[(nx+1):(nx+ny)]
    tval[nt] = mean.equal.test(x,y)$tval
  }
  t1 = 1 + 2 * sum(phi1^(1:length(x))*(1-(1:length(x)/ntot)))
  ftitle=paste('Histogram of T for phi1=',signif(phi1,2))
  hist(tval,col="grey",freq=FALSE,main=ftitle,ylim=yrange)
  dof = nx + ny - 2
  curve(dt(x,dof),add=TRUE)
}
```

The plot for a particular realization is shown in fig. 22.4. The histograms show that as the autocorrelation parameter ϕ_1 increases, the variance of t-values increases. This implies that performing the t-test on serially correlated data will tend to incorrectly reject the null hypothesis at a greater rate than the chosen significance level. Equivalently, a sample that is deemed "significant" by the standard decision rule may actually be insignificant when autocorrelation is taken into account.

Exercise 5.5, page 19

Taking the expectations of both sides of (5.2) gives

$$\mathbb{E}[X_t] = \phi_1 \mathbb{E}[X_{t-1}] + \phi_2 \mathbb{E}[X_{t-2}] + ... + \phi_p \mathbb{E}[X_{t-p}] + \mathbb{E}[W_t] + k. \qquad (22.27)$$

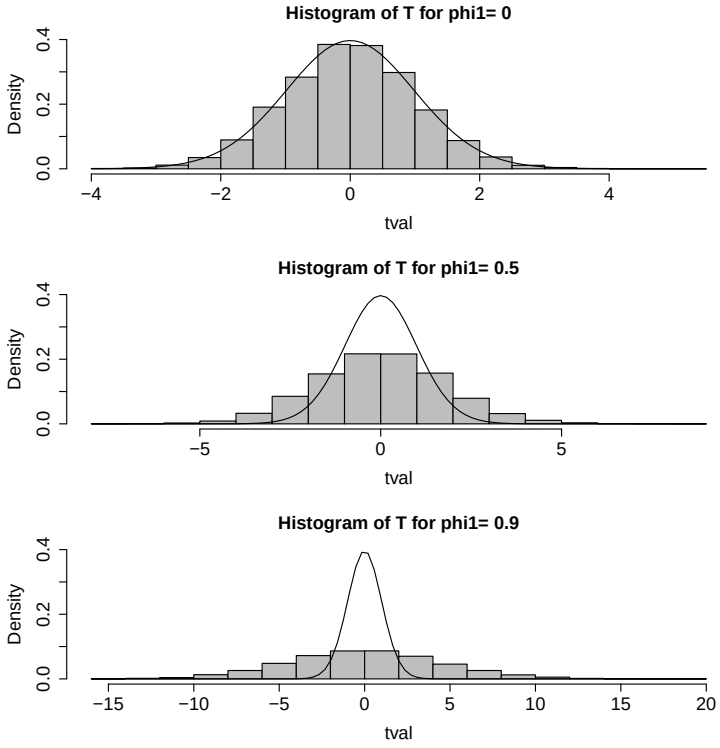


Figure 22.4 Histograms of the t-statistic for testing equality of means calculated from time series generated by an AR1 model with $\phi_1 = 0, 0.5, 0.9$. The time series are each of length 25.

Since the process is stationary, all expectations of $X_{t+\tau}$ are identical. Solving for the expectation therefore gives

$$\mathbb{E}[X_t] = \frac{k}{1 - \phi_1 - \phi_2 - \cdots - \phi_p}, \tag{22.28}$$

where $\mathbb{E}[W_t] = 0$ has been used.

To calculate the variance, first subtract the mean from both sides of (5.2), which yields

$$X_t - \mu = \phi_1 (X_{t-1} - \mu) + \phi_2 (X_{t-2} - \mu) + ... + \phi_p (X_{t-p} - \mu) + W_t, \tag{22.29}$$

where the k term vanishes because of (22.28). Multiply both sides by $X_t - \mu$ and take expectations:

$$\operatorname{var}[X_t] = \phi_1 \operatorname{cov}[X_t, X_{t-1}] + \phi_2 \operatorname{cov}[X_t, X_{t-2}] + \cdots + \phi_p \operatorname{cov}[X_t, X_{t-p}] + \operatorname{cov}[X_t, W_t]. \tag{22.30}$$

The last term can be evaluated by substituting (5.2) for X_t, which yields

$$\operatorname{cov}[X_t, W_t] = \operatorname{cov}[\phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + W_t + k, W_t] = \operatorname{var}[W_t], \tag{22.31}$$

since W_t is independent of $X_{t-\tau}$ for all $\tau \geq 1$. Substituting the relation $\operatorname{cov}[X_t, X_{t-\tau}] = \operatorname{var}[X_t]\rho_\tau$ and solving for the variance gives

$$\operatorname{var}[X_t] = \frac{\operatorname{var}[W_t]}{1 - \rho_1\phi_1 - \rho_2\phi_2 - \cdots - \rho_p\phi_p}. \tag{22.32}$$

Exercise 5.6, page 19

The expectation of Y_t is

$$\mathbb{E}[Y_t] = \frac{1}{2K+1} \sum_{k=-K}^{K} \mathbb{E}[X_{t+k}] = 0. \tag{22.33}$$

The variance of Y_t is

$$\operatorname{var}[Y_t] = \frac{1}{(2K+1)^2} \sum_{k=-K}^{K} \sum_{j=-K}^{K} \mathbb{E}\left[X_{t+k} X_{t+j}\right].$$

Since X_t is white noise, $\mathbb{E}\left[X_{t+k} X_{t+j}\right]$ vanishes except when $k = j$. Thus, the double sum collapses to a single sum:

$$\begin{aligned} \operatorname{var}[Y_t] &= \frac{1}{(2K+1)^2} \sum_{k=-K}^{K} \sigma^2 \\ &= \frac{\sigma^2}{(2K+1)}. \end{aligned}$$

The autocovariance function is

$$\operatorname{cov}[Y_{t+\tau}, Y_t] = \frac{1}{(2K+1)^2} \sum_{k=-K}^{K} \sum_{j=-K}^{K} \mathbb{E}\left[X_{t+\tau+k} X_{t+j}\right] \tag{22.34}$$

Since X_t is white noise, $\mathbb{E}\left[X_{t+\tau+k}X_{t+j}\right]$ vanishes except when $k+\tau=j$. When $\tau=0$, this identity holds $2K+1$ times. When $\tau=1$, this identity holds $2K$ times. For general τ, the identity holds $2K+1-|\tau|$ times. Thus

$$\operatorname{cov}[Y_{t+\tau},Y_t]=\frac{\sigma^2}{(2K+1)^2}\left(2K+1-|\tau|\right)\quad\text{for }|\tau|<=2K+1 \tag{22.35}$$

The autocorrelation is therefore

$$\operatorname{cor}[Y_{t+\tau},Y_t]=\frac{\operatorname{cov}[Y_{t+\tau},Y_t]}{\operatorname{var}[Y_t]}=1-\frac{|\tau|}{2K+1}\quad\text{for }|\tau|<=2K+1.$$

For $|\tau|>2K+1$, the autocorrelation of Y_t vanishes. Thus, the autocorrelation function of Y_t decreases linearly from 1 until it vanishes, and then remains 0 thereafter.

The fact that running means are correlated even when the underlying process is not follows from the fact that a running mean is computed from the time mean within a window, so when the two windows intersect they contain the same data, and hence are correlated.

Exercise 6.1, page 23

A spectral averaging function is the following:

```
spectrum.smooth = function(pgram,window,alpha=0.1) {
## COMPUTES SPECTRAL RUNNING MEAN AS:
##   SPEC_AVE = SMOOTHED PERIODOGRAM FROM M = -WINDOW TO M = WINDOW
## INPUT:
##    PGRAM[NTOT/2]: PERIODOGRAM VALUES FROM PERIODOGRAM.R
##    WINDOW: HALF-WIDTH OF AVERAGING WINDOW
##    ALPHA: THE (1-ALPHA)100% LEVEL FOR THE CONFIDENCE LIMITS
## OUTPUT: LIST
##    PGRAM.AVE[NTOT/2]: AVERAGED PERIODOGRAM
##    FREQ[NTOT/2]: FREQUENCIES (CYCLES PER TIME STEP)
##    BW[2]: BANDWIDTH C(-WINDOW/NTOT,WINDOW/NTOT)
##    CI.LIMITS[2]: PROPORTIONALITY CONSTANT FOR LOWER AND UPPER CONFIDENCE LIMITS
##       (DOF/CHI-SQUARE[ALPHA/2, DOF], DOF/CHI-SQUARE[1-ALPHA/2,DOF])
##    PGRAM[NTOT/2]: REPEATS PERIODOGRAM (FOR LATER PLOTTING)

  nhalf     = length(pgram)
  ntot      = nhalf * 2
  freq      = 1:(ntot/2) / ntot
  dof       = 2*(2*window + 1)

  pgram.ave = as.numeric(rep(NA,nhalf))
  for ( n in (window+1):(nhalf-window-1)) pgram.ave[n] =
      mean(pgram[(n-window):(n+window)])

  bw = c(-window,window)/ntot
  ci = dof/qchisq(c(alpha/2,1-alpha/2),dof)

  list(pgram.ave=pgram.ave,freq=freq,bw=bw,ci=ci,pgram=pgram)
}
```

Exercise 6.2, page 23

The output from the spectral estimation is:

```
> set.seed(1)
> x.sd        = 1
> window      = 5
> ntot        = 64
> freq        = 1:(ntot/2) / ntot
> x           = rnorm(ntot,sd=x.sd)
> x.pgram     = periodogram(x)
> x.pgram.smooth = spectrum.smooth(x.pgram,window)
> cbind(freq,x.pgram,x.pgram.smooth$pgram.ave)
           freq    x.pgram
 [1,] 0.015625 0.46954028        NA
 [2,] 0.031250 0.08678869        NA
 [3,] 0.046875 0.20078007        NA
 [4,] 0.062500 0.08812362        NA
 [5,] 0.078125 0.73401690        NA
 [6,] 0.093750 2.15621581 0.8100061
 [7,] 0.109375 0.40955937 0.7723600
 [8,] 0.125000 0.26816996 0.8389093
 [9,] 0.140625 1.14349297 0.8596130
[10,] 0.156250 1.03053739 0.9965911
[11,] 0.171875 2.32284234 1.0368931
[12,] 0.187500 0.05543276 0.9298071
[13,] 0.203125 0.81883090 1.0700647
[14,] 0.218750 0.42852123 1.0775075
[15,] 0.234375 1.59488238 1.0317232
[16,] 0.250000 1.17733897 0.9613558
[17,] 0.265625 0.97827006 0.7704560
[18,] 0.281250 1.95239234 0.8123769
[19,] 0.296875 0.35004103 0.8457826
[20,] 0.312500 0.63986523 0.9243640
[21,] 0.328125 0.25649685 0.8015068
[22,] 0.343750 0.22294430 0.7269105
[23,] 0.359375 0.51656297 0.7570746
[24,] 0.375000 1.18629292 0.6136065
[25,] 0.390625 1.29291736 0.6251425
[26,] 0.406250 0.24345292 0.5862000
[27,] 0.421875 0.35677905        NA
[28,] 0.437500 1.31007588        NA
[29,] 0.453125 0.37424337        NA
[30,] 0.468750 0.47693653        NA
[31,] 0.484375 0.21149808        NA
[32,] 0.500000 1.95934289        NA
```

The plot produced by

```
spectrum.plot(x.pgram.smooth)
abline(h=x.sd,lty="dashed",lwd=2)
ftitle      = paste("White Noise (N=",ntot,")",sep="")
title(main=ftitle)
```

is shown in fig. 22.5. To judge whether the spectrum is consistent with a white noise process, we must to decide whether the power spectrum is "flat," or constant. Technically, to do this, we would compare the ratio of spectral estimates with an appropriate threshold from an F-distribution. Instead of doing this, we crudely use the confidence interval by checking whether the *differences* between spectral estimates exceed the confidence interval. It can be seen that no two spectral estimates differ by more than a confidence interval, so we conclude the spectrum is consistent with a white noise process.

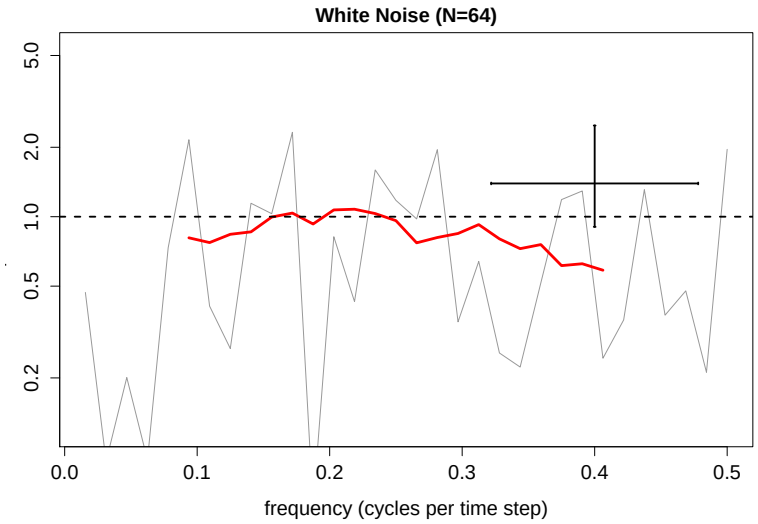


Figure 22.5 Periodogram and spectral average of a white noise process. The error bars in the upper right show the 90% confidence interval and the bandwidth. The horizontal dashed line shows the theoretical power spectrum of the process.

Exercise 6.3, page 23

The theoretical power spectrum for $GWN(0, \sigma^2)$ is derived from the Discrete Fourier Transform

$$p(\omega) = c_0 + 2 \sum_{\tau=1}^{\infty} c_\tau \cos(\omega\tau) \tag{22.36}$$

For GWN, $c_\tau = 0$ for all $\tau \geq 1$, so the power spectrum is easily evaluated to be

$$p(\omega) = c_0 = \sigma^2. \tag{22.37}$$

This power spectrum can be superimposed on the plot using the commands:

```
exact.pgram = x.sd^2
abline(h=exact.pgram,lwd=2,lty="dashed",col="black")
```

The difference between the theoretical and estimated power spectra lies within a confidence interval, so we conclude the estimated spectra is consistent with the theoretical power spectrum.

Exercise 6.4, page 24

The output from the spectral estimation is:

```
> window      = 5
> x.pgram     = periodogram(x)
> x.pgram.smooth = spectrum.smooth(x.pgram,window)
> cbind(freq,x.pgram,x.pgram.smooth$pgram.ave)
           freq    x.pgram
 [1,] 0.015625 4.12171567        NA
 [2,] 0.031250 4.28688721        NA
 [3,] 0.046875 2.18587695        NA
 [4,] 0.062500 0.80868492        NA
 [5,] 0.078125 5.00694613        NA
 [6,] 0.093750 8.68718662 3.0192022
 [7,] 0.109375 0.45912976 2.6531974
 [8,] 0.125000 0.69869530 2.3055704
 [9,] 0.140625 2.36259034 2.1226395
[10,] 0.156250 2.03132280 2.1816207
[11,] 0.171875 2.56218866 1.8167134
[12,] 0.187500 0.09566224 1.1014335
[13,] 0.203125 0.46299053 1.1621120
[14,] 0.218750 0.17363724 1.1047530
[15,] 0.234375 1.45747803 0.9041859
[16,] 0.250000 0.99296599 0.7397831
[17,] 0.265625 0.81910720 0.5225284
[18,] 0.281250 1.12659371 0.5375536
[19,] 0.296875 0.06774579 0.5196011
[20,] 0.312500 0.15635229 0.5302911
[21,] 0.328125 0.22289282 0.4021078
[22,] 0.343750 0.17238675 0.3240738
[23,] 0.359375 0.26093924 0.2755012
[24,] 0.375000 0.26551307 0.1774613
[25,] 0.390625 0.29122684 0.1816354
[26,] 0.406250 0.04746208 0.1769702
[27,] 0.421875 0.13459156        NA
[28,] 0.437500 0.28480903        NA
[29,] 0.453125 0.04815466        NA
[30,] 0.468750 0.11366153        NA
[31,] 0.484375 0.10503442        NA
[32,] 0.500000 0.81701638        NA
```

A plot of the estimated power spectra is shown in fig. 22.6.

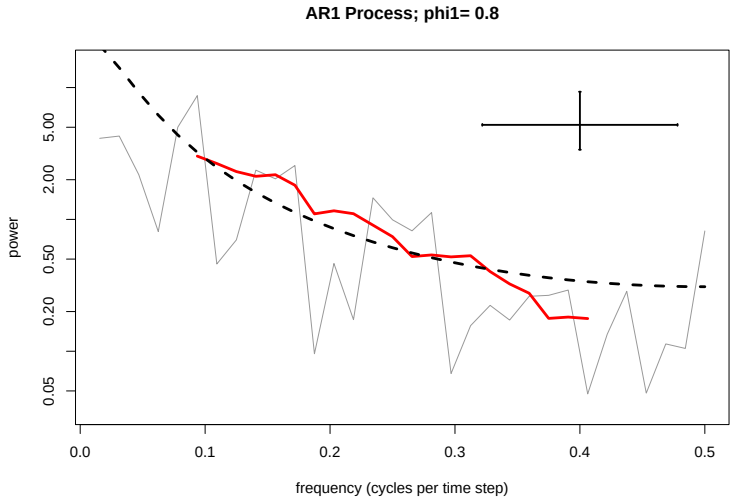


Figure 22.6 Periodogram and spectral average of an AR(1) process with $\phi = 0.8$. The error bars in the upper right show the 90% confidence interval and the bandwidth. The horizontal dashed line shows the theoretical power spectrum of the process.

Exercise 6.5, page 24

The theoretical power spectrum for an AR(1) process was derived in the text as

$$p(\omega) = \frac{\sigma_\epsilon^2}{1 + \phi_1^2 - 2\phi_1 \cos(2\pi f)}. \tag{22.38}$$

R commands for evaluating this power spectrum are as follows:

```
freq          = 1:(ntot/2) / ntot
x.spec.exact = 1/(1+phi1^2-2*phi1*cos(2*pi*freq))
lines(freq,x.spec.exact,lty="dashed",lwd=3)
```

The final plot is shown in fig. 22.6.

Exercise 6.6, page 24

The power spectrum for NINO3.4 can be estimated using the following:

```
fname       = paste(dir.enso,'nina34.data',sep='')
line1       = scan(fname,nlines=1,quiet=TRUE)
nino34.yr   = line1[1]:line1[2]
nino34.nyrs = length(nino34.yr)
data2       = read.table(fname,skip=1,nrows=nino34.nyrs,na.strings=-99.99,col.names=cnames)

## CREATE VECTOR WITH NINO34 TIME SERIES AND TIME AXIS
nino34.ts   = as.numeric(t(as.matrix(data2[,1:12+1])))
nino34.time = seq(from=data2[1,1],length.out=length(nino34.ts),by=1/12)

nomit = is.na(nino34.ts)
nino34.ts = nino34.ts[!nomit]
nino34.time = nino34.time[!nomit]

### FIND HIGHLY COMPOSITE NUMBER
chunk = 10
length.composite = length(nino34.time)
while ( nextn(length.composite) > length(nino34.time)) length.composite =
  length.composite - chunk
length.composite = nextn(length.composite)

### INCLUDE ONLY HIGHLY COMPOSITE NUMBER OF SAMPLES
nino34.ts   = nino34.ts  [1:length.composite + length(nino34.ts) - length.composite]
nino34.time = nino34.time[1:length.composite + length(nino34.ts) - length.composite]

nino34.pgram     = periodogram(nino34.ts)
nino34.pgram.smooth = spectrum.smooth(nino34.pgram,window)
spectrum.plot(nino34.pgram.smooth)
```

The resulting power spectra is shown in fig. 22.7.

There are significant peaks near $1/12 \approx 0.083$ and $1/6 \approx 0.17$, which are obviously associated with the annual and sub-annual cycles.

The largest peak occurs near $1/48 \approx 0.02$, which is near 4-years. However, the peak does not rise outside the confidence interval relative to its neighboring values, so it is not objectively clear that the peak is at 4-years or at even lower frequencies. The characterization that ENSO has dominant oscillatory periods around 4-7 years is not unambiguous.

Exercise 7.1, page 25

$$\mathbb{E}[\mathbf{y}] = \mathbb{E}\left[\sum_{j=1}^{N} c_j \mathbf{x}_j\right] = \sum_{j=1}^{N} c_j \mathbb{E}[\mathbf{x}_j] = c_1 \boldsymbol{\mu}_1 + \ldots c_N \boldsymbol{\mu}_N$$

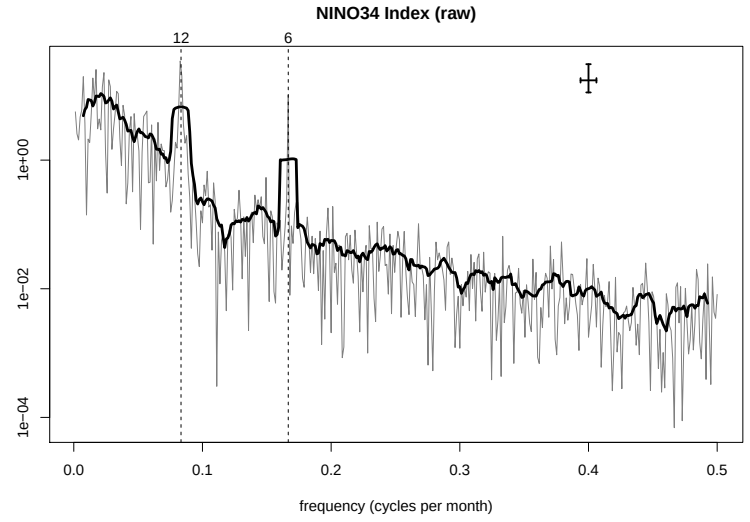


Figure 22.7 Periodogram and spectral average of the raw NINO3.4 index. The error bars in the upper right show the 90% confidence interval and the bandwidth.

$$\begin{aligned}
\mathrm{cov}[\mathbf{y}] =& \mathbb{E}\left[\left(\sum_{j=1}^{N} c_j\left(\mathbf{x}_j-\boldsymbol{\mu}_j\right)\right)\left(\sum_{k=1}^{N} c_k\left(\mathbf{x}_k-\boldsymbol{\mu}_k\right)\right)^T\right] \\
=& \sum_{j=1}^{N}\sum_{k=1}^{N} c_j c_k \mathbb{E}\left[\left(\mathbf{x}_j-\boldsymbol{\mu}_j\right)\left(\mathbf{x}_k-\boldsymbol{\mu}_k\right)^T\right] \\
=& \sum_{j=1}^{N} c_j^2 \boldsymbol{\Sigma}_j \qquad \text{due to independence of } \mathbf{x}_j \text{ and } \mathbf{x}_k
\end{aligned}$$

Exercise 7.2, page 25

Since $\mathbf{y}$ is a linear combination of Gaussian variables, it is itself Gaussian and hence only the mean and covariance need be computed. These are $\mathbb{E}[\mathbf{y}] = \mathbf{A}\boldsymbol{\mu} + \mathbf{b}$, and

$$\begin{aligned}
\mathrm{cov}(\mathbf{y}) =& \mathbb{E}\left[\left(\mathbf{y}-\mathbb{E}[\mathbf{y}]\right)\left(\mathbf{y}-\mathbb{E}[\mathbf{y}]\right)^T\right] \\
=& \mathbb{E}\left[\left(\mathbf{A}\mathbf{x}+\mathbf{b}-\mathbf{A}\boldsymbol{\mu}-\mathbf{b}\right)\left(\mathbf{A}\mathbf{x}+\mathbf{b}-\mathbf{A}\boldsymbol{\mu}-\mathbf{b}\right)^T\right] \\
=& \mathbb{E}\left[\mathbf{A}\left(\mathbf{x}-\boldsymbol{\mu}\right)\left(\mathbf{x}-\boldsymbol{\mu}\right)^T\mathbf{A}^T\right] \\
=& \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T
\end{aligned}$$

Thus, $\mathbf{y} \sim \mathrm{N}(\mathbf{A}\boldsymbol{\mu} + \mathbf{b}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T)$

Exercise 7.3, page 26

(a) $\mathbf{A}$ and $\mathbf{A}^T$ have the same diagonal, so they have the same trace.

(b) $\mathrm{tr}[\mathbf{AB}] = \mathrm{tr}\left[(\mathbf{AB})^T\right] = \mathrm{tr}\left[\mathbf{B}^T\mathbf{A}^T\right] = \mathrm{tr}[\mathbf{A}^T\mathbf{B}^T]$. The first equality follows from part (a), the others from properties of transpose and trace discussed in the chapter.

(c) $\mathrm{tr}[\mathbf{A}] = \mathrm{tr}[\mathbf{S\Lambda S}^{-1}] = \mathrm{tr}[\mathbf{\Lambda S}^{-1}\mathbf{S}] = \mathrm{tr}[\mathbf{\Lambda I}] = \mathrm{tr}[\mathbf{\Lambda}]$.

(d)

$$\begin{aligned}
\mathrm{tr}[\mathbf{A}^k] &= \mathrm{tr}[\left(\mathbf{S\Lambda S}^{-1}\right)^k] && \text{definition}\\
&= \mathrm{tr}[\left(\mathbf{S\Lambda S}^{-1}\right)\left(\mathbf{S\Lambda S}^{-1}\right)\ldots\left(\mathbf{S\Lambda S}^{-1}\right)] && \text{expand the power}\\
&= \mathrm{tr}[\mathbf{S\Lambda}^k\mathbf{S}^{-1}] && \text{the intermediate terms } \mathbf{S}^{-1}\mathbf{S} = \mathbf{I} \text{ cancel out}\\
&= \mathrm{tr}[\mathbf{\Lambda}^k]
\end{aligned}$$

Alternatively, $\mathbf{A}^k\mathbf{x} = \lambda^k\mathbf{x}$ implies that λ^k is an eigenvalue of $\mathbf{A}^k$, in which case the result follows from part c.

(e)

$$\mathrm{tr}[\mathbf{A}^{-1}] = \mathrm{tr}\left[\left(\mathbf{S\Lambda S}^{-1}\right)^{-1}\right] = \mathrm{tr}\left[\mathbf{S}^{-1}\mathbf{\Lambda}^{-1}\mathbf{S}\right] = \mathrm{tr}\left[\mathbf{\Lambda}^{-1}\left(\mathbf{SS}^{-1}\right)\right] = \mathrm{tr}\left[\mathbf{\Lambda}^{-1}\right] = \sum_{m=1}^{M}\lambda_m^{-1}.$$

The last equality follows from the fact that the inverse of a *diagonal* matrix is diagonal with diagonal elements equal to reciprocals of the original matrix.

(f) $\mathrm{tr}[\mathbf{AB}] = \sum_i\sum_j \mathbf{A}_{ij}\mathbf{B}_{ji}$ for any two matrices $\mathbf{A}$ and $\mathbf{B}$. Substituting $\mathbf{B} = \mathbf{A}^T$ gives $\mathrm{tr}[\mathbf{AA}^T] = \sum_i\sum_j \mathbf{A}_{ij}\mathbf{A}_{ij} = \sum_i\sum_j \mathbf{A}_{ij}^2$

(g) $|\mathbf{A}| = |\mathbf{S\Lambda S}^{-1}| = |\mathbf{S}||\mathbf{\Lambda}||\mathbf{S}^{-1}| = |\mathbf{S}||\mathbf{S}^{-1}||\mathbf{\Lambda}| = |\mathbf{SS}^{-1}||\mathbf{\Lambda}| = |\mathbf{I}||\mathbf{\Lambda}| = \lambda_1\lambda_2\ldots\lambda_P$. We use the fact that the determinant is a scalar so the order does not matter, and the determinant of a diagonal matrix equals the product of diagonals.

(h) By part (c), the trace equals the sum of eigenvalues values. Since the eigenvalues of a symmetric, positive definite matrix are positive, it follows the trace is positive. Similarly, by part (f), the determinant equals the product of eigenvalues. Since the eigenvalues are positive, the product is positive.

(i) A positive semi-definite matrix has only non-negative eigenvalues. Since the trace of a matrix equals the sum of eigenvalues, the sum of eigenvalues of a positive semi-definite matrix must also be non-negative, and hence the trace is non-negative. Since the determinant of a matrix equals the product of eigenvalues, the product of eigenvalues of a positive semi-definite matrix must be non-negative, and hence the determinant is non-negative.

Exercise 7.4, page 26

Consider the eigenvalue problem

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}.$$

Multiply both sides of this equation by $\mathbf{A}$, and invoke the fact that $\mathbf{A}$ is idempotent:

$$\mathbf{A}^2\mathbf{x} = \lambda\mathbf{A}\mathbf{x} \quad \Rightarrow \quad \mathbf{A}\mathbf{x} = \lambda^2\mathbf{x}.$$

Eliminating $\mathbf{A}\mathbf{x}$ from the above equations gives

$$\lambda\mathbf{x} = \lambda^2\mathbf{x}.$$

Since the eigenvector is non-zero (i.e., $\mathbf{x} \neq \mathbf{0}$), this equation can be satisfied in general only if $\lambda = \lambda^2$, which has the two solutions $\lambda = 0$ and $\lambda = 1$.

The trace of any matrix equals the sum of eigenvalues. Since the eigenvalues of $\mathbf{A}$ are either 0 or 1, the sum of eigenvalues equals the number that are equal to one.

Exercise 7.5, page 26

(a) $\boldsymbol{\xi}^T\boldsymbol{\xi} = (\mathbf{U}\mathbf{x})^T(\mathbf{U}\mathbf{x}) = \mathbf{x}^T\mathbf{U}^T\mathbf{U}\mathbf{x} = \mathbf{x}^T\mathbf{x}$ since $\mathbf{U}$ is orthogonal.

(b) $\boldsymbol{\xi}^T\boldsymbol{\eta} = (\mathbf{U}\mathbf{x})^T(\mathbf{U}\mathbf{y}) = \mathbf{x}^T\mathbf{U}^T\mathbf{U}\mathbf{y} = \mathbf{x}^T\mathbf{y}$. Thus, we have the identities $\boldsymbol{\xi}^T\boldsymbol{\xi} = \mathbf{x}^T\mathbf{x}$, $\boldsymbol{\eta}^T\boldsymbol{\eta} = \mathbf{y}^T\mathbf{y}$, and $\boldsymbol{\xi}^T\boldsymbol{\eta} = \mathbf{x}^T\mathbf{y}$. Since the angle depends only on these three quantities, it is preserved by the transformation.

(c) Since $\mathbf{U}$ is orthogonal, $\mathbf{U}\mathbf{U}^T = \mathbf{I}$. Taking the determinant of both sides gives $|\mathbf{I}| = 1 = |\mathbf{U}\mathbf{U}^T| = |\mathbf{U}||\mathbf{U}^T| = |\mathbf{U}|^2$. Thus, $|\mathbf{U}| = \pm 1$.

(d) $|\mathbf{U}\mathbf{A}\mathbf{U}^T| = |\mathbf{U}||\mathbf{A}^T||\mathbf{U}^T| = |\mathbf{U}||\mathbf{U}^T||\mathbf{A}| = |\mathbf{U}\mathbf{U}^T||\mathbf{A}| = 1|\mathbf{A}|$.

(e) $\mathrm{tr}[\mathbf{U}\mathbf{A}\mathbf{U}^T] = \mathrm{tr}[\mathbf{U}\mathbf{U}^T\mathbf{A}| = \mathrm{tr}[\mathbf{A}] = \mathrm{tr}[\mathbf{A}]$.

(f) Show that $(\mathbf{U}_1\mathbf{U}_2)(\mathbf{U}_1\mathbf{U}_2)^T = \mathbf{I}$ and $(\mathbf{U}_1\mathbf{U}_2)^T(\mathbf{U}_1\mathbf{U}_2) = \mathbf{I}$ (need to show both).

Exercise 7.6, page 26

By definition, $\boldsymbol{\Sigma}_Y = \mathbb{E}[(\mathbf{y} - \boldsymbol{\mu}_Y)(\mathbf{y} - \boldsymbol{\mu}_Y)^T] = \mathbb{E}[\mathbf{y}\mathbf{y}^T] - \boldsymbol{\mu}_Y\boldsymbol{\mu}_Y^T$. Therefore

$$\begin{aligned}\mathbb{E}[\mathbf{y}^T\mathbf{A}\mathbf{y}] &= \mathbb{E}[\sum_i \sum_j y_i y_j A_{ij}] = \sum_i \sum_j \mathbb{E}[y_i y_j] A_{ij} = \sum_i \sum_j (\boldsymbol{\Sigma}_{ij} + \mu_i\mu_j) A_{ij} \\ &= \mathrm{tr}[\mathbf{A}\boldsymbol{\Sigma}_Y] + \boldsymbol{\mu}^T\mathbf{A}\boldsymbol{\mu}.\end{aligned}$$

Exercise 7.7, page 27

Take the transpose of the hint:

$$\left(\mathbf{A}^{-1}\right)^T\mathbf{A}^T = \mathbf{I} \qquad \text{transpose of the hint} \qquad (22.39)$$

$$\left(\mathbf{A}^{-1}\right)^T = \left(\mathbf{A}^T\right)^{-1} \qquad \text{multiply both sides by the inverse of transpose.} \qquad (22.40)$$

Exercise 7.8, page 27

(a)

$$\|\mathbf{X}\|_F^2 = \mathrm{tr}\left[\mathbf{X}\mathbf{X}^T\right] = \sum_i \sum_j (\mathbf{X})_{ij} \left(\mathbf{X}^T\right)_{ji} = \sum_i \sum_j (\mathbf{X})_{ij} (\mathbf{X})_{ij} = \sum_i \sum_j (\mathbf{X})_{ij}^2 .$$

Taking the square root of both sides gives (7.6).

(b)

$$\begin{aligned} \|\mathbf{UAV}\|_F^2 &= \mathrm{tr}[(\mathbf{UAV})(\mathbf{UAV})^T] = \mathrm{tr}[\mathbf{UAVV}^T\mathbf{A}^T\mathbf{U}^T] \\ &= \mathrm{tr}[\mathbf{AA}^T\mathbf{U}^T\mathbf{U}] = \mathrm{tr}[\mathbf{AA}^T] = \|\mathbf{A}\|_F^2 . \end{aligned}$$

Exercise 7.9, page 27

Consider the eigenvalue equation

$$\mathbf{Ax} = \lambda\mathbf{x}.$$

Multiplying this equation by b gives

$$b\mathbf{Ax} = b\lambda\mathbf{x}.$$

Of course,

$$a\mathbf{Ix} = a\mathbf{x}.$$

Adding these two equations together gives:

$$\begin{aligned} a\mathbf{x} + b\mathbf{Ax} &= a\mathbf{x} + b\lambda\mathbf{x} \\ (a\mathbf{I} + \mathbf{A})\,\mathbf{x} &= (a + b\lambda)\,\mathbf{x}. \end{aligned}$$

The last equation shows that the eigenvalue of $a\mathbf{I} + \mathbf{A}$ is $a + b\lambda$.

Exercise 7.10, page 27

Positive definite means $\mathbf{x}^T\mathbf{Ax} > 0$ for all $\mathbf{x}$. Choose

$$\mathbf{x} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{22.41}$$

For this choice, $\mathbf{x}^T\mathbf{Ax} = A_{11}$. Hence, the first diagonal element must be positive. Next, let $\mathbf{x}$ be a vector of all zeros except for a one in the second row. Then in this case $\mathbf{x}^T\mathbf{Ax} = A_{22}$, which must be positive. Repeating this procedure for each row of $\mathbf{x}$ reveals that each and every diagonal element must be positive.

Exercise 7.11, page 28

(a) $\mathbf{\Sigma}^{-1} = \left(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\right)^{-1} = (\mathbf{U}^T)^{-1}\mathbf{\Lambda}^{-1}\mathbf{U}^{-1} = \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^T$, since $\mathbf{U}^T = \mathbf{U}^{-1}$.

(b)

$$\begin{aligned} Q &= (\mathbf{x}-\boldsymbol{\mu})^T \mathbf{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu}) = (\mathbf{x}-\boldsymbol{\mu})^T \mathbf{U}\mathbf{\Lambda}^{-1}\mathbf{U}^T (\mathbf{x}-\boldsymbol{\mu}) \\ &= \left((\mathbf{x}-\boldsymbol{\mu})^T \mathbf{U}\mathbf{\Lambda}^{-1/2}\right)\left(\mathbf{\Lambda}^{-1/2}\mathbf{U}^T(\mathbf{x}-\boldsymbol{\mu})\right) = \boldsymbol{\xi}^T\boldsymbol{\xi} \end{aligned}$$

where $\boldsymbol{\xi} = \mathbf{A}(\mathbf{x}-\boldsymbol{\mu})$ and $\mathbf{A} = \mathbf{\Lambda}^{-1/2}\mathbf{U}^T$. Since the covariance matrix is positive definite, all diagonal elements of $\mathbf{\Lambda}$ are positive and hence have an inverse and a positive square root.

(c) $\boldsymbol{\xi}$ is a linear combination of constants and Gaussians, so it is itself Gaussian. The mean is

$$\mathbb{E}[\boldsymbol{\xi}] = \mathbb{E}[\mathbf{A}(\mathbf{x}-\boldsymbol{\mu})] = \mathbf{A}(\mathbb{E}[\mathbf{x}]-\boldsymbol{\mu}) = 0$$

and the covariance is

$$\begin{aligned} \mathrm{cov}[\boldsymbol{\xi}] =& \mathbb{E}[(\boldsymbol{\xi})(\boldsymbol{\xi})^T] \\ =& \mathbb{E}[\mathbf{A}(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{A}^T] \\ =& \mathbf{A}\mathbb{E}[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T]\mathbf{A}^T \\ =& \mathbf{A}\mathbf{\Sigma}\mathbf{A}^T = \mathbf{\Lambda}^{-1}\mathbf{U}^T\left(\mathbf{U}\mathbf{\Lambda}\mathbf{U}^T\right)\mathbf{U}\mathbf{\Lambda}^{-1} \\ =& \mathbf{I} \end{aligned}$$

Thus, $\boldsymbol{\xi} \sim \mathrm{N}(0, \mathbf{I})$.

(d) The covariance matrix of $\boldsymbol{\xi}$ is diagonal, so $\mathrm{cov}[\boldsymbol{\xi}_i, \boldsymbol{\xi}_j] = 0$ if $i \neq j$. That is, each element of $\boldsymbol{\xi}$ is uncorrelated with every other element of $\boldsymbol{\xi}$. Since $\boldsymbol{\xi}$ also is Gaussian, uncorrelatedness implies independence.

(e) Since $\mathbf{Q} = \xi_1^2 + \xi_2^2 + \ldots \xi_P^2$ is a sum of squared independent Gaussians, each with zero mean and unit variance, it must have the chi-squared distribution with P degrees of freedom.

Exercise 8.1, page 30

An R function that solves the least squares problem using the normal equations is the following:

```
regress.normal = function(y,x,include.intercept=TRUE,alpha=0.05) {
#########################################################################
## DETERMINES THE LEAST SQUARES ESTIMATE OF B IN THE EQUATION Y = XB + E
## BASED ON THE NORMAL EQUATIONS
## INPUT:
##   Y[NTOT]:      N-DIMENSIONAL VECTOR OF PREDICTANDS
##   X[NTOT,MTOT]: N X M DIMENSIONAL MATRIX OF PREDICTORS
##   INCLUDE.INTERCEPT: INCLUDE THE INTERCEPT? (DEFAULT=TRUE)
## OUTPUT:
##   BHAT: M-DIMENSIONAL VECTOR OF ESTIMATES OF B
##   R2:  R-SQUARED
##   SSE: SUM SQUARE ERROR OF THE LEAST SQUARES PREDICTION
##   DOF: DEGREES OF FREEDOM OF THE SSE.
##   RES.SE: STANDARD ERROR OF THE RESIDUALS
#########################################################################

ntot = length(y)
if ( length(x) %% ntot != 0) stop('x not dimensioned correctly')
mtot = length(x)/ntot
if ( ntot <= mtot ) stop('regression problem is not over-determined')

### STRIP MISSING DATA
dim(x)     = c(ntot,mtot)
is.missing = is.na(y) | is.na(rowSums(x))
x.good     = x[!is.missing,]
y.good     = y[!is.missing ]
nsamp      = sum(!is.missing)

if (include.intercept) x.good = cbind(rep(1,nsamp),x.good)
mtot = dim(x.good)[2]

xtx       = t(x.good) %*% x.good
xty       = t(x.good) %*% y.good
xtx.inv   = chol2inv(chol(xtx))
bhat      = as.numeric(xtx.inv %*% xty)

sse       = sum((y.good - x.good %*% bhat)^2)
r2        = 1 - sse/sum((y.good-mean(y.good))^2)
dof       = nsamp - mtot
res.se    = sqrt(sse/dof)

list(bhat=bhat,r2=r2,sse=sse,dof=dof,res.se=res.se)
}
```

Exercise 8.2, page 31

Using this 'data,' we obtain least squares estimates of the model (8.1) as follows:

```
> set.seed(1)
> ntot  = 20
> y     = rnorm(ntot); pred1 = rnorm(ntot); pred2 = rnorm(ntot)
> x     = cbind(pred1,pred2)
> xy.normal = regress.normal(y,x)
> print(xy.normal)
$bhat
           [,1]
[1,]  0.1493806
[2,] -0.1516176
[3,]  0.2893589

$r2
[1] 0.1078292

$sse
[1] 14.13789

$dof
[1] 17

$res.se
[1] 0.9119432
```

Exercise 8.3, page 32

The result of fitting a trend line to CO_2 concentration is:

```
> ntot = length(co2)
> year = iyst + (1:ntot - 0.5)/12
> year.shift = year - 1960
> year.sqr = year.shift^2
> co2.trend.normal = regress.normal(co2,year.shift)
> print(co2.trend.normal)
$bhat
[1] 308.698631   1.565872

$r2
[1] 0.9792493

$sse
[1] 10031.19

$dof
[1] 689

$res.se
[1] 3.815633
```

Thus, the growth rate is 1.6 ppm/year.

Using the `lm` function gives:

```
> co2.year.lm = lm(co2~year.shift,na.action=na.omit)
> print(summary(co2.year.lm))

Call:
lm(formula = co2 ~ year.shift, na.action = na.omit)

Residuals:
    Min      1Q  Median      3Q     Max
-7.4921 -2.6494 -0.3407  2.3416 11.1495

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.087e+02  2.917e-01  1058.4   <2e-16 ***
year.shift  1.566e+00  8.684e-03   180.3   <2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 3.816 on 689 degrees of freedom
  (5 observations deleted due to missingness)
Multiple R-squared:  0.9792, Adjusted R-squared:  0.9792
F-statistic: 3.251e+04 on 1 and 689 DF,  p-value: < 2.2e-16
```

The coefficients, R^2, and dof are consistent.

Exercise 8.4, page 32

The residuals are:

```
> options(width=70)
> x = cbind(rep(1,ntot),year.shift)
> y.pred = x %*% co2.trend.normal$bhat
> resid  = co2 - y.pred
> print(as.numeric(resid[1:50]))
 [1]  7.666124  8.075635  8.555146  9.864656 10.744167 10.173678
 [7]  8.633188  6.232699  4.352210  3.891720  4.931231  5.990742
[13]  6.600252  7.239763  7.949274  8.758784  9.728295  8.787806
[19]  7.457316  5.546827  3.426338  3.875848  4.465359  5.244870
[25]  6.044380  6.533891  7.523402  8.342912  8.592423  8.001933
[31]  6.901444  4.590955  3.320465  2.349976  3.489487  4.358997
[37]  5.278508  5.488019  6.137529  7.537040  8.266551  7.356061
[43]  5.495572  3.395083  1.704593  1.354104  2.353615  3.413125
[49]  4.542636        NA
```

A plot of the residuals is shown in fig. 22.8.

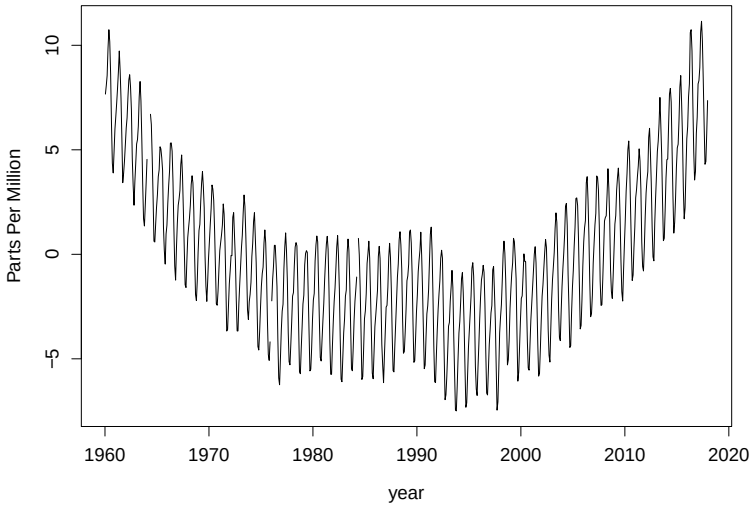


Figure 22.8 Residuals of CO_2 data after linear trend has been removed.

Exercise 8.5, page 33

The annual cycle can be estimated as follows:

```
> nharm = 2
>
> t = seq(year.shift)/12
> x = NULL
> for ( n in 1:nharm) x = cbind(x,cos(2*pi*t*n),sin(2*pi*t*n))
> colnames(x)  = c(paste(rep(c('cos','sin'),nharm),rep(1:nharm,each=2),sep=""))
> co2.cycle.normal = regress.normal(co2,x)
> print(co2.cycle.normal)
$bhat
[1] 354.3263100  -1.4843479   2.1128289   0.7733454  -0.3005125

$r2
[1] 0.005265222

$sse
[1] 480868.2

$dof
[1] 686

$res.se
[1] 26.47591
```

Exercise 8.6, page 33

The residuals after the annual cycle has been removed can be computed as

```
x = cbind(rep(1,ntot),x)
y.pred = x %*% co2.cycle.normal$bhat
resid  = co2 - y.pred
```

A plot of the residuals and corresponding raw data is produced by

```
resid  = co2 - y.pred + co2.cycle.normal$bhat[1]

fout = 'co2.cycle.residuals.eps'
if ( lplotfile) postscript(fout,horizontal=FALSE,
   onefile=FALSE,height=6,width=8,pointsize=12)
par(mfcol=c(1,1),mar=c(5,5,3,1))
par(cex.lab=1.2,cex.axis=1.2,cex.main=1.2)
yrange = range(co2,resid,na.rm=TRUE)
plot(year.say,resid,type='l',xlab='year',
   ylab='Parts Per Million',ylim=yrange,col='red',lwd=2)
par(new=TRUE)
plot(year.say,co2,type='l',xlab='',ylab='',ylim=yrange,axes=FALSE)
if ( lplotfile) dev.off()
```

The resulting plot is shown in fig. 22.9.

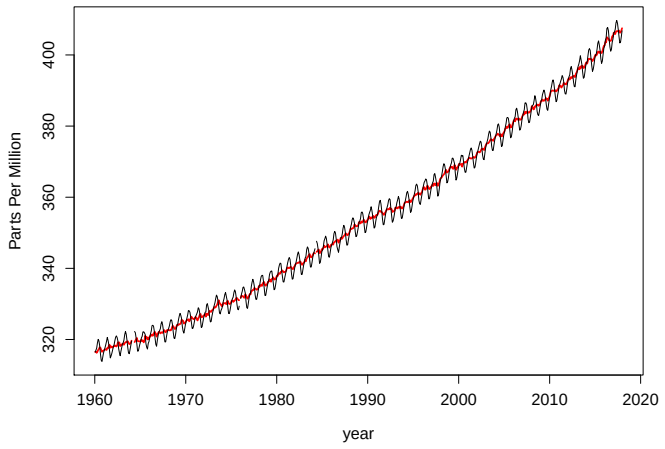


Figure 22.9 CO_2 (black curve) and CO_2 after annual cycle has been removed (red).

Exercise 9.1, page 36

A code for generating a plot is the following. The resulting figure is shown in fig. 22.10.

```
plot(x.year,y,type='b',pch=19,lwd=2,xlab='year',ylab='sea level (mm)')
ftitle = paste(station,'Sea Level During',season)
title(main=ftitle,line=0.5)
abline(lm(y~x.year),lty='dashed',lwd=3,col='grey50')
```

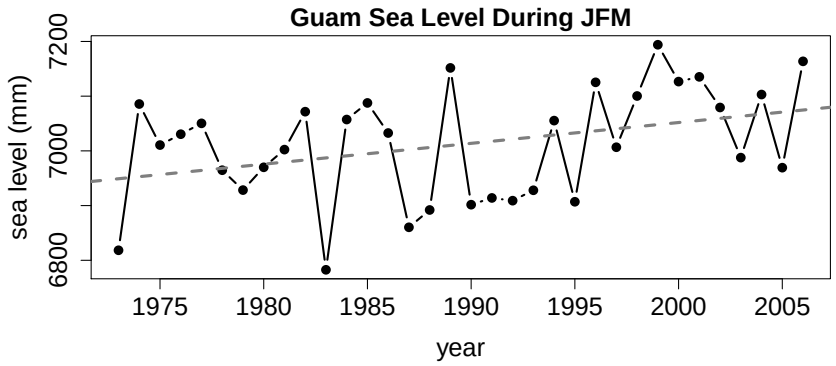


Figure 22.10 Sea level at Guam during JFM.

The figure suggests a slight increase in sea level over the past few decades.

Exercise 9.2, page 36

The result of fitting JFM Guam sea level to year is the following:

```
> print(regress.normal(y,x.year))
$bhat
[1] -560.046422    3.805933

$r2
[1] 0.135494

$sse
[1] 302447.4

$dof
[1] 32

$res.se
[1] 97.21873
```

Exercise 9.3, page 36

An R function that performs regression and computes the t-value, p-value, and confidence interval is the following:

```
regress.normal = function(y,x,include.intercept=TRUE,alpha=0.05) {
#########################################################################
## DETERMINES THE LEAST SQUARES ESTIMATE OF B IN THE EQUATION Y = XB + E
## BASED ON THE NORMAL EQUATIONS
## INPUT:
##   Y[NTOT]:      N-DIMENSIONAL VECTOR OF PREDICTANDS
##   X[NTOT,MTOT]: N X M DIMENSIONAL MATRIX OF PREDICTORS
##   INCLUDE.INTERCEPT: INCLUDE THE INTERCEPT? (DEFAULT=TRUE)
## OUTPUT:
##   BHAT: M-DIMENSIONAL VECTOR OF ESTIMATES OF B
##   R2:  R-SQUARED
##   SSE: SUM SQUARE ERROR OF THE LEAST SQUARES PREDICTION
##   DOF: DEGREES OF FREEDOM OF THE SSE.
##   RES.SE: STANDARD ERROR OF THE RESIDUALS
##   TVAL[MTOT]: T-STATISTIC FOR EACH REGRESSION COEFFICIENT
##   PVAL[MTOT]: P-VALUE FOR EACH REGRESSION COEFFICIENT
##   BHAT.CI[MTOT,2]: CONFIDENCE INTERVAL FOR EACH REGRESSION COEF.
#########################################################################

ntot = length(y)
if ( length(x) %% ntot != 0) stop('x not dimensioned correctly')
mtot = length(x)/ntot
if ( ntot <= mtot ) stop('regression problem is not over-determined')

### STRIP MISSING DATA
dim(x)     = c(ntot,mtot)
is.missing = is.na(y) | is.na(rowSums(x))
x.good     = x[!is.missing,]
y.good     = y[!is.missing ]
nsamp      = sum(!is.missing)

if (include.intercept) x.good = cbind(rep(1,nsamp),x.good)
mtot = dim(x.good)[2]

xtx       = t(x.good) %*% x.good
xty       = t(x.good) %*% y.good
xtx.inv   = chol2inv(chol(xtx))
bhat      = as.numeric(xtx.inv %*% xty)

sse       = sum((y.good - x.good %*% bhat)^2)
r2        = 1 - sse/sum((y.good-mean(y.good))^2)
dof       = nsamp - mtot
res.se    = sqrt(sse/dof)

### INFERENCE
st.err = sqrt(sse/dof*diag(xtx.inv))
tval   = bhat/st.err
pval   = 2*pt(abs(tval),dof,lower.tail=FALSE)
t.crit = qt(alpha/2,dof,lower.tail=FALSE)
bhat.ci = cbind(bhat - t.crit*st.err, bhat + t.crit*st.err)

list(bhat=bhat,r2=r2,sse=sse,dof=dof,res.se=res.se,tval=tval,pval=pval,bhat.ci=bhat.ci)
}
```

Exercise 9.4, page 36

The output is identical to earlier problems, except for the following added items

```
> print(regress.normal(y,x.year))
...
$tval
[1] -0.1656398  2.2394999

$pval
[1] 0.86948250 0.03220094

$bhat.ci
               [,1]        [,2]
[1,] -7447.1429574 6327.050113
[2,]     0.3442531    7.267614
```

The trend is significant at the 5% level, but just barely. The results agree with lm:

```
> trend.lm = lm(y~x.year)
> trend.sum = summary(trend.lm)
> print(trend.sum)

Call:
lm(formula = y ~ x.year)

Residuals:
    Min      1Q  Median      3Q     Max
-204.45 -100.43   24.59   78.49  145.99

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -560.046   3381.110  -0.166   0.8695
x.year         3.806      1.699   2.239   0.0322 *
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 97.22 on 32 degrees of freedom
Multiple R-squared:  0.1355, Adjusted R-squared:  0.1085
F-statistic: 5.015 on 1 and 32 DF,  p-value: 0.0322
```

And with `confint`:

```
> print(confint(trend.lm))
                     2.5 %      97.5 %
(Intercept) -7447.1429574 6327.050113
x.year          0.3442531    7.267614
```

Exercise 9.5, page 37

The output is given above.

Exercise 9.6, page 37

The result of fitting sea level to both year and NINO3.4 is the following:

```
> print(regress.normal(y,cbind(x.year,x.sst)))
$bhat
[1] -676.326254    3.865792  -77.019967

$r2
[1] 0.6148428

$sse
[1] 134747.3

$dof
[1] 31

$res.se
[1] 65.9294

$tval
[1] -0.2949533  3.3541617 -6.2113731

$pval
[1] 7.699959e-01 2.112795e-03 6.733968e-07

$bhat.ci
              [,1]        [,2]
[1,] -5352.919193 4000.266686
[2,]     1.515179    6.216405
[3,]  -102.309586  -51.730348
```

Trend is significant: p-value is less than 5% and confidence interval does not include 0. This result agrees with lm:

```
> trend.enso.lm = lm(y~x.year+x.sst)
> trend.enso.sum = summary(trend.enso.lm)
> print(trend.enso.sum)

Call:
lm(formula = y ~ x.year + x.sst)

Residuals:
   Min     1Q Median     3Q    Max
-99.02 -49.59   9.31  28.70 201.20

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -676.326   2292.994  -0.295  0.77000
x.year         3.866      1.153   3.354  0.00211 **
x.sst        -77.020     12.400  -6.211 6.73e-07 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 65.93 on 31 degrees of freedom
Multiple R-squared:  0.6148, Adjusted R-squared:   0.59
F-statistic: 24.74 on 2 and 31 DF,  p-value: 3.779e-07
```

And with `confint`:

```
> print(confint(trend.enso.lm))
                    2.5 %      97.5 %
(Intercept) -5352.919193 4000.266686
x.year          1.515179    6.216405
x.sst        -102.309586  -51.730348
```

Exercise 10.1, page 40

$$X_{n,j} = \begin{cases} 1 & \text{if } j = 1 \\ \cos\left(2\pi n(j/2)/12\right) & \text{if j is even} \\ \sin\left(2\pi n(j-1)/2/12\right) & \text{if j is odd and } j > 1 \end{cases}.$$

and

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ c_1 \\ s_1 \\ c_2 \\ s_2 \\ \vdots \\ c_H \\ s_H \end{pmatrix} \tag{22.42}$$

Exercise 10.2, page 41

A complete code for all parts of the exercises is the following:

```
modsel.loo.simple = function(y,x,kmax=dim(x)[2]+1) {
ntot    = length(y)
x.const = cbind(rep(1,ntot),x); # ADD COLUMN OF ONES IN PREDICTOR MATRIX
if ( dim(x)[1] != ntot ) stop("inconsistent dimension of y and x in modsel.loo")
if ( kmax > dim(x.const)[2] ) stop("kmax exceeds 1 + number of predictors")
yvar     = var(y)
ypss     = sum((y-mean(y))^2)

cvmse    = numeric(kmax); cvstd    = numeric(kmax); se.sqr   = numeric(kmax)
nse.sqr  = numeric(kmax); ic.std   = numeric(kmax)
aic      = numeric(kmax); aicc     = numeric(kmax); bic      = numeric(kmax)

for ( k in 1:kmax) {
x.pred      = x.const[,1:k]
xy.lm       = lm(y~x.pred-1)
yhat        = fitted(xy.lm)
yresid      = residuals(xy.lm)
se.sqr[k]   = sum(yresid^2)/(ntot-k)
nse.sqr[k]  = se.sqr[k]/yvar

## BRUTE FORCE LOO
# yhat.loo = numeric(ntot)
# for ( n in 1:ntot) {
# y.pred  = y[-n]
# x.pred  = x.const[-n,1:k]
# yhat.loo[n] = sum(coefficients(lm(y.pred~x.pred-1)) * x.const[n,1:k])
# }
# cverr       = (y-yhat.loo)^2

## FAST LOO
dim(x.pred) = c(ntot,k)
xtx.inv     = chol2inv(chol( t(x.pred) %*% x.pred ))
alpha       = numeric(ntot)
for (n in 1:ntot) alpha[n] = as.numeric(t(x.pred[n,]) %*% xtx.inv %*% x.pred[n,])
cverr       = (y - yhat)^2/(1-alpha)^2

cvmse[k]    = mean(cverr)
cvstd[k]    = sd(cverr)/sqrt(ntot)

nparm       = k + 1
aic[k]      = ntot *log(2*pi) + ntot*log(mean(yresid^2)) + ntot + 2*nparm
aicc[k]     = ntot *log(2*pi) + ntot*log(mean(yresid^2)) + ntot +
    2*nparm*(nparm+1)/(ntot-nparm-1)
bic[k]      = ntot *log(2*pi) + ntot*log(mean(yresid^2)) + ntot + nparm*log(ntot)
ic.std[k]   = ntot * sqrt(psigamma(ntot/2,deriv=1))
}

cvmse = cvmse/yvar
cvstd = cvstd/yvar

list(cvmse=cvmse,cvstd=cvstd,k.min=k.min,k.sel=k.sel,se.sqr=se.sqr,nse.sqr=nse.sqr,
  npcs.min=npcs.min,npcs.sel=npcs.sel,aic=aic,aicc=aicc,bic=bic,ic.std=ic.std)}
```

A plot of the normalized unbiased error variance as a function of the number of predictors is shown in fig. 22.11. “0” means only the intercept term is included, and in this case the normalized error variance is 1 because the model error variance equals the total variability. The minimum error variance occurs when 4 Fourier terms are included, which corresponds to 2 harmonics (since the sine and cosine count as one harmonic). The minimum normalized error variance is about 78%. This means the annual cycle explains about 22% of the variance of the NINO3.4 index, leaving 78% unexplained by the annual cycle. The values of the normalized unbiased error variance are

```
[1] "unbiased error variance"
 [1] 1.0000000 0.9431272 0.8116455 0.7987252 0.7796379 0.7806310
 [7] 0.7815282 0.7824943 0.7832677 0.7842617 0.7852547
```

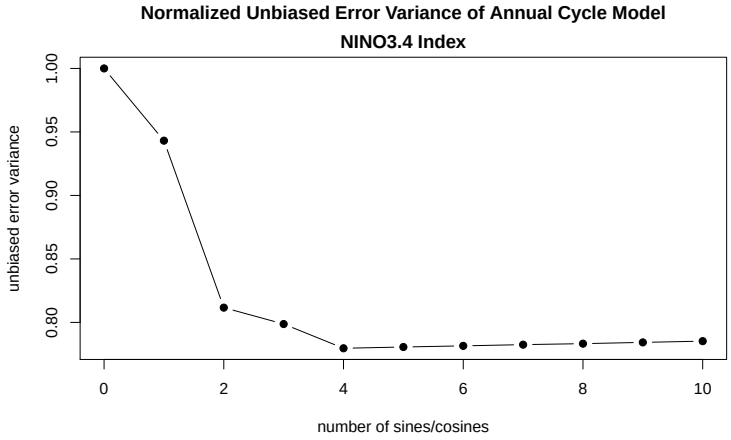


Figure 22.11 Normalized unbiased error variance of annual cycle model for NINO3.4.

Exercise 10.3, page 41

The normalized cross validated mean square error of annual cycle model for NINO3.4 is shown in fig. 22.12. The numerical values are

```
[1] "unbiased error variance"
 [1] 1.0012658 0.9454002 0.8147378 0.8027723 0.7846022 0.7865786
 [7] 0.7885135 0.7905072 0.7922921 0.7943716 0.7963414
```

Exercise 10.4, page 41

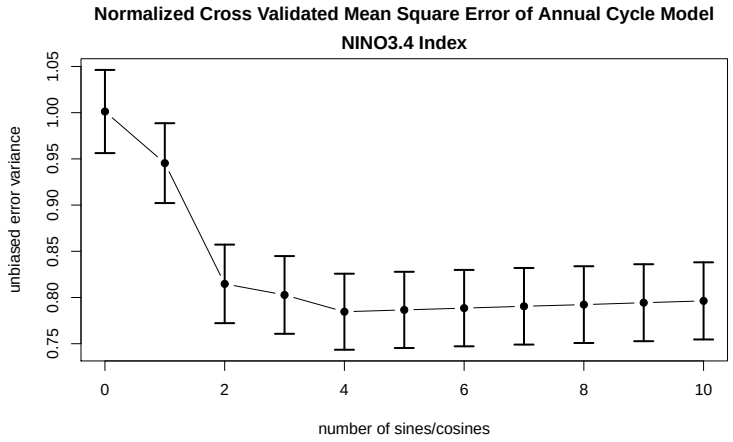


Figure 22.12 Normalized cross validated mean square error of annual cycle model for NINO3.4. Error bars show the standard error.

Exercise 10.5, page 42

See figure 22.12 for the error bars. The "one-standard-deviation rule" would select the model with 1 Fourier harmonic (but both sine and cosine).

Exercise 10.6, page 42

The numerical values of the information criteria are

```
[1] "AIC"
 [1] 2113.715 2068.396 1950.634 1938.937 1920.799 1922.799 1924.699
 [8] 1926.667 1928.438 1930.429 1932.416
[1] "AICC"
 [1] 2109.730 2062.427 1942.685 1929.013 1908.906 1908.942 1908.883
 [8] 1908.897 1908.720 1908.767 1908.817
[1] "BIC"
 [1] 2123.061 2082.416 1969.328 1962.304 1948.839 1955.512 1962.086
 [8] 1968.727 1975.171 1981.835 1988.496
```

Exercise 10.7, page 42

See figure. 22.13

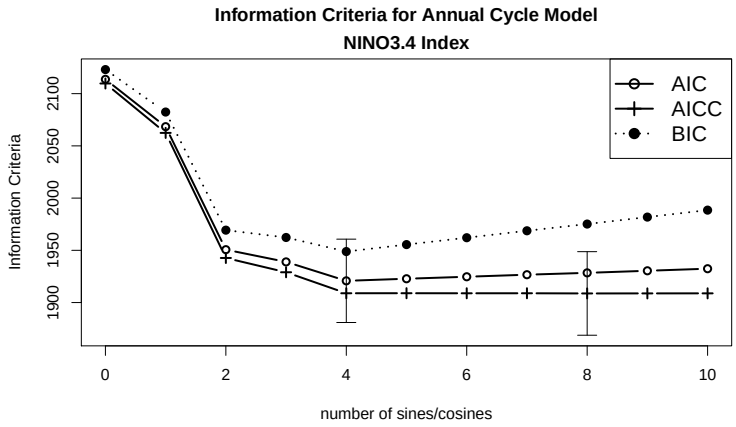


Figure 22.13 Information criteria for various annual cycle models for NINO3.4.

Exercise 10.8, page 42

The NINO3.4 anomaly can be computed and plotted as follows

```
y = nino34.ts
nh.sel = 4
x = x.pred[,1:nh.sel]
xy.lm = lm(y~x)

fout = 'nino34.anomaly.pdf'
if (lplotfile) pdf(fout,width=8,height=5)
par(mfcol=c(1,1),mar=c(5,5,5,1))
plot(nino34.time,residuals(xy.lm),type='l',col='blue',xlab='year',ylab='NINO3.4 Anomaly')
ftitle.top = paste('NINO3.4 Anomaly')
ftitle.bot = paste(nh.sel/2,'Annual Harmonics Removed')
title(main=ftitle.top,line=2.0)
title(main=ftitle.bot,line=0.5)
if (lplotfile) dev.off()
```

The resulting plot is shown in fig. 22.14

Exercise 10.9, page 42

Substituting $K = M + 1$ in (10.5) gives

$$AIC_c = N \log\left(\frac{SSE}{N}\right) + \frac{2(M+1)N}{N-M-2}. \tag{22.43}$$

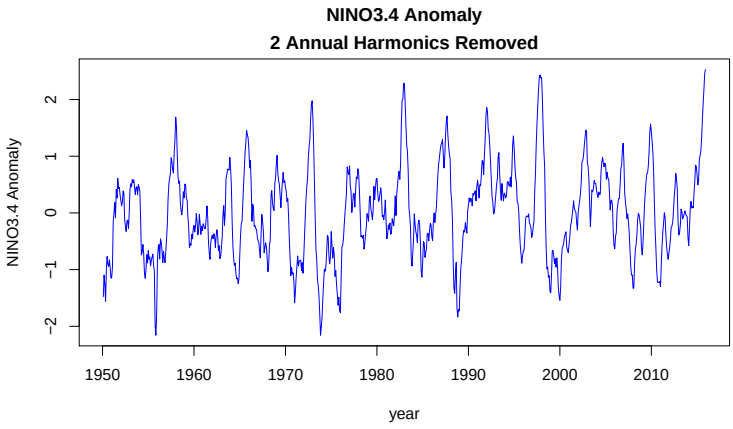


Figure 22.14 NINO3.4 anomaly after 2 annual harmonics have been removed.

Taking the difference of the two expressions gives

$$\Delta AIC_c = \frac{2(M+1)N}{N-M-2} - \left(2(M+1) + \frac{2(M+1)(M+2)}{N-M-2}\right) = 0$$

This shows that the two expressions differ by a constant. However, shifting a function upward or downward by a constant does not change the location of the minimum, so both expressions would be minimized by the same model and hence give identical model selections.

Exercise 10.10, page 43

Using the notation (**??**), it follows that

$$\mathbf{X}^T\mathbf{X} = \begin{pmatrix}\mathbf{x}_1^T & \mathbf{x}_2^T & \dots & \mathbf{x}_N^T\end{pmatrix}\begin{pmatrix}\mathbf{x}_1\\ \mathbf{x}_2\\ \vdots\\ \mathbf{x}_N\end{pmatrix} = \sum_{n=1}^{N}\mathbf{x}_n^T\mathbf{x}_n, \tag{22.44}$$

and

$$\mathbf{X}^T\mathbf{y} = \begin{pmatrix}\mathbf{x}_1^T & \mathbf{x}_2^T & \dots & \mathbf{x}_N^T\end{pmatrix}\begin{pmatrix}y_1\\ y_2\\ \vdots\\ y_N\end{pmatrix} = \sum_{n=1}^{N}\mathbf{x}_n^T y_n. \tag{22.45}$$

Let $\boldsymbol{\beta}_k$ denote the least squares estimate of $\boldsymbol{\beta}$ based on all samples except the k'th. This estimate can be expressed as

$$\boldsymbol{\beta}_k = \left(\mathbf{X}^T\mathbf{X} - \mathbf{x}_k^T\mathbf{x}_k\right)^{-1}\left(\mathbf{X}^T\mathbf{y} - \mathbf{x}_k^T y_k\right). \tag{22.46}$$

To simplify this expression, we invoke the Sherman-Morrison-Woodbury formula:

$$(\mathbf{A}-\mathbf{BDC})^{-1}=\mathbf{A}^{-1}+\mathbf{A}^{-1}\mathbf{B}\left(\mathbf{D}^{-1}-\mathbf{CA}^{-1}\mathbf{B}\right)^{-1}\mathbf{CA}^{-1}, \tag{22.47}$$

where it is assumed that the matrices are conformable and all matrix inverses are defined (i.e., $\mathbf{A}$ and $\mathbf{D}$ are not singular). This formula implies

$$\left(\mathbf{X}^T\mathbf{X}-\mathbf{x}_k^T\mathbf{x}_k\right)^{-1}=\left(\mathbf{X}^T\mathbf{X}\right)^{-1}+\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T\left(\mathbf{I}-\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T\right)^{-1}\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}. \tag{22.48}$$

Substituting this expression in (22.46), and noting that $\mathbf{I}-\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T$ is a scalar, gives

$$\begin{aligned}\boldsymbol{\beta}_k&=\left(\left(\mathbf{X}^T\mathbf{X}\right)^{-1}+\frac{\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}}{1-\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T}\right)\left(\mathbf{X}^T\mathbf{y}-\mathbf{x}_k^Ty_k\right)\\&=\hat{\boldsymbol{\beta}}+\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T\left(-\mathbf{y}_k+\frac{\mathbf{x}_k\hat{\boldsymbol{\beta}}-\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^Ty_k}{1-\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T}\right)\\&=\hat{\boldsymbol{\beta}}+\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T\left(\frac{\mathbf{x}_k\hat{\boldsymbol{\beta}}-y_k}{1-\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T}\right).\end{aligned} \tag{22.49}$$

The difference between the withheld sample y_k and its least squares prediction $\mathbf{x}_k\boldsymbol{\beta}_k$ is therefore

$$\begin{aligned}y_k-\mathbf{x}_k\boldsymbol{\beta}_k&=y_k-\mathbf{x}_k\hat{\boldsymbol{\beta}}-\frac{\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T\left(\mathbf{x}_k\hat{\boldsymbol{\beta}}-y_k\right)}{1-\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T}\\&=y_k-\mathbf{x}_k\hat{\boldsymbol{\beta}}+\frac{\alpha_k}{1-\alpha_k}\left(y_k-\mathbf{x}_k\hat{\boldsymbol{\beta}}\right)\\&=\frac{y_k-\mathbf{x}_k\hat{\boldsymbol{\beta}}}{1-\alpha_k},\end{aligned} \tag{22.50}$$

where we define the scalar

$$\alpha_k=\mathbf{x}_k\left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{x}_k^T. \tag{22.51}$$

Exercise 11.1, page 45

If the comparisons are independent, then the experimentwise error rate is related to the per-comparison significance level by

$$\alpha_{pc}=1-(1-\alpha_E)^{1/M}. \tag{22.52}$$

An experimentwise error rate of $\alpha_E=5\%$ therefore corresponds to a per-comparison significance level of 0.1%:

```
> nsamp = 30; nvarb = 50; alpha.e = 0.05; rho.obs = 0.6
> alpha.pc = 1 - (1-alpha.e)^(1/nvarb)
> alpha.pc
[1] 0.00102534
```

Under the null hypothesis of independent and normally distributed variables, the statistic

$$T = \frac{\hat{\rho}}{\sqrt{1-\hat{\rho}^2}}\sqrt{N-2}, \tag{22.53}$$

has a t distribution with $N-2$ degrees of freedom. The question is ambiguous because either the maximum value or maximum *absolute* value are appropriate, so we compute critical t-values and critical correlations for both interpretations:

```
> t.1tail = qt(alpha.pc  ,nsamp-2,lower.tail=FALSE)
> t.2tail = qt(alpha.pc/2,nsamp-2,lower.tail=FALSE)
> rho.1tail = t.1tail / sqrt(nsamp-2+t.1tail^2)
> rho.2tail = t.2tail / sqrt(nsamp-2+t.2tail^2)
> rho.1tail
[1] 0.5403947
> rho.2tail
[1] 0.5693206
```

Both critical correlations (i.e, 0.54, 0.57) are less than the observed correlation of 0.6, so we conclude that the predictor is significant at the 5% significance level, regardless of whether the forecaster is looking at the maximum, or maximum absolute, correlation.

Alternatively, we can compute the actual experimentwise error rate and compare with 5%. The observed correlation of 0.6 corresponds to a t-value of

```
> tval = rho.obs*sqrt(nsamp-2)/sqrt(1-rho.obs^2)
> tval
[1] 3.968627
```

The area to the right of this value is

```
> alpha.pc.1tail = pt(tval,nsamp-2,lower.tail=FALSE)
> alpha.pc.1tail
[1] 0.0002285276
```

Therefore, the experimentwise error rate is

```
> alpha.pc.2tail = 2*alpha.pc.1tail
> alpha.e.1tail = 1 - (1-alpha.pc.1tail)^nvarb
> alpha.e.2tail = 1 - (1-alpha.pc.2tail)^nvarb
> alpha.e.1tail
[1] 0.01136264
> alpha.e.2tail
[1] 0.02259872
```

Both of these error rates are less than 5%, so the predictor is significant at the 5% level even after taking screening into account, in agreement with the previous analysis.

If the variables are not independent, then we can use the Bonferroni bound $\alpha_E \leq M\alpha_{pc}$:

```
> alpha.e.bonferroni.1tail = nvarb * alpha.pc.1tail
> alpha.e.bonferroni.2tail = nvarb * alpha.pc.2tail
> alpha.e.bonferroni.1tail
[1] 0.01142638
> alpha.e.bonferroni.2tail
[1] 0.02285276
```

Both values of α_E are less than 5% and hence imply significance.

Exercise 11.2, page 46

The significance test is invalid because the selection of the four forecasts is based on the same data as used to test significance. The experimentwise error rate cannot be estimated from $\alpha_E = 1 - (1 - \alpha_{pc})^M$ because this equation pertains to selecting a *single maximum* and determining the probability of achieving *that value*. In contrast, we are choosing the best *four out of nine variables*, then we are applying some *other* calculation on those variables (i.e., computing their mean, and then computing the correlation using that mean).

To estimate the experimentwise error rate, we perform a numerical simulation. First, we generate random numbers to simulate 20 years of 9 forecasts, plus 20 years of observations. By construction, the random numbers are independent, so the true correlation between any pair of variables is zero. The correlation between each of the 9 "forecasts" and "observations" are computed and ranked, the four forecasts with largest correlation are averaged to construct a single forecast, and the correlation between this 4-member mean and "observations" is computed. This procedure is repeated 10,000 times to produce an empirical distribution for the correlation skill. An R code for this analysis is the following:

```
> set.seed(1)
> ntot      = 20
> ntrials   = 10000
> nmod      = 9
> nsel      = 4
> obs       = rnorm(ntot)
> cor.skill = numeric(ntrials)
> for ( nt in 1:ntrials) {
+  frcs   = array(rnorm(ntot*nmod),dim=c(ntot,nmod))
+  cor.xy = as.numeric(cor(frcs,obs))
+  npic   = order(cor.xy,decreasing=TRUE)
+  frcs.mean = rowMeans(frcs[,npic[1:nsel]])
+  cor.skill[nt] = cor(frcs.mean,obs)
+ }
> cor.95 = quantile(cor.skill,probs=0.95)
> cor.95
      95%
0.6145697
> pval.e = sum( cor.skill >= 0.55)/ntrials
> pval.e
[1] 0.1166
```

A histogram of the resulting correlations of the 4-member mean is shown in fig. 22.15. First, we see that the correlations are skewed above zero, even though none of the forecasts have skill. Second, the ninety-fifth percentile, indicated by the vertical dashed line, shows that the correlation would have to exceed 0.61 in order to reject the null hypothesis of zero correlation at the 5% level. Consistent with this, the experimentwise p-value for this procedure is 11.7% (i.e., greater than 5%). Therefore, the correlation skill of 0.55 is not significant when the selection process has been taken into account.

Exercise 11.3, page 46

The significance test is not valid because it did not account for the fact that the investigator looked at 3 correlations and chose the largest. The probability of finding *at least one* out of three independent correlations to exceed the 5% level is

$$p_e = 1 - (1 - 0.05)^3 \approx 14\%. \tag{22.54}$$

This calculation is not realistic because the correlations are not independent (e.g., the 3 correlations are computed using exactly the same observations). The Bonferroni bound, which can be applied even if the variables are dependent, implies $\alpha_E \leq M\alpha_{pc}$, or $\alpha_E \leq 15\%$. Both calculations suggest that the experimentwise error rate for this analysis could be as high as 14-15%, so there is little basis to assume the variable is significant.

To *ensure* an experimentwise error rate of 5%, the Bonferroni bound implies that the comparisonwise error rate should be

$$p_c = 0.05/3 \approx 1.7\%. \tag{22.55}$$

The critical value of the correlation for a 1.7% significance level is

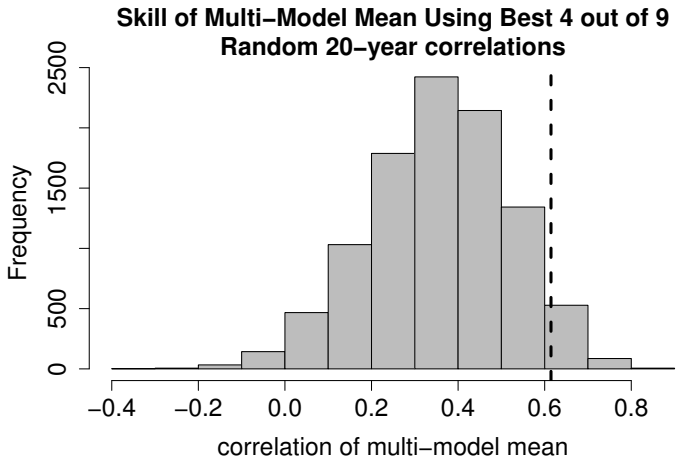


Figure 22.15 Histogram of correlation coefficients of a multi-model forecast derived from averaging the four most correlated "no-skill" forecasts out of nine no-skill forecasts. The vertical dashed line indicates the ninety-fifth percentile.

```
> alpha = 0.05
> nyrs  = 50
> nsel  = 3
>
> pe    = 1 - (1-alpha)^nsel
> pc    = 1 - (1-alpha)^(1/nsel)
> pc
[1] 0.01695243
> t.c   = qt(pc/2,nyrs-2,lower.tail=FALSE)
> rho.c   = t.c/sqrt(nyrs-2+t.c^2)
> rho.c
[1] 0.3362835
```

Therefore, the observed correlation of 0.3 is not significant, in an experimentwise sense, when screening is taken into account.

Also, one normally expects a predictor to become less useful as the lead time increases. However, in this example, the correlation is strong in October and weak in the months in closer proximity (November and December). Since this behavior contradicts expectations based on past experience and physical models, the investigator is obliged to explain how the value in October is a better predictor than the values in November or December.

Exercise 12.1, page 49

$$\begin{aligned}\mathbf{Y}' &= \mathbf{W}^{-1}\dot{\mathbf{U}}\dot{\mathbf{S}}\dot{\mathbf{V}}^T && (22.56)\\ &= \left(\frac{1}{\sqrt{N-1}}\mathbf{W}^{-1}\dot{\mathbf{U}}\dot{\mathbf{S}}\right)\left(\sqrt{N-1}\dot{\mathbf{V}}^T\right) && (22.57)\\ &= \mathbf{E}\mathbf{F}^T && (22.58)\end{aligned}$$

where

$$\begin{aligned}\mathbf{E} &= \frac{1}{\sqrt{N-1}}\mathbf{W}^{-1}\dot{\mathbf{U}}\dot{\mathbf{S}} && (22.59)\\ \mathbf{F} &= \sqrt{N-1}\dot{\mathbf{V}} && (22.60)\end{aligned}$$

The PCs satisfy

$$\frac{1}{N-1}\mathbf{F}^T\mathbf{F} = \mathbf{I}. \qquad (22.61)$$

Exercise 12.2, page 50

A function that performs PCA is the following:

```
eof.latlon.simple <- function(lon, lat, data.array, neof=30) {
# COMPUTE PRINCIPAL COMPONENTS OF A DATA ARRAY.
# INPUT:
# LON[NLON]: VECTOR SPECIFYING LONGITUDES
# LAT[NLAT]: VECTOR SPECIFYING LATITUDES
# DATA.ARRAY[NLON,NLAT,NTOT] OR DATA.ARRAY[NLON*NLAT,NTOT]:
#   THE DATA ARRAY IN [SPACE1,SPACE2,TIME] OR [SPACE,TIME] FORMAT
# NEOF = NUMBER OF SPATIAL EOFS (WITH MASK) TO BE OUTPUTTED.  DEFAULT = 30
# OUTPUT LIST:
# $EOF[NLON*NLAT,NEOF]: THE FIRST NEOF SCALED EOFS
# $PC[NTOT,NEOF]: THE PCS, NORMALIZED TO UNIT VARIANCE
# $FEXPVAR: FRACTION OF EXPLAINED VARIANCE FOR EACH EOF (ALL OF THEM).
# $FEXPVAR.CI: CONFIDENCE INTERVALS FOR FRACTION OF EXPLAINED VARIANCE.
# $EOFI[NLON*NLAT,NEOF]: PSEUDO INVERSE OF EOF (I.E.,T(EOFI) %*% EOF = I)
# $NEOF: MINIMUM OF (NEOF IN ARGUMENT LIST, RANK OF DATA.ARRAY)

# DEFINE PARAMETERS
nlon = length(lon)
nlat = length(lat)
ntot = length(data.array)/(nlon*nlat)
if ( length(data.array) %% nlon*nlat != 0 )
   stop("data.array dimension not integral multiple of nlon*nlat")

# DEFINE WEIGHTING
weight =rep(sqrt(cos(lat*pi/180)),each=nlon)

# IDENTIFY MISSING OR 0-WEIGHTED DATA
dim(data.array) = c(nlon*nlat,ntot)
lbad = is.na(rowSums(data.array)) | weight == 0

data.array = data.array - rowMeans(data.array) # REMOVE CLIMATOLOGY
data.array = data.array * weight # NORMALIZE/WEIGHT THE VARIABLES

# COMPRESS DATA BY ELIMINATING MISSING GRID POINTS
data.array = data.array[!lbad,]
ndef = sum(!lbad)

# COMPUTE SVD
mmin = min(ndef,ntot,neof)
data.svd = svd(data.array,nu=mmin,nv=mmin)

fexpvar = data.svd$d^2/sum(data.svd$d^2) # COMPUTE FEXPVAR

# COMPUTE CONFIDENCE INTERVAL OF FEXPVAR
fexpvar.ci = cbind(fexpvar * ( 1 + sqrt(2/ntot)) , fexpvar * (1 - sqrt(2/ntot)))

# COMPUTE PCS
pc = sqrt(ntot-1)*data.svd$v[,(1:mmin)]

# FILL IN MISSING POINTS IN EOF
eof  = array(NA,dim=c(nlon*nlat,mmin))
eofi = array(NA,dim=c(nlon*nlat,mmin))
for ( n in 1:mmin ) eof [!lbad,n] = data.svd$u[,n]/weight[!lbad]*data.svd$d[n]/sqrt(ntot-1)
for ( n in 1:mmin ) eofi[!lbad,n] = data.svd$u[,n]*weight[!lbad]/data.svd$d[n]*sqrt(ntot-1)

list(eof=eof,pc=pc,fexpvar=fexpvar,fexpvar.ci=fexpvar.ci,eofi=eofi,neof=mmin)
}
```

Exercise 12.3, page 50

The plots should be similar to those shown in the notes.

Exercise 12.4, page 51

(a) $\mathbf{b}^T\mathbf{b} = \mathbf{a}^T\mathbf{a}/\mathbf{a}^T\mathbf{a} = 1$.

(b)

$$\mathrm{var}[r] = \mathrm{var}\left[\frac{\mathbf{a}^T\mathbf{x}}{\sqrt{\mathbf{a}^T\mathbf{a}}}\right] = \left(\frac{1}{\sqrt{\mathbf{a}^T\mathbf{a}}}\right)^2 \mathrm{var}[\mathbf{a}^T\mathbf{x}] = \frac{\mathbf{a}^T\boldsymbol{\Sigma}\mathbf{a}}{\mathbf{a}^T\mathbf{a}}. \tag{22.62}$$

(c) Substitute $\mathbf{a} = \mathbf{U}\tilde{\mathbf{a}}$ in the above expression:

$$\mathrm{var}[r] = \frac{\mathbf{a}^T\boldsymbol{\Sigma}\mathbf{a}}{\mathbf{a}^T\mathbf{a}} = \frac{\mathbf{a}^T\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^T\mathbf{a}}{\mathbf{a}^T\mathbf{a}} = \frac{\tilde{\mathbf{a}}^T\boldsymbol{\Lambda}\tilde{\mathbf{a}}}{\tilde{\mathbf{a}}^T\mathbf{U}^T\mathbf{U}\tilde{\mathbf{a}}} = \frac{\tilde{\mathbf{a}}^T\boldsymbol{\Lambda}\tilde{\mathbf{a}}}{\tilde{\mathbf{a}}^T\tilde{\mathbf{a}}}. \tag{22.63}$$

$\boldsymbol{\Lambda}$ is diagonal, so variance of r can be written as

$$\mathrm{var}[r] = \frac{\tilde{a}_1^2\lambda_1 + \cdots + \tilde{a}_M^2\lambda_M}{\tilde{a}_1^2 + \cdots + \tilde{a}_M^2} \tag{22.64}$$

(d) Because $\lambda_1 > \lambda_2 > \cdots > \lambda_M$, we have the inequality

$$\frac{\tilde{a}_1^2\lambda_1 + \cdots + \tilde{a}_M^2\lambda_M}{\tilde{a}_1^2 + \cdots + \tilde{a}_M^2} < \frac{\tilde{a}_1^2\lambda_1 + \cdots + \tilde{a}_M^2\lambda_1}{\tilde{a}_1^2 + \cdots + \tilde{a}_M^2} = \lambda_1\left(\frac{\tilde{a}_1^2 + \cdots + \tilde{a}_M^2}{\tilde{a}_1^2 + \cdots + \tilde{a}_M^2}\right) = \lambda_1. \tag{22.65}$$

This maximum is obtained when

$$\tilde{\mathbf{a}} = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}. \tag{22.66}$$

This solution corresponds to the vector $\mathbf{a}$ equal to the leading eigenvector of $\boldsymbol{\Sigma}$.

(e) Let the SVD be $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$. Then, the sample covariance matrix becomes

$$\frac{1}{N-1}\mathbf{X}^T\mathbf{X} = \frac{1}{N-1}\mathbf{V}\mathbf{S}^T\mathbf{S}\mathbf{V}^T. \tag{22.67}$$

This implies that the orthogonal eigenvector matrix of $\boldsymbol{\Sigma}$ is $\mathbf{V}$, and the eigenvalues are related to the singular values as

$$\hat{\lambda}_i = \frac{1}{N-1}s_i^2. \tag{22.68}$$

(f) The leading principal component maximizes the variance of a standardized linear combination.

Exercise 13.1, page 55

An R code for doing the Livezey-Chen test is the following:

```
livezey.chen = function(lon,lat,xdata,ydata,ntrials=1000,alpha=0.05) {
### PERFORMS THE LIVEZEY-CHEN FIELD SIGNIFICANCE TEST
# INPUT:
#   LON[NLON]: LONGITUDE OF THE FIELD DATA
#   LAT[NLAT]: LATITUDE OF THE FIELD DATA
#   XDATA[NLON,NLAT,NTOT]: DATA ARRAY FOR THE FIELD
#   YDATA[NTOT]: REFERENCE TIME SERIES
#   NTRIALS: NUMBER OF MONTE CARLO TRIALS (DEFAULT = 10000)
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT:
#   HISTOGRAM OF AREAS WITH SIGNIFICANT CORRELATIONS
#   LIST:
#     $CRIT.NULL: SIGNIFICANCE THRESHOLD FROM LIVEZEY-CHEN TEST
#     $SIG.AREA.OBS: PERCENTAGE AREA WITH SIGNIFICANT CORRELATIONS

nlon         = length(lon)
nlat         = length(lat)
ntot         = length(ydata)
dim(xdata)   = c(nlon*nlat,ntot)

area.frac    = rep(cos(lat*pi/180),each=nlon)
lgood        = !is.na(xdata[,1])
area.frac[!lgood] = NA
area.frac    = area.frac/sum(area.frac,na.rm=TRUE)

cor.map      = as.numeric(cor(t(xdata),ydata))
sig.area.obs = sum(area.frac[abs(cor.map) > 2/sqrt(ntot)],na.rm=TRUE)*100

perc.mc = numeric(ntrials)
for ( n in 1:ntrials) {
signif.goodonly = abs(cor(t(xdata[lgood,]),rnorm(ntot))) > 2/sqrt(ntot)
perc.mc[n]    = sum(area.frac[lgood][signif.goodonly])*100
}
crit.null    = quantile(perc.mc,probs=1-alpha)
perc.hist    = hist(perc.mc,breaks=nbreaks,plot=FALSE)
xrange       = range(perc.hist$breaks,sig.area.obs)
xlab.say     = paste('Percent Area with Significant Correlations (p<',alpha,')',sep='')
hist(perc.mc,col='grey70',xlab=xlab.say,ylab='count',xlim=xrange,main="")
abline(v=crit.null,lwd=2,col='grey30',lty='dashed')
text(crit.null,mean(par()$yaxp[1:2]),'95%\npercentile',pos=2,cex=1.2)
abline(v=sig.area.obs,lwd=2)
text(sig.area.obs,mean(par()$yaxp[1:2]),paste('obs fraction= ',round(sig.area.obs),'%',sep=''),pos=4,cex=1.2)
ftitle.top = 'Percent Area with Significant Correlations'
title(ftitle.top,line=2.0)
list(crit.null=crit.null,sig.area.obs=sig.area.obs)
}
```

The figure should look like fig. 13.4 in the lecture notes. State the values of crit.null and `sig.area.obs` that you obtain.

Exercise 13.2, page 55

An R code that performs the permutation test is the following:

```
fieldsig.permutation = function(lon,lat,xdata,ydata,ntrials=1000,alpha=0.05) {
### PERFORMS PERMUTATION FIELD SIGNIFICANCE TEST
# INPUT:
#   LON[NLON]: LONGITUDE OF THE FIELD DATA
#   LAT[NLAT]: LATITUDE OF THE FIELD DATA
#   XDATA[NLON,NLAT,NTOT]: DATA ARRAY FOR THE FIELD
#   YDATA[NTOT]: REFERENCE TIME SERIES
#   NTRIALS: NUMBER OF TRIALS (DEFAULT = 10000)
#   ALPHA: SIGNIFICANCE LEVEL OF THE TEST (DEFAULT = 5%)
# OUTPUT:
#   HISTOGRAM OF AREAS WITH SIGNIFICANT CORRELATIONS
#   LIST:
#     $CRIT.NULL: SIGNIFICANCE THRESHOLD FROM LIVEZEY-CHEN TEST
#     $SIG.AREA.OBS: PERCENTAGE AREA WITH SIGNIFICANT CORRELATIONS

nlon         = length(lon)
nlat         = length(lat)
ntot         = length(ydata)
dim(xdata)   = c(nlon*nlat,ntot)

area.frac    = rep(cos(lat*pi/180),each=nlon)
lgood        = !is.na(xdata[,1])
area.frac[!lgood] = NA
area.frac    = area.frac/sum(area.frac,na.rm=TRUE)

cor.map      = as.numeric(cor(t(xdata),ydata))
sig.area.obs = sum(area.frac[abs(cor.map) > 2/sqrt(ntot)],na.rm=TRUE)*100

perc.mc = numeric(ntrials)
for ( n in 1:ntrials) {
temp  = TRUE
while(temp) {
nperm = sample(nyrs)
temp  = any(nperm == 1:nyrs)
}
signif.goodonly = abs(cor(t(xdata[lgood,]),ydata[nperm])) > 2/sqrt(ntot)
perc.mc[n]    = sum(area.frac[lgood][signif.goodonly])*100
}
crit.null    = quantile(perc.mc,probs=1-alpha)
perc.hist    = hist(perc.mc,plot=FALSE)
xrange       = range(perc.hist$breaks,sig.area.obs)
xlab.say     = paste('Percent Area with Significant Correlations (p<',alpha,')',sep='')
hist(perc.mc,col='grey70',xlab=xlab.say,ylab='count',xlim=xrange,main="")
abline(v=crit.null,lwd=2,col='grey30',lty='dashed')
text(crit.null,mean(par()$yaxp[1:2]),'95%\npercentile\nUnder H0',pos=2,cex=1.2)
abline(v=sig.area.obs,lwd=2)
text(sig.area.obs,mean(par()$yaxp[1:2]),paste('obs fraction= ',round(sig.area.obs),'%',sep=''),pos=4,cex=1.2)
ftitle.top = 'Percent Area with Significant Correlations Under Null Hypothesis'
title(ftitle.top,line=2.0)
list(crit.null=crit.null,sig.area.obs=sig.area.obs,ntrials=ntrials)
}
```

The figure should look like fig. 13.4 in the lecture notes.

Exercise 13.3, page 56

MIC can be computed as follows:

```
r.sqr = as.numeric(rep(NA,tem.eof$neof))
for ( n in 1:tem.eof$neof) if (nyrs -n - 1 - 1 > 0) r.sqr[n] =
  summary(lm(nino34~tem.eof$pc[,1:n]))$r.square

n = 1:tem.eof$neof
mic.rsqr = log(1-r.sqr)+
         (nyrs+1)*(n+1)/(nyrs-(n+1)-2) -
         (nyrs+1)/(nyrs-1-2) -
         (nyrs+1)*(n)/(nyrs-n-2)
```

Exercise 13.4, page 56

Exercise 13.5, page 56

An R code for calculating FDR is the following:

```
fdr.fieldsig = function(y,x,fdr=0.1) {
## APPLIES FALSE DISCOVERY RATE TO A CORRELATION MAP
## INPUT:
#     Y[NTOT]: REFERENCE TIME SERIES
#     X[NTOT,MTOT]: FIELD TIME SERIES
#     FDR: FALSE DISCOVERY RATE
## OUTPUT:
#     LREJECT[MTOT]: LOGICAL VECTOR INDICATING REJECTIONS

## DEFINE SHAPE OF ARRAYS
ntot    = length(y)
if ( length(x) %% ntot != 0 ) stop('x not dimensioned correctly')
mtot    = length(x)/ntot
dim(x)  = c(ntot,mtot)

## COMPUTE CORRELATION MAP AND ASSOCIATED P-VALUES
dof     = ntot - 2
cor.map = as.numeric(cor(x,y))
tval    = abs(cor.map) * sqrt( dof / (1-cor.map^2) )
pval    = 2 * pt(tval,dof,lower.tail=FALSE)

## SCREEN OUT MISSING DATA
lgood   = !is.na(x[1,])
ngood   = sum(lgood)

n.order = order(pval)

p.upper = (1:ngood)/ngood*alpha

lreject          = rep(NA,mtot)
lreject[n.order[1:ngood]] = pval[n.order[1:ngood]] <= p.upper

lreject

}
```

The plot should be similar to fig. 13.8 of the notes.

Exercise 14.1, page 59

To minimize SSE_W, differentiate (14.2) with respect to $\mathbf{B}$ and set the result to zero. The derivative is

$$\begin{aligned}\frac{\partial SSE_W}{\partial B_{ab}} &= (-2)\sum_{s'=1}^{S}\sum_{n=1}^{N} W_{bs'}\left(X_{na}\right)\left(Y_{ns'} - \sum_{m=1}^{M} X_{nm}B_{ms'}\right) \\ &= (-2)\mathbf{W}\left(\mathbf{Y} - \mathbf{XB}\right)^T\mathbf{X}\end{aligned}$$

Because $\mathbf{W}$ is positive definite, the derivative vanishes if and only if

$$\left(\mathbf{Y} - \mathbf{XB}\right)^T\mathbf{X} = \mathbf{0}. \tag{22.69}$$

The solution is independent of $\mathbf{W}$ and identical to the least squares estimate.

Exercise 14.2, page 61

An R function that computes MIC is the following:

```
micc.gaussian = function(x,y,equal.dim=TRUE,alpha=0.05) {
### THIS FUNCTION COMPUTES CORRECTED MUTUAL INFORMATION CRITERION (MICC)
###     FOR Y = XB + E
### INPUT:
##    X[NSAMP,XDIM]: X DATA ARRAY, OFTEN FORMATTED AS [TIME,EOF]
##    Y[NSAMP,YDIM]: Y DATA ARRAY, OFTEN FORMATTED AS [TIME,EOF]
##    EQUAL.DIM: LOGICAL INDICATING WHETHER
##      (TRUE) EQUAL NUMBER OF X'S, Y'S CHOSEN: MIC[MIN(XDIM,YDIM)];
##      (FALSE) MIC FOR ALL TRUNCATIONS ARE COMPUTED: MIC[XDIM,YDIM]
### OUTPUT: LIST (DIMENSIONS DEPEND ON EQUAL.DIM)
##    $MICC: MIC VALUES
##    $PENALTY: THE PENALTY TERM IN MIC
##    $CRIT: SIGNIFICANCE THRESHOLD OF MIC

if (is.null(dim(x))) dim(x) = c(length(x),1)
if (is.null(dim(y))) dim(y) = c(length(y),1)

nsamp   = dim(x)[1]
nx.dim  = dim(x)[2]
ny.dim  = dim(y)[2]

if ( nsamp != dim(y)[1]) stop('x and y have inconsistent leading dimensions')

cov.xy = cov(cbind(x,y))

if (equal.dim) {
max.dim      = min(nsamp-1,nx.dim,ny.dim)
micc         = as.numeric(rep(NA,max.dim))
micc.penalty = as.numeric(rep(NA,max.dim))
micc.crit    = as.numeric(rep(NA,max.dim))
for (n in 1:max.dim) if (nyrs -2*n-2>0){
nall    = c(1:n,1:n+nx.dim)
micc.penalty[n] = (nyrs+1) * (2*n/(nyrs-2*n-2)-2*n/(nyrs-n-2))
micc.crit[n]    = - qchisq(alpha,n^2,lower.tail=FALSE)/(nyrs-(2*n+3)/2) + micc.penalty[n]
micc[n] = log(det(cov.xy[nall,nall])) -
          log(det(cov.xy[1:n,1:n,drop=FALSE])) -
          log(det(cov.xy[1:n+nx.dim,1:n+nx.dim,drop=FALSE])) +
          micc.penalty[n]
}
} else {
[LEAVE BLANK]
}

list(micc=micc,penalty=micc.penalty,crit=micc.crit)

}
```

Exercise 14.3, page 61

The MIC values for prefix= "Sep2Dec" are

```
> micc
 [1]   -2.4045042   -3.0197617   -3.1685965   -3.5064433
 [5]   -3.8591420   -3.8506078   -3.6629287   -3.3292924
 [9]   -2.6104376   -2.2448622   -1.3491189   -0.4359491
[13]    1.2171204    3.1035460    5.3236336    8.2142499
[17]   11.2348781   15.7603783   20.8051143   27.8231220
[21]   36.3598677   46.9349010   59.7489209   76.9114337
[25]   98.6326493  128.3214484  169.2004940  226.5889051
[29]  310.8455055  450.1993069  707.8745492 1318.4198146
[33] 4431.5898013           NA
```

Exercise 14.4, page 61

A code for making predictions and plotting results is the following:

```
##################################
######### PLOT PREDICTED AND OBSERVED FIELDS
##################################
beta.hat     = chol2inv(chol(cov(pc[mon.init,year.keep,1:nmin]))) %*% cov(pc[mon.init,year.keep,1:nmin],pc[mon.targ,yea
ypred.tilde  = pc[mon.init,,1:nmin] %*% beta.hat

cbar.breaks = c(-3.5, -3.0, -2.5, -2.0, -1.5, -1.0, -0.5,  0.5,  1.0,  1.5,  2.0,  2.5,  3.0,  4.5)
cbar.cols   = c("white","white","white","white","white","white","white","#BEBEBE","#A7A7A7","#909090","#7A7A7A","#63636
cbar.list   = list(plot.breaks=cbar.breaks,plot.cols=cbar.cols)

if (leave.data.out) yhat.say = 'Prediction' else yhat.say = 'Regression Fit'

for ( year in c(1981,1982,1988)) {
nypic = year - iyst + 1
fout = paste('ersstv5.pred',year,prefix,sep='')
if (lplotfile) pdf.eps(fout,'pdf',height=3)
ypred = as.numeric(eof.list$eof[,1:nmin] %*% ypred.tilde[nypic,1:nmin])
if (year == 1981) cbar.list$plot.breaks = cbar.breaks/2 else cbar.list$plot.breaks = cbar.breaks
par(mfrow=c(1,2),mar=c(0.5,0.5,2,0.5))
plot_latlon_bw(lon.sst,lat.sst,sst.full[,mon.targ,nypic],lmaskzero=FALSE,shrinkdomain=TRUE,suppress.axislab=TRUE,plot.l
title(main=paste('Observed',month.abb[mon.targ],year),line=0.5)
plot_latlon_bw(lon.sst,lat.sst,ypred,lmaskzero=FALSE,shrinkdomain=TRUE,suppress.axislab=TRUE,plot.legend=FALSE,cbar.lis
title(main=paste(yhat.say,' for ',year,' (',month.abb[mon.init],' IC; ',nmin,'EOFs)',sep=''),line=0.5)
if (lplotfile) dev.off()
}
```

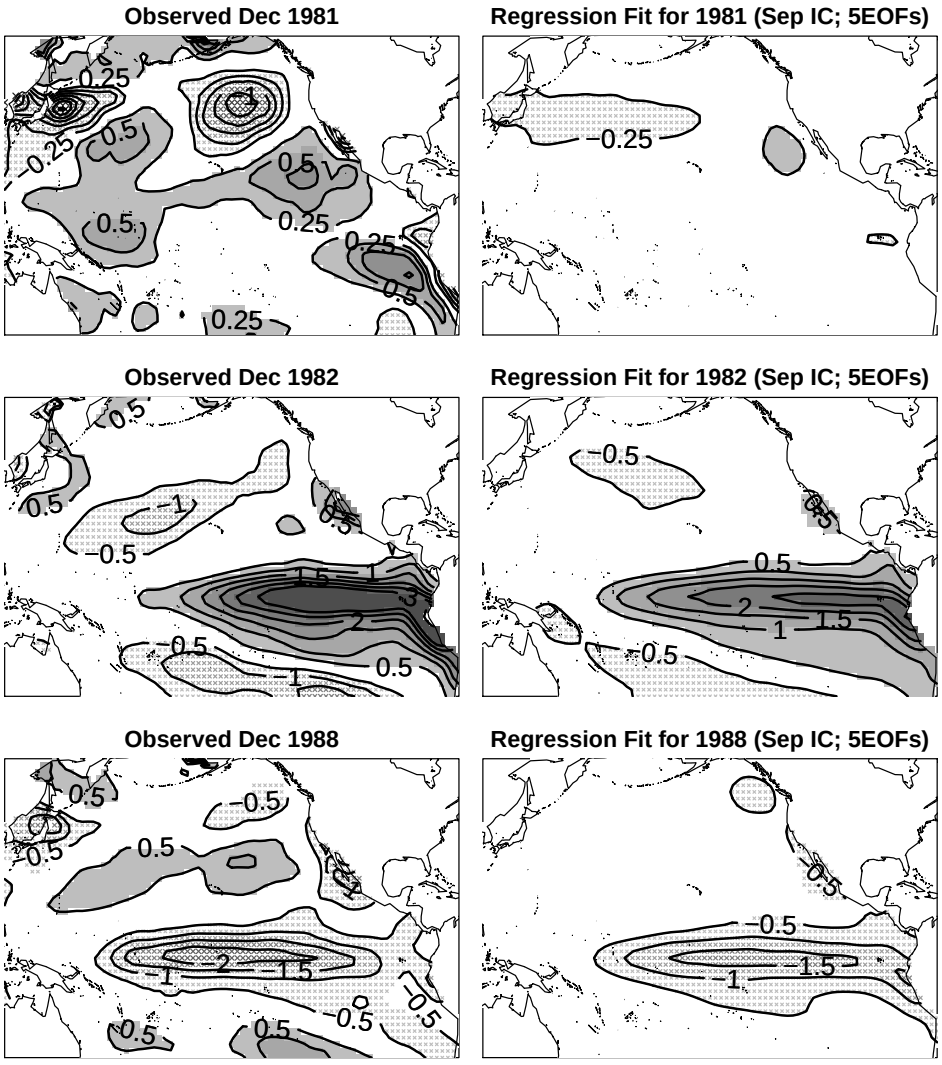


Figure 22.16 Comparison between the predicted (left) and observed (right) December anomaly SSTs for a neutral (top; 1981), warm (middle; 1982), and cold (bottom; 1988) ENSO year. The contour label is in units of degrees Celcius. The prediction is based on 22 EOFs with September initial conditions.

Exercise 15.1, page 65

The plot of MIC should be consistent with fig. 15.5 from the notes. The first 5 rows and columns are

```
> mic[1:5,1:5]
             [,1]        [,2]       [,3]       [,4]       [,5]
[1,] -0.001874504 -0.05766078 -0.2613093 -0.2377176 -0.2327597
[2,] -0.198177256 -0.42024605 -0.6405989 -0.5908572 -0.5438949
[3,] -0.185517754 -0.51092279 -0.6917943 -0.6108790 -0.6101217
[4,] -0.149274572 -0.65337786 -0.9198444 -0.8214891 -0.7746863
[5,] -0.111265453 -0.61204103 -0.8646096 -0.7910013 -0.6999580
```

The minimum occurs at $T_X = 4, T_Y = 3$ and is -0.9198444.

Exercise 15.2, page 65

An R function that performs all of CCA is given below. For $T_X = 4, T_Y = 3$, the canonical correlations are

```
> cca.list$can.cor
[1] 0.73858363 0.66234594 0.05366594
```

```
cca.pca = function(fx,fy,ex,ey,tx=NULL,ty=NULL) {
#########################################################
## PERFORMS CANONICAL CORRELATION ANALYSIS ON X AND Y.
## X AND Y ARE ASSUMED TO BE IN THE FOLLOWING FORMS:
## X = FX %*% EX^T   ### AND ### Y = FY %*% EY^T
## WHERE FX AND FY HAVE COVARIANCE MATRICES = I
## FOR EXAMPLE: FROM PRINCIPAL COMPONENT ANALYSIS
## IF TX OR TY = NULL, THEN BOTH (TX,TY) SELECTED USING MIC
## INPUT:
#    FX[NTOT,MX]: TIME SERIES FOR X-COMPONENTS: ORTHOGONAL
#    FY[NTOT,MY]: TIME SERIES FOR Y-COMPONENTS: ORTHOGONAL
#    EX[SX,MX]: SPATIAL STRUCTURES OF THE X-COMPONENTS
#    EY[SY,MY]: SPATIAL STRUCTURES OF THE Y-COMPONENTS
#    TX: TRUNCATION FOR X (TX <= MX); IF NULL, TX IS SELECTED
#    TY: TRUNCATION FOR Y (TY <= MY); IF NULL, TY IS SELECTED
## OUTPUT LIST:
#    MIC[MX,MY]: MUTUAL INFORMATION CRITERION
#    NMIN[1,2]: VALUES OF TX,TY THAT MINIMIZES MIC
#    CAN.COR[MIN(TX,TY)]: CANONICAL CORRELATIONS
#    RX[NTOT,MIN(TX,TY)]: CANONICAL VARIATES FOR X
#    RY[NTOT,MIN(TX,TY)]: CANONICAL VARIATES FOR Y
#    PX[SX  ,MIN(TX,TY)]: CANONICAL LOADING VECTORS FOR X
#    PY[SY  ,MIN(TX,TY)]: CANONICAL LOADING VECTORS FOR Y
#    QX.TILDE[TX,MIN(TX,TY)]: WEIGHTING VECTORS FOR X-FEATURES
#    QY.TILDE[TY,MIN(TX,TY)]: WEIGHTING VECTORS FOR Y-FEATURES
#    TX, TY: SELECTED VALUES OF TX AND TY
#########################################################

ntot  = dim(fx)[1]
mx    = dim(fx)[2]
my    = dim(fy)[2]
if (ntot != dim(fy)[1]) stop('fx and fy have inconsistent time dimension')

micc  = array(NA,dim=c(mx,my))
for (ty in 1:mx) for (tx in 1:my) if (ntot-tx-ty-2 > 0) {
penalty      = (ntot+1)*((tx+ty)/(ntot-tx-ty-2) - tx/(ntot-tx-2) - ty/(ntot-ty-2))
xy.svd       = svd(cov(fx[,1:tx],fy[,1:ty]))
micc [tx,ty] = sum(log(1-xy.svd$d^2)) + penalty
}

nmin    = which(micc == min(micc,na.rm=TRUE),arr.ind=TRUE)
tx      = nmin[1]
ty      = nmin[2]
cov.xy  = cov(fx[,1:tx],fy[,1:ty])
xy.svd  = svd(cov.xy)
can.cor = xy.svd$d
qx      = xy.svd$u
qy      = xy.svd$v
rx      = fx[,1:tx] %*% qx
ry      = fy[,1:ty] %*% qy
px      = ex[,1:tx] %*% qx
py      = ey[,1:ty] %*% qy

list(can.cor=can.cor,rx=rx,ry=ry,px=px,py=py,tx=tx,ty=ty,mic=micc)

}
```

Exercise 15.3, page 65

see fig. **??** of the lecture notes.

Exercise 15.4, page 65

see figs. 15.1 and 15.2 of the lecture notes.

Exercise 15.5, page 65

```
> cca.pca.list$fexpvar.x
[1] 0.1678503 0.2410055 0.1668347 0.1111007
> cca.pca.list$fexpvar.y
[1] 0.1997934 0.3069071 0.2975915
>
[1] "sum exp. var. x= 0.68679126577992 from EOFs= 0.686791265779918"
[1] "sum exp. var. y= 0.804291929099929 from EOFs= 0.804291929099928"
```

Exercise 15.6, page 65

An R code that computes the 5% significance levels of the canonical correlations based on 5000 trials of Monte Carlo experiments is the following:

```
## SIGNIFICANCE THRESHOLDS FOR CORRELATIONS
ntrials = 5000
alpha = 0.05
rho.mc  = array(NA,dim=c(min(nx.min,ny.min),ntrials))
for (nt in 1:ntrials) {
x = array(rnorm(nyrs*nx.min),dim=c(nx.min,nyrs))
y = array(rnorm(nyrs*ny.min),dim=c(ny.min,nyrs))
rho.mc[,nt] = svd( t(svd(x - rowMeans(x))$v) %*% svd(y - rowMeans(y))$v )$d
}
rho.crit = numeric(min(nx.min,ny.min))
for (n in 1:min(nx.min,ny.min)) rho.crit[n] = quantile(rho.mc[n,],probs=1-alpha)
```

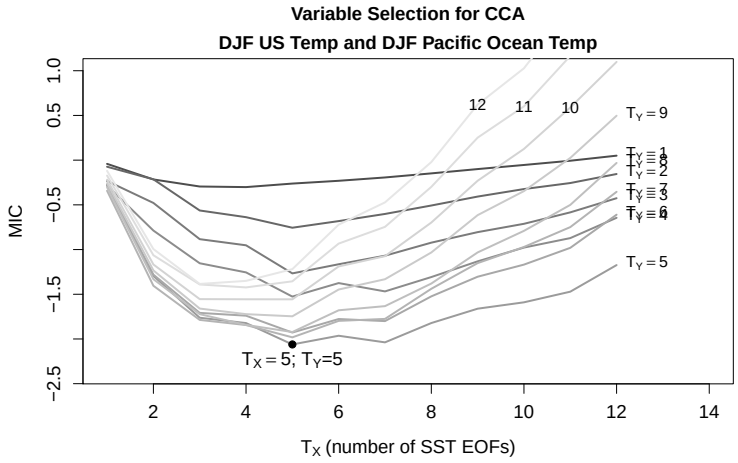
The resulting values are

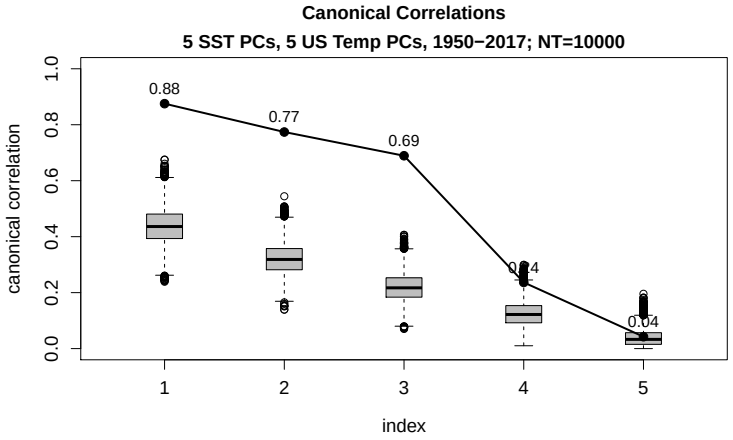
```
> rho.crit
[1] 0.4619982 0.3083790 0.1772026
```

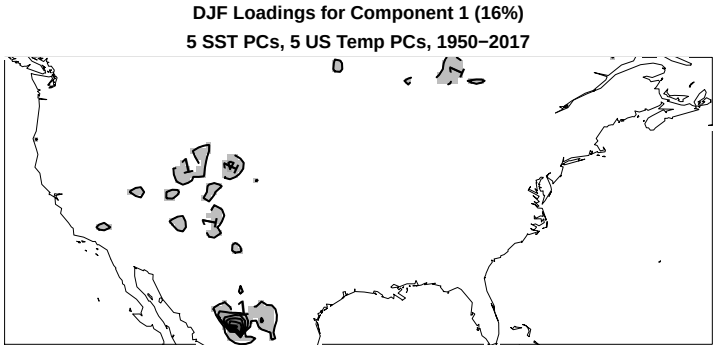
The first two canonical correlations exceed these 5% significance thresholds.

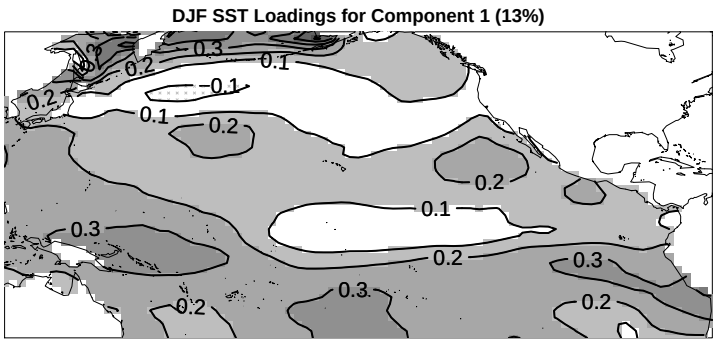
Exercise 15.7, page 65

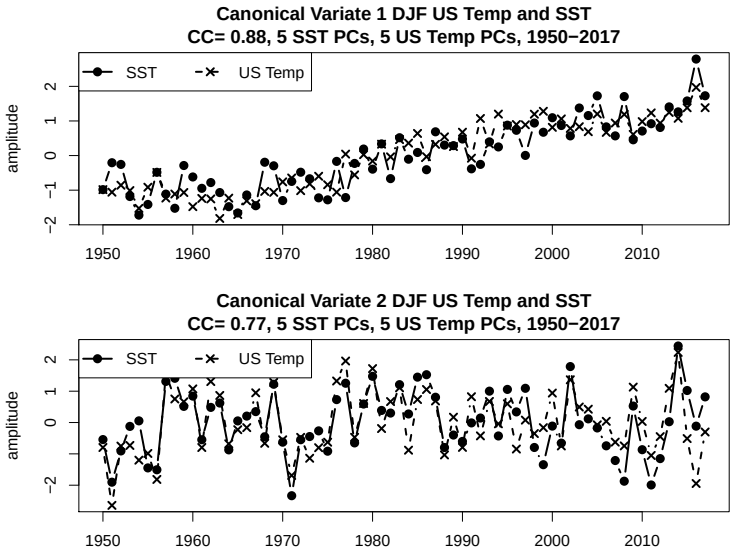
The figures for no detrending are shown below. The big change is that the leading component contains a strong trend component. This makes sense because trends are highly correlated, and CCA maximizes correlation.


Variable Selection for CCA
DJF US Temp and DJF Pacific Ocean Temp
MIC
1.0
0.5
−0.5
−1.5
−2.5
2
4
6
8
10
12
14
12
11
10
$T_Y = 9$
$T_Y = 1$
$T_Y = 2$
$T_Y = 7$
$T_Y = 3$
$T_Y = 4$
$T_Y = 5$
$T_X = 5$; T_Y=5
T_X (number of SST EOFs)


Canonical Correlations
5 SST PCs, 5 US Temp PCs, 1950−2017; NT=10000
canonical correlation
0.0
0.2
0.4
0.6
0.8
1.0
0.88
0.77
0.69
0.14
0.04
1
2
3
4
5
index


DJF Loadings for Component 1 (16%)
5 SST PCs, 5 US Temp PCs, 1950−2017


DJF SST Loadings for Component 1 (13%)
0.3
0.2
0.1
−0.1
0.2
0.1
0.2
0.2
0.1
0.3
0.2
0.3
0.3
0.2
0.2



Exercise 15.8, page 66

$$\begin{aligned}\tilde{\mathbf{\Sigma}}_{\epsilon Y} &= \left(\mathbf{R}_Y - \mathbf{R}_X\mathbf{S}_\rho\right)^T \left(\mathbf{R}_Y - \mathbf{R}_X\mathbf{S}_\rho\right)/(N-1) \\ &= \left(\mathbf{R}_Y^T\mathbf{R}_Y - \mathbf{S}_\rho\mathbf{R}_X^T\mathbf{R}_Y - \mathbf{R}_Y^T\mathbf{R}_X\mathbf{S}_\rho + \mathbf{S}_\rho\mathbf{R}_X^T\mathbf{R}_X\mathbf{S}_\rho\right)/(N-1) \\ &= \mathbf{I} - \mathbf{S}_\rho^2, \end{aligned} \tag{22.70}$$

where we have used

$$\frac{\mathbf{R}_Y^T\mathbf{R}_Y}{N-1} = \mathbf{I}, \quad \frac{\mathbf{R}_X^T\mathbf{R}_X}{N-1} = \mathbf{I}, \quad \frac{\mathbf{R}_Y^T\mathbf{R}_X}{N-1} = \mathbf{S}_\rho. \tag{22.71}$$

Exercise 16.1, page 67

$\mathbb{E}[\mathbf{C}] = \mathbf{0}$ because $\mathbb{E}[\mathbf{u}_t^* - \hat{\boldsymbol{\mu}}_Y] = \mathbf{0}$. Also

$$\mathbb{E}[\mathbf{A}] = \mathrm{cov}\left[\mathbf{u}_t^* - \hat{\boldsymbol{\mu}}_Y\right] = \mathrm{cov}\left[\mathbf{u}_t^*\right] + \mathrm{cov}\left[\hat{\boldsymbol{\mu}}_Y\right] = \mathbf{\Sigma}_U + \frac{1}{N_Y}\mathbf{\Sigma}_U = \left(1 + \frac{1}{N_Y}\right)\mathbf{\Sigma}_U.$$

Exercise 16.2, page 67

By the invariance property, X and Y may be re-scaled without affecting the measure. Accordingly, normalize each variable by the standard deviation of Y:

$$X^* = X/\sigma_Y \quad \text{and} \quad Y^* = Y/\sigma_Y. \tag{22.72}$$

Then, $\mathrm{var}[Y^*] = 1$, and $\mathrm{var}[X^*] = \sigma_X^2/\sigma_Y^2$, hence

$$d(\sigma_X^2, \sigma_Y^2) = d(\sigma_X^2/\sigma_Y^2, 1). \tag{22.73}$$

The last function clearly depends only on the variance ratio.

Exercise 16.3, page 67

First compute the economy SVD of $\mathbf{Y}$:

$$\underset{N_Y \times S}{\mathbf{Y}} = \underset{N_Y \times T}{\mathbf{U}_Y} \quad \underset{T \times T}{\dot{\mathbf{S}}_Y} \quad \underset{T \times S\text{'}}{\dot{\mathbf{V}}_Y^T} \tag{22.74}$$

where we assume $T < \min(N_Y, S)$. Now define the transformation matrix

$$\mathbf{L} = \dot{\mathbf{V}}_Y \dot{\mathbf{S}}_Y^{-1}. \tag{22.75}$$

The transformed variables become

$$\mathbf{Y}' = \mathbf{Y}\mathbf{L} = \mathbf{U}_Y \tag{22.76}$$

$$\mathbf{X}' = \mathbf{X}\mathbf{L}. \tag{22.77}$$

Thus the covariance matrices for the transformed variables become

$$\tilde{\mathbf{\Sigma}}_Y = \frac{1}{N_Y - 1} \mathbf{Y}'^T \mathbf{Y}' = \frac{1}{N_Y - 1} \mathbf{I} \quad \text{and} \quad \tilde{\mathbf{\Sigma}}_X = \frac{1}{N_X - 1} \mathbf{X}'^T \mathbf{X}'. \tag{22.78}$$

Substituting this into the generalized eigenvalue problem gives

$$(N_Y - 1)\tilde{\mathbf{\Sigma}}_X \tilde{\mathbf{q}} = \lambda \tilde{\mathbf{q}}. \tag{22.79}$$

Note, this is a standard eigenvalue problem. Thus, we compute the SVD

$$\sqrt{N_Y - 1}\mathbf{X}\mathbf{L} = \mathbf{U}_X \mathbf{S}_X \mathbf{V}_X^T. \tag{22.80}$$

It is straightforward to show that the eigenvectors $\tilde{\mathbf{q}}$ are the columns of $\mathbf{V}_X$, and the eigenvalues are the squared singular values. Thus, ignoring the normalization convention, the variates are

$$\mathbf{R}_X = \mathbf{X}\mathbf{L}\mathbf{V}_X \quad \text{and} \quad \mathbf{R}_Y = \mathbf{Y}\mathbf{L}\mathbf{V}_X. \tag{22.81}$$

The loading vectors can be found by regression.

Exercise 16.4, page 68

As usual, taking derivatives is straightforward when the quantity is written in index notation. For notational simplicity, we temporarily suppress the X and Y subscripts. Thus, the sum total variance is

$$\gamma_Y = E\left[(\mathbf{y}_s - E[\mathbf{y}_s] - \mathbf{P}_{sk}\mathbf{r}_k)\, \mathbf{W}_{ss'}\, (\mathbf{y}_{s'} - E[\mathbf{y}_{s'}] - \mathbf{P}_{s'k'}\mathbf{r}_{k'}) \right]. \tag{22.82}$$

Differentiating with respect to $\mathbf{P}$ yields

$$\frac{\partial \gamma_Y}{\partial \mathbf{P}_{ab}} = (-2)E\left[\mathbf{r}_b \mathbf{W}_{as'}\left(\mathbf{y}_{s'} - E[\mathbf{y}_{s'}] - \mathbf{P}_{s'k'}\mathbf{r}_{k'}\right)\right] \tag{22.83}$$

$$= (-2)\mathbf{W}\left(\operatorname{cov}[\mathbf{y}, \mathbf{r}] - \mathbf{P}\operatorname{cov}[\mathbf{r}, \mathbf{r}]\right). \tag{22.84}$$

Since $\mathbf{W}$ is positive definite, the derivative vanishes only if the term in parentheses vanishes. Setting this term to zero and solving for $\mathbf{P}$ yields the desired solution.

In the finite sample case, the sum total variance is

$$\dot{\gamma}_Y = \left(\dot{\mathbf{Y}}_{ns} - \mathbf{R}_{nk}\mathbf{P}_{sk}\right)\mathbf{W}_{ss'}\left(\dot{\mathbf{Y}}_{ns'} - \mathbf{R}_{nk'}\mathbf{P}_{s'k'}\right), \tag{22.85}$$

where products of repeated indices are implicitly summed. Differentiating with respect to $\mathbf{P}$ yields

$$\frac{\partial \dot{\gamma}_Y}{\partial P_{ab}} = (-2)\mathbf{R}_{nb}\mathbf{W}_{as'}\left(\dot{\mathbf{Y}}_{ns'} - \mathbf{R}_{nk'}\mathbf{P}_{s'k'}\right) \tag{22.86}$$

$$= (-2)\mathbf{W}\left(\dot{\mathbf{Y}} - \mathbf{R}\mathbf{P}^T\right)^T\mathbf{R} \tag{22.87}$$

$$= (-2)\mathbf{W}\left(\dot{\mathbf{Y}}^T\mathbf{R} - \mathbf{P}\mathbf{R}^T\mathbf{R}\right). \tag{22.88}$$

Since $\mathbf{W}$ is positive definite, the derivative vanishes only if the term in parentheses vanishes. Setting this term to zero and solving for $\mathbf{P}$ yields the desired solution.

Exercise 16.5, page 68

A useful trick is to re-write λ as

$$\lambda = \frac{\mathbf{q}^T\mathbf{\Delta}\mathbf{q}}{\mathbf{q}^T\mathbf{\Sigma}_Y\mathbf{q}} + 1 \tag{22.89}$$

where $\mathbf{\Delta} = \mathbf{\Sigma}_X - \mathbf{\Sigma}_Y$. If H_0 is true, then $\mathbf{\Delta} = \mathbf{0}$ and $\mathbf{q}^T\mathbf{\Delta}\mathbf{q}$ vanishes for all $\mathbf{q}$. On the other hand, if H_0 is false, $\mathbf{\Delta}$ must differ from zero in at least one element. Suppose this non-zero element occurs at the i'th diagonal element. Then, choosing $\mathbf{q}$ to be vector with a one at the i'th element and zero everywhere else yields $\mathbf{q}^T\mathbf{\Delta}\mathbf{q} = \Delta_{ii}$, which is non-zero by assumption. Moreover, this expression depends only on Δ_{ii}, so the value of the other elements of $\mathbf{\Delta}$ are irrelevant in this argument. Now, suppose all the diagonal elements of $\mathbf{\Delta}$ vanish. If H_0 is false, then $\mathbf{\Delta}$ must be non-zero for at least one off-diagonal element. Suppose $\Delta_{ij} \neq 0$. Then, choosing $\mathbf{q}$ to be a vector that has ones at the i'th and j'th elements and zero everywhere else yields $\mathbf{q}^T\mathbf{\Delta}\mathbf{q} = \Delta_{ij}$, which is non-zero by assumption. Moreover, this expression depends only on Δ_{ij}, so the value of the other off-diagonal elements are irrelevant. This proof exhausts all possible cases, therefore if the hypothesis H_0 is false, then λ must differ from one for at least one choice of $\mathbf{q}$.

Exercise 16.6, page 68

Substituting $\dot{\mathbf{Y}} = \mathbf{R}_Y \dot{\mathbf{P}}_Y^T$ and using standard properties of the trace gives

$$\begin{aligned} \|\dot{\mathbf{Y}}\|_W^2 &= \frac{1}{N_Y - 1} \operatorname{tr}\left[\mathbf{R}_Y \mathbf{P}^T \mathbf{W} \mathbf{P} \mathbf{R}_Y^T\right] = \operatorname{tr}\left[\mathbf{P}^T \mathbf{W} \mathbf{P} \left(\frac{\mathbf{R}_Y^T \mathbf{R}_Y}{N_Y - 1}\right)\right] \\ &= \operatorname{tr}\left[\mathbf{P}^T \mathbf{W} \mathbf{P}\right] = \sum_{k=1}^{T} \mathbf{p}_k^T \mathbf{W} \mathbf{p}_k. \end{aligned} \tag{22.90}$$

The last equation shows that the sum total variance equals the sum of inner products of the loading vectors *with no cross terms*. Thus, $\mathbf{p}_k^T \mathbf{W} \mathbf{p}_k$ may be interpreted as the variance explained by the k'th component.

Exercise 16.7, page 69

A function that computes MIC and performs CDA is the following

```
cda.eof = function(xdata,ydata,eof.list) {
### PERFORMS COVARIANCE DISCRIMINANT ANALYSIS ON X AND Y
### INPUT:
###   XDATA[NX,MDIM]
###   YDATA[NY,MDIM]
###   EOF.LIST: LIST FROM EOF CALCULATION
### OUTPUT:
###   MIC[NEOF]: MIC AS A FUNCTION OF NUMBER OF PCS
###   NMIN: LOCATION OF MINIMUM MIC
###   DISCR.RATIO[NEOF]: DISCRIMINANT RATIOS VS. NUMBER OF PCS
###   RX[NX,NMIN]: VARIATE TIME SERIES FOR X
###   RY[NY,NMIN]: VARIATE TIME SERIES FOR Y
###   PMAT[SPACE,NMIN]: LOADING VECTOR

neof       = eof.list$neof
if (length(xdata) %% neof != 0 ) stop('xdata not a multiple of neof')
if (length(ydata) %% neof != 0 ) stop('ydata not a multiple of neof')
nx         = length(xdata) / neof
ny         = length(ydata) / neof
dim(xdata) = c(nx,neof)
dim(ydata) = c(ny,neof)

nmax       = min(nx-1,ny-1,neof)

nt         = nx + ny
cov.x      = cov(xdata) * (nx-1)/nx
cov.y      = cov(ydata) * (ny-1)/ny
cov.t      = (nx * cov.x + ny * cov.y)/nt

mic        = as.numeric(rep(NA,nmax))
penalty    = as.numeric(rep(NA,nmax))
for (ne in 1:nmax) penalty[ne] = ne * ( nx*(nx+1)/(nx-ne-2) + ny*(ny+1)/(ny-ne-2) - nt*(nt+1)/(nt-ne-2))
for (ne in 1:nmax) {
mic[ne] = nx * log(det(cov.x[1:ne,1:ne,drop=FALSE])) +
          ny * log(det(cov.y[1:ne,1:ne,drop=FALSE])) -
          nt * log(det(cov.t[1:ne,1:ne,drop=FALSE])) + penalty[ne]
}

penalty = penalty / nt
mic     = mic     / nt
nmin    = which.min(mic)

### rescale covariances to be unbiased
cov.x   = cov.x * nx / (nx-1)
cov.y   = cov.y * ny / (ny-1)

gev.list    = gev(cov.x[1:nmin,1:nmin],cov.y[1:nmin,1:nmin])
discr.ratio = gev.list$lambda
rx          = xdata[,1:nmin] %*% gev.list$q
ry          = ydata[,1:nmin] %*% gev.list$q
pmat        = eof.list$eof[,1:nmin] %*% cov.y[1:nmin,1:nmin] %*% gev.list$q

list(mic=mic,nmin=nmin,discr.ratio=discr.ratio,rx=rx,ry=ry,pmat=pmat)
}
```

The MIC values for the data is

```
> mic
 [1] -0.19404378 -0.60912246 -0.60089462 -0.77768582 -0.97252532
```

Exercise 16.8, page 70

```
$discr.ratio
[1] 21.8064448  1.1885098  1.0114935  0.8586228  0.8169091
```

Exercise 16.9, page 70

The variates should look like those in fig. 16.3. The diagonal elements of the 20C variates match the discriminant ratios:

```
> diag(cov(cda.eof.list$rx))
[1] 21.8064448  1.1885098  1.0114935  0.8586228  0.8169091
> diag(cov(cda.eof.list$ry))
[1] 1 1 1 1 1
```

Exercise 16.10, page 70

The loading vector should look like that in fig. 16.3.

Exercise 16.11, page 70

An R code for Monte Carlo experiments is the following:

```
ntrials      = 1000
cov.identity = diag(1,nrow=nmin,ncol=nmin)
lambda.mc    = array(NA,dim=c(nmin,ntrials))
for ( nt in 1:ntrials) {
cov.x = rWishart(1,dof.20c-1,cov.identity)[,,1]/(dof.20c-1)
cov.y = rWishart(1,dof.ctr-1,cov.identity)[,,1]/(dof.ctr-1)
lambda.mc[,nt] = gev(cov.x,cov.y)$lambda
}
lambda.crit = array(NA,dim=c(nmin,2))
for ( n in 1:nmin) lambda.crit[n,] = quantile(lambda.mc[n,],probs=c(alpha/2,1-alpha/2))
```

The percentiles are

```
> lambda.crit
          [,1]      [,2]
[1,] 1.1030110 1.4559969
[2,] 0.9907935 1.2569631
[3,] 0.8839713 1.1269708
[4,] 0.7872901 1.0173159
[5,] 0.6806821 0.8910988
```

Exercise 16.12, page 70

The 95% percentiles as a function of T are

```
[1] "5% significance for univariate F-test= 1.59949546683544"
> lambda.crit
 [1]  1.603912  2.029011  2.235836  2.562755  2.863984  3.197075
 [7]  3.511517  3.839611  4.128509  4.473972  4.912822  5.326827
[13]  5.847943  6.402706  6.933168  7.538059  8.366698  9.042997
[19]  9.557897 10.486632 11.228660 12.446608 13.606677 14.982855
[25] 16.511167 18.575636 19.861070 22.171482 24.300277 27.020040
```

The critical values increase with dimension because each model is nested within the next, and the algorithm maximizes the discriminant ratio for each model. Looking in the reverse direction, as the dimension decreases, some of the parameters that were available for tuning become constrained, so then the maximum must decrease.

Exercise 17.1, page 73

An R function that performs ANOVA on a spatial field is 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

## COMPUTE GRAND MEAN
dim(x) = c(nspace,nens*ncon)
gmean  = rowMeans(x)

## COMPUTE ENSEMBLE MEAN
dim(x) = c(nspace,nens,ncon)
emean  = array(NA,dim=c(nspace,ncon))
for ( nc in 1:ncon) emean[,nc] = rowMeans(x[,,nc,drop=FALSE])

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

### COMPUTE NOISE
noise   = array(NA,dim=c(nspace,nens,ncon))
for ( nc in 1:ncon) noise[,,nc] = x[,,nc] - emean[,nc]

### COMPUTE NOISE VARIANCE
dim(noise) = c(nspace,nens*ncon)
var.nos    = rowSums(noise^2)/ncon/(nens-1)

### COMPUTE F
f = nens * var.sig / var.nos
f.crit = qf(alpha,ncon-1,ncon*(nens-1),lower.tail=FALSE)

list(f=f,f.crit=f.crit)

}
```

Exercise 17.2, page 74

The F-map should look like fig. 18.1 in the notes. The summary is

```
> summary(fval)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  1.512   2.698   3.700   4.064   4.918  32.843    3720
```

Exercise 17.3, page 74

The results should like figure 18.1 from the notes.

Exercise 17.4, page 75

Generally, F values decay as the number ensemble members decreases and as the lead time decreases.

Exercise 18.1, page 78

An R code for estimating significance levels of the eigenvalues from PrCA is the following:

```
prca.null.fixdim = function(ndim,ncon,nens,alpha=0.01,ntrial=10000) {
## ESTIMATES SIGNIFICANCE LEVEL OF THE ALL EIGENVALUES FROM PRCA/MANOVA
## FOR FIXED TRUNCATION NDIM
## BY MONTE CARLO METHODS
## INPUT:
#    NDIM: DIMENSION OF THE RANDOM VECTOR
#    NCOM: NUMBER OF CONDITIONS (E.G., NUMBER OF YEARS)
#    NENS: NUMBER OF ENSEMBLE MEMBERS (E.G., NUMBER OF REPITITIONS)
#    ALPHA: DESIRED SIGNIFICANCE LEVELS (CAN BE MORE THAN ONE)
#    NTRIALS: NUMBER OF MONTE CARLO TRIALS
## OUTPUT:
#    EVAL.CRIT[2,NDIM]: (alpha,1-alpha) SIGNIFICANCE THRESHOLDS FOR EACH EIGENVALUE

eval.max = array(NA,dim=c(ndim,ntrial))

dof.sig = ncon - 1
dof.nos = ncon*(nens-1)
for ( nt in 1:ntrial ) {
x       = array(rnorm(ndim*ncon*nens),dim=c(ndim*ncon,nens))
x.emean = rowMeans(x)
x.noise = x - x.emean

dim(x.emean) = c(ndim,ncon)
cov.sig      = nens * cov(t(x.emean))

dim(x.noise) = c(ndim,ncon*nens)
cov.nos      = x.noise %*% t(x.noise) / dof.nos

eval.max[,nt] = gev(cov.sig,cov.nos)$lambda
}

eval.crit = array(NA,dim=c(2,ndim))
for ( n in 1:ndim) eval.crit[,n] = quantile(eval.max[n,],probs=c(alpha,1-alpha))
rownames(eval.crit) = paste(c(alpha,1-alpha)*100,'%',sep='')

eval.crit
}
```

The resulting 1% significance thresholds for `ndim = 6` are

```
> fval.crit
         [,1]      [,2]      [,3]      [,4]      [,5]      [,6]
1%  1.274289 0.9646709 0.7218789 0.5190632 0.3530367 0.1903565
99% 2.728496 1.9477712 1.4799000 1.1425884 0.8754198 0.6339855
```

Exercise 18.2, page 78

```
> mic
 [1] -0.2422784468 -0.1859868967 -0.2475338825 -0.2474193296
 [5] -0.3759129623 -0.5164772016 -0.4870610152 -0.4322602904
 [9] -0.4153250531 -0.4605784420 -0.3959601760 -0.3631133884
[13] -0.3060199465 -0.2622818018 -0.1989315625 -0.1834919804
[17] -0.1291759564 -0.1230023572 -0.0768296814 -0.0253331366
[21]  0.0003693148 -0.1127739471 -0.0523793555  0.0607204885
[25]  0.1454873891  0.2427903034  0.2756050003  0.3125827453
[29]  0.2887010746  0.2935311458
```

Exercise 18.3, page 78

An R code that performs all steps in PrCA is the following:

```
prca.eof = function(data.eof,nens,ncon,alpha=0.01) {
[PREAMBLE OMITTED]

neof = data.eof$neof
ntot = dim(data.eof$pc)[1]
if ( nens * ncon != ntot) stop('time dimension inconsistent with nens,ncon')

######### COMPUTE MLE OF COVARIANCE MATRICES
fx         = data.eof$pc
dim(fx)    = c(nens,ncon*neof)
emean      = colMeans(fx)
noise      = t(t(fx)-emean)
dim(emean) = c(     ncon,neof)
dim(noise) = c(nens*ncon,neof)
dim(fx)    = c(nens*ncon,neof)
cov.s      = cov(emean) * (ncon-1)/ncon
cov.n      = cov(noise) * (ncon*nens-1)/(ncon*nens)
cov.t      = cov(fx   ) * (ncon*nens-1)/(ncon*nens)

######### COMPUTE MIC
mic        = as.numeric(rep(NA,neof))
penalty    = as.numeric(rep(NA,neof))
for ( n in 1:neof) if (nens*ncon-n-ncon-2>0){
penalty.random = (nens*ncon+1)*((n+ncon)/(nens*ncon-n-ncon-2) - n/(nens*ncon-n-2) - ncon/(nens*ncon-ncon-2))
penalty.fixed  = 2*(ncon-1)*n/(nens*ncon)
penalty.fixedc = (2*ncon*n + n*(n+1))/(ncon*nens-ncon-n-1) - (2*n+n*(n+1))/(ncon*nens-1-n-1)
penalty[n]     = penalty.fixedc
mic[n]         = log(det(cov.n[1:n,1:n,drop=FALSE])) -
                 log(det(cov.t[1:n,1:n,drop=FALSE])) +
                 penalty[n]
}
nmin = which.min(mic)

######### COMPUTE SIGNIFICANCE THRESHOLD FOR MIC
nu.e      = ncon*(nens-1)
nu.h      = ncon-1
mic.crit  = as.numeric(rep(NA,neof))
for ( n in 1:neof) if (nens*ncon-n-ncon-2>0) mic.crit[n] = -qchisq(1-alpha,n*nu.h)/(nu.e-(n-nu.h+1)/2) + penalty[n]

######### SOLVE PRCA/MANOVA
dof.s    = ncon - 1
dof.n    = ncon * (nens - 1)
dof.t    = ncon * nens - 1
cov.s    = cov.s * ncon        / dof.s
cov.n    = cov.n * ncon * nens / dof.n
cov.t    = cov.t * ncon * nens / dof.t
gev.list = gev(cov.s[1:nmin,1:nmin],cov.n[1:nmin,1:nmin])
q        = gev.list$q
f.max    = gev.list$lambda * nens
rsqr.adj = (f.max-1)/(f.max + ncon*(nens-1)/(ncon-1))
snr.adj  = (f.max-1)/nens

###  NORMALIZE Q
for ( n in 1:nmin) q[,n] = q[,n]/sqrt(as.numeric(t(q[,n])%*% cov.t[1:nmin,1:nmin] %*% q[,n]))

### COMPUTE VARIATES AND LOADING PATTERN
dim(fx)    = c(nens*ncon,neof)
r          = fx[,1:nmin] %*% q
p          = data.eof$eof[,1:nmin] %*% cov.t[1:nmin,1:nmin] %*% q

### FLIP SIGN
for ( n in 1:nmin) if (sum(p[,n],na.rm=TRUE) < 0) {p[,n] = -p[,n]; r[,n] = -r[,n]; q[,n] = -q[,n]}

### COMPUTE EXPLAINED VARIANCES
total.var = sum(data.eof$sval^2)/(ntot-1)
lgood     = !data.eof$lbad
fexpvar   = numeric(nmin)
for ( n in 1:nmin) fexpvar[n] = sum(data.eof$weight[lgood]^2 * p[lgood,n]^2) / total.var

##############################################
######### WRITE OUT RESULTS
##############################################
list(mic=mic,mic.crit=mic.crit,f.max=f.max,r=r,p=p,fexpvar=fexpvar,rsqr.adj=rsqr.adj,snr.adj=snr.adj,
    nmin=nmin,ncon=ncon,nens=nens,alpha=alpha)
}
```

The maximized F-ratios are

```
> prca.list$f.max
[1] 19.4746501  3.1956229  2.7206396  1.0809605  0.6636562  0.6385614
> prca.list$rsqr.adj
[1]  0.527391997  0.117092530  0.094146488  0.004866437 -0.020737369
[6] -0.022319127
> prca.list$snr.adj
[1]  1.154665632  0.137226433  0.107539975  0.005060034 -0.021021486
[6] -0.022589909
```

Exercise 18.4, page 79

See fig. 22.18.

Exercise 18.5, page 79

The following shows that the sample covariance matrix of the predictable components equals the identity matrix.

```
> signif(cov(r),4)
           [,1]       [,2]       [,3]       [,4]       [,5]
[1,]  1.000e+00  6.578e-18 -2.864e-16  8.649e-17  1.190e-17
[2,]  6.578e-18  1.000e+00 -1.558e-16  2.514e-16  1.173e-16
[3,] -2.864e-16 -1.558e-16  1.000e+00  8.631e-17 -1.065e-16
[4,]  8.649e-17  2.514e-16  8.631e-17  1.000e+00 -2.067e-16
[5,]  1.190e-17  1.173e-16 -1.065e-16 -2.067e-16  1.000e+00
[6,] -1.353e-16 -4.863e-17 -8.512e-17  1.338e-16 -3.679e-16
           [,6]
[1,] -1.353e-16
[2,] -4.863e-17
[3,] -8.512e-17
[4,]  1.338e-16
[5,] -3.679e-16
[6,]  1.000e+00
```

Exercise 18.6, page 79

See fig. **??**

Exercise 19.2, page 83

A function that solves the Kalman Filter equations is the following:

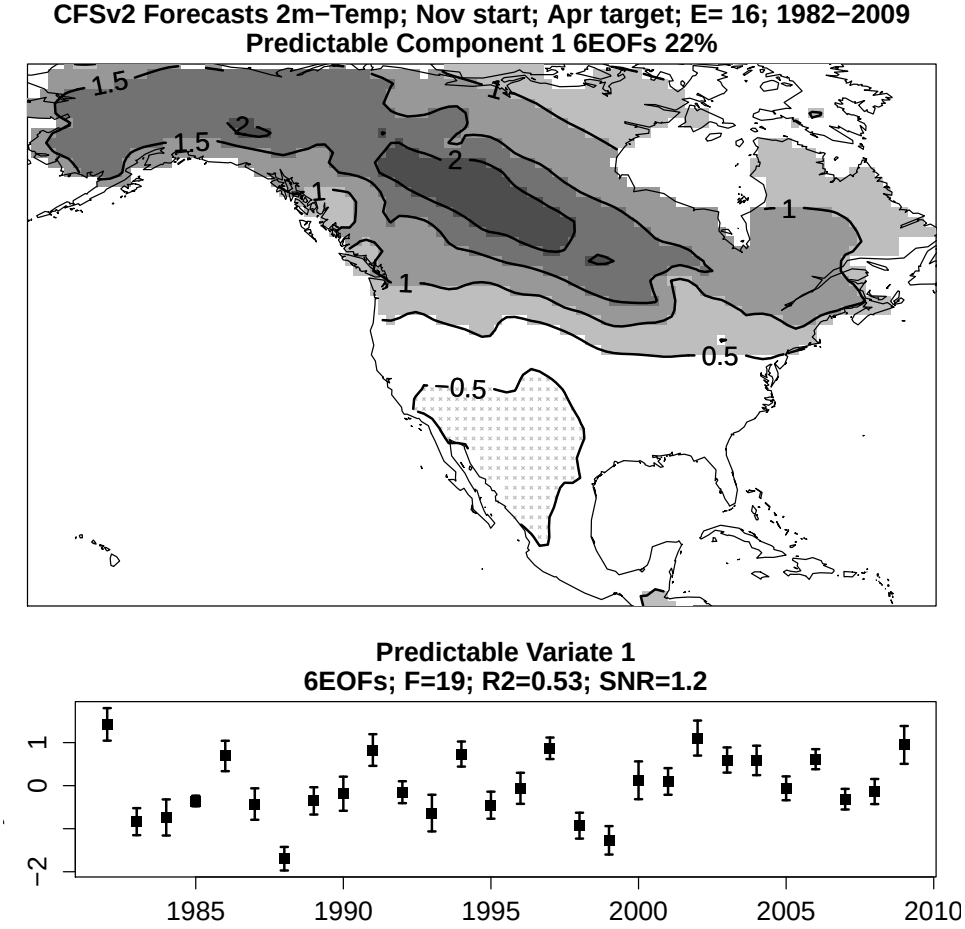


Figure 22.17 First predictable component.

```
kalman.population = function(mub,sigmab,hop,r.cov,obs) {
##########
## EVALUATES THE KALMAN FILTER EQUATIONS FOR THE MEAN AND COVARIANCE MATRIX OF THE ANALYSIS
## BASED ON *POPULATION* COVARIANCE MATRICES AND MEANS
## INPUT:
#    MUB:    [NDIM] MEAN OF THE BACKGROUND DISTRIBUTION
#    SIGMAB: [NDIM,NDIM] BACKGROUND COVARIANCE MATRIX
#    HOP:    [NOBS,NDIM] INTERPOLATION OPERATOR
#    R.COV:  [NOBS,NOBS] COVARIANCE MATRIX OF THE OBSERVATIONAL ERROR
#    OBS:    [NOBS] THE OBSERVATIONS
## OUTPUT
#    MUA:    [NDIM] VECTOR OF THE ANALYSIS DISTRIBUTION
#    SIGMAA: [NDIM,NDIM] COVARIANCE MATRIX OF THE ANALYSIS DISTRIBUTION

xo.cov    = sigmab %*% t(hop)
o.cov     = hop %*% xo.cov + r.cov
o.cov.inv = chol2inv(chol(o.cov))
kgain     = xo.cov %*% o.cov.inv

mua       = mub    + kgain %*% ( obs - hop %*% mub )
sigmaa    = sigmab - kgain %*% t(xo.cov)

list(mua=mua,sigmaa=sigmaa)
}
```

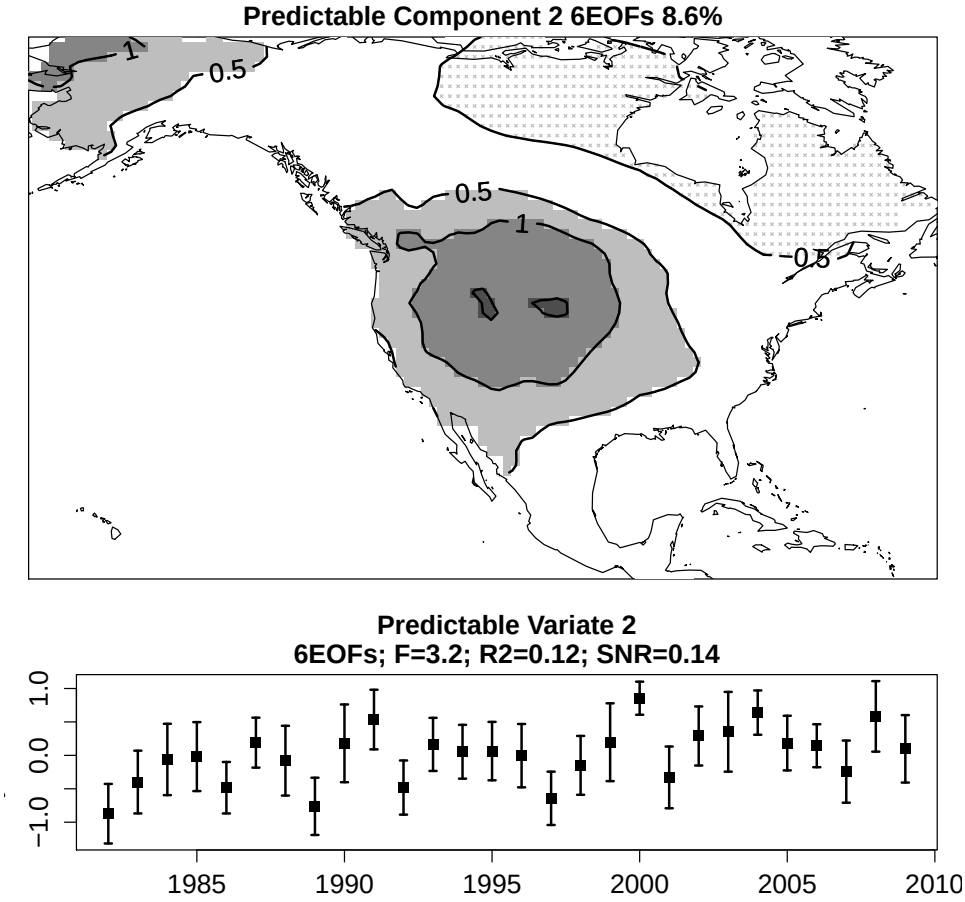


Figure 22.18 Second predictable component

Exercise 19.3, page 83

A loop that solves for the analysis based on observations and background distribution is the following:

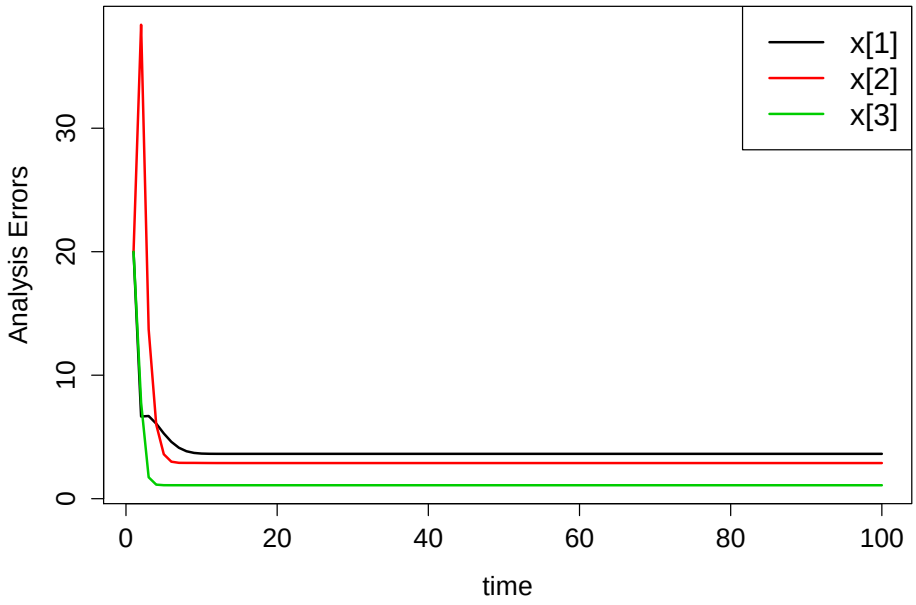


Figure 22.19 Time series of the standard error of the analysis.

```
### ASSIMILATE OBSERVATIONS
mua = matrix(rep(0,ndim*ntot),nrow=ndim)
pa  = array(rep(0,ndim^2*ntot),dim=c(ndim,ndim,ntot))
for ( n in 1:ndim) pa[n,n,1] = 400

for ( n in 2:ntot) {
  muf = dynop %*% mua[,n-1]
  pf  = dynop %*% pa[,,n-1] %*% t(dynop) + q.cov

  kf  = kalman.population(muf,pf,hop,r.cov,obs[,n])
  mua[,n] = kf$mua
  pa[,,n] = kf$sigmaa
}
```

Exercise 19.4, page 83

A plot of the analysis errors for the three elements of $\mathbf{x}_t$, as given by the square root of the diagonal elements of $\mathbf{\Sigma}_t^B$, is shown in fig. 22.19. The errors asymptote to a constant after a long time.

Exercise 19.5, page 83

Plots for the case $\mathbf{R}_t = 1$ are shown in fig. 22.20. The standard error of the analysis is much smaller in this case, because the observational uncertainty is much smaller and hence provides more information about the true state than in the previous case. The analysis errors are always smaller than either the background or observational uncertainties.

Exercise 19.6, page 83

To assimilate the linear combination $z(t) = (\mathbf{x}_t)_1 + (\mathbf{x}_t)_2$, we would use the following operators in the observation equation

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{R}_t = 25. \tag{22.91}$$

The result of assimilating these observations to estimate $(\mathbf{x}_t)_1$ is shown in fig. 22.21.

Exercise 19.7, page 85

The mean square error of the prediction model $\hat{\mathbf{y}} = \mathbf{A}\mathbf{o} + \mathbf{b}$ is

$$\text{MSE} = \mathbb{E}\left[(\mathbf{y} - \mathbf{A}\mathbf{o} - \mathbf{b})^T (\mathbf{y} - \mathbf{A}\mathbf{o} - \mathbf{b}) \right] \tag{22.92}$$

Taking the derivative with respect to $\mathbf{A}$ and $\mathbf{b}$ gives

$$\frac{\partial \text{MSE}}{\partial \mathbf{A}} = -2\mathbb{E}\left[(\mathbf{y} - \mathbf{A}\mathbf{o} - \mathbf{b})\, \mathbf{o}^T \right] = -2\mathbb{E}\left[\mathbf{y}\mathbf{o}^T - \mathbf{A}\mathbf{o}\mathbf{o}^T - \mathbf{b}\mathbf{o}^T \right] \tag{22.93}$$

$$\frac{\partial \text{MSE}}{\partial \mathbf{b}} = -2\mathbb{E}\left[\mathbf{y} - \mathbf{A}\mathbf{o} - \mathbf{b} \right]. \tag{22.94}$$

Setting the second equation to zero and solving yields

$$\mathbf{b} = \mathbb{E}[\mathbf{y}] - \mathbf{A}\mathbb{E}[\mathbf{o}]. \tag{22.95}$$

Substituting this into the first equation yields

$$\frac{\partial \text{MSE}}{\partial \mathbf{A}} = -2\mathbb{E}\left[\mathbf{y}\mathbf{o}^T - \mathbf{A}\mathbf{o}\mathbf{o}^T - (\mathbb{E}[\mathbf{y}] - \mathbf{A}\mathbb{E}[\mathbf{o}])\, \mathbf{o}^T \right] \tag{22.96}$$

$$= (-2)\left(\mathbb{E}[\mathbf{y}\mathbf{o}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{o}]^T \right) + 2\left(\mathbf{A}\mathbb{E}[\mathbf{o}\mathbf{o}^T] - \mathbf{A}\mathbb{E}[\mathbf{o}]\mathbb{E}[\mathbf{o}^T] \right) \tag{22.97}$$

$$= (-2)\left(\mathbf{\Sigma}_{YO} - \mathbf{A}\mathbf{\Sigma}_{OO} \right), \tag{22.98}$$

where we have used the fact that for any random vectors $\mathbf{x}$ and $\mathbf{y}$,

$$\mathbf{\Sigma}_{XY} = \mathbb{E}\left[(\mathbf{x} - \mathbb{E}[\mathbf{x}])\, (\mathbf{y} - \mathbb{E}[\mathbf{y}])^T \right] = \mathbb{E}[\mathbf{x}\mathbf{y}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}]^T. \tag{22.99}$$

Setting the derivative to zero and solving yields

$$\mathbf{A} = \mathbf{\Sigma}_{YO}\mathbf{\Sigma}_{OO}^{-1}, \tag{22.100}$$

as desired.

Given $\mathbf{o} = \mathbf{H}\mathbf{y} + \mathbf{r}$, it follows that

$$\mathbb{E}[\mathbf{o}] = \mathbf{H}\mathbb{E}[\mathbf{y}] + \mathbb{E}[\mathbf{r}] = \mathbf{H}\boldsymbol{\mu}_B \tag{22.101}$$

$$\operatorname{cov}[\mathbf{o}, \mathbf{y}] = \operatorname{cov}[\mathbf{H}\mathbf{y} + \mathbf{r}, \mathbf{y}] = \mathbf{H}\operatorname{cov}[\mathbf{y}] + \operatorname{cov}[\mathbf{r}, \mathbf{y}] = \mathbf{H}\boldsymbol{\Sigma}_B \tag{22.102}$$

$$\operatorname{cov}[\mathbf{o}] = \operatorname{cov}[\mathbf{H}\mathbf{y} + \mathbf{r}, \mathbf{H}\mathbf{y} + \mathbf{r}] = \mathbf{H}\operatorname{cov}[\mathbf{y}]\mathbf{H}^T + \operatorname{cov}[\mathbf{r}] = \mathbf{H}\boldsymbol{\Sigma}_B\mathbf{H}^T + \mathbf{R}. \tag{22.103}$$

where we used $\mathbb{E}[\mathbf{r}] = \mathbf{0}$ and $\operatorname{cov}[\mathbf{r}, \mathbf{y}] = \mathbf{0}$. Substituting these expressions into the least squares estimates for $\mathbf{A}$ gives

$$\mathbf{A} = (\mathbf{H}\boldsymbol{\Sigma}_B)^T \left(\mathbf{H}\boldsymbol{\Sigma}_B\mathbf{H}^T + \mathbf{R}\right)^{-1} = \boldsymbol{\Sigma}_B\mathbf{H}^T \left(\mathbf{H}\boldsymbol{\Sigma}_B\mathbf{H}^T + \mathbf{R}\right)^{-1}. \tag{22.104}$$

Substituting the expressions into the least squares estimate for $\mathbf{b}$ gives

$$\mathbf{b} = \boldsymbol{\mu}_B - \boldsymbol{\Sigma}_B\mathbf{H}^T \left(\mathbf{H}\boldsymbol{\Sigma}_B\mathbf{H}^T + \mathbf{R}\right)^{-1} \mathbf{H}\boldsymbol{\mu}_B \tag{22.105}$$

Therefore, the least squares prediction $\hat{\mathbf{y}}$ is

$$\hat{\mathbf{y}} = \mathbf{A}\mathbf{o} + \mathbf{b} = \boldsymbol{\mu}_B + \boldsymbol{\Sigma}_B\mathbf{H}^T \left(\mathbf{H}\boldsymbol{\Sigma}_B\mathbf{H}^T + \mathbf{R}\right)^{-1} (\mathbf{o} - \mathbf{H}\boldsymbol{\mu}_B) \tag{22.106}$$

Exercise 20.1, page 88

The symmetric square root requires

$$\mathbf{I} - \beta\mathbf{w}\mathbf{w}^T = \left(\mathbf{I} - \delta\beta\mathbf{w}\mathbf{w}^T\right)\left(\mathbf{I} - \delta\beta\mathbf{w}\mathbf{w}^T\right). \tag{22.107}$$

Expanding the right hand side yields

$$\mathbf{I} - \beta\mathbf{w}\mathbf{w}^T = \mathbf{I} - 2\delta\beta\mathbf{w}\mathbf{w}^T + \delta^2\beta^2\left(\mathbf{w}^T\mathbf{w}\right)\mathbf{w}\mathbf{w}^T. \tag{22.108}$$

Simplifying this equation yields

$$\left(\delta^2\beta\left(\mathbf{w}^T\mathbf{w}\right) - 2\delta\beta + \beta\right)\mathbf{w}\mathbf{w}^T = 0. \tag{22.109}$$

This equation is satisfied only for the choices

$$\delta = \frac{1 \pm \sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}}{\beta\mathbf{w}^T\mathbf{w}}. \tag{22.110}$$

Multiplying the top and bottom by the conjugate gives

$$\begin{aligned}
\delta &= \frac{1 \pm \sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}}{\beta\mathbf{w}^T\mathbf{w}} \frac{1 \mp \sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}}{1 \mp \sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}} \\
&= \frac{1 - \left(1 - \beta\mathbf{w}^T\mathbf{w}\right)}{\beta\mathbf{w}^T\mathbf{w}} \frac{1}{1 \mp \sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}} \\
&= \frac{1}{1 \mp \sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}}
\end{aligned} \tag{22.111}$$

To obtain a positive semi-definite square root matrix, we chose δ to render

$$\mathbf{x}^T \left(\mathbf{I} - \delta\beta \mathbf{w}\mathbf{w}^T\right) \mathbf{x} \geq 0 \quad \text{for all } x. \tag{22.112}$$

Expanding the left hand side yields

$$\mathbf{x}^T \left(\mathbf{I} - \delta\beta \mathbf{w}\mathbf{w}^T\right) \mathbf{x} = \mathbf{x}^T\mathbf{x} - \delta\beta \left(\mathbf{x}^T\mathbf{w}\right)^2. \tag{22.113}$$

The Cauch-Schwartz inequality implies

$$\epsilon = \frac{\left(\mathbf{x}^T\mathbf{w}\right)^2}{\left(\mathbf{x}^T\mathbf{x}\right)\left(\mathbf{w}^T\mathbf{w}\right)} \leq 0. \tag{22.114}$$

Therefore, we have

$$\begin{aligned}
\mathbf{x}^T\mathbf{x} - \delta\beta \left(\mathbf{x}^T\mathbf{w}\right)^2 &= \left(\mathbf{x}^T\mathbf{x}\right)\left(\mathbf{w}^T\mathbf{w}\right)\left(\frac{1}{\mathbf{w}^T\mathbf{w}} - \delta\beta\epsilon\right) \\
&= \left(\mathbf{x}^T\mathbf{x}\right)\left(1 - \left(1 \pm \sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}\right)\epsilon\right) \\
&= \left(\mathbf{x}^T\mathbf{x}\right)\left(1 - \epsilon \mp \epsilon\sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}\right)
\end{aligned} \tag{22.115}$$

Since $\beta \leq \mathbf{w}^T\mathbf{w}$, the term in the radical is between 0 and 1. Since ϵ also is between 0 and 1, the quadratic form is positive for all ϵ, including $\epsilon = 1$, only if δ is chosen as

$$\delta = \frac{1 - \sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}}{\beta\mathbf{w}^T\mathbf{w}} = \frac{1}{1 + \sqrt{1 - \beta\mathbf{w}^T\mathbf{w}}} \tag{22.116}$$

Exercise 20.2, page 88

Proof. First consider the case in which $\mathbf{B}$ is rank-1, which implies that $\mathbf{B} = \mathbf{u}\mathbf{u}^T$ for some vector $\mathbf{u}$. Then

$$\mathbf{C} \circ \mathbf{B} = \mathbf{C} \circ \mathbf{u}\mathbf{u}^T = C_{ij}u_iu_j = \mathbf{D}_u\mathbf{C}\mathbf{D}_u, \tag{22.117}$$

where $\mathbf{D}_u$ is a diagonal matrix whose diagonal elements equal the vector elements of $\mathbf{u}$. The last matrix on the right hand side is positive semi-definite, as shown below:

$$\mathbf{x}^T \left(\mathbf{C} \circ \mathbf{B}\right) \mathbf{x} = \mathbf{x}^T\mathbf{D}_u\mathbf{C}\mathbf{D}_u\mathbf{x} = \mathbf{y}^T\mathbf{C}\mathbf{y} \geq 0, \tag{22.118}$$

where we have defined $\mathbf{y} = \mathbf{D}_u\mathbf{x}$. Since the above inequality holds for arbitrary $\mathbf{x}$, $\mathbf{C} \circ \mathbf{B}$ is positive semi-definite. Now consider rank greater than 1. By the spectral factorization theorem, $\mathbf{B} = \sum \mathbf{u}_i\mathbf{u}_i^T$ for some set of vectors $\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_N$. Then

$$\mathbf{C} \circ \mathbf{B} = \mathbf{C} \circ \sum_i \mathbf{u}_i\mathbf{u}_i^T = \mathbf{C} \circ \mathbf{u}_1\mathbf{u}_1^T + \mathbf{C} \circ \mathbf{u}_2\mathbf{u}_2^T + \cdots + \mathbf{C} \circ \mathbf{u}_N\mathbf{u}_N^T. \tag{22.119}$$

Each term is positive semi-definite, so therefore the sum is too. *QED*

Exercise 21.1, page 90

```
> cold.x

fevd(x = t.cold)
...
 Estimated parameters:
  location      scale      shape
 6.5115237  3.5662767 -0.2347719

 Standard Error Estimates:
  location      scale      shape
0.58044433 0.40333521 0.09324019

 Estimated parameter covariance matrix.
             location       scale        shape
location  0.33691562  0.01740978 -0.020276953
scale     0.01740978  0.16267929 -0.019456804
shape    -0.02027695 -0.01945680  0.008693734
```

The return levels are

```
> return.level(cold.x)
fevd(x = t.cold)
get(paste("return.level.fevd.", newcl, sep = ""))(x = x, return.period = return.period)

 GEV model fitted to  t.cold
Data are assumed to be  stationary
[1] "Return Levels for period units in years"
  2-year level  20-year level 100-year level
      7.763954      14.138334      16.543253
```

Exercise 21.2, page 90

The shape parameter is

$$\xi = -0.23 \pm 0.09, \tag{22.120}$$

so it is negative even for 2 standard errors, which means it is a Weibull distribution.

Exercise 21.3, page 90

```
> cold.x.tele

fevd(x = t.cold, location.fun = ~index)

[1] "Estimation Method used: MLE"

...
 Estimated parameters:
       mu0        mu1      scale      shape
 6.4879640 -1.4483469  3.0933949 -0.1545121

 Standard Error Estimates:
      mu0       mu1     scale     shape
0.5231300 0.4817077 0.3851407 0.1244832

 Estimated parameter covariance matrix.
               mu0         mu1       scale       shape
mu0    0.27366503  0.03729961  0.05315883 -0.02825529
mu1    0.03729961  0.23204226  0.04143058 -0.02199465
scale  0.05315883  0.04143058  0.14833340 -0.02640878
shape -0.02825529 -0.02199465 -0.02640878  0.01549606
```

The coefficient for the location parameter is

$$\mu = -1.45 \pm 0.48. \tag{22.121}$$

This is non-zero even for 2 standard errors, so it is significant. A more accurate test is the likelihood ratio test:

```
> lr.test(cold.x,cold.x.tele)

Likelihood-ratio Test

data:  t.coldt.cold
Likelihood-ratio = 8.2806, chi-square critical value = 3.8415,
alpha = 0.0500, Degrees of Freedom = 1.0000, p-value = 0.004007
alternative hypothesis: greater
```

The p-value is less than 5%, hence significant.

The shape parameter is not significantly different from zero or positive at the 5% level, so it could be any of the 3 distributions.

Exercise 21.4, page 90

The probability that an exceedance occurs in year 1 is p. The probability that an exceedance occurs in year 2 is the probability that it did not occur in year 1, times the probability that it did occur in year 2, which gives $(1-p)p$. Similarly, exceedance in year 3 has probability $(1-p)^2p$, and exceedance in year y has probability $p(1-p)^{y-1}$. Thus, the average waiting

time is

$$\text{return period} = p \sum_{y=1}^{\infty} y\,(1-p)^{y-1}\,. \tag{22.122}$$

To evaluate this sum, recall the sum of an infinite geometric series is

$$\sum_{k=0}^{\infty} q^k = \frac{1}{1-q} \quad \text{for } |q| < 1. \tag{22.123}$$

Differentiating this equation with respect to q yields the identity

$$\sum_{k=0}^{\infty} k q^{k-1} = \frac{1}{(1-q)^2} \quad \text{for } |q| < 1. \tag{22.124}$$

This identity gives the sum in (22.122) in the case $q = 1 - p$, thus

$$\text{return period} = p \frac{1}{(1-(1-p))^2} = \frac{1}{p}. \tag{22.125}$$

Thus, if the probability of exceeding x_p in any given year is p, then the return period is $1/p$ years.

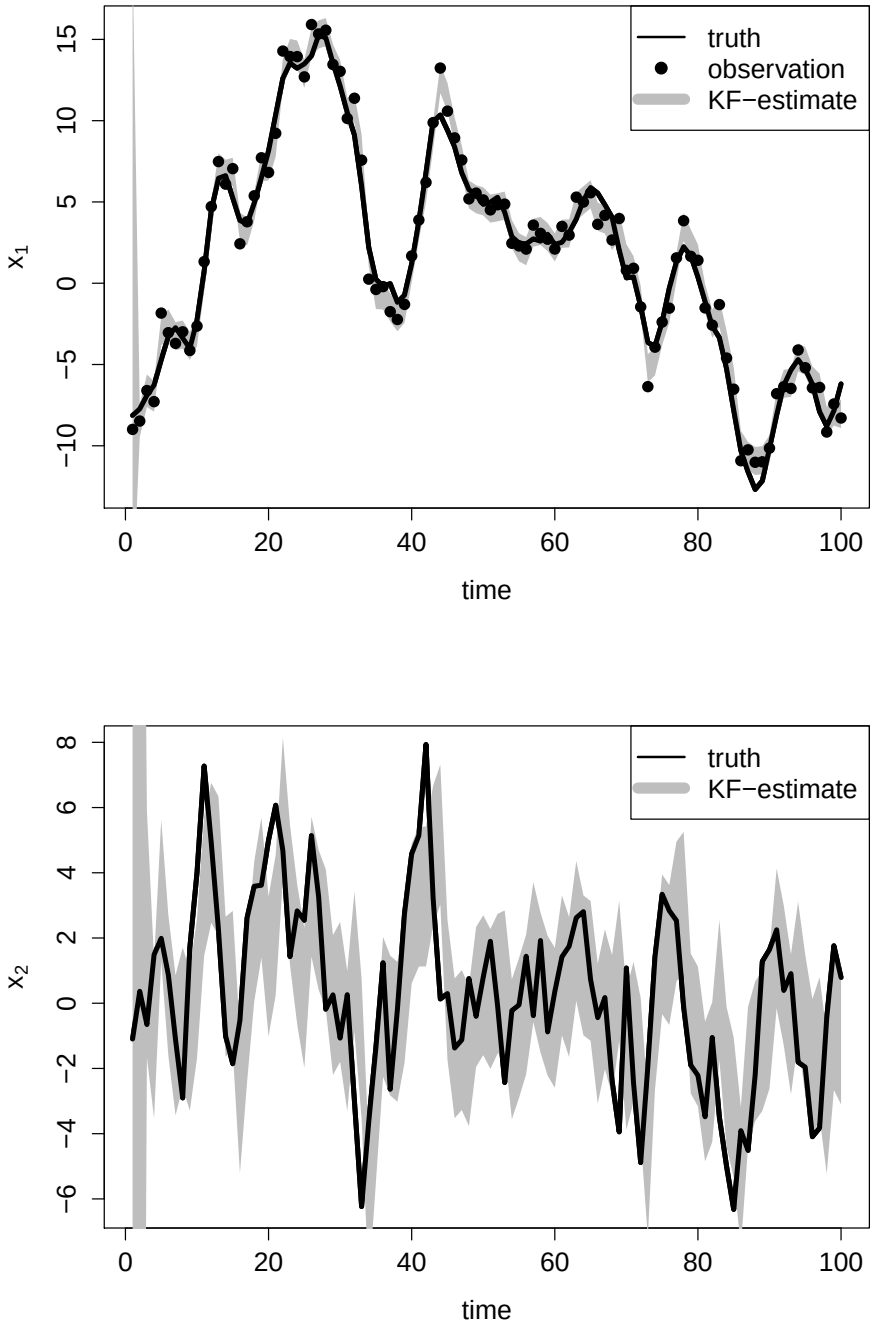


Figure 22.20 A particular realization of the first (top) and second (bottom) elements of $\mathbf{y}(t)$ generated by the vector autoregressive model (solid curve), corresponding observations (dots), and the analysis mean plus a standard deviation, as estimated state by the Kalman Filter (grey shading), but using $\mathbf{R}_t = 1$

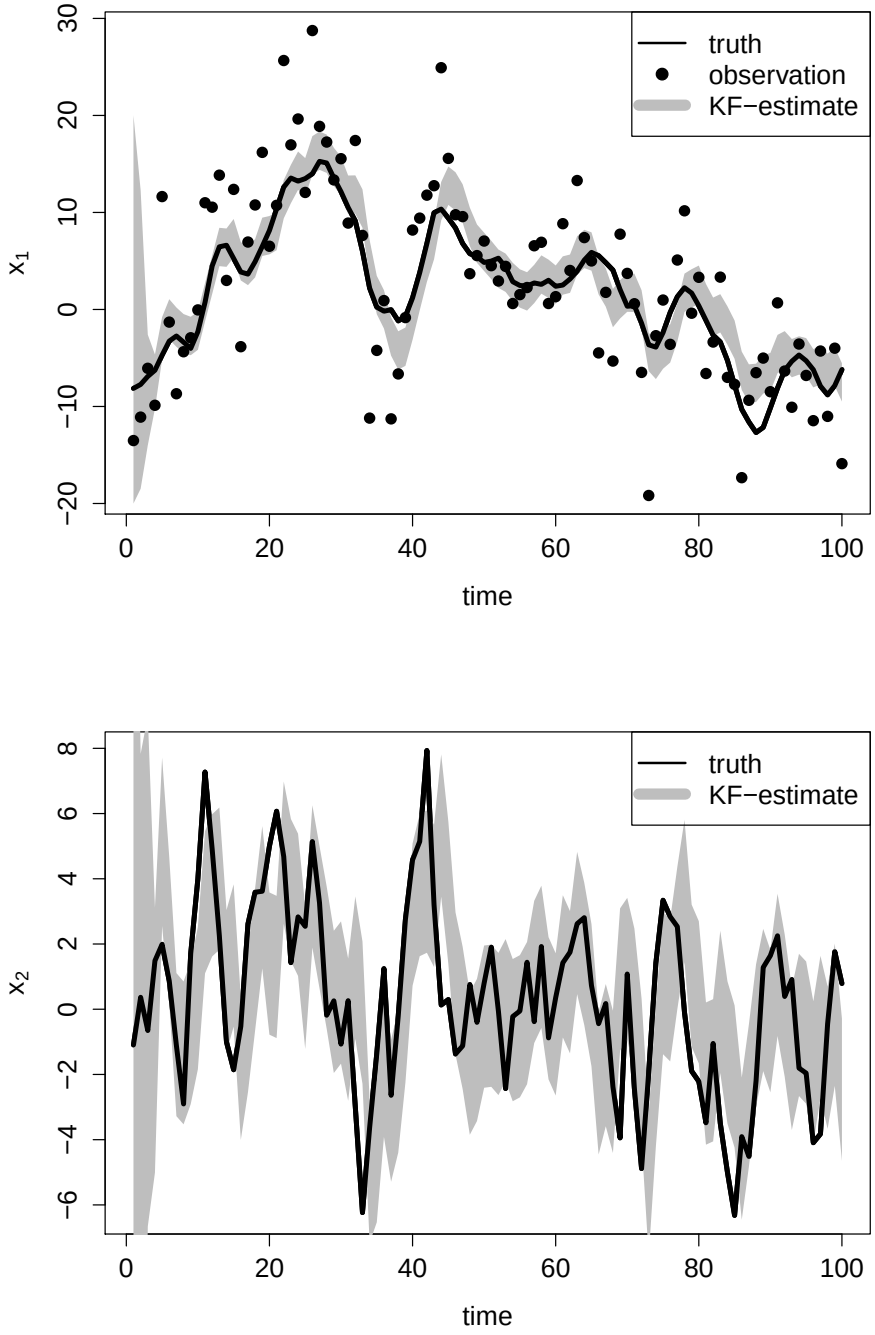


Figure 22.21 A particular realization of the first (top) and second (bottom) elements of $\mathbf{y}(t)$ generated by the vector autoregressive model (solid curve), corresponding observations (dots), and the analysis mean plus a standard deviation, as estimated state by the Kalman Filter (grey shading), but using $\mathbf{R}_t = 25$, and observing the sum of the first two elements.