

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: $\text{tr}(\mathbf{A}) = \sum_{i=1}^p a_{ii} = a_{11} + a_{22} + \dots + a_{pp}$

- Determinant of 2×2 : $\begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}$

- Eigenvalues: Roots of the polynomial equation $|\mathbf{A} - \lambda\mathbf{I}| = 0$ where \mathbf{I} is an identity matrix
- Eigenvectors: Each eigenvalue of \mathbf{A} has a corresponding nonzero vector \mathbf{b} that satisfies $\mathbf{Ab} = \lambda\mathbf{b}$
- For eigenvalues λ_i of \mathbf{A} : $\text{tr}(\mathbf{A}) = \sum_{i=1}^p \lambda_i$ and $|\mathbf{A}| = \prod_{i=1}^p \lambda_i = \lambda_1\lambda_2 \dots \lambda_p$

- 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

- $\rho_{ij} = \frac{\sigma_{ij}}{\sqrt{\sigma_{ii}\sigma_{jj}}} = \frac{\text{Cov}(x_i, x_j)}{\sqrt{\text{Var}(x_i)\text{Var}(x_j)}}$

- $\boldsymbol{\mu} = E(\mathbf{x}) = \begin{bmatrix} E(x_1) \\ \vdots \\ E(x_p) \end{bmatrix} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_p \end{bmatrix}$

- $\boldsymbol{\Sigma} = \text{Cov}(\mathbf{x}) = E[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})'] = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \dots & \sigma_{1p} \\ \sigma_{21} & \sigma_{22} & \dots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \dots & \sigma_{pp} \end{bmatrix}$

- $\boldsymbol{\Sigma} = E(\mathbf{xx}') - \boldsymbol{\mu}\boldsymbol{\mu}'$

- $\mathbf{P} = \text{Corr}(\mathbf{x}) = \begin{bmatrix} 1 & \rho_{12} & \dots & \rho_{1p} \\ \rho_{21} & 1 & \dots & \rho_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{p1} & \rho_{p2} & \dots & 1 \end{bmatrix}$

- Multivariate normal distribution, $\mathbf{x} \sim N_p(\boldsymbol{\mu}, \boldsymbol{\Sigma})$: $f(\mathbf{x} | \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{p/2} |\boldsymbol{\Sigma}|^{1/2}} e^{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})' \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})}$ for $-\infty < x_i < \infty$ for

$i=1, \dots, p$ and $|\boldsymbol{\Sigma}| > 0$

- $\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_{r=1}^N \mathbf{x}_r = \frac{1}{N} (\mathbf{x}_1 + \mathbf{x}_2 + \dots + \mathbf{x}_N)$

- $\hat{\boldsymbol{\Sigma}} = \frac{1}{N-1} \sum_{r=1}^N (\mathbf{x}_r - \hat{\boldsymbol{\mu}})(\mathbf{x}_r - \hat{\boldsymbol{\mu}})'$

- $\hat{\sigma}_{ij} = \widehat{\text{Cov}}(x_i, x_j) = \frac{1}{N-1} \sum_{r=1}^N (x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)$

- $r_{ij} = \frac{\hat{\sigma}_{ij}}{\sqrt{\hat{\sigma}_{ii}\hat{\sigma}_{jj}}} = \frac{\frac{1}{N-1} \sum_{r=1}^N (x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)}{\sqrt{\left[\frac{1}{N-1} \sum_{r=1}^N (x_{ri} - \bar{x}_i)^2 \right] \left[\frac{1}{N-1} \sum_{r=1}^N (x_{rj} - \bar{x}_j)^2 \right]}} = \frac{\sum_{r=1}^N (x_{ri} - \bar{x}_i)(x_{rj} - \bar{x}_j)}{\sqrt{\left[\sum_{r=1}^N (x_{ri} - \bar{x}_i)^2 \right] \left[\sum_{r=1}^N (x_{rj} - \bar{x}_j)^2 \right]}}$

- $\mathbf{R} = \begin{bmatrix} 1 & r_{12} & \dots & r_{1p} \\ r_{21} & 1 & \dots & r_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ r_{p1} & r_{p2} & \dots & 1 \end{bmatrix}$

- $z_{rj} = \frac{x_{rj} - \hat{\mu}_j}{\sqrt{\hat{\sigma}_{jj}}}$

Graphics

- None

PCA

- $y_j = \mathbf{a}_j'(\mathbf{x} - \boldsymbol{\mu})$ for $j = 1, \dots, p$

- Total variance: $\text{tr}(\boldsymbol{\Sigma}) = \sum_{i=1}^p \sigma_{ii} = \sigma_{11} + \sigma_{22} + \dots + \sigma_{pp}$

- $\hat{y}_j = \hat{\mathbf{a}}_j'(\mathbf{x} - \hat{\boldsymbol{\mu}})$ for $j = 1, \dots, p$

- $\hat{y}_{rj}^* = \hat{\mathbf{a}}_j' \mathbf{z}_r$ and $\hat{y}_{rj} = \hat{\mathbf{a}}_j'(\mathbf{x}_r - \hat{\boldsymbol{\mu}})$ for $j = 1, \dots, p$ and $r = 1, \dots, N$

- $\hat{\mathbf{c}}_j^* = \hat{\lambda}_j^{1/2} \hat{\mathbf{a}}_j^*$

- $r_{z_i, y_j} = \hat{\mathbf{a}}_j^* \sqrt{\hat{\lambda}_j}$

- Orthogonal matrix: Individual columns within a matrix are orthogonal to each other

- $\tilde{\mathbf{X}} = \mathbf{U} \hat{\boldsymbol{\Lambda}} \hat{\mathbf{A}}'$ 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, $\hat{\boldsymbol{\Lambda}} = \text{Diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_p)$, \mathbf{U} is an orthogonal matrix, and each row of $\mathbf{U}\hat{\boldsymbol{\Lambda}}$ provides the PC scores

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

FA

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

CA

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

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 $\mathbf{b} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$ and $k = (1/2)(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)' \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2)$.
- $d_i = (\mathbf{x} - \boldsymbol{\mu}_i)' \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}_i)$

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

NNC

- None

Logistic regression

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

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

Introduction to R

Function	Description
<code>pnorm()</code>	Finds a cumulative probability from a univariate normal distribution
<code>qnorm()</code>	Finds a quantile from a univariate normal distribution
<code>ls()</code> and <code>objects()</code>	List items in R's database
<code>c()</code>	Combine items into a vector
<code>sd()</code>	Calculate a standard deviation
<code>var()</code>	Calculate a variance
<code>sqrt()</code>	Calculate a square root
<code>pt()</code>	Finds a cumulative probability from a univariate t distribution
<code>qt()</code>	Finds a quantile from a univariate t distribution
<code>t.test()</code>	Calculates quantities associated with a t-test
<code>read.table(file = "c:\\chris\\datafile.txt", header = TRUE, sep = ",")</code>	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 <code>sep=", "</code> option. If the first row does not contain the variable names, use <code>header = FALSE</code> and <code>col.names = c("var1", ..., "var.c")</code> to name the variables yourself. The <code>read.csv()</code> function provides a shortcut way to read in comma delimited data files.
<code>summary()</code>	Summarize information in a data frame or list
<code>head()</code>	Print the first few rows of a data frame
<code>write.table(x = set1, file = "C:\\out_file.csv", quote = FALSE, row.names = FALSE, sep=",")</code>	Save data in a data frame to a file on the hard drive. The data was in the data frame <code>set1</code> and it will be written as a comma delimited file named <code>out_file.csv</code> .
<code>library(RODBC)</code> <code>z<-odbcConnectExcel("C:\\gpa.xls")</code> <code>gpa.excel<-sqlFetch(z, "sheet1")</code> <code>close(z)</code>	Read in data from an Excel file called <code>gpa.xls</code> . The data is located on sheet1 of the Excel file.
<code>plot(x = x, y = y)</code>	Plots <code>y</code> on the y-axis and <code>x</code> on the x-axis
<code>lm(formula = y ~ x, data = set1)</code>	Find the sample regression model (and various other measures) with the response variable <code>y</code> and predictor variable <code>x</code> within <code>set1</code>
<code>names()</code>	Provide the names of items in a list
<code>class()</code>	State the class of an object
<code>win.graph(width = 6, height = 6, pointsize = 10)</code>	Opens a new graphics window that is 6"x6" with font size of 10
<code>segments()</code>	Draw a line segment on a plot
<code>curve()</code>	Plot a function of <code>x</code> , like $f(x) = x^2$
<code>expression()</code>	Can be used to put Greek letters and mathematical symbols on a plot
<code>axis()</code>	Allows for finer control of an x or y-axis in a plot

Function	Description
<code>methods()</code>	List the method or generic functions

Matrix algebra

Function	Description
<code>matrix(data = c(1, 2, 3, 4, 5, 6), nrow = 2, ncol = 3, byrow = TRUE)</code>	Create a matrix of size 2x3 by row
<code>t()</code>	Transpose a matrix
<code>A+B</code>	Matrix addition for matrices A and B
<code>A%*%B</code>	Matrix multiplication for A and B
<code>A*B</code>	Elementwise multiplication for A and B
<code>cbind()</code>	Combine elements by column
<code>solve(A)</code>	Find the inverse of A
<code>diag(A)</code>	Extract the diagonal elements of A
<code>sum(A)</code>	Sum the elements of A
<code>det(A)</code>	Determinant of A
<code>eigen(A)</code>	Find the eigenvalues and eigenvectors of A
<code>abline(h = y)</code>	Plots a horizontal line at <code>y</code> . A vertical line is plotted with the argument <code>v</code> .
<code>arrows()</code>	Draw an arrow on a plot

Data, distributions, and correlation

Function	Description
<code>cov2cor()</code>	Calculate a correlation matrix from a covariance matrix
<code>dmvnorm()</code>	$f(\mathbf{x})$ for a multivariate normal distribution; this is in the <code>mvtnorm</code> package
<code>seq()</code>	Create a sequence of numbers
<code>persp3d()</code>	3D surface plot; this function is in the <code>rgl</code> package
<code>contour()</code>	Contour plot
<code>cov()</code>	Calculate estimated covariance matrix
<code>cor()</code>	Calculate estimated correlation matrix
<code>colMeans()</code>	Find the means of each column in a matrix
<code>apply()</code>	Apply a function to every row or column of a matrix
<code>set.seed()</code>	Set a seed number
<code>rmvnorm()</code>	Simulate random vectors from a multivariate normal distribution; this function is in the <code>mvtnorm</code> package
<code>points()</code>	Add points to a plot
<code>scale()</code>	Standardize columns of data
<code>expand.grid()</code>	Create all possible combinations of items within separate vectors
<code>par()</code>	Graphics parameters; <code>pty = "s"</code> creates a square plot, <code>mfrow = c(2,2)</code> creates a 2x2 matrix of plots

Graphics

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

PCA

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

FA

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

CA

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

DA

Function	Description
<code>lda()</code>	Linear discriminant analysis; use the <code>cv = TRUE</code> argument for cross-validation; this function is in the <code>MASS</code> package
<code>predict()</code>	The corresponding method function calculates posterior probabilities for resubstitution
<code>summarize.class()</code>	Calculates the accuracy of the classification methods; this function is written by your instructor, and it is available in <code>Placekick_DA.r</code>
<code>qda()</code>	Quadratic discriminant analysis; use the <code>cv = TRUE</code> argument for cross-validation; this function is in the <code>MASS</code> package
<code>rbind()</code>	Combine two or more data frames or matrices by rows

<code>runif()</code>	Simulate observations from a uniform distribution
<code>floor()</code>	Remove the values after the decimal place (the mathematical floor function)
<code>rank()</code>	Find the ordering of a set of values

NNC

Function	Description
<code>knn()</code>	Nearest neighbor classification using resubstitution; this function is in the class package
<code>knn.cv()</code>	Nearest neighbor classification using cross-validation; this function is in the class package

Logistic regression

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