Final Exam – Formulas
STAT 875

Section 1.1

- Binomial distribution: P(W = w) = $\frac{n!}{w!(n-w)!}\pi^{w}(1-\pi)^{n-w}$ for w = 0,1,…,n with E(W) = nπ and Var(W) = nπ(1-π)
- $\hat{\pi}$ = w/n
- Wald confidence interval for π is $\hat{\pi} \pm Z_{1-\alpha/2}\sqrt{\frac{\hat{\pi}(1-\hat{\pi})}{n}}$
- Wilson interval for π is $\tilde{\pi} \pm \frac{Z_{1-\alpha/2}n^{1/2}}{n+Z_{1-\alpha/2}^{2}}\sqrt{\hat{\pi}(1-\hat{\pi})+\frac{Z_{1-\alpha/2}^{2}}{4n}}$ with $\tilde{\pi} = \frac{w + Z_{1-\alpha/2}^{2}/2}{n+Z_{1-\alpha/2}^{2}}$
- Agresti-Coull interval for π is $\tilde{\pi} \pm Z_{1-\alpha/2}\sqrt{\frac{\tilde{\pi}(1-\tilde{\pi})}{n+Z_{1-\alpha/2}^{2}}}$
- True confidence level is $\sum_{w=0}^{n} I(w)\binom{n}{w}\pi^{w}(1-\pi)^{n-w}$ where I(w) = 1 if the interval for a w contains π and 0 otherwise
- Clopper-Pearson confidence interval for π is Beta(α/2; w, n-w+1) < π < Beta(1-α/2; w+1, n-w)
- Score test statistic for testing $\pi = \pi_0$ is $Z_0 = \frac{\hat{\pi}-\pi_0}{\sqrt{\frac{\pi_0(1-\pi_0)}{n}}}$ where a standard normal distributional approximation is used
- General form of LRT statistic: $\Lambda = \frac{\text{Max. lik. when parameters satisfy } H_0}{\text{Max. lik. when parameters satisfy } H_0 \text{ or } H_a}$ and -2log(Λ) is approximately a χ^2 random variable for large samples under H_0 with degrees of freedom equal to the difference in dimension between the alternative and null hypothesis parameter spaces

Section 1.2

- $\hat{\pi}_j = w_j / n_j$
- Wald confidence interval for $\pi_1 - \pi_2$ is $\hat{\pi}_1 - \hat{\pi}_2 \pm Z_{1-\alpha/2}\sqrt{\frac{\hat{\pi}_1(1-\hat{\pi}_1)}{n_1}+\frac{\hat{\pi}_2(1-\hat{\pi}_2)}{n_2}}$
- Agresti-Caffo confidence interval is $\tilde{\pi}_1 - \tilde{\pi}_2 \pm Z_{1-\alpha/2}\sqrt{\frac{\tilde{\pi}_1(1-\tilde{\pi}_1)}{n_1+2}+\frac{\tilde{\pi}_2(1-\tilde{\pi}_2)}{n_2+2}}$ where $\tilde{\pi}_1 = \frac{w_1+1}{n_1+2}$ and $\tilde{\pi}_2 = \frac{w_2+1}{n_2+2}$
- Score test statistic for testing $\pi_1 - \pi_2 = 0$ is $Z_0 = \frac{\hat{\pi}_1 - \hat{\pi}_2}{\sqrt{\bar{\pi}(1-\bar{\pi})\left(\frac{1}{n_1}+\frac{1}{n_2}\right)}}$ where a standard normal distributional approximation is used and $\bar{\pi} = w_+ / n_+$

- Pearson chi-square test statistic for testing $\pi_1 - \pi_2 = 0$ is $X^2 = \sum_{j=1}^{2}\frac{(w_j - n_j\bar{\pi})^2}{n_j\bar{\pi}(1-\bar{\pi})}$ where a χ_1^2 distributional approximation is used
- Transformed LRT statistic for testing $\pi_1 - \pi_2 = 0$ is
$$-2\log(\Lambda) = -2\left[w_1\log\left(\frac{\bar{\pi}}{\hat{\pi}_1}\right)+(n_1-w_1)\log\left(\frac{1-\bar{\pi}}{1-\hat{\pi}_1}\right)+w_2\log\left(\frac{\bar{\pi}}{\hat{\pi}_2}\right)+(n_2-w_2)\log\left(\frac{1-\bar{\pi}}{1-\hat{\pi}_2}\right)\right]$$
- $RR = \frac{\pi_1}{\pi_2}$ and $\widehat{RR} = \frac{\hat{\pi}_1}{\hat{\pi}_2} = \frac{w_1 n_2}{w_2 n_1}$
- Wald confidence interval for RR is $\exp\left[\log(\hat{\pi}_1/\hat{\pi}_2) \pm Z_{1-\alpha/2}\sqrt{\widehat{Var}(\log(\hat{\pi}_1/\hat{\pi}_2))}\right]$ where $\widehat{Var}(\log(\hat{\pi}_1/\hat{\pi}_2)) = \frac{1-\hat{\pi}_1}{n_1\hat{\pi}_1}+\frac{1-\hat{\pi}_2}{n_2\hat{\pi}_2} = \frac{1}{w_1}-\frac{1}{n_1}+\frac{1}{w_2}-\frac{1}{n_2}$
- $OR = \frac{odds_1}{odds_2} = \frac{\pi_1/(1-\pi_1)}{\pi_2/(1-\pi_2)} = \frac{\pi_1(1-\pi_2)}{\pi_2(1-\pi_1)}$ and $\widehat{OR} = \frac{\widehat{odds}_1}{\widehat{odds}_2} = \frac{\hat{\pi}_1(1-\hat{\pi}_2)}{\hat{\pi}_2(1-\hat{\pi}_1)} = \frac{w_1(n_2-w_2)}{w_2(n_1-w_1)}$
- Wald confidence interval for OR is $\exp\left[\log(\widehat{OR}) \pm Z_{1-\alpha/2}\sqrt{\frac{1}{w_1}+\frac{1}{n_1-w_1}+\frac{1}{w_2}+\frac{1}{n_2-w_2}}\right]$
- $\widetilde{OR} = \frac{(w_1+0.5)(n_2-w_2+0.5)}{(w_2+0.5)(n_1-w_1+0.5)}$

Chapter 2

- Logistic regression model is $\log\left(\frac{\pi}{1-\pi}\right) = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p$
- Estimated covariance matrix form when there is only one explanatory variable:
$$-E\left(\begin{bmatrix}\frac{\partial^2 \log\left[L(\beta_0,\beta_1 \mid y_1,\ldots,y_n)\right]}{\partial\beta_0^2} & \frac{\partial^2 \log\left[L(\beta_0,\beta_1 \mid y_1,\ldots,y_n)\right]}{\partial\beta_0\partial\beta_1} \\ \frac{\partial^2 \log\left[L(\beta_0,\beta_1 \mid y_1,\ldots,y_n)\right]}{\partial\beta_0\partial\beta_1} & \frac{\partial^2 \log\left[L(\beta_0,\beta_1 \mid y_1,\ldots,y_n)\right]}{\partial\beta_1^2}\end{bmatrix}\right)^{-1}\Bigg|_{\beta_0=\hat{\beta}_0,\beta_1=\hat{\beta}_1}$$
- Wald test statistic for testing $\beta_r = 0$ is $Z_0 = \frac{\hat{\beta}_r}{\sqrt{\widehat{Var}(\hat{\beta}_r)}}$ where a standard normal approximation is used
- Transformed LRT statistic for testing a set of β's are equal to 0 is
$$-2\log(\Lambda) = -2\log\left(\frac{L(\beta^{(0)} \mid y_1,\ldots,y_n)}{L(\beta^{(a)} \mid y_1,\ldots,y_n)}\right) = -2\left[\sum_{i=1}^{n} y_i\log\left(\frac{\hat{\pi}_i^{(0)}}{\hat{\pi}_i^{(a)}}\right)+(1-y_i)\log\left(\frac{1-\hat{\pi}_i^{(0)}}{1-\hat{\pi}_i^{(a)}}\right)\right]$$ where a χ_q^2 distributional approximation is used (q is number of β's set to 0 in the null hypothesis)
- $OR = e^{c\beta_1}$ for a one explanatory variable logistic regression model and a c-unit increase in x_1; Wald confidence interval for OR is $e^{c\hat{\beta}_1 \pm cZ_{1-\alpha/2}\sqrt{\widehat{Var}(\hat{\beta}_1)}}$
- Profile likelihood interval for β_1 uses $-2\log\left(\frac{L(\tilde{\beta}_0,\beta_1 \mid y_1,\ldots,y_n)}{L(\hat{\beta}_0,\hat{\beta}_1 \mid y_1,\ldots,y_n)}\right)$

- Wald confidence interval for π is $\frac{e^{\hat\beta_0+\hat\beta_1x_1+\cdots+\hat\beta_px_p \pm Z_{1-\alpha/2}\sqrt{\widehat{Var}(\hat\beta_0+\hat\beta_1x_1+\cdots+\hat\beta_px_p)}}}{1+e^{\hat\beta_0+\hat\beta_1x_1+\cdots+\hat\beta_px_p \pm Z_{1-\alpha/2}\sqrt{\widehat{Var}(\hat\beta_0+\hat\beta_1x_1+\cdots+\hat\beta_px_p)}}}$ where
  $\widehat{Var}(\hat\beta_0+\hat\beta_1x_1+\cdots+\hat\beta_px_p)=\sum_{i=0}^{p}\widehat{Var}(\hat\beta_i)+2\sum_{i=0}^{p-1}\sum_{j=i+1}^{p}x_ix_j\widehat{Cov}(\hat\beta_i,\hat\beta_j)$
- Convergence of the parameter estimates occurs when $\frac{|G^{(i)}-G^{(i-1)}|}{0.1+|G^{(i)}|}<\epsilon$ where $G^{(i)}$ denotes the residual deviance at iteration i
- Probit regression model is probit(π) = $\beta_0 + \beta_1x_1 + \cdots \beta_px_p$
- Complementary log-log regression model is log[-log(1 – π)] = $\beta_0 + \beta_1x_1 + \cdots \beta_px_p$

<u>Chapter 3</u>

- Multinomial PMF for $n_1, \ldots, n_J$: $\frac{n!}{\prod_{j=1}^{J} n_j!}\prod_{j=1}^{J}\pi_j^{n_j}$
- P(X = i, Y = j) = π_{ij}
- Independence: $\pi_{ij} = \pi_{i+}\pi_{+j}$
- P(Y = j | X = i) = $\pi_{j|i}$
- $X^2=\sum_{i=1}^{I}\sum_{j=1}^{J}\frac{(n_{ij}-n_{i+}n_{+j}/n)^2}{n_{i+}n_{+j}/n}$; under independence, X^2 has a $\chi^2_{(I-1)(J-1)}$ distribution for a large sample
- $-2\log(\Lambda)=2\sum_{i=1}^{I}\sum_{j=1}^{J}n_{ij}\log\left(\frac{n_{ij}}{n_{i+}n_{+j}/n}\right)$; under independence, -2log(Λ) has a $\chi^2_{(I-1)(J-1)}$ distribution for a large sample
- P-value for Monte Carlo test involving X^2: $(\#\, X^{2*}\ge X^2)/B$
- Multinomial regression model: log(π_j/π_1) = $\beta_{j0} + \beta_{j1}x_1 + \ldots + \beta_{jp}x_p$ for j = 2, …, J
- For one explanatory variable: $\pi_1=\frac{1}{1+\sum_{j=2}^{J}e^{\beta_{j0}+\beta_{j1}x}}$ and $\pi_j=\frac{e^{\beta_{j0}+\beta_{j1}x}}{1+\sum_{j=2}^{J}e^{\beta_{j0}+\beta_{j1}x}}$
- Proportional odds model: $\text{logit}[P(Y\le j)]=\beta_{j0}+\beta_1x_1+\cdots+\beta_px_p$
- For one explanatory variable: $\pi_j=\frac{e^{\beta_{j0}+\beta_1x}}{1+e^{\beta_{j0}+\beta_1x}}-\frac{e^{\beta_{j-1,0}+\beta_1x}}{1+e^{\beta_{j-1,0}+\beta_1x}}$ for j = 2, …, J – 1, π_1 = $e^{\beta_{10}+\beta_1x}/(1+e^{\beta_{10}+\beta_1x})$, and π_J = $1-e^{\beta_{J-1,0}+\beta_1x}/(1+e^{\beta_{J-1,0}+\beta_1x})$
- Nonproportional odds model: $\text{logit}(P(Y\le j))=\beta_{j0}+\beta_{j1}x_1+\cdots+\beta_{jp}x_p$
- Adjacent-categories model: $\log(\pi_j/\pi_{j+1})=\beta_{j0}+\beta_{j1}x_1+\cdots+\beta_{jp}x_p$

<u>Chapter 4</u>

- Poisson PMF: $P(Y=y)=\frac{e^{-\mu}\mu^y}{y!}$ where E(Y) = μ and Var(Y) = μ
- $L(\mu;y_1\ldots,y_n)=\prod_{k=1}^{n}\frac{e^{-\mu}\mu^{y_k}}{y_k!}$
- $\hat\mu=n^{-1}\sum_{k=1}^{n}y_k$

- $\widehat{Var}(\hat\mu)=\frac{\hat\mu}{n}$
- Score test statistic for testing $\mu = \mu_0$ is $Z_0=\frac{\hat\mu-\mu_0}{\sqrt{\mu_0/n}}$ where a standard normal distributional approximation is used
- Score confidence interval for μ: $\left(\hat\mu+\frac{Z^2_{1-\alpha/2}}{2n}\right)\pm Z_{1-\alpha/2}\sqrt{\frac{\hat\mu+Z^2_{1-\alpha/2}/4n}{n}}$
- Wald interval for μ using log(μ) transformation: $e^{\log(\hat\mu)\pm Z_{1-\alpha/2}\sqrt{1/(\hat\mu n)}}$
- Poisson regression model: log(μ) = $\beta_0 + \beta_1x_1 + \cdots \beta_px_p$
- $\mu(x+c)/\mu(x)=e^{c\beta_1}$ for the model log(μ) = $\beta_0 + \beta_1x_1$
- PC = $100(e^{c\beta_1}-1)$%
- Poisson rate regression model: $\mu=te^{\beta_0+\beta_1x}$

<u>Chapter 5</u>

- IC(k) = -2log(Likelihood function evaluated at parameter estimates) + kr
- $AIC=IC(2)$
- $AIC_c=IC(2n/(n-r-1))=AIC+\frac{2r(r+1)}{n-r-1}$
- $BIC=IC(\log(n))$
- $\Delta_m = BIC_m - BIC_0$
- $\hat\tau_A\approx\frac{\exp(-0.5\Delta_A)}{\sum_{m=1}^{M}\exp(-0.5\Delta_m)}$
- Model averaged estimate: $\hat\theta_{MA}=\sum_{m=1}^{M}\hat\tau_m\hat\theta_m$
- $\widehat{Var}(\hat\theta_{MA})=\sum_{m=1}^{M}\hat\tau_m[(\hat\theta_m-\hat\theta_{MA})^2+\widehat{Var}(\hat\theta_m)]$
- Pearson residual: $e_m=\frac{y_m-\hat y_m}{\sqrt{\widehat{Var}(Y_m)}}$
- Standardized Pearson residual: $r_m=\frac{y_m-\hat y_m}{\sqrt{\widehat{Var}(Y_m-\hat Y_m)}}=\frac{y_m-\hat y_m}{\sqrt{\widehat{Var}(Y_m)(1-h_m)}}$
- Deviance residual: $e_m^D=\text{sign}(y_m-\hat y_m)\sqrt{d_m}$
- Standardized deviance residual: $r_m^D=e_m^D/\sqrt{(1-h_m)}$
- Pearson statistic: $X^2=\sum_{m=1}^{M}e_m^2=\sum_{m=1}^{M}\frac{(y_m-\hat y_m)^2}{\widehat{Var}(y_m)}$
- $D/(M-\tilde p)$
- Cook's distance: $CD_m=\frac{r_m^2h_m}{(p+1)(1-h_m)^2}$
- $\Delta X_m^2=r_m^2$
- $\Delta D_m=(e_m^D)^2+h_mr_m^2$
- Var(Y) = $\gamma\mu$

- $\frac{1}{\gamma}\sum_{m=1}^{M}(Y_m-\mu_m)x_{mr}$
- $r_m=\frac{y_m-\hat{y}_m}{\sqrt{\widehat{Var}(Y_m)(1-h_m)\gamma}}$
- $\hat{\gamma}$ = $X^2/(M-\tilde{p})$
- Negative binomial PMF: $\binom{y+k-1}{y}\left(\frac{k}{\mu+k}\right)^k\left(1-\frac{k}{\mu+k}\right)^y$ where E(Y) = μ and Var(Y) = $\mu + \mu^2/k$

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

Syntax	Description
`dbinom(x = y, size = n, prob = π)`	Finds P(W = w) for W that has a binomial distribution with parameter π and number of trials n
`plot(x = x, y = y)`	Plots y on the y-axis and x on the x-axis
`abline(h = y)`	Plots a horizontal line at y
`set.seed(number)`	Set a seed of value number
`rbinom(n = sample size, size = n, prob = π)`	Simulates a sample of size "sample size" from a binomial distribution with parameter π and number of trials n
`mean(x)`	Finds the mean of the values inside of x
`var(x)`	Finds the variance of the values inside of x
`hist(x)`	Plots a histogram of the values inside of x
`table(x)`	Finds the frequency distribution for the values inside of x
`curve(expr = function, xlim = c(lower,upper))`	Plots a function of x within the limits specified by the `xlim` argument.
`qnorm(p = 1-α/2)`	Finds 1-α/2 quantile from a standard normal distribution
`binom.confint(x = # of successes, n = n, conf.level = confidence level, methods = "all")`	Finds 11 different confidence intervals for π with a particular observed # of successes, n trials, and confidence level; note that this function is in the *binom* package
`ifelse(test = condition to test, yes = yes value, no = no value)`	Returns a vector of values corresponding to the result of the test
`qbeta(p = 1-α/2, shape1 = a, shape2 = b)`	Finds the 1-α/2 quantile from a beta distribution with parameters a and b
`array(data = c(w1, w2, n1-w1, n2-w2), dim = c(2,2), dimnames = list(group = c("1", "2"), Response = c("1", "2")))`	Create a 2×2 contingency table
`rowSums(c.table)`	Find the sums of each row in a contingency table
`head(data frame)`	Show the first 6 rows in a data frame
`table(x = row variable raw data, y = column variable raw data)`	Another use of the `table()` function – creates a contingency table from the raw data
`xtabs(formula = ~ row variable raw data + column variable raw data, data = data frame)`	Creates a contingency table from the raw data
`wald2ci(x1 = w1, n1 = n1, x2 = w2, n2 = n2, conf.level = 1-α, adjust = "Wald")`	Calculate a Wald confidence interval for $\pi_1 - \pi_2$; note that this function is in the *PropCIs* package. The Agresti-Coull interval is calculated when `adjust = "AC"`.
`prop.test(x = c.table, conf.level = 1-α, correct = FALSE)`	Calculate X^2 for a test of $\pi_1 - \pi_2 = 0$ vs. $\neq 0$ and calculate a Wald confidence interval for $\pi_1 - \pi_2$
`chisq.test(x = c.table, correct = FALSE)`	Calculate X^2 for a test of $\pi_1 - \pi_2 = 0$ vs. $\neq 0$

Syntax	Description
`qchisq(p = 1-α, df = degrees of freedom)`	Finds 1-α quantile from a chi-square distribution for a given degrees of freedom
`glm(formula = y ~ x1 + x2, data = data frame, family = binomial(link = logit))`	Estimate a logistic regression model with response variable y and explanatory variables x1 and x2. Other types of binary regression models include: 1. Probit regression model – *binomial(link = probit)* 2. Complementary log-log model – *binomial(link = cloglog)* Changing the response variable to `w/n` and adding a `weight = n` where n is the number of trials allows one to fit a binary regression model to binomial responses. This function can be used for Poisson regression, Poisson rate regression, and quasi-Poisson regression models as well.
`names(mod.fit)`	Allows one to see the elements in the list object called `mod.fit`
`summary(mod.fit)`	Summarizes information in `mod.fit`
`class(mod.fit)`	Shows the class of `mod.fit`
`methods(class = glm)`	Provides list of method functions associated with a class named `glm`
`vcov(mod.fit)`	Calculates the covariance matrix of the parameter estimates as given in `mod.fit`
`Anova(mod.fit, test = "LR")`	Calculates -2log(Λ) for explanatory variables within `mod.fit`; note that this function is in the *car* package
`anova(mod.fit, test = "Chisq")`	Calculates -2log(Λ) for explanatory variables within `mod.fit`; note that the tests performed by this function is not necessarily the same as the tests performed by `Anova()`
`confint(object = mod.fit, parm = "variable name", level = 1-α)`	Calculates a profile likelihood ratio interval for the β parameter corresponding to the variable name given in `parm` when using `glm()`
`confint.default(object = mod.fit, parm = "variable name", level = 1-α)`	Same as `confint()`, but calculates a Wald interval
`predict(object = mod.fit, newdata = data frame, type = “response”, se.fit = TRUE)`	Predicts π at the explanatory variable values in the data frame using model information in `mod.fit`. The data frame needs to have the same explanatory variables as the original data set specified in `mod.fit`. Standard errors of the prediction are also produced. The `type = “link”` argument value can be used to predict the linear predictor part of the model.
`aggregate(formula = y ~ x, data = data frame, FUN = summary function)`	Summarizes the data in a data frame using a given function and by a variable in the data frame

Syntax	Description
`symbols(x = x, y = y, circles = z, inches = max size)`	Plots y on the y-axis and x on the x-axis with the plotting point proportional in size to z; the inches argument specifies the maximize size of a plotting point
`levels(variable)`	Shows the qualitative levels of a factor
`contrasts(variable)`	Shows how R will create indicator variables for a factor
`relevel(x = variable, ref = "New level”)`	Changes the base or reference level of a factor to what is specified in `ref`
`factor(x)`	Create a factor using the information in x; this function can also be used to re-order the factor levels by using the `level` argument.
`dmultinom(x = n.j, prob = pi.j)`	Finds the probability of observing a set of n_j values with probability parameters given in `pi.j`
`rmultinom(n = 1, size = sample size, prob = pi.j)`	Simulates one sample of size “sample size” from a multinomial distribution with probability parameters given in `pi.j`
`assocstats(x = contingency table)`	Calculates Pearson and LRT statistics needed for a test of independence; note that this function is in the `vcd` package
`plot.ecdf(x = data vector, verticals = TRUE, do.p = FALSE)`	Plots an empirical CDF
`parcoord(x = data frame)`	Creates a parallel coordinate plot
`multinom(formula = y ~ x1 + x2, data = data frame)`	Estimates a multinomial regression model with response y and explanatory variables x1 and x2; the `weights` argument can be used when counts for each x1 and x2 combination are available (like for contingency tables); this function is in the `nnet` package
`polr(formula = y ~ x1 + x2, data = data frame, method = "logistic")`	Estimates a proportional odds regression model with response y and explanatory variables x1 and x2; the `weights` argument can be used when counts for each x1 and x2 combination are available (like for contingency tables); note that the function actually estimates $\text{logit}(P(Y \le j)) = \beta_{j0} - \eta_1 x_1 - \cdots - \eta_p x_p$ so adjustments need to be made to match our notation; this function is in the MASS package
`vglm(formula = y ~ x, family = cumulative(parallel = FALSE), data = data frame)`	Estimates a non-proportional odds model with response y and explanatory variable x; the `weights` argument can be used when counts for x are available (like for contingency tables; make sure there are no 0 counts); `family = cumulative(parallel = TRUE)` estimates the proportional odds model; this function is in the VGAM package
`dpois(x = y, lambda = mu)`	Find P(Y = y) where Y ~ Po(μ)

Syntax	Description
offset(log(t))	Use in a `formula` argument of `glm()` to create an offset
glm.nb(formula = y ~ x1 + x2, data = data frame, link = log)	Estimate a negative binomial regression model with response y and explanatory variables x1 and x2; this function is in the MASS package
AIC()	Calculate information criteria
glmulti(y = y ~ ., data = set1, fitfunction = "glm", level = 1, method = "h", crit = "aicc", family = binomial(link="logit"))	Use all-subsets regression with all variables in the data frame; arguments specified include the `glm()` model fit function, main effects only (`level = 2` includes pairwise interactions), exhaustive method ("g" for genetic algorithm), AIC_c information criterion, and logistic regression
weightable(<object from glmulti()>)	Summarize best models from a `glmulti()` returned object
step(object = mod.fit.empty, direction = "forward", scope = list(upper = mod.fit.full), k = 2)	Perform forward selection using the AIC
residuals(object = mod.fit, type = "pearson")	Calculates the Pearson residuals corresponding to a model fit object `mod.fit`
rstandard(model = mod.fit, type = "pearson")	Calculates the standardized Pearson residuals corresponding to a model fit object `mod.fit`
loess(formula = y ~ x, data = set1, weights = trials)	Estimates a loess regression model where each observation is weighted by the variable trials
order()	Find the numerical order of observations in a vector; for example, running `x<-c(2,3,1)` and then `x[order(x)]` results in 1, 2, 3
lines(x = , y =)	Connect points with a line
cooks.distance(model = mod.fit)	Calculates the Cook's distance values corresponding to a model fit object `mod.fit`
hatvalues(model = mod.fit)	Calculates the hat matrix diagonal values corresponding to a model fit object `mod.fit`
glmInflDiag()	Performs a number of residual and influence value calculations for a GLM
examine.logistic.reg(mod.fit.obj = mod.fit)	Performs a number of residual and influence value calculations for a binary or binomial regression model with model fit object `mod.fit`

Additional general R functions and examples

Syntax	Description
> placekick<-read.table(file = "C:\\data\\placekick.csv", header = TRUE, sep = ",")	Read in a comma delimited file "placekick.csv" where the first row contains the variable names. The `read.csv()` function can also be used.

> library(package = mcprofile) > K<-matrix(data = c(1, 20), nrow = 1, ncol = 2) > K > linear.combo<-mcprofile(object = mod.fit, CM = K) #Calculate -2log(Lambda) > ci.logit.profile<-confint(object = linear.combo, level = 0.95) #CI for beta_0 + beta_1 * x > exp(ci.logit.profile$confint)/(1 + exp(ci.logit.profile$confint))	Example of how to calculate a profile likelihood ratio interval with functions in the mcprofile package

ci.pi<-function(newdata,
mod.fit.obj, alpha){
linear.pred<-predict(object =
mod.fit.obj, newdata =
newdata, type = "link", se =
TRUE)
CI.lin.pred.lower<-linear.pred$fit
- qnorm(p = 1-alpha/2) *
linear.pred$se
CI.lin.pred.upper<-linear.pred$fit
+ qnorm(p = 1-alpha/2) *
linear.pred$se
CI.pi.lower<-
exp(CI.lin.pred.lower) / (1 +
exp(CI.lin.pred.lower))
CI.pi.upper<-
exp(CI.lin.pred.upper) / (1 +
exp(CI.lin.pred.upper))
list(lower = CI.pi.lower, upper =
CI.pi.upper)
}
	Example function for calculating a confidence interval for π

ci.mu<-function(newdata,
mod.fit.obj, alpha) {
lin.pred.hat<-predict(object =
mod.fit.obj, newdata = newdata,
type = "link", se = TRUE)
lower<-exp(lin.pred.hat$fit –
qnorm(1 - alpha/2) *
lin.pred.hat$se)
upper<-exp(lin.pred.hat$fit +
qnorm(1 - alpha/2) *
lin.pred.hat$se)
list(lower = lower, upper = upper)
}
	Example function for calculating a confidence interval for μ in a Poisson regression model