Logic Regression

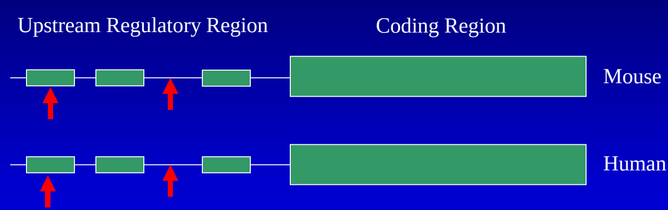
Earl F. Glynn

2 March 2005

Logic Regression

1. Biological Motivation
2. Background
   - Regression
   - Boolean Algebra
3. Logic Regression Formalities
4. Simple Example in "R"
5. Take Home Message

Logic Regression: Biological Motivation
Cyclic Gene Study



Problem: Understand four families of genes and how they bind in the upstream regulatory region. “Bind” and “Don’t Bind” can be interpreted as binary variables. *Combinatorial effects should be considered.*

Logic Regression: Biological Motivation
Regulatory Motif Finding by Logic Regression

U.C. Berkeley Division of Biostatistics Working Paper Series

Year 2004 *Paper* 145

Regulatory Motif Finding by Logic Regression

Sunduz Keles* Mark J. van der Laan† Chris Vulpe‡

BIOINFORMATICS

Regulatory motif finding by logic regression

Sunduz Keles, Mark J. van der Laan and Chris Vulpe

Logic Regression: Background

Regression

- Linear Regression
- Logistic Regression
- Linear Discriminant Analysis

Logic Regression: Background

Linear Regression

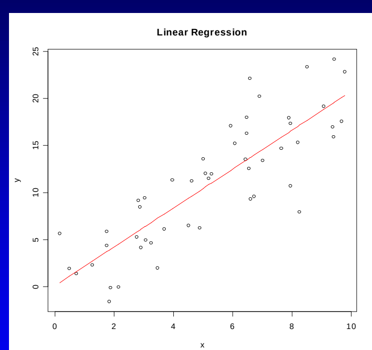
X = continuous-valued explanatory variable

Simple Linear Regression

$$Y = \beta_0 + \beta_1 X$$

Multiple Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots$$




```
# R Code
N <- 50
x <- sort(runif(N,0,10))
y <- 2*x + 3*rnorm(N)
plot(y~x, main="Linear Regression")
LineFit <- lm(y~x)
lines(x, fitted(LineFit), col="red")
summary(LineFit)
```

Logic Regression: Background

Linear Regression Formalities

E = expected response

Mean	$E[Y] = \mu$
Simple Regression	$E[Y \mid X1] = \beta_0 + \beta_1 X_1$
Multiple Regression	$E[Y \mid X1, X2] = \beta_0 + \beta_1 X_1 + \beta_2 X_2$

Logic Regression: Background

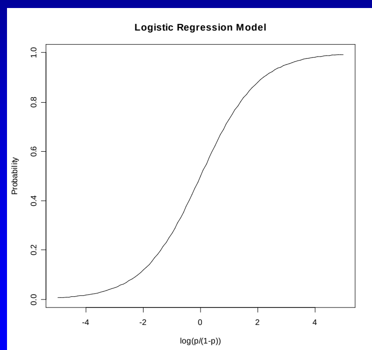
Logistic Regression

- Outcome is like flipping a coin (a Bernoulli trial):
  e.g, binary result: 0 = No, 1 = Yes
- “Predictors” (continuous or discrete) determine how “loaded” coin is
- Want to estimate how much a predictor loads the coin, i.e., changes the probability
- Use “odds” of an event: p/(1-p)
- Log(odds) = Log[p/(1-p)] = logit(p) = “logistic function”
- Preferred by statisticians when dependent variable is binary

Logic Regression: Background
Logistic Regression

Multiple Regression $Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots$

Logistic Regression $\text{Log}[p/(1-p)] = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots$



- Interpretation of coefficients is complicated since they relate to log-odds (logit) and not probability directly.
- Relationship between p and logit(p) is non-linear, but is nearly linear in the middle range of logit(p).

```
R code:  curve(1/(1+exp(-x)), from=-5, to=5, xlab="log(p/(1-p))",
            ylab="Probability", main="Logistic Regression Model")
```


Logic Regression: Background
Linear Discriminant Analysis

- Have observed data for two or more groups and want to distinguish among the groups.
- Discriminant analysis works by creating a new variable that is a linear combination of the original variables.
- Differences among groups are maximized.
- Discriminant functions can be applied to classify unknown cases. Score $= \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots$

Example: from http://obelia.jde.aca.mmu.ac.uk/multivar/da2.htm

discriminant score = 0.0029 height + 1.028 weight - 0.096 age

>0 male

<0 female

In general, a separate discriminant function for each group

Logic Regression: Background
Truth Tables and Boolean Algebra

NOT $\neg A = \overline{A}$ $\neg\neg A = A$

A	ØA
False	True
True	False

AND

$A \wedge B$		B	
		False	**True**
A	**False**	False	False
	True	False	True

OR

$A \vee B$		B	
		False	**True**
A	**False**	False	True
	True	True	True

Commutative Laws

x Ú y = y Ú x
x Ù y = y Ù x

Associative Laws

(x Ú y) Ú z = x Ú (y Ú z)
(x Ù y) Ù z = x Ù (y Ù z)

Distributive Laws

x Ú (y Ù z) = (x Ú y) Ù (x Ú z)
x Ù (y Ú z) = (x Ù y) Ú (x Ù z)

DeMorgan Laws

$$\overline{x \vee y} = \overline{x} \wedge \overline{y}$$

$$\overline{x \wedge y} = \overline{x} \vee \overline{y}$$

Logic Regression Formalities

Let ***Y*** denote "outcome of interest"
• could be continuous quantity, e.g., log ratio mRNA
• could be binary variable for class of genes,
e.g., 0=downregulated gene, 1=upregulated gene

Assume ***N*** independent observations of ***Y*** from same distribution.

Define vector for each gene ***n*** for any given binding site set of size ***M***:

$$\vec{S}_n = (S_{n,1}, \ldots, S_{n,M}),$$

The entries of this vector are defined as

$$S_{n,m} = \begin{cases} 1 & \text{if gene } n \text{ has at least one copy of motif } m, \\ 0 & \text{o.w.} \end{cases}$$

Adapted from Keles (2004), *Regulatory motif finding by logic regressions*

Logic Regression Formalities

Dependent: Y

Independent: $\vec{S}_n = (S_{n,1}, \ldots, S_{n,M}),$

Linear regression model:

$$E[Y \mid \vec{S}] = \beta_0 + \beta_1 S_1 + \cdots + \beta_m S_m$$

Logistic regression model:
(might be more appropriate if Y is binary variable):

$$E\left[\log\left(\frac{P(Y=1 \mid \vec{S})}{1 - P(Y=1 \mid \vec{S})}\right)\right] = \beta_0 + \beta_1 S_1 + \cdots + \beta_m S_m$$

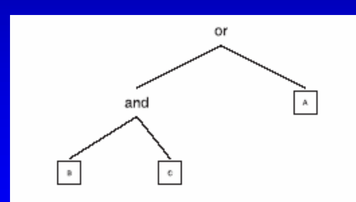
Could use with classification problem, too.
Need to estimate β coefficients, and 'most relevant' motifs.
But, no combinatorial effects considered yet …

Logic Regression
Assumptions for Motif Finding

- Few interacting transcription factors and these require binding to different sites on the transcription control regions
- Interaction of transcription factors, i.e., bind sites, can be reduced to a Boolean expression

Example:

Logic Tree:
L = (B Ù C) Ú A



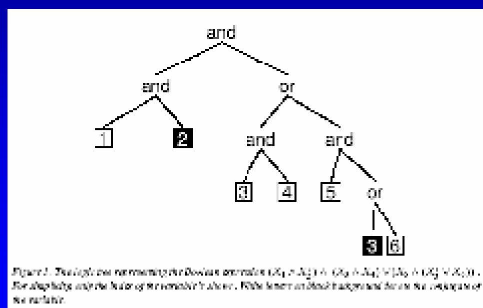
A transcription process might require a gene to have binding sites for factors B and C, or a binding site for factor A, in order to be regulated.

Logic Regression

Assumptions for Motif Finding

Adapted from Ruczinksi, et al, (2003), Logic Regression,
Journal of Computational and Graphical Statistics, 12(3), 475-511.

A more complicated logic tree



Need to decide complexity of desired logic trees

Logic Regression Formalities

Extend model to include combinatorial effects, by letting ***L*** be Boolean expression based on motif scores.

$$Y = \beta_0 + \beta_1 L_1 + \beta_2 L_2$$

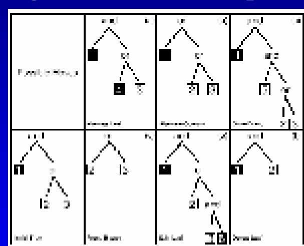
Where L_1 and L_2 are Boolean expressions obtained from vector ***S***. Each ***L*** can be represented by logic tree.

Logic regression identifies combinations of predictors (usually high dimensional) associated with an outcome.

Method works with linear regression, logistic regression, or classification problem.

Logic Regression Formalities

- Ruczinki et al (2003) provide LogicReg "R" package
- Uses simulated annealing algorithm to search high-dimensional space, with well-defined move set:



Adapted from Ruczinksi, et al, (2003),
Logic Regression,
Journal of Computational and Graphical Statistics,
12(3), 475-511.

- Proposed move accepted or rejected based on "score" and "temperature"
- Ruczinksi uses cross-validation and randomization-based hypothesis testing to choose among different model sizes

Logic Regression: Simple Example

Use Ruczinski's "LogicReg" in R

```
library(LogicReg)

X <- matrix(as.numeric(runif(160) < 0.5), 20,8)
colnames(X) <- paste("X", 1:ncol(X), sep="")
rownames(X) <- paste("case", 1:nrow(X), sep="")
# Y = (NOT X2) AND X6
Y <- as.numeric(!X[,2] & X[,6])
cbind(Y, X)
       Y X1 X2 X3 X4 X5 X6 X7 X8
case1  1  0  0  0  0  0  1  1  1
case2  0  0  1  1  1  0  0  1  0
case3  0  0  1  1  1  0  1  0  0
case4  0  1  1  1  1  1  1  0  0
case5  1  1  0  0  0  1  1  1  0
case6  0  0  1  1  0  0  0  1  1
case7  0  0  1  0  0  0  1  1  0
case8  0  0  0  1  0  1  0  0  1
case9  1  1  0  1  0  1  1  0  0
case10 0  0  0  0  1  1  0  0  0
case11 0  0  0  1  0  0  0  0  1
case12 0  0  1  1  1  0  1  1  0
case13 1  1  0  1  1  1  1  0  1
case14 0  0  1  0  1  0  0  1  0
case15 0  1  0  0  1  0  0  0  0
case16 0  1  1  1  1  0  1  0  1
case17 0  0  1  1  0  1  1  1  1
case18 0  0  1  0  0  1  0  1  0
case19 1  0  0  1  1  0  1  0  0
case20 0  1  1  0  0  0  0  0  1
```

Define Simulated Dataset:
Y = (NOT X_2) AND X_6

What kind of regression model is most appropriate here?

Logic Regression: Simple Example

```
SearchIterations <- 1000
Annealing <- logreg.anneal.control(start  = -1, end = -4,
                                   iter   = SearchIterations,
                                   update = SearchIterations/10)
Annealing
$start
[1] -1
$end
[1] -4
$iter
[1] 1000
$earlyout
[1] 0
$update
[1] 100

TreeControl <- logreg.tree.control(treesize=2,  # 2 leaves per node, e.g., (X1 OR X2)
                                   opers=2)     # "and" and "or"
TreeControl
$treesize
[1] 2
$opers
[1] 2
$minmass
[1] 0
```

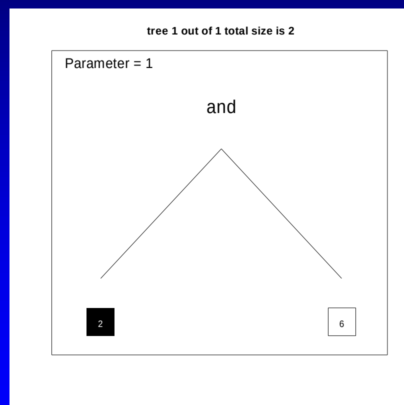

Logic Regression: Simple Example

```
logicfit <- logreg(resp=Y, bin=X,
                   type = REGRESSION.TYPE<-2,
                   select = FIT.SINGLE.MODEL<-1,
                   ntrees=1,
                   nleaves=2,   # force shape of final tree
                   anneal.control=Annealing,
                   tree.control=TreeControl)

log-temp current score    best score        acc / rej /sing    current parameters
  -1.000        0.4894        0.4894     0(   0)     0     0    0.350    0.000
  -1.300        0.4640        0.3145    64(  10)    26     0    0.125    0.375
  -1.600        0.3145        0.3145    55(   4)    41     0    0.077    0.780
  -1.900        0.0000        0.0000     1(   4)    95     0    0.000    1.000
  -2.200        0.0000        0.0000     0(   1)    99     0    0.000    1.000
  -2.500        0.0000        0.0000     0(   6)    94     0    0.000    1.000
  -2.800        0.0000        0.0000     0(   8)    92     0    0.000    1.000
  -3.100        0.0000        0.0000     0(   7)    93     0    0.000    1.000
  -3.400        0.0000        0.0000     0(   4)    96     0    0.000    1.000
  -3.700        0.0000        0.0000     0(   8)    92     0    0.000    1.000
  -4.000        0.0000        0.0000     0(   9)    91     0    0.000    1.000

logicfit
score 0
 +1 * ((not X2) and X6)
```

But what if other sets of random values are used that define the same problem?

Logic Regression: Simple Example

plot(logicfit)



Logic Regression: Simple Example

But what if other sets of random values are used that define the same problem?

Using TreeControl
Repeat 1000 times with 100 Iterations in Simulated Annealing Chain

```
LogicFit1Driver(1000, 20,  100)
**********************************************
      Equations                                         counts
 [1,] "score 0      +1 * (X6 and (not X2))"            "385"
 [2,] "score 0.389  -0.545 * (not X6)"                 "150"
 [3,] "score 0.389  +0.545 * X6"                       "135"
 [4,] "score 0.37   +0.824 * (X4 and X8)"              "105"
 [5,] "score 0      +1 * ((not X2) and X6)"             "76"
 [6,] "score 0.37   +0.824 * (X8 and X4)"               "65"
 [7,] "score 0.375  +0.667 * (X6 and X8)"               "48"
 [8,] "score 0.375  +0.667 * (X8 and X6)"               "18"
 [9,] "score 0.389  -0.545 * ((not X2) and (not X6))"  "5"
[10,] "score 0.416  +0.778 * ((not X4) and X7)"         "5"
[11,] "score 0.389  -0.545 * ((not X6) and (not X2))"  "4"
[12,] "score 0.408  -0.5 * ((not X6) and (not X7))"     "1"
[13,] "score 0.408  -0.5 * ((not X7) and (not X6))"     "1"
[14,] "score 0.416  +0.778 * (X7 and (not X4))"         "1"
[15,] "score 0.416  +0.778 * (X8 and X1)"               "1"

46.1% are "correct"
```

Logic Regression: Simple Example

How does solution vary by size of dataset and by iterations in simulated annealing chain?

Key:

- Number of logic equations (may not be distinct)
- % correct
- computation time[sec]

Iterations in Simulated Annealing Chain	Size of Dataset			
	20	**100**	**1000**	**10,000**
100	15 46.1% 6.3	4 63.2% 6.8	4 59.0 13.8	5 56.1 89.6
500	7 88.9% 8.0	4 95.0% 9.6	4 93.4% 31.2	4 93.7% 268.4
1,000	5 98.1% 9.3	2 100.0% 12.6	2 100.0% 48.3	3 99.9% 480.8
5,000	2 100.0% 20.0	2 100.0% 35.9	2 100.0% 217.8	2 100.0% 2167.2
10,000	2 100.0% 33.2	2 100.0% 64.8	2 100.0% 423.2	2 100.0% 4244.3

Ruczinksi: In practice use 25,000 iterations or more; 2,500 for a fast run.

Logic Regression: Example 2

Not yet Completed

Define "Problem"

$$Y = 0.5 + 2L_1 - 3L_2 - 0.5L_3$$

where

$$L_1 = (X_7 \vee X_{12}) \wedge \overline{X_{73}}$$

$$L_2 = \overline{X_{14}} \wedge X_{23}$$

$$L_3 = \overline{X_{99}}$$

Assume:

```
CaseCount <- 200
BinaryPredictorCount <- 100
SimulatedAnnealingIterations <- 50000

Annealing <- logreg.anneal.control(start = -1, end = -4,
                                   iter =   SimulatedAnnealingIterations,
                                   update = SimulatedAnnealingIterations/10)

TreeControl <- logreg.tree.control(treesize=2,  # 2 leaves per node, or 1 operators

                                   opers=2)     # "and" and "or"
```

Logic Regression: Example 2

Expecting	Observed	Comment
not X99	+0 * (not X84) +1 * (not X99)	OK
0.5 * (not X99)	+0 * (not X84) +0.5 * (not X99)	OK
-0.5 * (not X99)	+1.79e-37 * 1	Error. Cannot use negative coefficients?
0.5 * X99	+0 * (X72 and (not X43)) +0.5 * X99	OK
(not X14) and X23	+1 * ((not X14) and X23)	OK
-3 * ((not X14) and X23)	+1.75e-37 * 1	Error. Cannot use negative coefficients?
3 * ((not X14) and X23)	+3 * ((not X14) and X23)	OK
3 * (X14 or (notX23))	-3 * ((not X14) and X23)	OK (applied DeMorgan's Rule to previous line)

Logic Regression: Example 2

Refine "Problem" by using all positive coefficients and taking "-1" as negative of the logic expression:

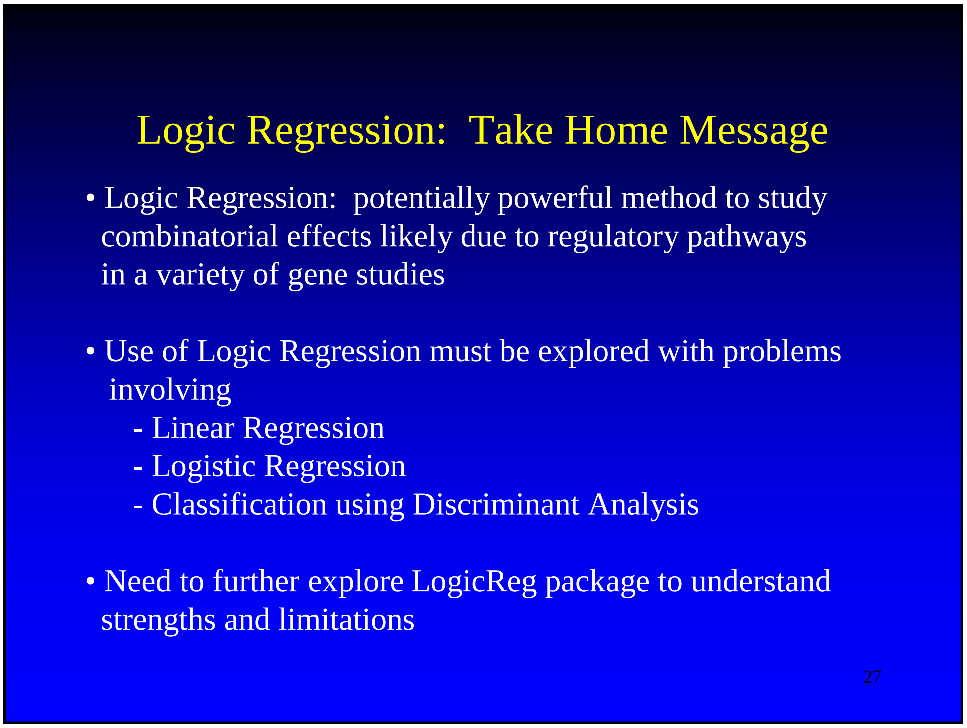
Expecting	Observed	Comment
X7 or X12	-1 * ((not X12) and (not X7))	OK (apply DeMorgan's)
(X7 or X12) and (not X73)	+0.775 * (not X73) -0.564 * ((not X7) and (not X12))	?

Needed to modify TreeControl:

```
TreeControl <- logreg.tree.control(treesize=4,  # 4 leaves per node, or 2 operators
                                   opers=2)     # "and" and "or"
```

Expecting	Observed	Comment
(X7 or X12) and (not X73)	+1 * (not X73) -1 * ((not X12) and ((not X7) and (not X73)))	Can be shown to be equivalent
2 * [(X7 or X12) and (not X73)]	+2 * (((not X73) and X7) and (not X12)) +2 * ((not X73) and X12)	?

Ongoing …


Logic Regression: Take Home Message
• Logic Regression: potentially powerful method to study combinatorial effects likely due to regulatory pathways in a variety of gene studies
• Use of Logic Regression must be explored with problems involving
- Linear Regression
- Logistic Regression
- Classification using Discriminant Analysis
• Need to further explore LogicReg package to understand strengths and limitations
27