University of California, Riverside
ENVIRONMENTAL STATISTICS

Course Number: ENSC 110
Quarter: Fall (2019–2020 academic year)
Units: 4 (Lecture: 3 hours; Laboratory: 3 hours)
Lecture: MWF 9:00 AM – 9:50 AM (Sci Labs 301)
Laboratory: W 4:00 PM – 6:50 PM (Sproul 2225)

Topics covered in Fall 2019

(Note: R will be used instead of Matlab starting from Academic Year 2020/2021)

Matrix algebra
Definitions of row/column vectors and $N \times M$ matrices, matrix multiplication, transposes, determinants, cofactors, inverse, identity matrices.

Matlab programming
Storing data as vectors and matrices in Matlab, matrix operations, matrix multiplication versus elementwise multiplication, plotting, input/output files, logics and loops, Fibonacci sequence, Bisection method.

Summary statistics
Sample mean, sample standard deviation, standard error of sample mean, Central Limit Theorem, weighted mean, weighted standard deviation, weighted standard error, deseasonalization of climate data

Hypothesis testing
Generating random number using Lehman's scheme, verifying the Central Limit Theorem by Monte Carlo simulation, testing H_0 and H_1 hypotheses using Monte Carlo simulation and Bootstrap resampling, testing the toxicity of the environment

Linear methods
Linear/quadratic/Lagrange interpolation, system of linear equations, Gaussian elimination, linear matrix equations, linear/non-linear regression, least-squares minimization.

Mathematical formulae and concepts discussed in class:

Matrix transpose of a 3×2 matrix: $\begin{pmatrix} a & b \\ c & d \\ e & f \end{pmatrix}^T = \begin{pmatrix} a & c & e \\ b & d & f \end{pmatrix}$

Matrix multiplication of a 2×2 matrix: $\begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} e & f \\ g & h \end{pmatrix} = \begin{pmatrix} ae+bg & af+bh \\ ce+dg & cf+dh \end{pmatrix}$

Determinant of a 2×2 matrix: $\Delta = \det\begin{pmatrix} a & b \\ c & d \end{pmatrix} = ad - bc$

Matrix inverse of a 2×2 matrix: $\begin{pmatrix} a & b \\ c & d \end{pmatrix}^{-1} = \frac{1}{\Delta}\begin{pmatrix} d & -b \\ -c & a \end{pmatrix}$

Bisection method: For a continuous function $f(x)$, if $a<b$ and $f(a)f(b)\le 0$, then there must exist at least one root *r* lying between *a* and *b*.

Fibonacci sequence in ecology: $F_1 = F_2 = 1;\ F_n = F_{n-1} + F_{n-2},\ n \ge 2$.

Weighted sample mean: $\bar{x} = \dfrac{\sum_{j=1}^{N} w_j x_j}{\sum_{j=1}^{N} w_j}$.

Weighted sample variance:

$$s^2 = \frac{\sum_{j=1}^{n} w_j \left(x_j - \bar{x}\right)^2}{\sum_{j=1}^{n} w_j} \qquad \textbf{(biased)}$$

$$\hat{s}^2 = \frac{s^2}{1 - \left(\sum_{j=1}^{n} w_j^2\right) \Big/ \left(\sum_{j=1}^{n} w_j\right)^2} \qquad \textbf{(unbiased)}$$

Gaussian (normal) distribution: $f(x;\mu,\sigma) = \dfrac{1}{\sigma\sqrt{2\pi}}\exp\left[-\dfrac{1}{2}\left(\dfrac{x-\mu}{\sigma}\right)^2\right]$, where $-\infty \le x \le \infty$.

Lehmer's scheme: $r_n = a r_{n-1} \bmod m,\ n \ge 2.\ r_1$ is the seed.

Monte Carlo/Bootstrap tests: Given a dataset $\mathbf{X}$, the statistical significance of a sample statistics $\theta(\mathbf{X})$ can be tested against the distribution of $\{\theta(\mathbf{X}^*)\}$, where $\{\mathbf{X}^*\}$ are simulated samples or resamples.

Interpolation: Given *n* data points, the unique polynomial that passes through all data points is given by $y = \sum_{j=1}^{n} p_j x^{n-1}$.

Vandermonde matrix: Given a vector $\vec{x} = [x_1 \ \cdots \ x_n]^T$, the Vandermonde matrix is a *n*×*n* square matrix, whose (*i*,*j*)-th element V_{ij} is given by x_i^{j-1}.

The least-squares solutions for:

(a) Line-fit $y = p_1 x + p_2$: $p_1 = \dfrac{\overline{xy} - \bar{x}\,\bar{y}}{\overline{x^2} - \bar{x}^2}$ and $p_2 = \bar{y} - p_1\bar{x}$.

(b) Multi-linear regression $\mathbf{y} = \mathbf{Xp} + \boldsymbol{\varepsilon}$: $\mathbf{p} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{y}$.

95% confidence interval for p_1: $p_1 \pm z_{0.025}\sigma_{p_1}$, where $z_{0.025} = 1.96$ for large samples,

$$\sigma_{p_1}^2 = \frac{n\sigma^2}{n\sum_{j=1}^{n} x_j^2 - \left(\sum_{j=1}^{n} x_j\right)^2}, \text{ and } \sigma^2 = \frac{1}{n-2}\sum_{k=1}^{n} \varepsilon_k^2 .$$

Matlab commands discussed in class:

save filename var format stores the specified variable **var** in a file named **filename**. **format** may be **'-mat'** for a binary MAT-file format (default) or **'-ascii'** 8-digit ASCII format.

plot(X,Y,LineSpec,value) plots vector **X** versus vector **Y** on the same axes. **LineSpec** and **value** together specify the line type, marker symbol, and color.

[j:i:k] creates a regularly-spaced vector going from **j** to **k** at a step **i**.

abs(x) returns the absolute value of **x**.

log10(x) returns the common logarithm (base 10) of **x**.

X' returns the complex-conjugate transpose of the 2D matrix **X**.

inv(X) returns the inverse of the square matrix **X**.

find(X) returns a vector containing the linear indices of each nonzero element in array **X**.

zeros(sz1,...,szN) and **ones(sz1,...,szN)** returns an **sz1**-by-**sz2**-by-…-by-**szN** array of 0's and 1's, respectively. e.g., **zeros(2,3)** returns a 2-by-3 array of 0's; **ones(5)** returns a 5-by-5 array of 1's.

diag(A) returns the diagonal of the matrix **A** or creates a square matrix with the vector **A** on the diagonal.

A*B performs a matrix multiplication of A and B.

A.*B and **A./B** performs element-by-element multiplication and division of A and B.

A\B solves the system of linear equations **A*x=B**. The matrices **A** and **B** must have the same number of rows. If **A** is a rectangular **m**-by-**n** matrix with **m~=n**, and **B** is a column vector with **m** elements or a matrix with **m** rows, then **A\B** returns a least-squares solution.

sum(A,dim) and **prod(A,dim)** returns the sum and the product of the array elements of **A**, respectively, along the dimension **dim**.

mean(A,dim) returns the mean dimension **dim**. For example, if **A** is a matrix, then **mean(A,2)** is a column vector containing the mean of each row.

std(A,flag,dim) and **var(A,flag,dim)** returns the standard deviation and the variance, respectively, along dimension **dim**. When **flag=0** (default), the return value is normalized by $N-1$, where N is the number of observations. When **flag=1**, the return value is normalized by N.

length(X) returns the length of the largest array dimension in **X**.

numel(A) returns the number of elements, **n**, in array **A**, equivalent to **prod(size(A))**.

rem(a,m) returns the remainder after division of **a** by **m**, where **a** is the dividend and **m** is the divisor.

rand([sz1,...,szN]) returns uniformly distributed pseudorandom numbers over the interval (0,1) in an **sz1**-by-**sz2**-by-…-by-**szN** array.

randi(imax,[sz1,...,szN]) returns pseudorandom integers between 1 and **imax** in an **sz1**-by-**sz2**-by-…-by-**szN** array.

randn([sz1,...,szN]) returns Gaussian distributed pseudorandom numbers with a mean 0 and a standard deviation 1 in an **sz1**-by-**sz2**-by-…-by-**szN** array.

normrnd(mu,sigma,[sz1,...,szN]) generates an **sz1**-by-**sz2**-by-…-by-**szN** array of normal random numbers drawn from the normal distribution with mean **mu** and standard deviation **sigma**.

p=polyfit(x,y,n) finds the coefficients of a polynomial $p(x)$ of degree n that fits the data in a least squares sense. The result **p** is a row vector of length **n+1** containing the polynomial coefficients in descending powers: $y = p_1 x^n + p_2 x^{n-1} + p_3 x^{n-3} + \cdots + p_n x + p_{n+1} + \text{residual}$.

y=polyval(p,x) returns the value of a polynomial of degree **n** evaluated at **x**. **p** is a vector of length **n+1** whose elements are the coefficients in descending powers of the polynomial to be evaluated.

vq=interp1(x,v,xq) returns interpolated values of a 1-D function at specific query points using linear interpolation. Vector **x** contains the sample points, and **v** contains the corresponding values, $v(x)$. Vector **xq** contains the coordinates of the query points.

y0=spline(x,y,x0) uses a cubic Hermite spline interpolation to find **y0** at **x0** given **y** as a function of **x**.

[p,pint] = regress(y,X) returns a vector **p** of coefficient estimates for a multiple linear regression of the responses in vector **y** on the predictors in matrix **X**. The matrix **X** must include a column of ones. The matrix **pint** returns the 95% confidence intervals for the coefficient estimates in rows.

p=lsqcurvefit(fun,p0,x,y) starts at **p0** and finds coefficients **p** to best fit the nonlinear function given by the function handle **fun** to the data **y** (in the least-squares sense). **fun** must accept two input variables, **p** and **x**, in that order, and return a vector of the same size as **y**.