BIOS 6312: Modern Regression Analysis

Andrew J. Spieker, Ph.D.

Assistant Professor of Biostatistics
Vanderbilt University Medical Center

Set 16: Examples for R Enthusiasts!

Version: 04/17/2023

Table of Contents

Analysis of HbA1c and age

Reading in the REACH data:

- Read in data:

```
reach.data <- read.csv("reach.csv",
                       header = TRUE,
                       stringsAsFactors = FALSE)
```

Analysis of HbA1c and age

Scatterplot:

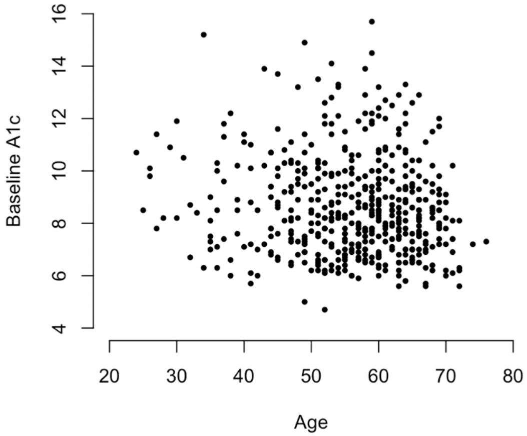
- Without a LOWESS smoother.

```
plot(reach.data$age, reach.data$a1c.0,
     xlim = c(20, 80), ylim = c(4, 16),
     xlab = "Age", ylab = "Baseline A1c",
     cex = 0.8, pch = 20,
     frame.plot = FALSE)
```

- Highly customizable.

Analysis of HbA1c and age

Scatterplot:



Analysis of HbA1c and age

Scatterplot:

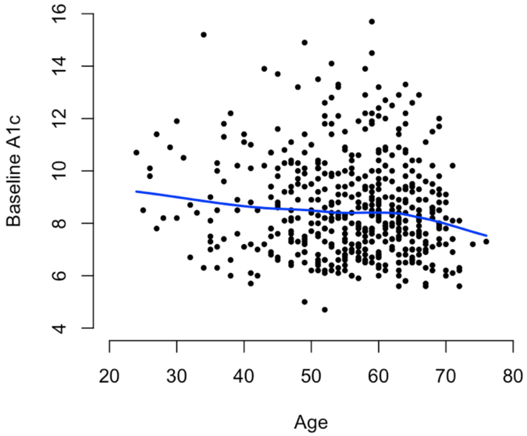
- With a LOWESS smoother.

```
scatter.smooth(reach.data$age, reach.data$a1c.0,
               xlim = c(20, 80), ylim = c(4, 16),
               xlab = "Age", ylab = "Baseline A1c",
               cex = 0.8, pch = 20,
               lpars = list(lwd = 2, col = "blue"),
               frame.plot = FALSE)
```

- Option `lpars` contains options specific to the smoothing ling.

Analysis of HbA1c and age

Scatterplot: With LOWESS smoother



Analysis of HbA1c and age

Regression fit:

- Fit regression model and print summary of results.

```
regr.a1c <- lm(a1c.0 ~ age, data = reach.data)
summary(regr.a1c)
```

Analysis of HbA1c and age

Regression fit:

- Print summary of results

```
Call:
lm(formula = a1c.0 ~ age, data = reach.data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0097 -1.4270 -0.2879  1.1100  7.1426

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  9.840478   0.490875  20.047   <2e-16 ***
age         -0.021747   0.008641  -2.517   0.0122 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.883 on 493 degrees of freedom
  (10 observations deleted due to missingness)
Multiple R-squared:  0.01268,Adjusted R-squared:  0.01068
F-statistic: 6.333 on 1 and 493 DF,  p-value: 0.01217
```

- Be warned: output is based on non-robust standard errors!

Analysis of HbA1c and age

Extracting robust standard errors:

- Need to install (and load) the `sandwich` package.

```
## Load library
library("sandwich")

## Huber-White variance
robust.var <- vcovHC(regr.a1c, type = "HC1")
> robust.var
             (Intercept)           age
(Intercept)  0.234640771 -3.986932e-03
age         -0.003986932  6.985083e-05

## Print standard errors for coefficients of interest
> sqrt(diag(robust.var))
(Intercept)         age
0.484397327 0.008357681
```

Diagnostic plots: age and HbA1c

Fitted/predicted values:

- This is a continuation of the previous example.
- Extract fitted values from regression fit:

```
fitted <- regr.a1c$fitted.values
```

Diagnostic plots: age and HbA1c

Studentized residuals:

- Studentized residuals not readily available.

```
## Residuals from regression model
resid <- regr.a1c$residuals

## Estimate error variance
sigma.hat <- sd(regr.a1c$residuals)

## Create hat matrix
dsn.X <- cbind(1, regr.a1c$model$age)
H <- dsn.X %*% solve(t(dsn.X) %*% dsn.X) %*% t(dsn.X)

## Diagonal entries (leverage)
lvg <- diag(H)

## Create studentized residuals
st.resid <- resid/(sigma.hat * sqrt(1 - lvg))
```

Diagnostic plots: age and HbA1c

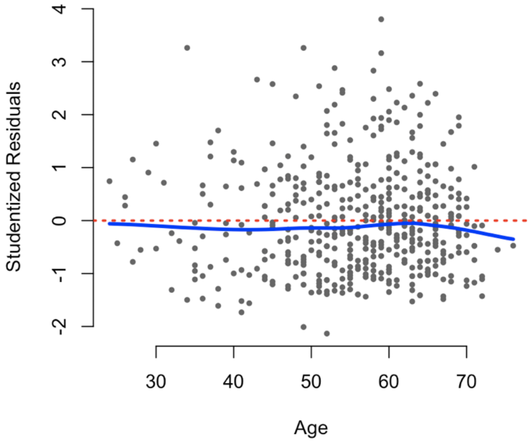
Residual-versus-predictor plot:

- Include LOWESS and an indicator of the *x*-axis.

```
scatter.smooth(regr.a1c$model$age, st.resid,
     xlab = "Age",
     ylab = "Studentized Residuals",
     cex = 0.8, pch = 20, col = "gray40",
     lpars = list(lwd = 3, col = "blue"),
     frame.plot = FALSE)
abline(0,0, lty = 3, lwd = 2, col = "red")
```

Diagnostic plots: age and HbA1c

Residual-versus-predictor plot:



Diagnostic plots: age and HbA1c

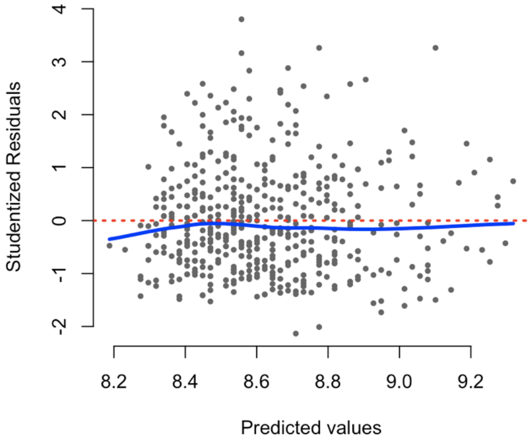
Residual-versus-fitted plot:

- Include LOWESS and an indicator of the *x*-axis.

```
scatter.smooth(fitted, st.resid,
               xlab = "Predicted values",
               ylab = "Studentized Residuals",
               cex = 0.8, pch = 20, col = "gray40",
               lpars = list(lwd = 3, col = "blue"),
               frame.plot = FALSE)
abline(0,0, lty = 3, lwd = 2, col = "red")
```

Diagnostic plots: age and HbA1c

Residual-versus-fitted plot:



Diagnostic plots: age and HbA1c

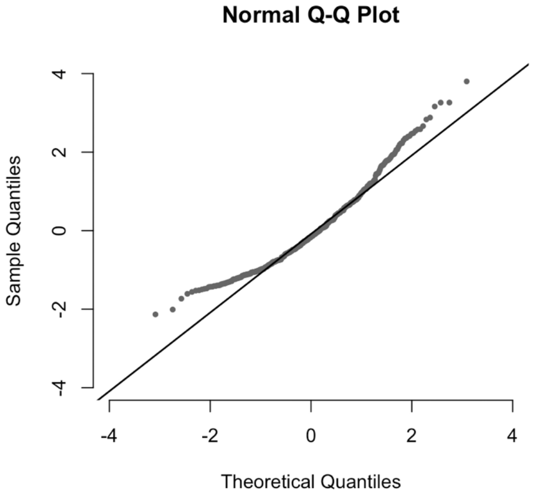
Quantile-quantile plot:

- Include reference line.

```
qqnorm(st.resid, frame = FALSE,
       cex = 0.8, pch = 20, col = "gray40",
       xlim = c(-4,4), ylim = c(-4, 4))
qqline(st.resid, lwd = 1.5)
```

Diagnostic plots: age and HbA1c

Quantile-quantile plot:



Subgroup effects in REACH

Reading in the REACH data:

- Read in data:

```
reach.data <- read.csv("reach.csv",
                       header = TRUE,
                       stringsAsFactors = FALSE)
```

Subgroup effects in REACH

Example: Subgroup effect with a continuous interaction term

- Model: $\text{E}[Y|X = x, Z = z] = \beta_0 + \beta_1 x + \beta_2 z + \beta_3 xz$.
  - X: REACH (0 = control; 1 = REACH).
  - Z: baseline A1c.
  - Y: six-month A1c.
- Goal: learn about REACH effect among subgroup with $Z = z_0$.

SUBGROUP EFFECTS IN REACH

Regression fit:

- Fit the regression model and extract sandwich variance.

```
library("sandwich")
regr.a1c <- lm(a1c.6 ~ reach * a1c.0,
               data = reach.data)
robust.var <- vcovHC(regr.a1c, type = "HC1")
```

- R knows to include lower-order interaction term.

Subgroup effects in REACH

Regression fit: Output

```
> summary(regr.a1c)

Call:
lm(formula = a1c.6 ~ reach * a1c.0, data = reach.data)

Residuals:
   Min     1Q Median     3Q    Max
-5.125 -1.108 -0.170  0.719  9.455

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   3.4990     0.5434    6.44  3.2e-10 ***
reach         0.7069     0.7729    0.91     0.36
a1c.0         0.6058     0.0615    9.85  < 2e-16 ***
reach:a1c.0  -0.1638     0.0869   -1.89     0.06 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.73 on 431 degrees of freedom
  (70 observations deleted due to missingness)
Multiple R-squared:  0.276,Adjusted R-squared:  0.271
F-statistic: 54.8 on 3 and 431 DF,  p-value: <2e-16
```

Subgroup effects in REACH

Regression fit: Robust variance

- Robust variance matrix is a 4 $\times$ 4 matrix.

```
> robust.var
            (Intercept)   reach    a1c.0 reach:a1c.0
(Intercept)      0.4396 -0.4396 -0.05116     0.05116
reach           -0.4396  0.7552  0.05116    -0.08620
a1c.0           -0.0512  0.0512  0.00618    -0.00618
reach:a1c.0      0.0512 -0.0862 -0.00618     0.01021
```

Subgroup effects in REACH

Example: Subgroup effect with a continuous interaction term

- Model: $\mathrm{E}[Y|X = x, Z = z] = \beta_0 + \beta_1 x + \beta_2 z + \beta_3 xz$.
  - X: REACH (0 = control; 1 = REACH).
  - Z: baseline A1c.
  - Y: six-month A1c.
- How do I learn about the REACH effect among the subgroup with $Z = z_0$?
  - $\mathrm{E}[Y|X = x + 1, Z = z_0] - \mathrm{E}[Y|X = x, Z = z_0] = \beta_1 + \beta_3 z_0$.

Subgroup effects in REACH

Linear combinations: Extra work in R

- To the best of my knowledge, R does not have a generalizable analog to Stata's `lincom`.
- To save you the agony, I created one for use in R:

```
lincom.R <- function(par, mults, coefs, vcov, N, alpha = 0.05) {
  R <- matrix(0, nrow = 1, ncol = length(coefs))
  for (q in 1:length(par)) {R[1,par[q]] <- mults[q]}
  w <- sqrt(as.numeric(t(R %*% coefs) %*%
                  solve(R %*% vcov %*% t(R)) %*%
                  (R %*% coefs)))
  p <- 2*(1 - pt(w, df = N - length(coefs)))
  Est <- R %*% coefs
  tol <- qt(1 - alpha/2, df = N - length(coefs))
  CI.Lo <- R %*% coefs - tol*sqrt(R %*% vcov %*% t(R))
  CI.Hi <- R %*% coefs + tol*sqrt(R %*% vcov %*% t(R))
  return(c(EST = Est, CI.LO = CI.Lo, CI.HI = CI.Hi, P = p))
}
```

- Won't catch mistakes, but will work when used correctly.
- Mimics Stata's *t*-statistic formulation.

Subgroup effects in REACH

Parameters for R function: `lincom.R`

- `par`: indices of parameters to combine.
- `mults`: multiples of those parameters noted by `par`.
- `coefs`: vector of model coefficients.
- `vcov`: variance-covariance matrix.
- `N`: number of observations used in analysis.
- `alpha`: confidence level (0.05 by default).

Subgroup effects in REACH

Linear combinations:

- Subgroup effect among those with baseline A1c of 7.5%.

```
lincom.R(par = c(2,4),
         mults = c(1,7.5),
         coefs = regr.a1c$coefficients,
         vcov = robust.var,
         N = dim(regr.a1c$model)[1])

EST      CI.LO     CI.HI          P
-0.52160 -0.89603 -0.14717  0.00322
```

- If you want to test $\beta_1 + 7.5\beta_3$, then the *indices* are 2 and 4 (not 1 and 3). The multiples are 1 and 7.5.

Three-way interactions in REACH

Reminder of setup:

- This example also makes use of the REACH data.
- Allow interaction by REACH, gender, baseline A1c.
  - X: REACH.
  - Z: gender.
  - W: baseline A1c.
  - Y: six-month A1c.

$$\begin{aligned} \mathrm{E}[Y|X=x, Z=z, W=w] &= \beta_0 + \beta_1 x + \beta_2 z + \beta_3 w \\ &\quad + \beta_4 xz + \beta_5 xw + \beta_6 wz + \beta_7 xzw \end{aligned}$$

Three-way interactions in REACH

Regression fit:

- Fit the regression model and extract sandwich variance.

```
library("sandwich")
regr.a1c <- lm(a1c.6 ~ reach * gender * a1c.0,
               data = reach.data)
robust.var <- vcovHC(regr.a1c, type = "HC1")
```

- R knows to include all lower-order interaction terms.

Three-way interactions in REACH

Regression fit: Output

```
> summary(regr.a1c)

Call:
lm(formula = a1c.6 ~ reach * gender * a1c.0, data = reach.data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.6703 -1.0592 -0.1864  0.7220  9.3823

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)         2.70617    0.79717   3.395 0.000751 ***
reach               1.89309    1.18664   1.595 0.111376
gender              1.47667    1.09984   1.343 0.180110
a1c.0               0.69993    0.09327   7.504 3.63e-13 ***
reach:gender       -2.11828    1.57526  -1.345 0.179430
reach:a1c.0        -0.29522    0.13492  -2.188 0.029198 *
gender:a1c.0       -0.17057    0.12521  -1.362 0.173838
reach:gender:a1c.0  0.23074    0.17739   1.301 0.194044
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.733 on 427 degrees of freedom
  (70 observations deleted due to missingness)
Multiple R-squared:  0.2799,Adjusted R-squared:  0.2681
F-statistic: 23.71 on 7 and 427 DF,  p-value: < 2.2e-16
```

Three-way interactions in REACH

Regression fit: Robust variance

- Robust variance matrix is an 8 $\times$ 8 matrix (too big to report, but I'll show you the first five rows and columns below).

```
                (Intercept)   reach  gender   a1c.0 reach:gender   . . .
(Intercept)          0.8172 -0.8172 -0.8172 -0.0959       0.8172
reach               -0.8172  1.3806  0.8172  0.0959      -1.3806
gender              -0.8172  0.8172  1.6166  0.0959      -1.6166
a1c.0               -0.0959  0.0959  0.0959  0.0117      -0.0959
reach:gender         0.8172 -1.3806 -1.6166 -0.0959       2.8024
                       .
                       .
                       .
```

Three-way interactions in REACH

Joint testing: Extra work in R

- To the best of my knowledge, R does not have a generalizable analog to `testparm`. I created one for your convenience:

```
testparm.R <- function(par, coefs, vcov, N = NULL, type = "F") {
  R <- matrix(0, nrow = length(par), ncol = length(coefs))
  for (q in 1:length(par)) {R[q,unlist(par[q])] <- 1}
  if (type == "F") {
    if (is.null(N)) {stop("Please provide a value for N")}
    f <- as.numeric(t(R %*% coefs) %*%
                      solve(R %*% vcov %*% t(R)) %*%
                      (R %*% coefs)/(length(par)))
    p <- 1 - pf(f, df1 = length(par),
                df2 = N - (length(coefs)))
    return(c(F = f, P = p)) }
  if (type == "W") {
    w <- as.numeric(t(R %*% coefs) %*%
                      solve(R %*% vcov %*% t(R)) %*%
                      (R %*% coefs))
    p <- 1 - pchisq(w, df = length(par))
    return(c(W = w, P = p))}
}
```

Three-way interactions in REACH

Parameters for R function: `testparm.R`

- `par`: list of combinations of parameters for joint test.
- `coefs`: vector of model coefficients.
- `vcov`: variance-covariance matrix.
- `N`: number of observations used in analysis.
- `type`: either "F" for *F*-test or "W" for Wald test.

Three-way interactions in REACH

Example: Testing overall effect of REACH

- $H_0 : \beta_1 = \beta_4 = \beta_5 = \beta_7 = 0$.

```
testparm.R(par = list(2,5,6,8),
           coefs = regr.a1c$coefficients,
           vcov = robust.var,
           N = dim(regr.a1c$model)[1])
```

```
## Output
       F        P
5.397209 0.000301
```

Three-way interactions in REACH

Example: Testing effect of REACH among females

- $H_0 : \beta_1 = \beta_5 = 0$.

```
testparm.R(par = list(2,6),
           coefs = regr.a1c$coefficients,
           vcov = robust.var,
           N = dim(regr.a1c$model)[1])
```

```
## Output
      F       P
5.25470 0.00557
```

Three-way interactions in REACH

Example: Testing effect of REACH among males

- $H_0 : \beta_1 + \beta_4 = \beta_5 + \beta_7 = 0.$

```
testparm.R(par = list(c(2,5), c(6,8)),
           coefs = regr.a1c$coefficients,
           vcov = robust.var,
           N = dim(regr.a1c$model)[1])
```

```
## Output
      F       P
5.53972 0.00422
```

Three-way interactions in REACH

Example: Testing interaction between baseline A1c and REACH

- $H_0 : \beta_5 = \beta_7 = 0$.

```
testparm.R(par = list(6,8),
           coefs = regr.a1c$coefficients,
           vcov = robust.var,
           N = dim(regr.a1c$model)[1])

## Output
     F      P
2.3293 0.0986
```

Table of Contents

Weighted least squares in MRI

Example: DSST and age

- X: age.
- Y: DSST.
- Model: $\mathrm{E}[Y|X = x] = \beta_0 + \beta_1 x$.
- Consider unweighted model, and model weighting inversely to age (as an example).

Weighted least squares in MRI

Reading in the MRI data:

- Read in data:

```
mri.data <- read.csv("mri.csv",
                      header = TRUE,
                      stringsAsFactors = FALSE)
```

Weighted least squares in MRI

Generating weights:

- Create and attach weights for weighted model:

```
mri.data$wts <- mri.data$age
```

Weighted least squares in MRI

Model fitting:

- Fit unweighted and weighted regression models.

```
model.u <- lm(dsst ~ age,
              data = mri.data)
model.w <- lm(dsst ~ age, weights = wts,
              data = mri.data)
```

- Make note of option `weights`.

Weighted least squares in MRI

Results:

- Unweighted model: ordinary standard errors.

```
> summary(model.u)

Call:
lm(formula = dsst ˜ age, data = mri.data)

Residuals:
   Min     1Q Median     3Q    Max
-41.45  -7.61  -0.14   7.55  44.00

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 105.3395     6.1570    17.1   <2e-16 ***
age          -0.8633     0.0825   -10.5   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 11.8 on 721 degrees of freedom
  (12 observations deleted due to missingness)
Multiple R-squared:  0.132,Adjusted R-squared:  0.131
F-statistic:  110 on 1 and 721 DF,  p-value: <2e-16
```

Weighted least squares in MRI

Results:

- Weighted model: ordinary standard errors.

```
> summary(model.w)

Call:
lm(formula = dsst ~ age, data = mri.data, weights = wts)

Weighted Residuals:
   Min     1Q Median     3Q    Max 
-356.7  -64.6   -0.8   65.5  388.9 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 106.2011     5.9753    17.8   <2e-16 ***
age          -0.8748     0.0796   -11.0   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 102 on 721 degrees of freedom
  (12 observations deleted due to missingness)
Multiple R-squared:  0.143,Adjusted R-squared:  0.142 
F-statistic:  121 on 1 and 721 DF,  p-value: <2e-16
```

Weighted least squares in MRI

Results:

- Unweighted model: sandwich standard errors.

```
> sqrt(diag(vcovHC(model.u, type = "HC1")))
(Intercept)         age
    5.70871     0.07551
```

Weighted least squares in MRI

Results:

- Weighted model: sandwich standard errors.

```
> sqrt(diag(vcovHC(model.w, type = "HC1")))
(Intercept)         age
    5.61944     0.07428
```

Table of Contents

Height and FEV

Reading in the FEV data:

- Read in data:

```
fev.data <- read.csv("fev.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
```

Height and FEV

Natural cubic splines:

- You're free to use my function:

```
ncs.R <- function(x, knots, stub = "b")
{
  N <- length(x); P <- length(knots)
  zP <- knots[P]; zP.1 <- knots[P - 1]
  bmat <- matrix(0, nrow = N, ncol = P - 1)
  bmat[,1] <- as.numeric(x)
  nms <- c(paste(stub, 1, sep = ""))
  for (j in 1:(P - 2))
  {
    zp <- knots[j]
    dp.num <- pmax(0, (x - zp)^3) - pmax(0, (x - zP)^3)
    dp <- dp.num/(zP - zp)
    dP.1.num <- pmax(0, (x - zP.1)^3) - pmax(0, (x - zP)^3)
    dP.1 <- dP.1.num/(zP - zP.1)
    bmat[,j + 1] <- dp - dP.1
    nms <- c(nms, paste(stub, j + 1, sep = ""))
  }
  bmat <- data.frame(cbind(1, bmat))
  names(bmat) <- c(paste(stub, 0, sep = ""), nms)
  return(bmat)
}
```

Height and FEV

Parameters for R function: `ncs.R`

- `x`: variable on which you seek to create spline basis functions.
- `knots`: values of the knots.
- `stub`: stub for variable name (helpful if you want to create basis functions for more than one variable and still be able to distinguish between them later).

Height and FEV

Natural cubic splines:

- Create matrix of basis functions (appending original variables for convenience of coding).

```
hmat <- ncs.R(fev.data$height, knots = c(50, 60, 70))
hmat$FEV <- fev.data$fev
hmat$height <- fev.data$height
```

Height and FEV

Natural cubic splines:

- Regression with basis splines included; extract coefficients.

```
zz.ncs <- lm(FEV ~ b1 + b2, data = hmat)
cfs <- coef(zz.ncs)
```

Height and FEV

Adding natural cubic splines to a plot:

- You're free to use my function:

```
line.ncs <- function(range, knots, coefs, col = "blue", lwd = 2, lty = 1)
{
  N <- length(range); P <- length(knots)
  zP <- knots[P]; zP.1 <- knots[P - 1]
  bmat <- matrix(0, nrow = N, ncol = P - 1)
  bmat[,1] <- as.numeric(range)
  for (j in 1:(P - 2))
  {
    zp <- knots[j]
    dp.num <- pmax(0, (range - zp)^3) - pmax(0, (range - zP)^3)
    dp <- dp.num/(zP - zp)
    dP.1.num <- pmax(0, (range - zP.1)^3) - pmax(0, (range - zP)^3)
    dP.1 <- dP.1.num/(zP - zP.1)
    bmat[,j + 1] <- dp - dP.1
  }
  bmat <- cbind(1, bmat)
  prdct <- bmat %*% coefs
  lines(range, prdct, col = col, lwd = lwd, lty = lty)
}
```

Height and FEV

Parameters for R function: `line.ncs`

- `range`: all *x*-values you want to use to generate the curve.
- `knots`: values of the knots (must be same as for basis spline generation).
- `coefs`: coefficients from regression model

Height and FEV

Natural cubic splines for height and FEV:

- Plot data and add splines (should match plot from notes).

```
plot(hmat$height, hmat$FEV, frame.plot = FALSE,
     xlab = "Height (in)", ylab = "FEV (L)",
     ylim = c(0,6), cex = 0.75, pch = 20,
     col = "gray40", main = "Natural cubic spline")

rng <- seq(min(fev.data$height),
           max(fev.data$height), 0.1)
line.ncs(rng, knots = c(50, 60, 70),
         coefs = cfs, lwd = 3)
```

Table of Contents

Diabetes and gender in MRI study

Example:

- X: $0 =$ female; $1 =$ male.
- Y: $0 =$ no diabetes; $1 =$ diabetes.

	Diabetes	No diabetes	Total
Male	53	313	366
Female	26	343	369
Total	79	656	735

- Estimated prevalence difference: 0.0743
- Estimated odds ratio (OR): 2.234
- Estimated prevalence ratio (RR): 2.055

Diabetes and gender in MRI study

Reading in the MRI data:

- Read in data:

```
mri.data <- read.csv("mri.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
```

Diabetes and gender in MRI study

Binary outcome regression: Identity link

- Function `glm` in R (must specify family and link):

```
model.1 <- glm(diabetes ~ male,
               family = binomial(link = "identity"),
               data = mri.data)
```

Diabetes and gender in MRI study

Binary outcome regression: Identity link (Results)

```
> summary(model.1)

Call:
glm(formula = diabetes ~ male, family = binomial(link = "identity"),
    data = mri.data)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-0.5593  -0.5593  -0.3823  -0.3823   2.3034

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.07046    0.01332   5.289 1.23e-07 ***
male         0.07435    0.02271   3.273  0.00106 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 501.59  on 734  degrees of freedom
Residual deviance: 490.82  on 733  degrees of freedom
AIC: 494.82

Number of Fisher Scoring iterations: 2
```

Diabetes and gender in MRI study

Binary outcome regression: Identity link

- Sandwich variance:

```
robust.var <- vcovHC(model.1, type = "HC1")

## Output
> sqrt(diag(robust.var))
(Intercept)        male
 0.01334092  0.02274339
```

- Does *not* agree perfectly with Stata output.
- The reason is a different degrees of freedom correction.
  - Stata uses $N - 1$; R uses $N - K$ ($K = 2$ in this case).

Diabetes and gender in MRI study

Binary outcome regression: Identity link

- Sandwich variance (calibrating degrees of freedom):

```
N <- dim(model.1$model)[1]
robust.var.df <- vcovHC(model.1, type = "HC1") * ((N - 2)/(N - 1))

## Output
> sqrt(diag(robust.var.df))
(Intercept)        male
 0.01333183  0.02272789
```

Diabetes and gender in MRI study

Side note:

- Stata seems to often use $N - 1$ irrespective of the number of model parameters.
- If you are using R for assignments, you are not expected to change the degrees of freedom correction to match Stata's. I'm just illustrating why there is a discrepancy here.

Diabetes and gender in MRI study

Binary outcome regression: Logit link

- Function `glm` in R (must specify family and link):

```
model.2 <- glm(diabetes ~ male,
               family = binomial(link = "logit"),
               data = mri.data)
robust.var <- vcovHC(model.2, type = "HC1") * (N - 2)/(N - 1)
```

Diabetes and gender in MRI study

Binary outcome regression: Logit link (salient results)

```
exp(c(OR = coef(model.2)[2],
      CI.Low = coef(model.2)[2] - qnorm(0.975) * sqrt(diag(robust.var))[2],
      CI.High = coef(model.2)[2] + qnorm(0.975) * sqrt(diag(robust.var))[2]))

## Output
    OR.male  CI.Low.male CI.High.male
    2.233841     1.363051     3.660940
```

Diabetes and gender in MRI study

Binary outcome regression: Log link

- Function `glm` in R (must specify family and link):

```
model.3 <- glm(diabetes ~ male,
               family = binomial(link = "log"),
               data = mri.data)
robust.var <- vcovHC(model.3, type = "HC1") * ((N - 2)/(N - 1))
```

Diabetes and gender in MRI study

Binary outcome regression: Log link (salient results)

```
exp(c(RR = coef(model.3)[2],
      CI.Low = coef(model.3)[2] - qnorm(0.975) * sqrt(diag(robust.var))[2],
      CI.High = coef(model.3)[2] + qnorm(0.975) * sqrt(diag(robust.var))[2]))

## Output
    RR.male  CI.Low.male CI.High.male
    2.055170     1.314688     3.212721
```

Diabetes and race in MRI study

Example:

- X: 1 = white; 2 = black; 3 = Asian; 4 = other.
- Y: 0 = no diabetes; 1 = diabetes.
- Model:
$$\log(\mathrm{P}(Y=1|X=x)) = \beta_0 + \beta_1 1(x=2) + \beta_2 1(x=3) + \beta_3 1(x=4)$$
- Hypothesis test: $H_0 : \beta_1 = \beta_2 = \beta_3 = 0$ vs. H_1 : (not H_0).

Diabetes and race in MRI study

Binary outcome regression: Joint testing

- Fit model and extract robust variance (note that we're re-calibrating the degrees of freedom correction).

```
model.race <- glm(diabetes ~ factor(race),
                  family = binomial(link = "log"),
                  data = mri.data)

N <- dim(model.race$model)[1]
robust.var <- vcovHC(model.race, type = "HC1") * ((N - 4)/(N - 1))
```

Diabetes and race in MRI study

Binary outcome regression: Joint testing

- The `testparm.R` function will work in this context.

```
testparm.R(par = list(2,3,4),
           coefs = coef(model.race),
           vcov = robust.var,
           type = "W")

## Output
         W           P
6.62942677 0.08469562
```

  - Note the use of a Wald test rather than an F-test.

Table of Contents

Diabetes and CHD in MRI study

Example:

- Multinomial regression model using MRI data:
    - X_1: $0 =$ no diabetes; $1 =$ diabetes.
    - X_2: age (years).
    - X_3: $0 =$ female; $1 =$ male.
    - Y: $0 =$ no CHD; $1 =$ angina; $2 =$ myocardial infarction.

Diabetes and CHD in MRI study

Multinomial regression:

- The `multinom` function is the most reliable one I could find, and requires the `nnet` package.

```
mreg <- multinom(chd ~ diabetes + age + male,
                 data = mri.data)
```

Diabetes and CHD in MRI study

Multinomial regression: Results

```
> exp(summary(mreg)$coefficients)

  (Intercept) diabetes      age     male
1  0.00250757 1.095997 1.048926 1.431986
2  0.06102624 1.773780 1.006461 2.003295
```

Diabetes and CHD in MRI study

Multinomial regression: Testing

- I am unaware of a method to obtain robust standard errors with `multinom` other than hard-coding. In this class, not worth effort to hard-code robust standard error for this model.
- Note: `testparm.R` works with non-robust variance (`vcov`).

```
testparm.R(par = list(2,6),
           coefs = c(coef(mreg)[1,],
                     coef(mreg)[2,]),
           vcov = vcov(mreg),
           type = "W")

## Output
        W         P
3.3446370 0.1878111
```

- Does not agree exactly with Stata output.

Table of Contents

General health in MRI study

Example:

- Proportional odds model:
  - X_1: age (years).
  - X_2: 0: female; 1: male.
  - Y: view of own health (1:5)
    - Higher values indicate poorer view of health.

General health in MRI study

Ordinal regression:

- The `polr` function is the most reliable one I could find, and requires the `MASS` package.

```
model.gh <- polr(factor(genhlth) ˜ age + male,
                 data = mri.data)
```

General health in MRI study

Ordinal regression: Results

```
exp(summary(model.gh)$coef[1:2,1])

## Output
      age      male
1.0277373 0.9203347
```

- Odds ratios do not agree perfectly with Stata output, but close.
  - Reason for discrepancy not clear (likely numeric in nature rather than the result of a substantive modeling assumption).

General health in MRI study

Ordinal regression: Standard errors

- The `vcovHC` function is not compatible with `polr` command, but the `sandwich` function is (does not include a degrees of freedom correction).

```
N <- dim(model.gh$model)[1]

> sqrt(diag(sandwich(model.gh) * (N)/(N - 1)))[1:2]

Re-fitting to get Hessian

       age       male
0.01339874 0.13542919
```

- Does not agree perfectly with Stata output, but close.

Table of Contents

Age and lymph nodes in endometrial study

Example:

- Y: # of nodes removed (count).
- X: age (years).
- Model: $\log(\mathrm{E}[Y|X = x]) = \beta_0 + \beta_1 x$

Age and lymph nodes in endometrial study

Reading in the endometrial data:

- Read in data:

```
endo.data <- read.csv("endometrial.csv",
                      header = TRUE,
                      stringsAsFactors = FALSE)
```

Age and lymph nodes in endometrial study

Poisson regression: Log link

- Function `glm` in R (must specify family and link):

```
model.nodes <- glm(nodes ~ age,
                   family = poisson(link = "log"),
                   data = endo.data)
```

Age and lymph nodes in endometrial study

Poisson regression: Log link

- Results:

```
N <- dim(regr.pois$model)[1]
robust.var <- vcovHC(regr.pois, type = "HC1") * ((N - 2)/(N - 1))

exp(c(IRR = coef(model.nodes)[2],
    CI.Low = coef(model.nodes)[2] - qnorm(0.975) * sqrt(diag(robust.var))[2],
    CI.High = coef(model.nodes)[2] + qnorm(0.975) * sqrt(diag(robust.var))[2]))

## Output
    IRR.age  CI.Low.age CI.High.age
   1.012862    1.002544    1.023286
```

Table of Contents

Kaplan-Meier by gender

Reading in the MRI data:

- Read in data:

```
mri.data <- read.csv("mri.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
```

Kaplan-Meier by gender

Survival library:

- Many methods to model time-to-event data rely on the `survival` package.

```
library("survival")
```

Kaplan-Meier by gender

Kaplan-Meier estimation:

- The `survfit` function allows us to perform Kaplan-Meier estimation.

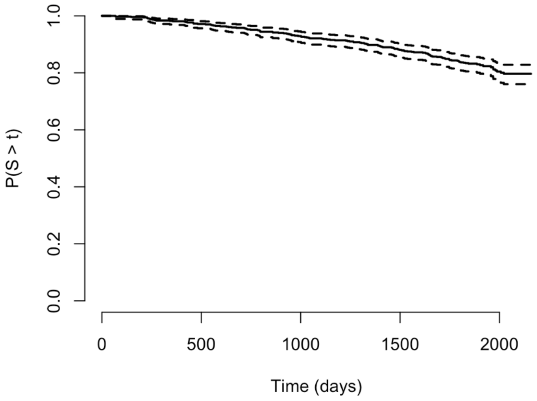
```
s.overall <- survfit(Surv(obstime, death) ~ 1,
                     conf.type = "log-log",
                     data = mri.data)

plot(s.overall,
     frame.plot = FALSE,
     col = c("black"),
     lwd = 2,
     xlab = "Time (days)",
     ylab = "P(S > t)")
```

- Confidence intervals from a log-log transformation will allow us to come close to Stata's results.

Kaplan-Meier by gender

Kaplan-Meier estimation: Overall



- Confidence intervals included if only presenting one group.

Kaplan-Meier by gender

Kaplan-Meier estimation:

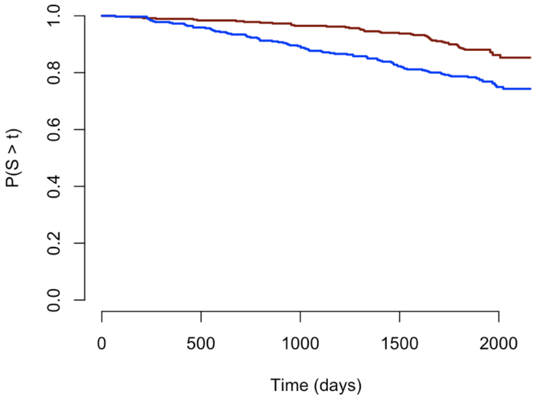
- Curves stratified by gender.

```
s.gender <- survfit(Surv(obstime, death) ~ male,
                    conf.type = "log-log",
                    data = mri.data)

plot(s.gender,
     frame.plot = FALSE,
     conf.int = FALSE,
     col = c("darkred", "blue"),
     lwd = c(2,2),
     xlab = "Time (days)",
     ylab = "P(S > t)")
```

Kaplan-Meier by gender

Kaplan-Meier estimation: By gender



- Confidence intervals not included if presenting 2+ groups.

Kaplan-Meier by gender

Kaplan-Meier estimation:

- Extracting restricted mean (overall).

```
summary(s.overall)$table[5:6]

## Output
    *rmean *se(rmean)
1974.46913   16.56625
```

Kaplan-Meier by gender

Kaplan-Meier estimation:

- Extracting restricted mean (by gender).

```
summary(s.gender)$table[,5:6]

## Output
         *rmean *se(rmean)
male=0 2049.954   17.55878
male=1 1899.064   27.59734
```

- Approximately agrees with Stata output.

Kaplan-Meier by gender

Kaplan-Meier estimation:

- Extracting quantiles (overall).

```
quantile(s.overall, 0.10)

## Output
$quantile
  10
1338

$lower
  10
1045

$upper
  10
1519
```

Kaplan-Meier by gender

Kaplan-Meier estimation:

- Extracting quantiles (by gender).

```
quantile(s.gender, 0.20)

## Output
$quantile
         20
male=0   NA
male=1 1707

$lower
         20
male=0   NA
male=1 1457

$upper
         20
male=0   NA
male=1 1988
```

- Approximately agrees with Stata.

Log-rank test for gender

Log-rank test:

- Function for log-rank test: `survdiff`.

```
logrank.gender

## Output
Call:
survdiff(formula = Surv(obstime, death) ~ male, data = mri.data)

         N Observed Expected (O-E)^2/E (O-E)^2/V
male=0 369       47     68.8      6.89      14.3
male=1 366       86     64.2      7.38      14.3

 Chisq= 14.3  on 1 degrees of freedom, p= 2e-04
```

Cox regression by gender

Proportional hazards regression:

- Function for Cox model: `coxph`.

```
model.gen <- coxph(Surv(obstime, death) ~ male,
                   ties = "breslow",
                   data = mri.data)
N <- length(model.gen$residuals)
robust.var <- sandwich(model.gen) * N/(N - 1)
```

Cox regression by gender

Proportional hazards regression:

- Results:

```
exp(c(HR = model.gen$coef,
      CI.Low = model.gen$coef - qnorm(0.975) * sqrt(robust.var),
      CI.Hi = model.gen$coef + qnorm(0.975) * sqrt(robust.var)))

## Output
 HR.male   CI.Low    CI.Hi
1.961765 1.378102 2.792624
```

Table of Contents

Time-dependent Cox for heart transplant

Reading in the transplant data:

- Read in data:

```
heart.data <- read.csv("transplant.csv",
                        header = TRUE,
                        stringsAsFactors = FALSE)
```

Time-dependent Cox for heart transplant

Structuring data:

- Create variable for initial time windows.

```
N <- dim(heart.data)[1]
heart.data$ptime <- c(0, heart.data$time[1:(N - 1)])
heart.data$ptime[heart.data$time == 1] <- 0
```

Time-dependent Cox for heart transplant

Time-dependent covariates:

- Account for clustering.

```
model.heart <- coxph(Surv(ptime, time, death) ~ transplant + cluster(id),
                     method = "breslow",
                     data = heart.data)
robust.var <- summary(model.heart)$coef[4]^2 * 21/20
```

Time-dependent Cox for heart transplant

Time-dependent covariates:

- Results:

```
exp(c(HR = model.heart$coef,
      CI.Low = model.heart$coef - qnorm(0.975) * sqrt(robust.var),
      CI.High = model.heart$coef + qnorm(0.975) * sqrt(robust.var)))

## Output
     HR.transplant  CI.Low.transplant CI.High.transplant
          0.261495           0.086369           0.791713
```

Table of Contents

Cumulative incidence of cardiovascular death

Subdistribution hazard in R:

- Competing risks regression can be performed using the `crr` function in the library `cmprsk`.
- We again use the MRI data.

```
library("cmprsk")

mri.data <- read.csv("mri.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
```

Cumulative incidence of cardiovascular death

Structuring data:

- Create variable indicating censoring, event of interest, and competing event(s).

```
mri.data$event <- 0
mri.data$event[mri.data$death == 1 & mri.data$cvd == 1] <- 1
mri.data$event[mri.data$death == 1 & mri.data$cvd == 0] <- 2
```

Cumulative incidence of cardiovascular death

Subdistribution hazard regression:

- Account for competing risks.

```
model.diab <- crr(mri.data$obstime,
                  mri.data$event,
                  mri.data$diabetes)
N <- model.diab$n
robust.var <- summary(model.diab)$coef[3]^2 * N/(N - 1)
```

Cumulative incidence of cardiovascular death

Subdistribution hazard regression:

- Results:

```
exp(c(summary(model.diab)$coef[1],
      summary(model.diab)$coef[1] - qnorm(0.975) * sqrt(robust.var),
      summary(model.diab)$coef[1] + qnorm(0.975) * sqrt(robust.var)))

## Output
[1] 2.44409 1.34119 4.45391
```

Cumulative incidence of cardiovascular death

Plot cumulative incidence:

- Create figures:

```
plot(predict(model.diab, cov1 = 0),
     frame.plot = FALSE,
     ylim = c(0, 0.2),
     col = "darkblue",
     xlab = "Analysis time",
     ylab = "Cumulative incidence")

lines(predict(model.diab, cov1 = 1),
      col = "darkred")

legend(0, 0.20, col = c("darkblue", "darkred"),
       lwd = c(1,1), lty = c(1,1),
       c("diabetes = 0", "diabetes = 1"))
```

Cumulative incidence of cardiovascular death

Cumulative incidence: By diabetes status

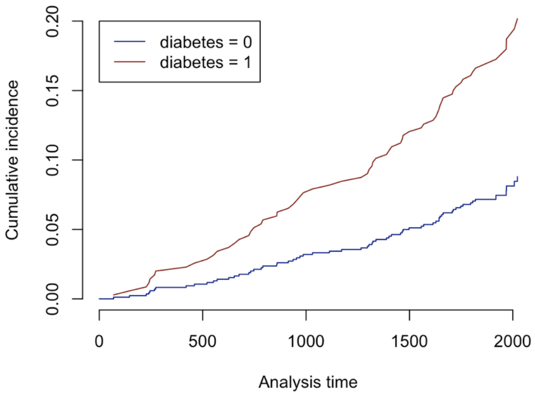


Table of Contents

DSST with LASSO

Reading in the MRI data:

- Read in data:

```
mri.data <- read.csv("mri.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
```

DSST with LASSO

Elastic net for GLMs:

- Many use `glmnet` to perform penalized regression.

```
library("glmnet")
```

DSST WITH LASSO

Split the data:

- Split data into training and test set:

```
N <- dim(mri.data)[1]
set.seed(111)
idx <- sample(1:N, size = floor(N/2), replace = FALSE)
mri.train <- mri.data[idx,]
mri.test <- mri.data[-idx,]
```

DSST WITH LASSO

Split the data:

- Formulate training set:

```
XY.train <- model.matrix(~ age + male + factor(race) + weight +
                           height + packyrs + yrsquit + alcoh +
                           physact + factor(chf) + diabetes +
                           atrophy + factor(numinf) + volinf + dsst,
                           data = mri.train)

Y.train <- as.numeric(XY.train[,dim(XY.train)[2]])
X.train <- XY.train[,2:(dim(XY.train)[2] - 1)]
```

DSST WITH LASSO

Cross-validation:

- Cross-validation on training data (also extract the optimal penalty).
- Note that $\alpha = 0$ corresponds to LASSO.

```
set.seed(4)
zz <- cv.glmnet(x = X.train, y = Y.train,
                family = "gaussian",
                alpha = 1,
                nfolds = 10,
                standardize = TRUE)

lmin <- zz$lambda.min
```

DSST WITH LASSO

Training error:

- Generate and display training error. This will not match the Stata results because the seeds are different.

```
Y.pred.train <- predict(zz, newx = X.train, s = lmin)
rmse.train <- sqrt(mean((Y.pred.train - Y.train)^2))

> rmse.train
[1] 10.1
```

DSST WITH LASSO

Test error:

- Generate test data; generate and display test error. This will not match the Stata results because the seeds are different.

```
XY.test <- matrix(model.matrix(~ age + male + factor(race) + weight +
                                 height + packyrs + yrsquit + alcoh +
                                 physact + factor(chf) + diabetes +
                                 genhlth + ldl + alb + crt + plt + sbp +
                                 aai + fev + atrophy + factor(numinf) +
                                 volinf + dsst, data = mri.test), ncol = 30)

Y.test <- as.numeric(XY.test[,dim(XY.test)[2]])
X.test <- XY.test[,2:(dim(XY.test)[2] - 1)]

Y.pred.test <- predict(zz, newx = X.test, s = lmin)

rmse.test <- sqrt(mean((Y.pred.test - Y.test)^2))

> rmse.test
[1] 11.9
```

Table of Contents

Predicting diabetes in MRI data

Reading in the MRI data:

- Read in data:

```
mri.data <- read.csv("mri.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE)
```

Predicting diabetes in MRI data

Split the data:

- Split data into training and test set; append indices:

```
N <- dim(mri.data)[1]
set.seed(111)
idx <- sample(1:N, size = floor(N/2), replace = FALSE)
mri.data$split <- 0
mri.data$split[idx] <- 1
mri.train <- mri.data[idx,]
mri.test <- mri.data[-idx,]
```

PREDICTING DIABETES IN MRI DATA

Train the model:

- Logistic regression model in training set:

```
zz.diab <- glm(diabetes ~ age + male + packyrs +
               physact + weight + height,
               family = binomial(link = "logit"),
               data = mri.train)
```

Predicting diabetes in MRI data

Generate predictions:

- Create full data set for purposes of generating predictions:

```
full.mat <- data.frame(model.matrix(~ age + male + packyrs + physact +
                                      weight + height + diabetes + split,
                                      data = mri.data))
```

Predicting diabetes in MRI data

AUC: Scaled Mann-Whitney *U*-statistic

- You're welcome to use my function, `lroc.R`:

```
lroc.R <- function(Y, Y.hat, split)
{
  Y.hat.00 <- Y.hat[Y == 0 & split == 0]
  Y.hat.10 <- Y.hat[Y == 1 & split == 0]
  Y.hat.01 <- Y.hat[Y == 0 & split == 1]
  Y.hat.11 <- Y.hat[Y == 1 & split == 1]
  N00 <- length(Y.hat.00); N10 <- length(Y.hat.10)
  N01 <- length(Y.hat.01); N11 <- length(Y.hat.11)
  U0 <- wilcox.test(Y.hat.00, Y.hat.10)$statistic
  U1 <- wilcox.test(Y.hat.01, Y.hat.11)$statistic
  AUC0 <- U0/(N00 * N10)
  if (AUC0 < 0.5) {AUC0 <- 1 - AUC0}
  AUC1 <- U1/(N01 * N11)
  if (AUC1 < 0.5) {AUC1 <- 1 - AUC1}
  return(c(Train.AUC = as.numeric(AUC0),
           Test.AUC = as.numeric(AUC1)))
}
```

Predicting diabetes in MRI data

Parameters for R function: `lroc.R`

- `Y`: true outcome.
- `Y.hat`: predicted value.
- `split`: indicator of training (0) and test (1) set.

Predicting diabetes in MRI data

Predictive ability:

- Training and test AUC. Will not match Stata output as the seed is different.

```
lroc.R(Y = full.mat$diabetes,
      Y.hat = predict(zz.diab, type = "response", newdata = full.mat),
      split = full.mat$split)

## Train.AUC  Test.AUC
##     0.580     0.681
```

Table of Contents

GEE in REACH

Reading in the REACH data:

- Read in data:

```
reach.data <- read.csv("reach.csv",
                       header = TRUE,
                       stringsAsFactors = FALSE)
```

GEE in REACH

Convert REACH data to long form:

- Don't forget to reorder the data and rename...

```
reach.long <- reshape(reach.data,
                      idvar = "id",
                      direction = "long",
                      varying = list(c(12:14), c(15:17)),
                      timevar = "t")

reach.long <- reach.long[order(reach.long$id),]

K <- dim(reach.long)[2]
names(reach.long) <- c(names(reach.long)[1:(K - 2)],
                       "a1c", "sdsca")
```

GEE in REACH

GEE (working independence):

- `geese` function in R (honk, honk!)

```
library("geepack")

zz.indep <- geese(a1c ~ factor(t)*factor(reach),
                  corstr = "independence",
                  data = reach.long)
```

GEE IN REACH

GEE (working independence):

- Output should match Stata results.

```
> summary(zz.indep)

Call:
geese(formula = a1c ~ factor(t) * factor(reach), data = reach.long,
    corstr = "independence")

Mean Model:
 Mean Link:                 identity
 Variance to Mean Relation: gaussian

 Coefficients:
                           estimate  san.se       wald          p
(Intercept)                8.539357 0.11842 5.1997e+03 0.0000e+00
factor(t)2                 0.178742 0.13653 1.7140e+00 1.9047e-01
factor(t)3                 0.044121 0.14029 9.8908e-02 7.5314e-01
factor(reach)1             0.169586 0.16987 9.9668e-01 3.1811e-01
factor(t)2:factor(reach)1 -0.802617 0.18568 1.8684e+01 1.5424e-05
factor(t)3:factor(reach)1 -0.192031 0.20088 9.1384e-01 3.3910e-01
```

GEE IN REACH

GEE (working exchangeable):

- Update the correlation structure!

```
zz.exch <- geese(a1c ~ factor(t)*factor(reach),
                 corstr = "exchangeable",
                 data = reach.long)
```

GEE in REACH

GEE (working exchangeable):

- Output should match Stata results.

```
> summary(zz.exch)

Call:
geese(formula = a1c ~ factor(t) * factor(reach), data = reach.long,
    corstr = "exchangeable")

Mean Model:
 Mean Link:                 identity
 Variance to Mean Relation: gaussian

 Coefficients:
                           estimate  san.se       wald          p
(Intercept)                8.534138 0.11806 5224.99181 0.0000e+00
factor(t)2                 0.178341 0.13523    1.73913 1.8725e-01
factor(t)3                 0.044573 0.13986    0.10157 7.4996e-01
factor(reach)1             0.178013 0.16942    1.10405 2.9338e-01
factor(t)2:factor(reach)1 -0.819732 0.18268   20.13617 7.2120e-06
factor(t)3:factor(reach)1 -0.194139 0.19812    0.96025 3.2713e-01
```

Mixed model in DFMO

Reading in the DFMO data:

- Read in data (subsetted to twelve months):

```
dfmo.data <- read.csv("dfmo.csv",
                      header = TRUE,
                      stringsAsFactors = FALSE)

dfmo.data <- subset(dfmo.data, dfmo.data$time != 15)
```

Mixed model in DFMO

Random intercepts model:

- Including cluster-robust standard errors:

```
library("lme4")
library("clubSandwich")

zz.1 <- lmer(spd ~ dose*time + (1|ptid),
             REML = FALSE,
             data = dfmo.data)

summary(zz.1)
sqrt(diag(vcovCR(zz.1, type = "CR0")))
```

Mixed model in DFMO

Random intercepts model:

- Output:

```
Fixed effects:
            Estimate Std. Error t value
(Intercept)  3.26550    0.17790   18.36
dose         0.52476    0.79889    0.66
time        -0.00714    0.02074   -0.34
dose:time   -0.31906    0.09611   -3.32

> sqrt(diag(vcovCR(zz.1, type = "CR0")))
(Intercept)        dose        time   dose:time
   0.184968    0.911460    0.017868    0.102780
```

- The standard errors do not agree with Stata's exactly.

Mixed model in DFMO

Mixed-effects model:

- Including cluster-robust standard errors:

```
library("lme4")
library("clubSandwich")

zz.2 <- lmer(spd ~ dose*time + (time|ptid),
             REML = FALSE,
             data = dfmo.data)

summary(zz.2)
sqrt(diag(vcovCR(zz.2, type = "CR0")))
```

Mixed model in DFMO

Mixed-effects model:

- Output:

```
Fixed effects:
            Estimate Std. Error t value
(Intercept)  3.26737    0.19527   16.73
dose         0.51463    0.87632    0.59
time        -0.00809    0.02103   -0.38
dose:time   -0.31524    0.09689   -3.25

> sqrt(diag(vcovCR(zz.2, type = "CR0")))
(Intercept)        dose        time   dose:time
   0.185329    0.911482    0.017949    0.102898
```

- The standard errors do not agree with Stata's exactly.