Final exam – Formulas
STAT 873

Introduction to R

- None

Matrix algebra

- Matrix multiplication:

$$\mathbf{AB} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} & a_{11}b_{12} + a_{12}b_{22} + a_{13}b_{32} \\ a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31} & a_{21}b_{12} + a_{22}b_{22} + a_{23}b_{32} \end{bmatrix}$$

- Inverse: For $\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$, $\mathbf{A}^{-1} = \frac{1}{a_{11}a_{22} - a_{12}a_{21}} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}$
- Trace: $\text{tr}(\mathbf{A}) = \sum_{i=1}^{p} a_{ii} = a_{11} + a_{22} + \ldots + a_{pp}$
- Determinant of 2×2: $\left| \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \right| = a_{11}a_{22} - a_{12}a_{21}$
- Eigenvalues: Roots of the polynomial equation $|\mathbf{A} - \lambda \mathbf{I}| = 0$ where **I** is an identity matrix
- Eigenvectors: Each eigenvalue of **A** has a corresponding nonzero vector **b** that satisfies $\mathbf{Ab} = \lambda \mathbf{b}$
- For eigenvalues λ_i of **A**: $\text{tr}(\mathbf{A}) = \sum_{i=1}^{p} \lambda_i$ and $|\mathbf{A}| = \prod_{i=1}^{p} \lambda_i = \lambda_1 \lambda_2 \cdots \lambda_p$
- Quadratic formula: The roots of the equation $ax^2 + bx + c = 0$ are $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- Vector length: $\sqrt{\sum_{i=1}^{p} a_i^2}$
- Positive definite matrices have all eigenvalues greater than 0 and positive semidefinite matrices are the same but with at least one eigenvalue equal to 0

Data, distributions, and correlation

- $\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} = \frac{\text{Cov}(x_i, x_j)}{\sqrt{\text{Var}(x_i)\text{Var}(x_j)}}$
- $\mu = E(\mathbf{x}) = \begin{bmatrix} E(x_1) \\ \vdots \\ E(x_p) \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_p \end{bmatrix}$
- $\Sigma = \text{Cov}(\mathbf{x}) = E[(\mathbf{x}-\mu)(\mathbf{x}-\mu)'] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \cdots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \cdots & \sigma_{pp} \end{bmatrix}$
- $\Sigma = E(\mathbf{xx}') - \mu\mu'$

- $\mathbf{P} = \text{Corr}(\mathbf{x}) = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1p} \\ \rho_{21} & 1 & \cdots & \rho_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{p1} & \rho_{p2} & \cdots & 1 \end{bmatrix}$
- Multivariate normal distribution, $\mathbf{x} \sim N_p(\mu, \Sigma)$: $f(\mathbf{x} \mid \mu, \Sigma) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} e^{-\frac{1}{2}\left[(\mathbf{x}-\mu)'\Sigma^{-1}(\mathbf{x}-\mu)\right]}$ for $-\infty < x_i < \infty$ for $i = 1, \ldots, p$ and $|\Sigma| > 0$
- $\hat{\mu} = \frac{1}{N}\sum_{r=1}^{N} \mathbf{x}_r = \frac{1}{N}\left(\mathbf{x}_1 + \mathbf{x}_2 + \ldots + \mathbf{x}_N\right)$
- $\hat{\Sigma} = \frac{1}{N-1}\sum_{r=1}^{N} (\mathbf{x}_r - \hat{\mu})(\mathbf{x}_r - \hat{\mu})'$
- $\hat{\sigma}_{ij} = \widehat{\text{Cov}}(x_i, x_j) = \frac{1}{N-1}\sum_{r=1}^{N}(x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)$
- $r_{ij} = \frac{\hat{\sigma}_{ij}}{\sqrt{\hat{\sigma}_{ii}\hat{\sigma}_{jj}}} = \frac{\frac{1}{N-1}\sum_{r=1}^{N}(x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)}{\sqrt{\left[\frac{1}{N-1}\sum_{r=1}^{N}(x_{ri} - \bar{x}_i)^2\right]\left[\frac{1}{N-1}\sum_{r=1}^{N}(x_{rj} - \bar{x}_j)^2\right]}} = \frac{\sum_{r=1}^{N}(x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)}{\sqrt{\left[\sum_{r=1}^{N}(x_{ri} - \bar{x}_i)^2\right]\left[\sum_{r=1}^{N}(x_{rj} - \bar{x}_j)^2\right]}}$
- $\mathbf{R} = \begin{bmatrix} 1 & r_{12} & \cdots & r_{1p} \\ r_{21} & 1 & \cdots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \cdots & 1 \end{bmatrix}$
- $z_{rj} = \frac{x_{rj} - \hat{\mu}_j}{\sqrt{\hat{\sigma}_{jj}}}$

Graphics

- None

PCA

- $y_j = \mathbf{a}_j'(\mathbf{x} - \mu)$ for j = 1, …, p
- Total variance: $\text{tr}(\Sigma) = \sum_{i=1}^{p}\sigma_{ii} = \sigma_{11} + \sigma_{22} + \ldots + \sigma_{pp}$
- $\hat{y}_j = \hat{\mathbf{a}}_j'(\mathbf{x} - \hat{\mu})$ for j = 1, …, p
- $\hat{y}_{rj}^* = \hat{\mathbf{a}}_j^{*\prime}\mathbf{z}_r$ and $\hat{y}_{rj} = \hat{\mathbf{a}}_j'(\mathbf{x}_r - \hat{\mu})$ for j = 1, …, p and r = 1, …, N
- $\hat{\mathbf{c}}_j^* = \hat{\lambda}_j^{*1/2}\hat{\mathbf{a}}_j^*$
- $r_{z_i, y_j} = \hat{a}_{ij}^*\sqrt{\hat{\lambda}_j}$
- Orthogonal matrix: Individual columns within a matrix are orthogonal to each other
- $\underset{N\times p}{\tilde{\mathbf{X}}} = \underset{N\times p}{\mathbf{U}}\,\underset{p\times p}{\hat{\Lambda}}\,\underset{p\times p}{\hat{\mathbf{A}}'}$ where $\tilde{\mathbf{X}}$ is a matrix with mean adjusted values for each variable, $\hat{\mathbf{A}}$ is a matrix with all of the eigenvectors, $\hat{\Lambda} = \text{Diag}(\hat{\lambda}_1, \ldots, \hat{\lambda}_p)$, $\mathbf{U}$ is an orthogonal matrix, and each row of $\mathbf{U}\hat{\Lambda}$ provides the PC scores

- $x_{rj} - \hat{\mu}_j \approx \hat{y}_{r1}\hat{a}_{j1} + \hat{y}_{r2}\hat{a}_{j2}$

FA

- $x_j = \mu_j + \lambda_{j1}f_1 + \lambda_{j2}f_2 + \ldots + \lambda_{jm}f_m + \eta_j$ for $j = 1, \ldots, p$
- $\tilde{x}_j = \lambda_{j1}f_1 + \lambda_{j2}f_2 + \ldots + \lambda_{jm}f_m + \eta_j$ for $j = 1, \ldots, p$; $\underset{p\times1}{\tilde{\mathbf{x}}} = \underset{p\times m}{\Lambda}\,\underset{m\times1}{\mathbf{f}} + \underset{p\times1}{\eta}$
- $z_j = \lambda_{j1}f_1 + \lambda_{j2}f_2 + \ldots + \lambda_{jm}f_m + \eta_j$ for $j = 1, \ldots, p$; $\underset{p\times1}{\mathbf{z}} = \underset{p\times m}{\Lambda}\,\underset{m\times1}{\mathbf{f}} + \underset{p\times1}{\eta}$
- $Var(ay_1+by_2) = a^2Var(y_1) + b^2Var(y_2) + 2abCov(y_1,y_2)$
- $\Sigma = \Lambda\Lambda' + \psi$; $Var(x_j) = \sum_{k=1}^{m}\lambda_{jk}^2 + \psi_j$ and $Cov(x_j, x_{j'}) = \sum_{k=1}^{m}\lambda_{jk}\lambda_{j'k}$
- With standardized variables, $\mathbf{P} = \Lambda\Lambda' + \psi$, $\sum_{k=1}^{m}\lambda_{jk}^2 + \psi_j = 1$, and $Corr(z_j, f_k) = \lambda_{jk}$
- LRT: $A = (N-1-(2p+4m+5)/6)\log\left(\dfrac{|\hat{\Lambda}\hat{\Lambda}' + \hat{\Psi}|}{|[(N-1)/N]\hat{\Sigma}|}\right)$ can be approximated by $\chi^2_{[(p-m)^2-p-m]/2}$
- AIC: $-2\log(L(\tilde{\mathbf{x}}\,|\,\hat{\Lambda},\hat{\Psi}))$ + 2(degrees of freedom for model)
- $\underset{p\times m}{\mathbf{B}} = \underset{p\times m}{\Lambda}\,\underset{m\times m}{\mathbf{T}}$
- $V = \dfrac{1}{p^2}\sum_{q=1}^{m}\left(p\sum_{j=1}^{p}\dfrac{b_{jq}^4}{h_j^4} - \left(\sum_{j=1}^{p}\dfrac{b_{jq}^2}{h_j^2}\right)^2\right)$ where $h_j^2 = \sum_{k=1}^{m}\lambda_{jk}^2$
- Bartlett's method: $\hat{\mathbf{f}}_r = \left(\hat{\Lambda}'\hat{\Psi}^{-1}\hat{\Lambda}\right)^{-1}\hat{\Lambda}'\hat{\Psi}^{-1}\mathbf{z}_r$
- Thompson's method: $\hat{\mathbf{f}}_r = \hat{\Lambda}'(\hat{\Lambda}\hat{\Lambda}' + \hat{\Psi})^{-1}\mathbf{z}_r$

CA

- $d_{rs} = \left[(\mathbf{x}_r - \mathbf{x}_s)'(\mathbf{x}_r - \mathbf{x}_s)\right]^{1/2}$
- $d_{rs} = \left[(\mathbf{z}_r - \mathbf{z}_s)'(\mathbf{z}_r - \mathbf{z}_s)\right]^{1/2}$
- $d_{ab} = \dfrac{1}{\dfrac{1}{n_a} + \dfrac{1}{n_b}}(\bar{\mathbf{x}}_a - \bar{\mathbf{x}}_b)'(\bar{\mathbf{x}}_a - \bar{\mathbf{x}}_b)$
- $\sum_{k=1}^{K}\sum_{r=1}^{N}\sum_{i=1}^{p}(x_{rik} - \bar{x}_{ik})^2$
- $W = \dfrac{\text{Within sum of squares}}{\text{Total sum of squares}}$

DA

- Choose Π_1 if $L(\mu_1, \Sigma_1|\,\mathbf{x}) > L(\mu_2, \Sigma_2|\,\mathbf{x})$ and choose Π_2 otherwise
- Suppose $\Sigma_1 = \Sigma_2$. Choose Π_1 if $\mathbf{b}'\mathbf{x} - k > 0$ and choose Π_2 otherwise, where $\mathbf{b} = \Sigma^{-1}(\mu_1-\mu_2)$ and $k = (1/2)(\mu_1-\mu_2)'\Sigma^{-1}(\mu_1+\mu_2)$.
- $d_i = (\mathbf{x} - \mu_i)'\Sigma^{-1}(\mathbf{x} - \mu_i)$

- $P(\Pi_i \mid \mathbf{x}) = \dfrac{e^{-\frac{1}{2}d_i}}{e^{-\frac{1}{2}d_1} + e^{-\frac{1}{2}d_2}}$
- $\hat{\Sigma} = \dfrac{(N_1 - 1)\hat{\Sigma}_1 + (N_2 - 1)\hat{\Sigma}_2}{N_1 + N_2 - 2}$
- $p_1 * C(2|1) * P(2|1) + p_2 * C(1|2) * P(1|2)$
- $p_1^* = \dfrac{p_1 C(2|1)}{p_1 C(2|1) + p_2 C(1|2)}$, $p_2^* = \dfrac{p_2 C(1|2)}{p_1 C(2|1) + p_2 C(1|2)}$
- $d_i^{**} = ½(\mathbf{x} - \mu_i)' \Sigma_i^{-1} (\mathbf{x} - \mu_i) + ½\log(|\Sigma_i|) - \log[p_i * C(j|i)]$
- $Y_{ij} = \mu + \alpha_i + \beta_i x_{ij} + \varepsilon_{ij}$

NNC

- None

Logistic regression

- $\pi = \dfrac{e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}}{1 + e^{\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p}}$
- $\text{logit}(\pi) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$
- Sensitivity: $\hat{P}(\text{classify positive} \mid \text{actual positive})$
- Specificity: $\hat{P}(\text{classify negative} \mid \text{actual negative})$
- PPV: $\hat{P}(\text{actual positive} \mid \text{classify positive})$
- NPV: $\hat{P}(\text{actual negative} \mid \text{classify negative})$

Multinomial regression

- $\dfrac{n!}{\prod_{j=1}^{J} n_j!} \prod_{j=1}^{J} \pi_j^{n_j}$
- $\prod_{r=1}^{N} \dfrac{n_r!}{\prod_{j=1}^{J} n_{rj}!} \prod_{j=1}^{J} \pi_j^{n_{rj}}$
- $\log(\pi_j/\pi_1) = \beta_{j0} + \beta_{j1}x_1 + \ldots + \beta_{jp}x_p$ for $j = 2, \ldots, J$
- $\pi_1 = \dfrac{1}{1 + \sum_{j=2}^{J} e^{\beta_{j0} + \beta_{j1}x_1 + \cdots + \beta_{jp}x_p}}$, $\pi_j = \dfrac{e^{\beta_{j0} + \beta_{j1}x_1 + \cdots + \beta_{jp}x_p}}{1 + \sum_{j=2}^{J} e^{\beta_{j0} + \beta_{j1}x_1 + \cdots + \beta_{jp}x_p}}$ for $j = 2, \ldots, J$
- $\text{logit}[P(Y \le j)] = \log\left[\dfrac{P(Y \le j)}{1 - P(Y \le j)}\right] = \beta_{j0} + \beta_1 x_1 + \cdots + \beta_p x_p$
- $\pi_1 = e^{\beta_{10} + \beta_1 x_1 + \cdots + \beta_p x_p} / (1 + e^{\beta_{10} + \beta_1 x_1 + \cdots + \beta_p x_p})$, $\pi_J = 1 - e^{\beta_{J-1,0} + \beta_1 x_1 + \cdots + \beta_p x_p} / (1 + e^{\beta_{J-1,0} + \beta_1 x_1 + \cdots + \beta_p x_p})$, and

  $\pi_j = \dfrac{e^{\beta_{j0} + \beta_1 x_1 + \cdots + \beta_p x_p}}{1 + e^{\beta_{j0} + \beta_1 x_1 + \cdots + \beta_p x_p}} - \dfrac{e^{\beta_{j-1,0} + \beta_1 x_1 + \cdots + \beta_p x_p}}{1 + e^{\beta_{j-1,0} + \beta_1 x_1 + \cdots + \beta_p x_p}}$ for $j = 2, \ldots, J - 1$

R functions – These functions are listed mostly in the order they were introduced in the notes

Introduction to R

Function	Description
`pnorm()`	Finds a cumulative probability from a univariate normal distribution
`qnorm()`	Finds a quantile from a univariate normal distribution
`ls()` and `objects()`	List items in R's database
`c()`	Combine items into a vector
`sd()`	Calculate a standard deviation
`var()`	Calculate a variance
`sqrt()`	Calculate a square root
`pt()`	Finds a cumulative probability from a univariate t distribution
`qt()`	Finds a quantile from a univariate t distribution
`t.test()`	Calculates quantities associated with a t-test

`read.table(file =`
`"c:\\chris\\datafile.txt",`
`header = TRUE, sep = "")`
	Read in a text data file with variable names in the first row and spaces separating the variable names and their values. Comma delimited data files can be read in using the `sep=","` option. If the first row does not contain the variable names, use `header = FALSE` and `col.names = c("var1", …, "var.c")` to name the variables yourself. The `read.csv()` function provides a shortcut way to read in comma delimited data files.

`summary()`	Summarize information in a data frame or list
`head()`	Print the first few rows of a data frame

`write.table(x = set1, file =`
`"C:\\out_file.csv", quote =`
`FALSE, row.names =`
`FALSE, sep=",")`
	Save data in a data frame to a file on the hard drive. The data was in the data frame `set1` and it will be written as a comma delimited file named `out_file.csv`.

`library(RODBC)`
`z<-odbcConnectExcel("C:\\gpa.xls")`
`gpa.excel<-sqlFetch(z, "sheet1")`
`close(z)`
	Read in data from an Excel file called `gpa.xls`. The data is located on sheet1 of the Excel file.

`plot(x = x, y = y)`	Plots y on the y-axis and x on the x-axis
`lm(formula = y ~ x, data = set1)`	Find the sample regression model (and various other measures) with the response variable y and predictor variable x within `set1`
`names()`	Provide the names of items in a list
`class()`	State the class of an object
`win.graph(width = 6, height = 6,` `pointsize = 10)`	Opens a new graphics window that is 6"×6" with font size of 10
`segments()`	Draw a line segment on a plot
`curve()`	Plot a function of x, like $f(x) = x^2$
`expression()`	Can be used to put Greek letters and mathematical symbols on a plot

Function	Description
`axis()`	Allows for finer control of an x or y-axis in a plot
`methods()`	List the method or generic functions

Matrix algebra

Function	Description
`matrix(data = c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3, byrow = TRUE)`	Create a matrix of size 2×3 by row
`t()`	Transpose a matrix
`A+B`	Matrix addition for matrices **A** and **B**
`A%*%B`	Matrix multiplication for **A** and **B**
`A*B`	Elementwise multiplication for **A** and **B**
`cbind()`	Combine elements by column
`solve(A)`	Find the inverse of **A**
`diag(A)`	Extract the diagonal elements of **A**
`sum(A)`	Sum the elements of **A**
`det(A)`	Determinant of **A**
`eigen(A)`	Find the eigenvalues and eigenvectors of **A**
`abline(h = y)`	Plots a horizontal line at `y`. A vertical line is plotted with the argument `v`.
`arrows()`	Draw an arrow on a plot

Data, distributions, and correlation

Function	Description
`cov2cor()`	Calculate a correlation matrix from a covariance matrix
`dmvnorm()`	f(**x**) for a multivariate normal distribution; this is in the mvtnorm package
`seq()`	Create a sequence of numbers
`persp3d()`	3D surface plot; this function is in the rgl package
`contour()`	Contour plot
`cov()`	Calculate estimated covariance matrix
`cor()`	Calculate estimated correlation matrix
`colMeans()`	Find the means of each column in a matrix
`apply()`	Apply a function to every row or column of a matrix
`set.seed()`	Set a seed number
`rmvnorm()`	Simulate random vectors from a multivariate normal distribution; this function is in the mvtnorm package
`points()`	Add points to a plot
`scale()`	Standardize columns of data
`expand.grid()`	Create all possible combinations of items within separate vectors

`par()`	Graphics parameters; `pty = "s"` creates a square plot, `mfrow = c(2,2)` creates a 2×2 matrix of plots

Graphics

Function	Description
`pairs()`	Side-by-side scatter plots
`scatterplotMatrix()`	Side-by-side scatter plots
`symbols()`	Bubble plot; `circles` argument specifies the third variable; `inches` argument controls the maximum size of the bubble
`identify()`	Interactively identifies points on a plot
`text()`	Puts text on a plot
`plot3d()`	3D scatter plot; this function is within the rgl package
`stars()`	Star plot
`parcoord()`	Parallel coordinate plot; this function is within the MASS package
`ipcp()`	This function is within the iplots package
`reshape()`	Changes a data frame from a wide to long format and vice versa
`histogram()`	Trellis histogram; this function is within the lattice package
`xyplot()`	Trellis scatter plot; this function is within the lattice package
`cloud()`	Trellis 3D scatter plot; this function is within the lattice package
`equal.count()`	Creates shingles for a trellis plot

PCA

Function	Description
`princomp()`	Performs PCA; `cor` argument specifies whether to use the covariance (FALSE) or correlation (TRUE) matrix
`summary()`	This function can be used to summarize the information with an object created by `princomp()`; the argument values of `loadings = TRUE` and `cutoff = 0.0` will lead to the printing of all eigenvectors
`predict()`	Computes PC scores when using an object created by `princomp()`; see my programs for how to calculate the scores correctly

FA

Function	Description
`factanal()`	Performs FA; the `rotation = "varimax"` argument specifies the varimax rotation method; the `scores` argument can be used to specify the type of scores (`"regression"` or `"Bartlett"`) to be calculated
`print()`	This function can be used to summarize the information with an object created by `factanal()`; the argument value of `cutoff = 0.0` will lead to the printing of all common factor loadings

CA

Function	Description
`dist()`	Calculates distances between observation pairs
`hclust()`	Performs agglomerative clustering
`plot()`	This function can be used with an object created by `hclust()` to create a hierarchical tree diagram
`palette()`	Provides a listing of eight colors corresponding to the numbers 1, 2, …, 8.
`cutree()`	Gives the cluster memberships when used with an object created by `hclust()`; the `k` argument specifies the number of clusters
`rect.hclust()`	Puts rectangles on a hierarchical tree diagram corresponding to clusters when used with an object created by `hclust()`.
`agnes()`	Performs agglomerative clustering
`kmeans()`	Performs K-means clustering
`aggregate()`	Applies a desired function to groups of observations within a data frame

DA

Function	Description
`lda()`	Linear discriminant analysis; use the `cv = TRUE` argument for cross-validation; this function is in the MASS package
`predict()`	The corresponding method function calculates posterior probabilities for resubstitution
`summarize.class()`	Calculates the accuracy of the classification methods; this function is written by your instructor, and it is available in Placekick_DA.r
`qda()`	Quadratic discriminant analysis; use the `cv = TRUE` argument for cross-validation; this function is in the MASS package

`rbind()`	Combine two or more data frames or matrices by rows
`runif()`	Simulate observations from a uniform distribution
`floor()`	Remove the values after the decimal place (the mathematical floor function)
`rank()`	Find the ordering of a set of values

NNC

Function	**Description**
`knn()`	Nearest neighbor classification using resubstitution; this function is in the class package
`knn.cv()`	Nearest neighbor classification using cross-validation; this function is in the class package

Logistic regression

Function	**Description**
`glm()`	Estimate a logistic regression model when `family = binomial(link = logit)` is given as an argument
`Anova()`	Perform LRTs; this function is in the car package
`predict()`	The corresponding method function estimates π when `type = "response"` is given as an argument
`cv()`	Calculates the cross-validation estimates of π; this function is written by your instructor, and it is available in Placekick_logisticreg.r
`prediction()`	Calculates the sensitivity and specificity for a number of cut-off probabilities; this function is in the ROCR package
`performance()`	Calculates the x and y-axis items for an ROC curve when the optional `"sens"` and `"fpr"` argument values are given; this function can also be used to calculate the area under the ROC curve by giving an optional "`auc`" argument value; this function is in the ROCR package
`plot()`	The corresponding method function plots the ROC curve; the `print.cutoffs.at` argument specifies the specific cut-off probabilities to include on a plot; this function is in the ROCR package
`slotNames()`	View components of an S4 object

Multinomial regression

Function	**Description**
`multinom()`	Estimates a multinomial regression model; this function is in the nnet package

`predict()`	The corresponding method function estimates π_j when `type = "probs"` is given as an argument, and it gives the classifications when `type = "class"` is given as an argument
`cv2()`	Calculates the cross-validation estimates of π_j and the corresponding classifications; this function is written by your instructor, and it is available in wheat_mult_reg.r