Generalized Linear Models Classification

SPH 247

Statistical Analysis of Laboratory Data

Generalized Linear Models

- The type of predictive model one uses depends on a number of issues; one is the type of response.
- Measured values such as quantity of a protein, age, weight usually can be handled in an ordinary linear regression model, possibly after a log transformation.
- Patient survival, which may be censored, calls for a different method (survival analysis, Cox regression).

- If the response is binary, then can we use logistic regression models
- If the response is a count, we can use Poisson regression
- If the count has a higher variance than is consistent with the Poisson, we can use a negative binomial or overdispersed Poisson
- Other forms of response can generate other types of generalized linear models

Generalized Linear Models

- We need a *linear predictor* of the same form as in linear regression βx
- In theory, such a linear predictor can generate any type of number as a prediction, positive, negative, or zero
- We choose a suitable distribution for the type of data we are predicting (normal for any number, gamma for positive numbers, binomial for binary responses, Poisson for counts)
- We create a *link function* which maps the mean of the distribution onto the set of all possible linear prediction results, which is the whole real line $(-\infty, \infty)$.
- The inverse of the link function takes the linear predictor to the actual prediction

- Ordinary linear regression has identity link (no transformation by the link function) and uses the normal distribution
- If one is predicting an inherently positive quantity, one may want to use the log link since e^x is always positive.
- An alternative to using a generalized linear model with an log link, is to transform the data using the log or maybe glog. This is a device that works well with measurement data and may be usable in other cases

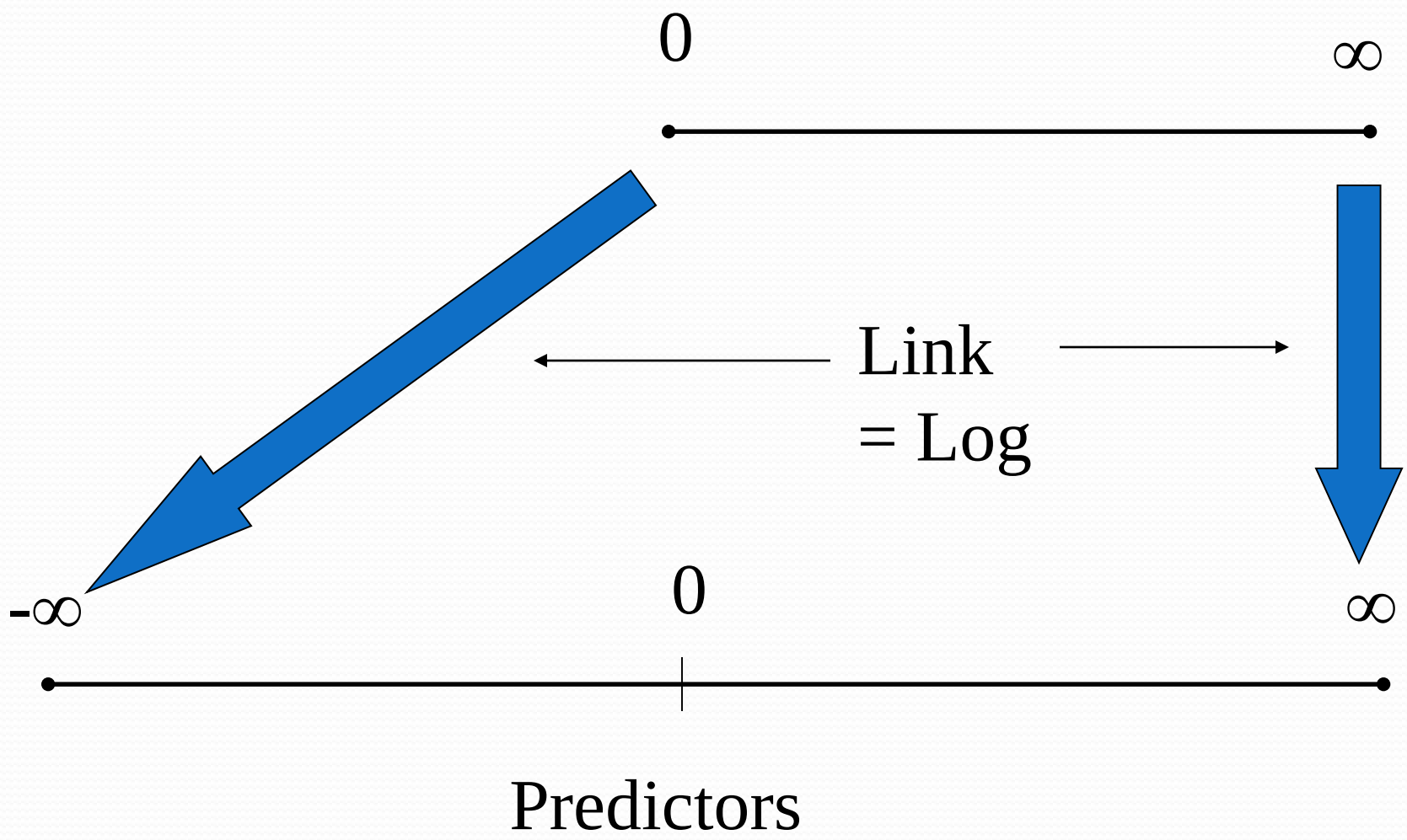
R glm() Families

Family	Links
gaussian	**identity**, log, inverse
binomial	**logit**, probit, cauchit, log, cloglog
Gamma	**inverse**, identity, log
inverse.gaussian	**1/mu^2**, inverse, identity, log
poisson	**log**, identity, sqrt
quasi	**identity**, logit, probit, cloglog, inverse, log, 1/mu^2 and sqrt
quasibinomial	**logit**, probit, identity, cloglog, inverse, log, 1/mu^2 and sqrt
quasipoisson	**log**, identity, logit, probit, cloglog, inverse, 1/mu^2 and sqrt

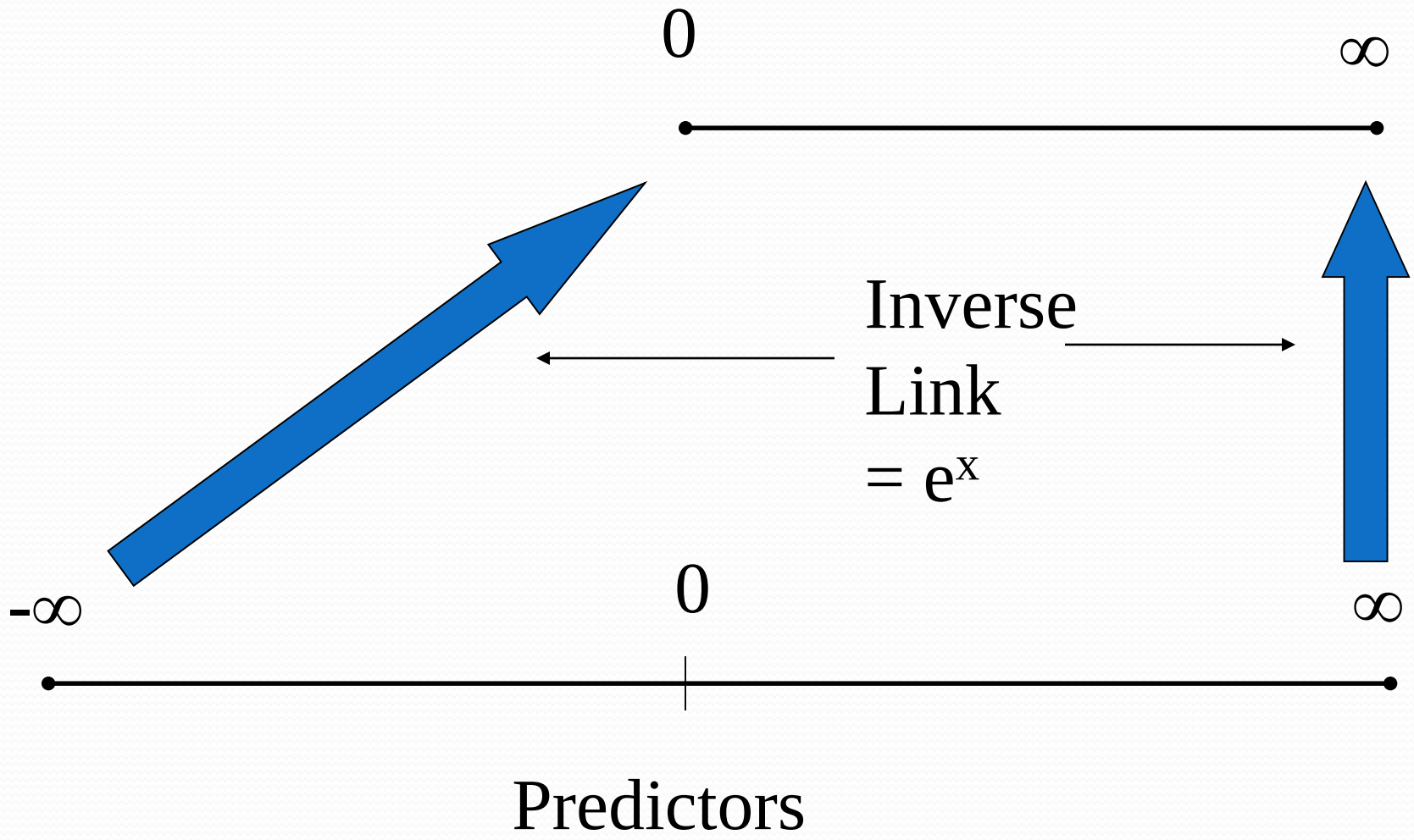
R glm() Link Functions

Links	Domain	Range	
identity	$(-\infty, \infty)$	$(-\infty, \infty)$	$\eta = X\beta = g(\mu) = \mu$
log	$(0, \infty)$	$(-\infty, \infty)$	$\eta = X\beta = g(\mu) = \log(\mu)$
inverse	$(0, \infty)$	$(0, \infty)$	$\eta = X\beta = g(\mu) = 1/\mu$
logit	$(0, 1)$	$(-\infty, \infty)$	$\eta = X\beta = g(\mu) = \log\left(p/(1-p)\right)$
probit	$(0, 1)$	$(-\infty, \infty)$	$\eta = X\beta = g(\mu) = \Phi^{-1}(p)$
cloglog	$(0, 1)$	$(-\infty, \infty)$	$\eta = X\beta = g(\mu) = \log(-\log(1-p))$
1/mu^2	$(0, \infty)$	$(0, \infty)$	$\eta = X\beta = g(\mu) = 1/\mu^2$
sqrt	$(0, \infty)$	$(0, \infty)$	$\eta = X\beta = g(\mu) = \sqrt{\mu}$

Possible Means



Possible Means

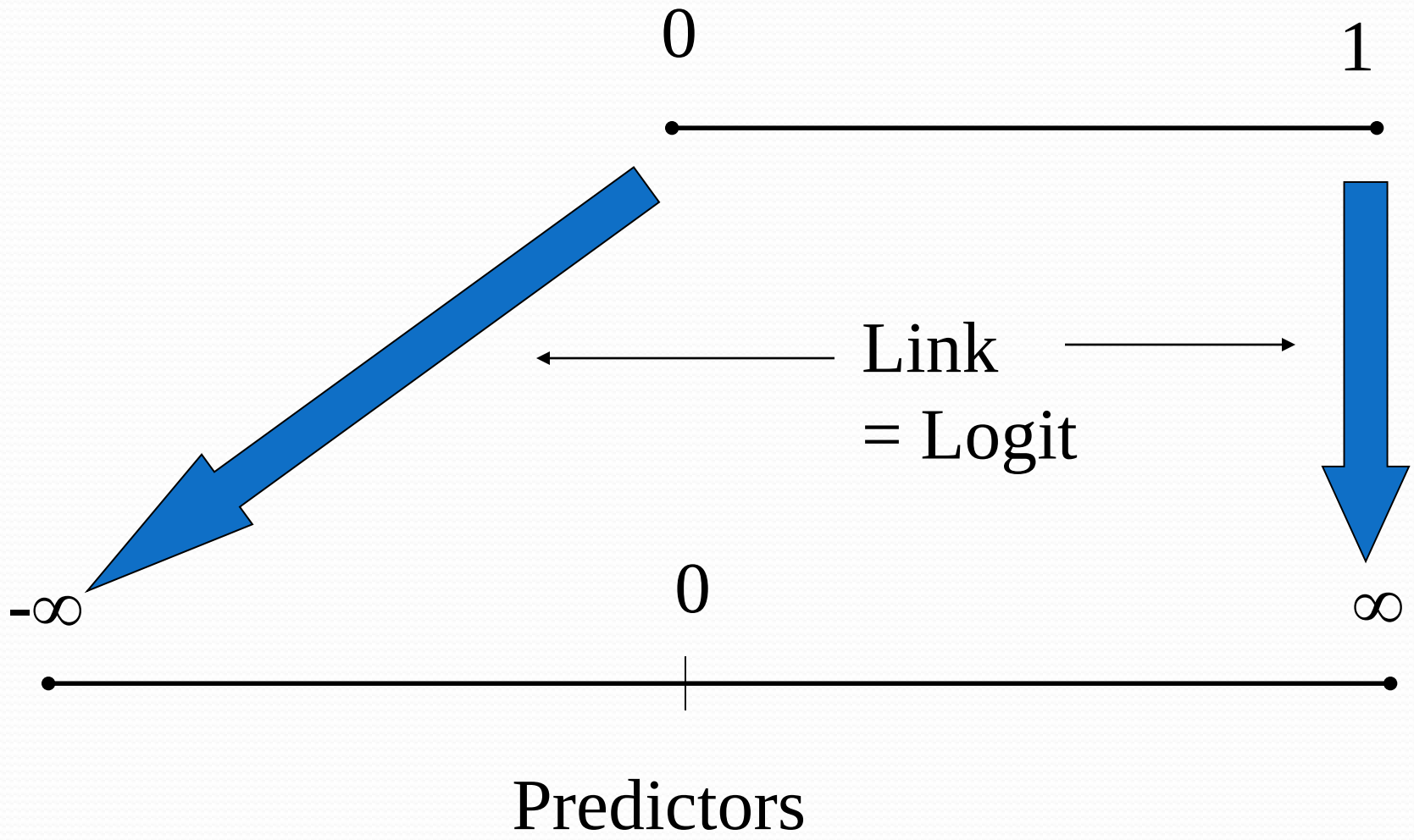


Logistic Regression

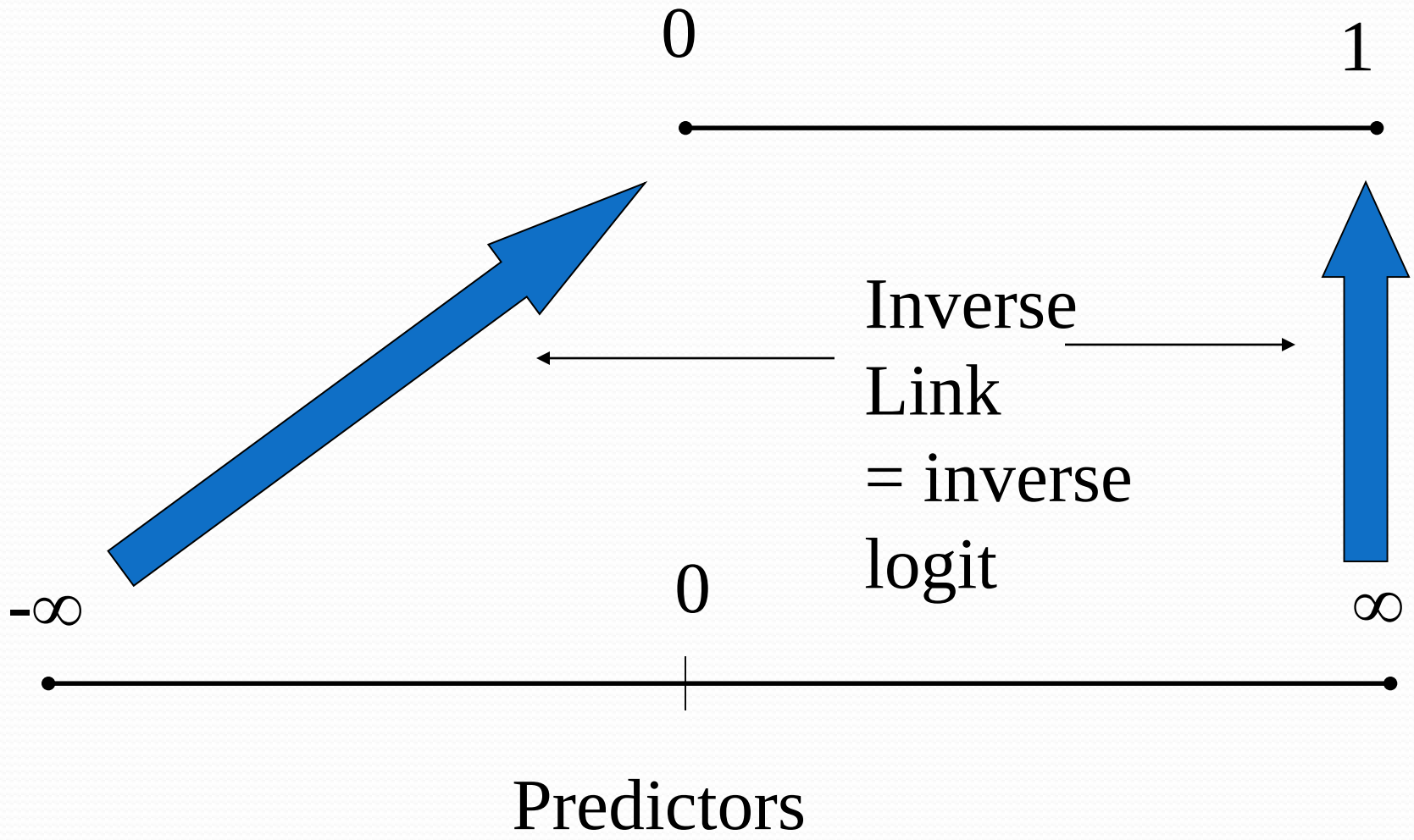
- Suppose we are trying to predict a binary variable (patient has ovarian cancer or not, patient is responding to therapy or not)
- We can describe this by a 0/1 variable in which the value 1 is used for one response (patient has ovarian cancer) and 0 for the other (patient does not have ovarian cancer
- We can then try to predict this response

- For a given patient, a prediction can be thought of as a kind of probability that the patient does have ovarian cancer. As such, the prediction should be between 0 and 1. Thus ordinary linear regression is not suitable
- The logit transform takes a number which can be anything, positive or negative, and produces a number between 0 and 1. Thus the logit link is useful for binary data

Possible Means



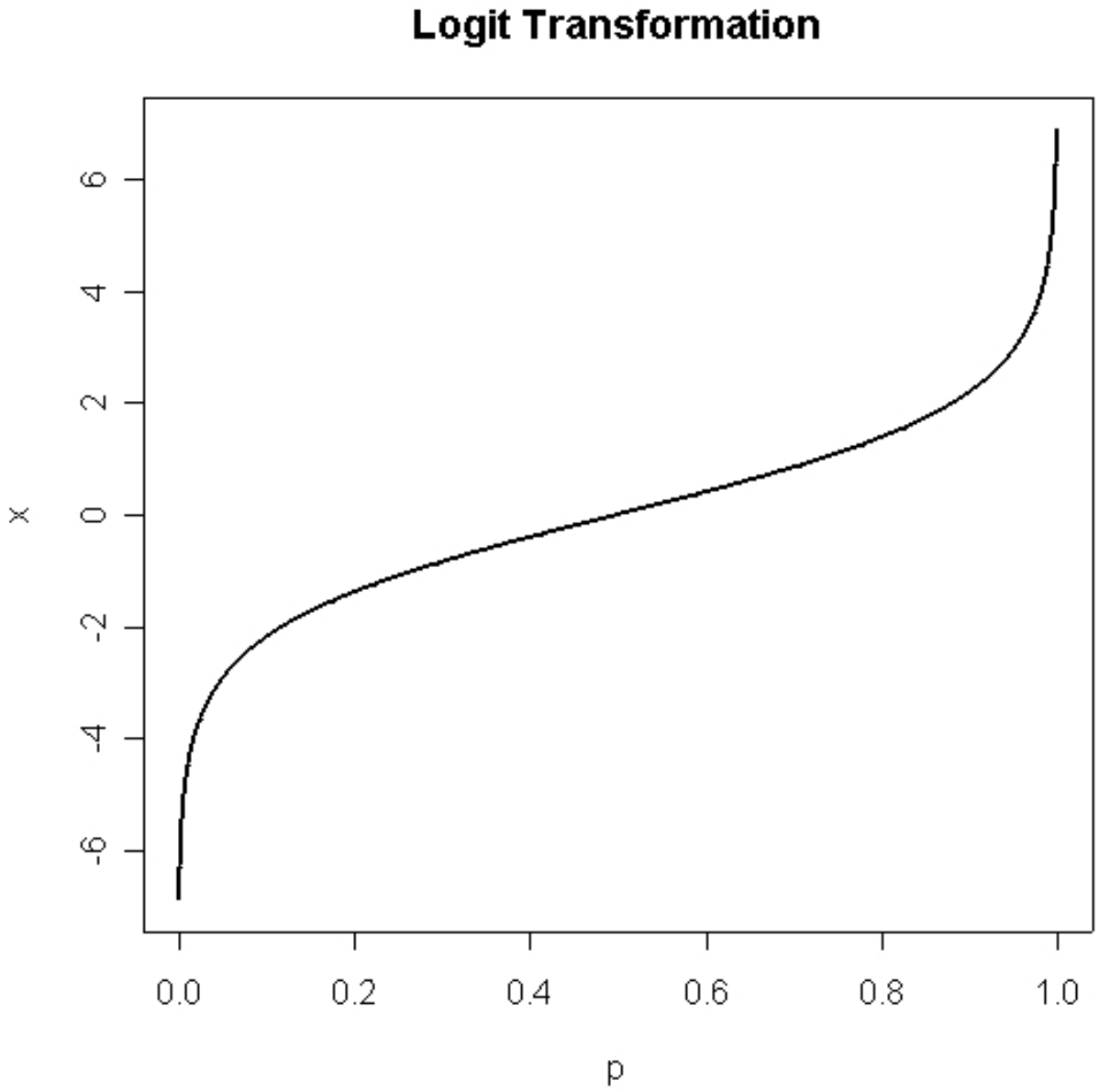
Possible Means

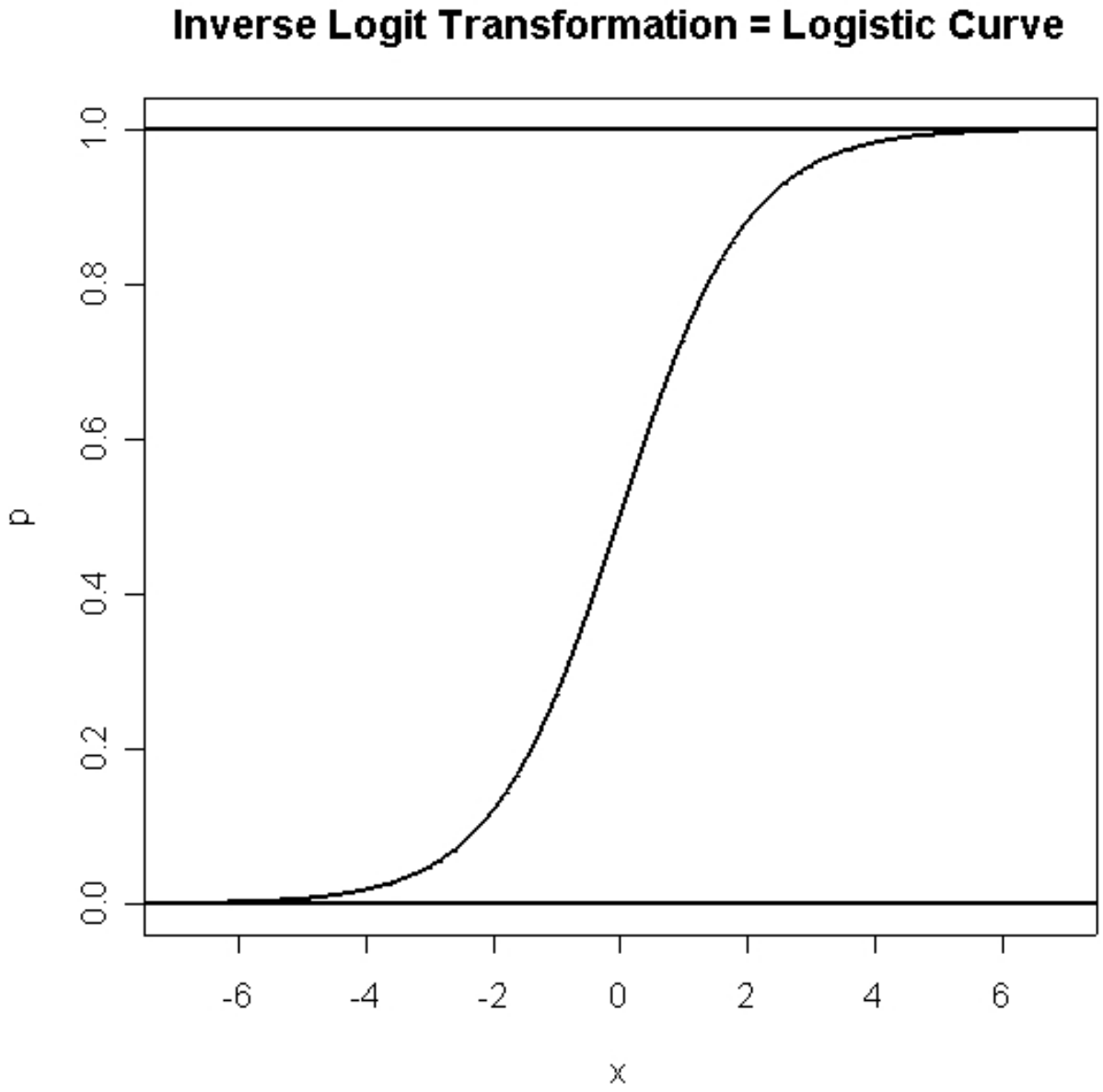


$$\text{logit}(p) = \log\left(\frac{p}{1-p}\right) \quad \text{if } p \to 0 \text{ then } \text{logit}(p) \to -\infty \quad \text{if } p \to 1 \text{ then } \text{logit}(p) \to \infty$$

$$\text{logit}^{-1}(x) = \frac{e^x}{1+e^x} \quad \text{if x} \to -\infty \text{ then } \text{logit}^{-1}(x) \to 0 \quad \text{if x} \to \infty \text{ then } \text{logit}^{-1}(x) \to 1$$

$$\log\left(\frac{\frac{e^x}{1+e^x}}{1-\frac{e^x}{1+e^x}}\right) = \log\left(\frac{\frac{e^x}{1+e^x}}{\frac{1+e^x-e^x}{1+e^x}}\right) = \log\left(\frac{\frac{e^x}{1+e^x}}{\frac{1}{1+e^x}}\right) = \log\left(e^x\right) = x$$


Logit Transformation
x
6
4
2
0
-2
-4
-6
0.0
0.2
0.4
0.6
0.8
1.0
p


Inverse Logit Transformation = Logistic Curve
p
0.0
0.2
0.4
0.6
0.8
1.0
-6
-4
-2
0
2
4
6
x

Analyzing Tabular Data with Logistic Regression

- Response is hypertensive y/n
- Predictors are smoking (y/n), obesity (y/n), snoring (y/n) [coded as 0/1 for Stata, R does not care]
- How well can these 3 factors explain/predict the presence of hypertension?
- Which are important?

```
no.yes <- c("No","Yes")
smoking <- gl(2,1,8,no.yes)
obesity <- gl(2,2,8,no.yes)
snoring <- gl(2,4,8,no.yes)
n.tot <- c(60,17,8,2,187,85,51,23)
n.hyp <- c(5,2,1,0,35,13,15,8)
hyp <- data.frame(smoking,obesity,snoring,n.tot,n.hyp,n.hyp/n.tot)
print(hyp)

  smoking obesity snoring n.tot n.hyp n.hyp.n.tot
1      No      No      No    60     5  0.08333333
2     Yes      No      No    17     2  0.11764706
3      No     Yes      No     8     1  0.12500000
4     Yes     Yes      No     2     0  0.00000000
5      No      No     Yes   187    35  0.18716578
6     Yes      No     Yes    85    13  0.15294118
7      No     Yes     Yes    51    15  0.29411765
8     Yes     Yes     Yes    23     8  0.34782609
```

Specifying Logistic Regressions in R

- For each 'cell', we need to specify the diseased and normals, which will be what we try to fit.
- This can be specified either as a matrix with one column consisting of the number of diseased persons, and the other the number of normals (not the total).
- Or we can specify the proportions as a response, with weights equal to the sample size

```
hyp.tbl <- cbind(n.hyp, n.tot-n.hyp)
print(hyp.tbl)
glm.hyp1 <- glm(hyp.tbl ~ smoking+obesity+snoring,family=binomial("logit"))
glm.hyp2 <- glm(hyp.tbl ~ smoking+obesity+snoring,binomial)
prop.hyp <- n.hyp/n.tot
glm.hyp2 <- glm(prop.hyp ~ smoking+obesity+snoring,binomial,weights=n.tot)
```

```
 n.hyp
[1,]     5  55
[2,]     2  15
[3,]     1   7
[4,]     0   2
[5,]    35 152
[6,]    13  72
[7,]    15  36
[8,]     8  15
```

```
> summary(glm.hyp1)

Call:
glm(formula = hyp.tbl ~ smoking + obesity + snoring, family = binomial("logit"))

Deviance Residuals:
       1         2         3         4         5         6         7         8
-0.04344   0.54145  -0.25476  -0.80051   0.19759  -0.46602  -0.21262   0.56231

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.37766    0.38018  -6.254    4e-10 ***
smokingYes  -0.06777    0.27812  -0.244   0.8075
obesityYes   0.69531    0.28509   2.439   0.0147 *
snoringYes   0.87194    0.39757   2.193   0.0283 *
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 14.1259  on 7  degrees of freedom
Residual deviance:  1.6184  on 4  degrees of freedom
AIC: 34.537

Number of Fisher Scoring iterations: 4
```

```
> anova(glm.hyp1,test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: hyp.tbl

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                        7    14.1259
smoking  1   0.0022         6    14.1237    0.9627
obesity  1   6.8274         5     7.2963    0.0090
snoring  1   5.6779         4     1.6184    0.0172
```

```
> predict(glm.hyp1)
         1          2          3          4          5          6          7
-2.3776615 -2.4454364 -1.6823519 -1.7501268 -1.5057221 -1.5734970 -0.8104126
         8
-0.8781874
> predict(glm.hyp1,type="response")
         1          2          3          4          5          6          7
0.08489206 0.07977292 0.15678429 0.14803121 0.18157364 0.17171843 0.30780259
         8
0.29355353
> rbind(predict(glm.hyp1,type="response"),prop.hyp)
                  1          2         3         4         5         6         7
         0.08489206 0.07977292 0.1567843 0.1480312 0.1815736 0.1717184 0.3078026
prop.hyp 0.08333333 0.11764706 0.1250000 0.0000000 0.1871658 0.1529412 0.2941176
                 8
         0.2935535
prop.hyp 0.3478261
> rbind(predict(glm.hyp1,type="response")*n.tot,n.hyp)
             1        2        3         4        5        6        7        8
      5.093524 1.356140 1.254274 0.2960624 33.95427 14.59607 15.69793 6.751731
n.hyp 5.000000 2.000000 1.000000 0.0000000 35.00000 13.00000 15.00000 8.000000
```

Logistic Regression with Raw Data

- Sometimes the data are in the form of individual cases with the covariates and resulting binary classification variable as a 0/1 variable or two-level factor. It is convenient not to have to tabulate
- Also, if any of the covariates is continuous, categorization is not possible without discretizing the variable

juul(ISwR) R Documentation

Juul's IGF data

Description

The juul data frame has 1339 rows and 6 columns. It contains a reference sample of the distribution of insulin-like growth factor (IGF-1), one observation per subject in various ages with the bulk of the data collected in connection with school physical examinations.

Format

This data frame contains the following columns:

```
age:       a numeric vector (years).
menarche: a numeric vector. Has menarche occurred (code 1: no, 2: yes)?
sex:       a numeric vector (1: boy, 2: girl).
igf1:      a numeric vector. Insulin-like growth factor ($µ$g/l).
tanner:    a numeric vector. Codes 1–5: Stages of puberty a.m. Tanner.
testvol:  a numeric vector. Testicular volume (ml).
```

Source

Original data.

```
> library(ISwR)
> data(juul)
> juul1 <- subset(juul,age > 8 & age < 20 & complete.cases(menarche))
```

Girls between 8 and 20.

```
> summary(juul1)
      age            menarche          sex          igf1            tanner
 Min.   : 8.03   Min.   :1.000   Min.   :2   Min.   : 95.0   Min.   : 1.000
 1st Qu.:10.62   1st Qu.:1.000   1st Qu.:2   1st Qu.:280.5   1st Qu.: 1.000
 Median :13.17   Median :2.000   Median :2   Median :409.0   Median : 4.000
 Mean   :13.44   Mean   :1.507   Mean   :2   Mean   :414.1   Mean   : 3.307
 3rd Qu.:16.48   3rd Qu.:2.000   3rd Qu.:2   3rd Qu.:514.0   3rd Qu.: 5.000
 Max.   :19.75   Max.   :2.000   Max.   :2   Max.   :914.0   Max.   : 5.000
                                             NA's   :108.0   NA's   :83.000
    testvol
 Min.   : NA
 1st Qu.: NA
 Median : NA
 Mean   :NaN
 3rd Qu.: NA
 Max.   : NA
 NA's   :519
```

```
> juul1$menarche <- factor(juul1$menarche,labels=c("No","Yes"))
> juul1$tanner <- factor(juul1$tanner)
> attach(juul1)
> summary(glm(menarche ~ age,binomial))

Call:
glm(formula = menarche ~ age, family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.32759  -0.18998   0.01253   0.12132   2.45922

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -20.0132     2.0284  -9.867   <2e-16 ***
age           1.5173     0.1544   9.829   <2e-16 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 719.39  on 518  degrees of freedom
Residual deviance: 200.66  on 517  degrees of freedom
AIC: 204.66

Number of Fisher Scoring iterations: 7
```

```
> summary(glm(menarche ~ age+tanner,binomial))

Call:
glm(formula = menarche ~ age + tanner, family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.56180  -0.12461   0.02475   0.08055   2.86120

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -13.7758     2.7630  -4.986 6.17e-07 ***
age           0.8603     0.2311   3.723 0.000197 ***
tanner2      -0.5211     1.4846  -0.351 0.725609
tanner3       0.8264     1.2377   0.668 0.504313
tanner4       2.5645     1.2172   2.107 0.035132 *
tanner5       5.1897     1.4140   3.670 0.000242 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 604.2  on 435  degrees of freedom
Residual deviance: 106.6  on 430  degrees of freedom
AIC: 118.6

Number of Fisher Scoring iterations: 8
```

```
> anova(glm(menarche ~ age+tanner,binomial),test="Chisq")
Analysis of Deviance Table

Model: binomial, link: logit

Response: menarche

Terms added sequentially (first to last)


        Df Deviance Resid. Df Resid. Dev P(>|Chi|)
NULL                      435     604.19
age      1   442.31       434     161.88 3.396e-98
tanner   4    55.28       430     106.60 2.835e-11

> drop1(glm(menarche ~ age+tanner,binomial),test="Chisq")
Single term deletions

Model:
menarche ~ age + tanner
       Df Deviance     AIC     LRT   Pr(Chi)
<none>     106.599 118.599
age     1  124.500 134.500  17.901 2.327e-05 ***
tanner  4  161.881 165.881  55.282 2.835e-11 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
```

Class prediction from gene expression

- One common use of omics data is to try to develop predictions for classes of patients, such as
  - cancer/normal
  - type of tumor
  - grading or staging of tumors
  - many other disease/healthy or diagnosis of disease type

Two-class prediction

- Linear regression
- Logistic regression
- Linear or quadratic discriminant analysis
- Partial least squares
- Fuzzy neural nets estimated by genetic algorithms and other buzzwords
- Many such methods require fewer variables than cases, so dimension reduction is needed

Dimension Reduction

- Suppose we have 20,000 variables and wish to predict whether a patient has ovarian cancer or not and suppose we have 50 cases and 50 controls
- We can only use a number of predictors much smaller than 50
- How do we do this?

- Two distinct ways are selection of genes and selection of “supergenes” as linear combinations
- We can choose the genes with the most significant t-tests or other individual gene criteria
- We can use forward stepwise logistic regression, which adds the most significant gene, then the most significant addition, and so on, or other ways of picking the best subset of genes

Supergenes are linear combinations of genes. If g_1, g_2, g_3, …, g_p are the expression measurements for the p genes in an array, and a_1, a_2, a_3, …, a_p are a set of coefficients then $g_1 a_1 + g_2 a_2 + g_3 a_3 + \ldots + g_p a_p$ is a supergene. Methods for construction of supergenes include PCA and PLS

Choosing Subsets of Genes

- Suppose we have 50 cases and 50 controls and an array of 20,000 gene expression values for each of the 100 observations
- In general, any arbitrary set of 100 genes will be able to predict perfectly in the data if a logistic regression is fit to the 100 genes
- Most of these will predict poorly in future samples

- This is a mathematical fact
- A statistical fact is that even if there is no association at all between any gene and the disease, often a few genes will produce apparently excellent results, that will not generalize at all
- We must somehow account for this, and cross validation is the usual way

```
y <- rep(0:1,each=50)
x <- matrix(rnorm(100*20000),ncol=100)
ts <- vector("numeric",20000)
for (i in 1:20000)
{
  ts[i] <- (t.test(x[i,] ~ y)$statistic)^2
}
ind <- order(ts,decreasing=T)

sp.glm <- glm(y ~ x[ind[1],],binomial)
print(summary(sp.glm))
yp <- predict.glm(sp.glm,type="response")
yp[yp < 0.5] <- 0
yp[yp >= 0.5] <- 1
print("Number of Misclassifications out of 100")
print(sum(y != yp))
```

```
> source("spuriousprediction2.r")

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.96156  -1.07483   0.08347   0.99583   1.68009

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -0.03078    0.22122  -0.139 0.889342
x[ind[1], ] -1.15034    0.30385  -3.786 0.000153 ***

    Null deviance: 138.63  on 99  degrees of freedom
Residual deviance: 119.00  on 98  degrees of freedom
AIC: 123.00

Number of Fisher Scoring iterations: 4

[1] "Number of Misclassifications out of 100"
[1] 36
```

```
[1] "Number of variables/Misclassifications out of 100"
[1]  1 36
[1] "Number of variables/Misclassifications out of 100"
[1]  2 32
[1] "Number of variables/Misclassifications out of 100"
[1]  3 27
[1] "Number of variables/Misclassifications out of 100"
[1]  4 19
[1] "Number of variables/Misclassifications out of 100"
[1]  5 17
[1] "Number of variables/Misclassifications out of 100"
[1]  6 21
[1] "Number of variables/Misclassifications out of 100"
[1]  7 16
[1] "Number of variables/Misclassifications out of 100"
[1] 20  0
Warning messages:
1: Algorithm did not converge in: glm.fit(x = X, y = Y,
  weights = weights, start = start, etastart = etastart,
2: fitted probabilities numerically 0 or 1 occurred in:
glm.fit(x = X, y = Y, weights = weights, start = start,
etastart = etastart,
```

Now with the first 20 variables instead of the 20/20000 with the Biggest t-scores:

```
[1] "Number of variables/Misclassifications out of 100"
[1] 20 26

Call:
glm(formula = y ~ x[1, ] + x[2, ] + x[3, ] + x[4, ] + x[5, ] +
    x[6, ] + x[7, ] + x[8, ] + x[9, ] + x[10, ] + x[11, ] + x[12,
    ] + x[13, ] + x[14, ] + x[15, ] + x[16, ] + x[17, ] + x[18,
    ] + x[19, ] + x[20, ], family = binomial)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.20702  -0.89041   0.01297   0.92103   1.90446
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  0.06041    0.24533   0.246   0.8055
x[1, ]      -0.43297    0.30242  -1.432   0.1522
x[2, ]       0.60087    0.28979   2.074   0.0381 *
x[3, ]       0.11777    0.23215   0.507   0.6119
x[4, ]       0.22212    0.24727   0.898   0.3690
x[5, ]      -0.15468    0.26043  -0.594   0.5526
x[6, ]       0.31370    0.24938   1.258   0.2084
x[7, ]      -0.43456    0.30462  -1.427   0.1537
x[8, ]      -0.41751    0.29113  -1.434   0.1515
x[9, ]      -0.45591    0.29228  -1.560   0.1188
x[10, ]      0.50699    0.28279   1.793   0.0730 .
x[11, ]     -0.54391    0.27250  -1.996   0.0459 *
x[12, ]      0.38480    0.26215   1.468   0.1422
x[13, ]     -0.04257    0.24281  -0.175   0.8608
x[14, ]      0.13996    0.25947   0.539   0.5896
x[15, ]      0.41957    0.23650   1.774   0.0761 .
x[16, ]     -0.20779    0.29312  -0.709   0.4784
x[17, ]      0.57632    0.30106   1.914   0.0556 .
x[18, ]      0.02833    0.27818   0.102   0.9189
x[19, ]      0.25862    0.25417   1.018   0.3089
x[20, ]      0.45244    0.23562   1.920   0.0548 .
```

```
Null deviance: 138.63  on 99  degrees of freedom
Residual deviance: 112.35  on 79  degrees of freedom
-----------------------------------------------------
(138.63 – 112.35) = 26.28 ~ chisq(20) p ~ .32
-----------------------------------------------------
```

	Df	Deviance	Resid. Df	Resid. Dev	P(>|Chi|)
NULL			99	138.629	
x[1,]	1	0.467	98	138.163	0.494
x[2,]	1	1.376	97	136.787	0.241
x[3,]	1	0.217	96	136.570	0.641
x[4,]	1	0.135	95	136.435	0.713
x[5,]	1	0.962	94	135.473	0.327
x[6,]	1	0.603	93	134.870	0.437
x[7,]	1	1.622	92	133.248	0.203
x[8,]	1	0.575	91	132.672	0.448
x[9,]	1	0.574	90	132.099	0.449
x[10,]	1	1.509	89	130.589	0.219
x[11,]	1	2.262	88	128.327	0.133
x[12,]	1	1.557	87	126.771	0.212
x[13,]	1	0.006	86	126.764	0.937
x[14,]	1	0.598	85	126.166	0.439
x[15,]	1	2.902	84	123.264	0.088
x[16,]	1	0.328	83	122.936	0.567
x[17,]	1	5.015	82	117.921	0.025
x[18,]	1	0.011	81	117.909	0.916
x[19,]	1	1.704	80	116.205	0.192
x[20,]	1	3.855	79	112.350	0.050

Consequences of many variables

- If there is no effect of any variable on the classification, it is still the case that the number of cases correctly classified increases in the sample that was used to derive the classifier as the number of variables increases
- But the statistical significance is usually not there

- If the variables used are selected from many, the apparent statistical significance and the apparent success in classification is greatly inflated, causing end-stage delusionary behavior in the investigator
- This problem can be improved using cross validation or other resampling methods

Overfitting

- When we fit a statistical model to data, we adjust the parameters so that the fit is as good as possible and the errors are as small as possible
- Once we have done so, the model may fit well, but we don't have an unbiased estimate of how well it fits if we use the same data to assess as to fit

Training and Test Data

- One way to approach this problem is to fit the model on one dataset (say half the data) and assess the fit on another
- This avoids bias but is inefficient, since we can only use perhaps half the data for fitting
- We can get more by doing this twice in which each half serves as the training set once and the test set once
- This is two-fold cross validation

- It may be more efficient to use 5- 10-, or 20-fold cross validation depending on the size of the data set
- Leave-out-one cross validation is also popular, especially with small data sets
- With 10-fold CV, one can divide the set into 10 parts, pick random subsets of size 1/10, or repeatedly divide the data

```
ind <- order(ts,decreasing=T)

n.tot <- 0
n.wrong <- 0

for (i in 1:100)
{
  test.set.list <- sample(100,10)
  test.seti <- rep(F,100)
  test.seti[test.set.list] <- T
  train.seti <- !test.seti
  y1 <- y[train.seti]
  x1 <- x[ind[1],train.seti]
  sp.glm <- glm( y1 ~ x1,binomial)
  yp <- predict.glm(sp.glm,data.frame(x1=x[ind[1],test.seti]),type="response")
  yp[yp < 0.5] <- 0
  yp[yp >= 0.5] <- 1
  n.tot <- n.tot+10
  n.wrong <- n.wrong+sum(y[test.seti] == yp)
}
```

```
print("Number of variables/Misclassifications out of 100")
print(c(1,n.wrong,n.tot,100*n.wrong/n.tot))

> source("spuriousprediction3.r")
[1] "Number of variables/Misclassifications out of 100"
[1]    1.0  637.0 1000.0   63.7
```

Cf. missclass within the 100 for this variable was 36
It should have been about 50 since the predictors are random
Cross validation does not solve the problem if the whole data
Set was used to find the variable(s)

Stepwise Logistic Regression

- Another way to select variables is stepwise
- This can be better than individual variable selection, which may choose many highly correlated predictors that are redundent
- A generic function step() can be used for many kinds of predictor functions in R

Using step()

- step(glm.model) is sufficient
- It uses steps either backward (using drop1) or forward (using add1) until a model is reached that cannot be improved
- Criterion is AIC = Akaiki Information Criterion, which tries to account for the effect of extra variables, more so than MSE or R^2

- You may also specify a scope in the form of a list(lower=model1, upper =model2)
- For expression arrays, with thousands of variables one should start with y ~ 1 and use scope =list(lower=y~1, upper=**)

```
for (i in 1:100)
{
  assign(paste("x",i,sep=""),x[ind[i],])
}
fchar <- "y~x1"
for (i in 2:100)
{
  fchar <- paste(fchar,"+x",i,sep="")
}
form <- as.formula(fchar)
step(glm(y ~ 1),list(lower=(y~1),upper=form))
```

assign creates a variable with a name and a value

paste makes a character string by pasting together parts

The first loop creates variables x1 to x100

The second loop creates a formula of the form

```
y~x1+x2+x3+…+x100
```

```
Step:  AIC= -288.12
 y ~ x29 + x13 + x60 + x17 + x47 + x3 + x50 + x30 + x26 + x16 +
    x78 + x9 + x37 + x89 + x52 + x6 + x46 + x75 + x83 + x62 +
    x28 + x14 + x98 + x22 + x8 + x56 + x81 + x53 + x65 + x5 +
    x23 + x27 + x44 + x99 + x90 + x92 + x93 + x71 + x70 + x40 +
    x10 + x77 + x20 + x15 + x4 + x33 + x61 + x25 + x68 + x35 +
    x67 + x55 + x96 + x19 + x87 + x39 + x42 + x64 + x100 + x94 +
    x18 + x63 + x2 + x11 + x86 + x7 + x12 + x57 + x24 + x80 +
    x31 + x32 + x21 + x51 + x49 + x72 + x58 + x41 + x69 + x36
```

Given that there is no variable here actually related to the Response, this cannot be said to have done very well. Partly The problem is that we started with the 100 accidentally highest t-scores

Poisson Distributions

- The Poisson distribution can be used to model unbounded count data, 0, 1, 2, 3, …
- An example would be the number of cases of sepsis in each hospital in a city in a given month.
- The Poisson distribution has a single parameter λ, which is the mean of the distribution and also the variance. The standard deviation is

$$\sqrt{\lambda}$$

Poisson Regression

- If the mean λ of the Poisson distribution depends on variables $x_1, x_2, \ldots, x_p$ then we can use a generalized linear model with Poisson distribution and log link.
- We have that $\log(\lambda)$ is a linear function of $x_1, x_2, \ldots, x_p$.
- This works pretty much like logistic regression, and is used for data in which the count has no specific upper limit (number of cases of lung cancer at a hospital) whereas logistic regression would be used when the count is the number out of a total (number of emergency room admissions positive for C. dificile out of the known total of admissions).

Lung cancer incidence in four Danish cities 1968-1971

This data set contains counts of incident lung cancer cases and population size in four neighbouring Danish cities by age group.

A data frame with 24 observations on the following 4 variables:

‘city’ a factor with levels ‘Fredericia’, ‘Horsens’, ‘Kolding’, and ‘Vejle’.
‘age’ a factor with levels ‘40-54’, ‘55-59’, ‘60-64’, ‘65-69’, ‘70-74’, and ‘75+’.
‘pop’ a numeric vector, number of inhabitants.
‘cases’ a numeric vector, number of lung cancer cases.

Details:

These data were “at the center of public interest in Denmark in 1974”, according to Erling Andersen's paper. The city of Fredericia has a substantial petrochemical industry in the harbour area.

```
> library(ISwR)
> data(eba1977)
> help(eba1977)
> dim(eba1977)
[1] 24  4
> eba1977
         city   age  pop cases
1  Fredericia 40-54 3059    11
2     Horsens 40-54 2879    13
3     Kolding 40-54 3142     4
4       Vejle 40-54 2520     5
5  Fredericia 55-59  800    11
..........
20      Vejle 70-74  539     8
21 Fredericia   75+  605    10
22    Horsens   75+  782     2
23    Kolding   75+  659    12
24      Vejle   75+  619     7
```

```
> attach(eba1977)
> eba.glm <- glm(cases ~
  city+age+offset(log(pop)),family=poisson)
> summary(eba.glm)

Call:
glm(formula = cases ~ city + age + offset(log(pop)),
  family = poisson)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.63573  -0.67296  -0.03436   0.37258   1.85267
```

Having offset(x) in a formula is like having x in the formula except the coefficient is fixed to 1. Having offset(log(pop)) in the formula, with the log link, makes the parameter lambda proportional to the population. A similar effect would come from analyzing the ratio of cases to population, but then we would not have count data.

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.6321     0.2003 -28.125  < 2e-16 ***
cityHorsens  -0.3301     0.1815  -1.818   0.0690 .
cityKolding  -0.3715     0.1878  -1.978   0.0479 *
cityVejle    -0.2723     0.1879  -1.450   0.1472
age55-59      1.1010     0.2483   4.434 9.23e-06 ***
age60-64      1.5186     0.2316   6.556 5.53e-11 ***
age65-69      1.7677     0.2294   7.704 1.31e-14 ***
age70-74      1.8569     0.2353   7.891 3.00e-15 ***
age75+        1.4197     0.2503   5.672 1.41e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for poisson family taken to be 1)

    Null deviance: 129.908  on 23  degrees of freedom
Residual deviance:  23.447  on 15  degrees of freedom
AIC: 137.84

Number of Fisher Scoring iterations: 5
```

$$\text{predictor}_{ij} = \text{city}_i + \log\left(\text{pop}_i\right) + \text{age}_j$$

$$\lambda_{ij} = \exp\left[\text{city}_i + \log\left(\text{pop}_i\right) + \text{age}_j\right]$$

$$= \exp\left[\text{city}_i\right]\exp\left[\text{age}_j\right]\text{pop}_i$$

Goodness of Fit

- If the model fits well, the residual deviance should be in the neighborhood of the df of the residual deviance.
- 23.447 on 15 df
- Under the null hypothesis that the model fits, and if the smallest fitted value is > 5, then the null distribution is approximately chi-squared

```
> min(fitted(eba.glm))
[1] 6.731286
> pchisq(deviance(eba.glm),
          df.residual(eba.glm),lower=F)
[1] 0.07509017
```

```
> drop1(eba.glm,test="Chisq")
Single term deletions

Model:
cases ~ city + age + offset(log(pop))
       Df Deviance    AIC     LRT Pr(Chi)
<none>      23.447 137.84
city    3   28.307 136.69   4.859  0.1824
age     5  126.515 230.90 103.068  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
  ’ 1
```

The test of the city effect would not be correct if we had individual patient data, since it then would be a characteristic of a group of patients, not of a patient. This would require a hierarchical model as in glmer() or Proc Mixed

```
> cf <- coef(summary(eba.glm))
> cf
              Estimate Std. Error    z value      Pr(>|z|)
(Intercept) -5.6320645  0.2002545 -28.124529 4.911333e-174
cityHorsens -0.3300600  0.1815033  -1.818479  6.899094e-02
cityKolding -0.3715462  0.1878063  -1.978348  4.788946e-02
cityVejle   -0.2723177  0.1878534  -1.449629  1.471620e-01
age55-59     1.1010140  0.2482858   4.434463  9.230223e-06
age60-64     1.5186123  0.2316376   6.555985  5.527587e-11
age65-69     1.7677062  0.2294395   7.704455  1.314030e-14
age70-74     1.8568633  0.2353230   7.890701  3.004950e-15
age75+       1.4196534  0.2502707   5.672472  1.407514e-08
```

```
> est <- cf[,1]
> se <- cf[,2]
> rr <- exp(cbind(est, est-se*qnorm(.975),
                      est+se*qnorm(.975)))
colnames(rr) <- c("RateRatio","LowerCL","UpperCL")
> rr
              RateRatio     LowerCL      UpperCL
(Intercept) 0.003581174 0.002418625  0.005302521
cityHorsens 0.718880610 0.503687146  1.026012546
cityKolding 0.689667168 0.477285856  0.996553318
cityVejle   0.761612264 0.527026991  1.100613918
age55-59    3.007213795 1.848515376  4.892215085
age60-64    4.565884929 2.899710957  7.189442499
age65-69    5.857402508 3.735990951  9.183417356
age70-74    6.403619032 4.037552548 10.156236043
age75+      4.135686847 2.532309969  6.754270176
```

These are rates per 4 person years.
The confidence intervals use an asymptotic approximation. A more accurate method in some cases is

```
> exp(cbind(coef(eba.glm),confint(eba.glm)))
Waiting for profiling to be done...
                             2.5 %       97.5 %
(Intercept) 0.003581174 0.002373629  0.005212346
cityHorsens 0.718880610 0.502694733  1.025912422
cityKolding 0.689667168 0.475568043  0.995045687
cityVejle   0.761612264 0.525131867  1.098950868
age55-59    3.007213795 1.842951851  4.901008833
age60-64    4.565884929 2.907180919  7.236296972
age65-69    5.857402508 3.748295295  9.248885425
age70-74    6.403619032 4.043044796 10.211923083
age75+      4.135686847 2.522891909  6.762422572
```

Breast cancer mortality

Danish study on the effect of screening for breast cancer.

Format:

A data frame with 24 observations on 4 variables.

‘age’ a factor with levels ‘50-54’, ‘55-59’, ‘60-64’, ‘65-69’, ‘70-74’, and ‘75-79’

‘cohort’ a factor with levels ‘Study gr.’, ‘Nat.ctr.’, ‘Hist.ctr.’, and ‘Hist.nat.ctr.’.

‘bc.deaths’ numeric, number of breast cancer deaths.

‘p.yr’ a numeric vector, person-years under study.

Details:

Four cohorts were collected. The "study group" consists of the population of women in the appropriate age range in Copenhagen and Frederiksberg after the introduction of routine mammography screening. The "national control group" consisted of the population in the parts of Denmark in which routine mammography screening was not available. These two groups were both collected in the years 1991-2001. The "historical control group" and the "historical national control group" are similar cohorts from 10 years earlier (1981-1991), before the introduction of screening in Copenhagen and Frederiksberg. The study group comprises the entire population, not just those accepting the invitation to be screened.

A.H. Olsen et al. (2005), Breast cancer mortality in Copenhagen after introduction of mammography screening. British Medical Journal, 330: 220-222.

Exercise

- In the bcmort data set, the four-level factor cohort can be considered the product of two two-level factors, say "period" (1981-1991 or 1991-2001) and "area" (Copenhagen/Fredriksberg and National). Generate those two factors.
- Fit a Poisson regression model to the data with age, period, and area as descriptors, as well as the three two-factor interaction terms. The interaction between period and area can be interpreted as the effect of screening (explain why).