Data-based Statistical Decision Model

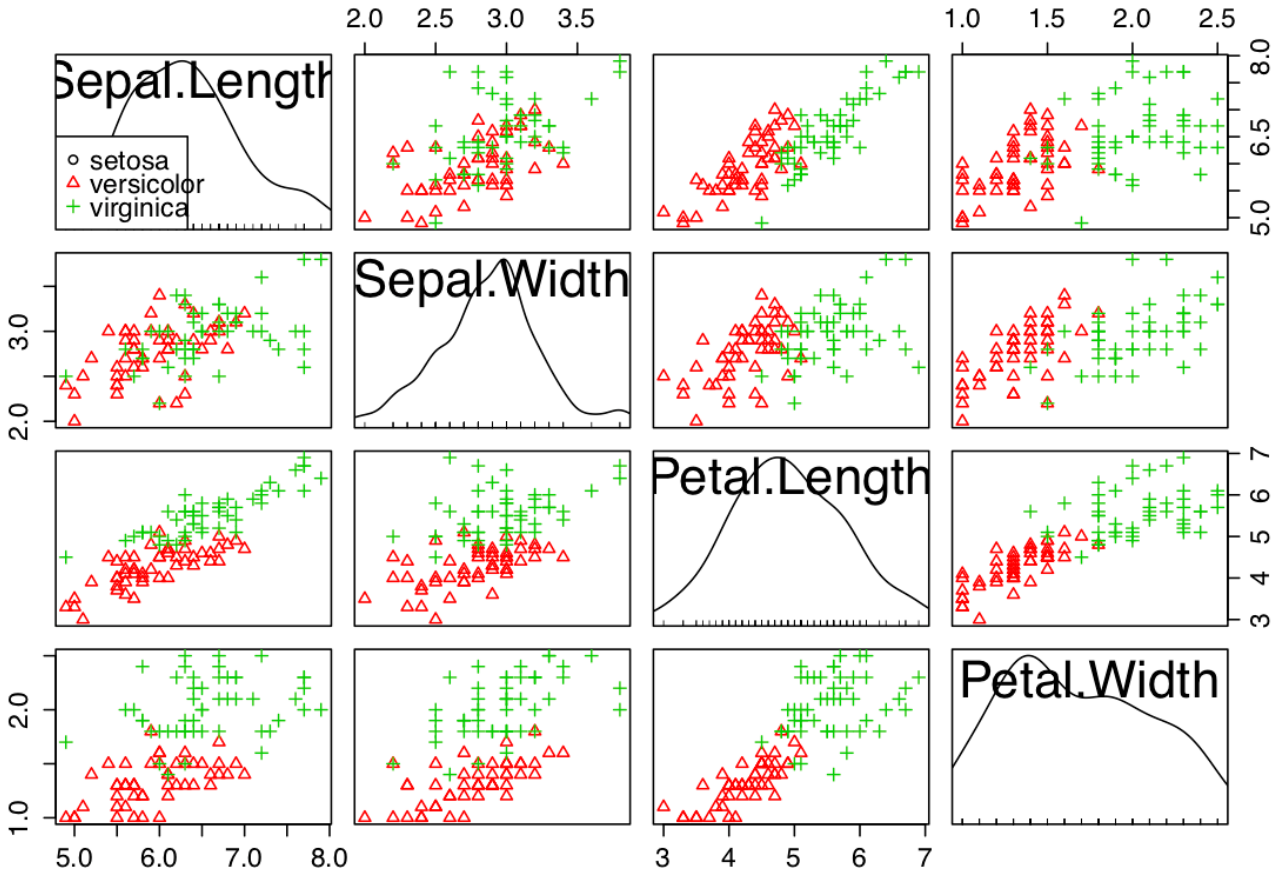
Lecture 9 - Predicting binary outcomes

Sungkyu Jung

Classification

- Classification, like regression, is a predictive task, but one in which the outcome takes only values across discrete categories; classification problems are very common (arguably just as or perhaps even more common than regression problems!)
- Examples:
    - Predicting whether a patient will develop breast cancer or remain healthy, given genetic information
    - Predicting whether or not a user will like a new product, based on user covariates and a history of his/her previous ratings
    - Predicting the region of Italy in which a brand of olive oil was made, based on its chemical composition
    - Predicting the next elected president, based on various social, political, and historical measurements
- Classification is fundamentally a different problem than regression, and so, we will need different tools. In this lecture we will learn one of the most common tools: *logistic regression*. You should know that there are many, many more methods beyond this one (just like there are many methods for estimating the regression function).

Predict Species!



Think about this:

- Regression: $Y = m(x) + \epsilon$, or $Y|X = x \sim N(m(x), \sigma^2)$.
- Classification: $X|Y = y \sim N(\mu_y, \Sigma_y)$.

Why not just use least squares?

- Before we present logistic regression, we address the (reasonable) question: why not just use least squares?
- Consider a classification problem in which we are given samples (x_i, y_i), $i = 1, \ldots, n$, and as usual x_i denotes predictor measurements, and y_i discrete outcomes. In this classification problem, we will assume that y_i can take two values, which we will write (without a loss of generality) as 0 and 1. Then, to predict a future outcome Y from predictor measurement X, we might consider performing linear regression. I.e., we fit coefficients β according to the familiar criterion

$$\min_{\beta} \sum_{i=1} (y_i - \beta' x_i)^2$$

and to predict the class of Y from X, we round $\hat{\beta}' X$ to whichever class is closest, 0 or 1.

- What is the problem with this, if any?

For purely predictive purposes, this actually is not a crazy idea—it tends to give decent predictions. But there are two drawbacks:

1. We cannot use any of the well-established routines for statistical inference with least squares (e.g., confidence intervals, etc.), because these are based on a model in which the outcome is continuously distributed.
2. We cannot use this method when the number of classes exceeds 2. If we were to simply code the response as $1, \ldots, K$ for a number of classes $K > 2$, then the ordering here would be arbitrary—but it actually matters.

Logistic regression

- A simple-minded understanding of logistic regression is to predict $y \in \{0, 1\}$ using $x_1, \ldots, x_p$ by a formula

$$y \sim \phi(x_1, \ldots, x_p) = \phi(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p) = \phi(\beta' X)$$

- Logistic regression starts with different model setup than linear regression: instead of modeling Y as a function of $X = (x_1, \ldots, x_p)$ directly, we model the probability that Y is equal to class 1, given X. First, abbreviate $p(X) = P(Y = 1|X)$. Then the logistic model is

$$p(X) = \frac{\exp(\beta' X)}{1 + \exp(\beta' X)}.$$

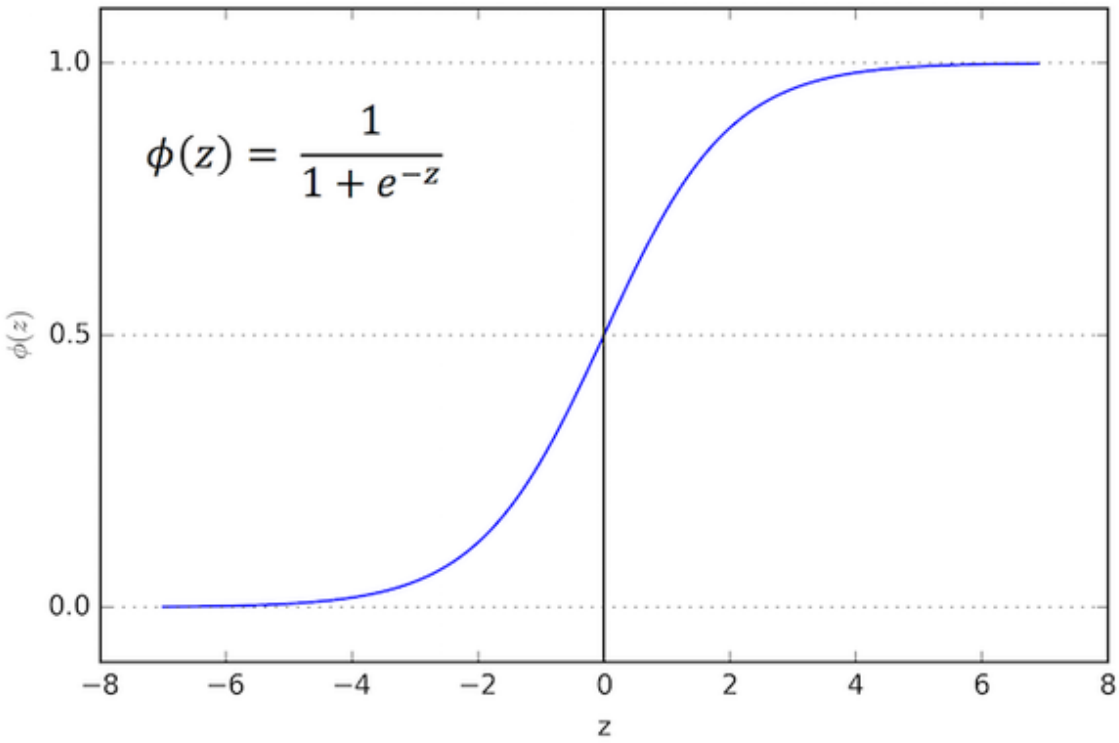
Logistic regression as a *generalized* linear regression

- "Linear": Explanatory variables to an intermediate predictor z:

$$(x_1, \ldots, x_p) \to z = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

- "Generalized": z to estimated probability p:

$$p = \phi(z)$$



Interpreting coefficients

- How can we interpret the role of the coefficients β?
- The logistic model is rearranged:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta' X$$

- The left-hand side $\log\left(\frac{p(X)}{1-p(X)}\right)$ is called *log-odds* of Class 1.
  - $p(X) = P(Y = 1 \mid X)$ = Probability of Class 1 (given X).
  - $p(X)/(1-p(X))$ = *Odds* of Class 1 (given X)

Interpreting odds

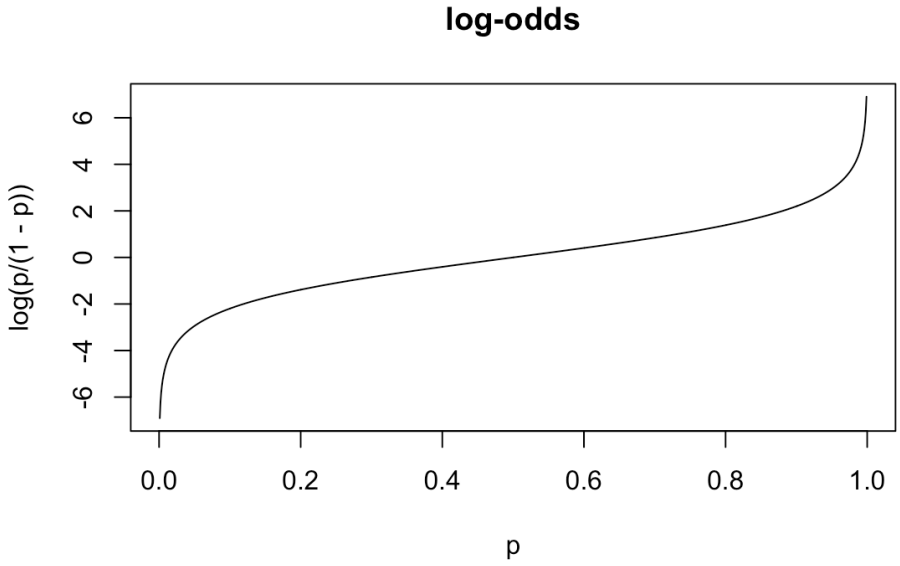
Odds are an alternative scale to probability for representing chance

- As a way to express the payoffs for bets
- *Evens* bet: Winner gets paid an equal amount to that staked [Odds = 1]
- 3-1 *against* bet: pay $3 for every $1 [Odds = 1/3]
- 3-1 *on* bet: pay $1 for every $3 [Odds = 3]
- if the games are fair, if you win with probability p, then you would make in the long run

$$E(payout) = (1-p)(-1) + p(\frac{1}{\text{Odds}}) = 0$$

- That is, Odds = $\frac{p}{1-p}$.

```
p = seq(0,1,length = 1000)
plot(p, log(p/(1-p)), type = "l", main = "log-odds")
```



- Can also write

$$\frac{p(X)}{1-p(X)} = \exp(\beta' X) = \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)$$

- Increasing X_j by one unit, while keeping all other predictors fixed, multiplies the *odds* by e^{β_j}.
- Increasing X_j by one unit, while keeping all other predictors fixed, changes the *log-odds* by β_j.

Model, with assumptions

Logistic regression model assumes:

1. The conditional mean of Y given X is of the form

$$P(Y=1|X) = p(X) = \frac{\exp(\beta' X)}{1+\exp(\beta' X)}.$$

2. The data $(Y_i, X_i) \in \{0,1\} \times \mathbb{R}^p$ are i.i.d.

That's it.

What happened to noises?

- Noise is already described by the model.
- Think about the type of noise you can add here. Since Y is either 0 or 1, the form of noise is quite restrictive; a noise can either make Y 0 or 1.
- Compare the following information (from the model)

$$P(Y=1|X) = p(X), \quad P(Y=0|X) = 1-p(X),$$

or $Y|X \sim Bernoulli(p(X))$ with

$$Y|X \sim N(m(X), \sigma^2),$$

from a regression model $Y = m(X) + \epsilon, \epsilon \sim N(0, \sigma^2)$.

Maximum likelihood estimation

Recall the binary distribution for 1 trial (i.e., Bernoulli distribution)

- For $Y \sim Bernoulli(p)$,

$$p(Y=y) = p^y (1-p)^{1-y}.$$

- For the ith observation (x_i, y_i),

$$p(Y_i = y_i | X_i = x_i) = p(x_i)^{y_i} (1-p(x_i))^{1-y_i}.$$

- For the data (x_i, y_i), $i = 1, \ldots, n$,

$$p(\{Y_i = y_i, i=1,\ldots,n\} | \{X_i = x_i, i=1,\ldots,n\}) = \prod_{i=1}^{n} p(x_i)^{y_i} (1-p(x_i))^{1-y_i}.$$

- The likelihood $L(\beta)$ is the joint mass function $p(\{Y_i = y_i, i=1,\ldots,n\} | \{X_i = x_i, i=1,\ldots,n\})$ as a function of β

- Recall $p(X) = \frac{\exp(\beta' X)}{1+\exp(\beta' X)}$.

- Maximizing the likelihood is the same as maximizing the log-likelihood

$$\ell(\beta) = \frac{1}{n}\sum_{i=1}^{n} y_i \cdot (\beta' x_i) - \log(1 + e^{\beta' x_i}).$$

- There is no closed-form formula for $\hat{\beta} = \text{argmax}\ell(\beta)$, but we can maximize it by running repeated weighted least squares regressions!

Inference

- A lot of the standard machinery for inference in linear regression carries over to logistic regression. Recall that we can solve for the logistic regression coefficients $\hat{\beta}$ by performing repeated weighted linear regressions; hence we can simply think of the logistic regession estimates as the result of a single weighted linear regression - the last one in this sequence (upon convergence). Confidence intervals for β_j and so forth, are then all obtained from this weighted linear regression perspective. We will not go into detail here, but such inferential tools are implemented in software, and it helps to be aware of where they come from.
- We will see some options shortly.

R Example: Logistic regression

- Use `glm` (R core function) and `glmnet` package to numerically compute the fitted coefficients (βs) of the logistic regression.
- `model.matrix()` transforms the categorical variables in the `data` into appropriate numeric variables (by creating *dummy* variables if needed).
- Logistic regression is a special case of Generalized Linear Models, where the response is binary. To indicate this, in the argument of `glmnet()` and `glm()`, we specify `family = "binomial"`.

Data Example: The Obama-Clinton divide

- The following graphic, published in New York Times, by Amada Cox, in 2008 gained popularity for its beautiful use of graphics in displaying a statistical analysis.

> http://www.nytimes.com/imagepages/2008/04/16/us/20080416_OBAMA_GRAPHIC.html (http://www.nytimes.com/imagepages/2008/04/16/us/20080416_OBAMA_GRAPHIC.html)

- We see an entire story between Obama and Clinton - positions taken, counties won, and counties lost.
- This is an example of classification: For each given county, using measurements of the county (e.g. %black population, %high-school grad, geographic location, etc), you can predict whether Clinton wins or not.

The data: 2008 Democratic primaries

- Read the data, and divide the data into training and testing set.

```
primary <- read.csv(url("http://www.stat.pitt.edu/sungkyu/course/pds/material/primaries.csv"),head=T)
primary <- as_tibble(primary) %>%
  mutate(AFprop = black06 / pop06 * 100) %>% filter(complete.cases(.))

set.seed(1)
train <- primary %>% sample_frac(size = 0.8)
test <- primary %>% setdiff(train)
```

- See if the response is dichotomous:

```
str(primary$winner)
```

```
##  Factor w/ 2 levels "clinton","obama": 2 1 2 1 1 2 2 1 2 1 ...
```

Fitting an ordinary logistic regression using `glm()`

```
form <- as.formula(
  "winner ~ AFprop + region + pres04winner")
fit0 <- glm(form, data = train, family = "binomial")
summary(fit0)
```

```
##
## Call:
## glm(formula = form, family = "binomial", data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8945  -0.9151  -0.4792   0.9714   2.1722
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -0.667292   0.098259  -6.791 1.11e-11 ***
## AFprop             0.107875   0.006577  16.402  < 2e-16 ***
## regionNE          -1.098688   0.236502  -4.646 3.39e-06 ***
## regionS           -1.618590   0.151599 -10.677  < 2e-16 ***
## regionW            1.124208   0.162157   6.933 4.12e-12 ***
## pres04winnerkerry  0.369703   0.147418   2.508   0.0121 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2635.9  on 1925  degrees of freedom
## Residual deviance: 2047.9  on 1920  degrees of freedom
## AIC: 2059.9
##
## Number of Fisher Scoring iterations: 5
```

- Interpretation of the last coefficient: All others fixed, the odds for Clinton increases by $e^{\beta_5} = 1.447305$ times.

AIC

- AIC is defined in the same way as in linear regression:

$$AIC = -2\ell + 2p.$$

- It's an important indicator of model fit. It follows the rule: Smaller the better.

Residuals and diagnostics

- After a model has been fit, it is wise to check the model to see how well it fits the data
- In linear regression, these diagnostics were build around residuals and the residual sum of squares
- In logistic regression (and all generalized linear models), there are a few different kinds of residuals (and thus, different equivalents to the residual sum of squares)
- Raw residual = $y_i - \hat{y}_i$, where y_i is either 0 or 1 and $\hat{y}_i = \hat{P}(Y = 1|X = x_i)$.

Pearson residuals

The first kind is called the Pearson residual, and is based on the idea of subtracting off the mean and dividing by the standard deviation

- For a logistic regression model,

$$r_i = \frac{y_i - \hat{y}_i}{\hat{y}_i(1 - \hat{y}_i)}.$$

- Note that if we replace $\hat{y}_i = \hat{P}(Y = 1|X = x_i)$ with $P(Y = 1|X = x_i)$, then r_i has mean 0 and variance 1.

Deviance residuals

The other approach is based on the contribution of each point to the likelihood.

- For linear regression,

$$\ell \approx \sum_{i=1}^{n} -\frac{(y_i - \hat{y_i})^2}{2\hat{\sigma}^2}$$

  For each observation, the standarized residual is sign$(y_i - \hat{y}_i)$

- For logistic regression,

$$\begin{aligned}\ell &= \frac{1}{n}\sum_{i=1}^{n} y_i \cdot (\beta' x_i) - \log(1 + e^{\beta' x_i}) \\ &= \frac{1}{n}\sum_{i=1}^{n} y_i \log \hat{y}_i + (1 - y_i)\log(1 - \log \hat{y}_i)\end{aligned}$$

By analogy with linear regression, the deviance residual:

$$d_i = s_i\sqrt{-2(y_i \log \hat{y}_i + (1 - y_i)\log(1 - \log \hat{y}_i))}$$

where $s_i = 1$ if $y_i = 1$ and $s_i = -1$ if $y_i = 0$.

Deviance and Pearson's statistic

- Each of these types of residuals can be squared and added together to create an RSS-like statistic
- Combining the deviance residuals produces the deviance:

$$D = \sum d_i^2 = -2\ell.$$

- Combining the Pearson residuals produces the Pearson statistic:

$$X^2 = \sum_{i=1}^{n} r_i^2.$$

- They both should follow χ^2 distribution (if the model is correct).

Hat matrix, leverage, Cooks' distance

- As you may recall, in linear regression it was important to divide the residual by $\sqrt{1-h_{ii}}$ to account for the leverage that a point had over its own fit
- Similar steps can be taken for logistic regression; here, the projection matrix is

$$\mathbf{H} = \mathbf{W}^{1/2}\mathbf{X}(\mathbf{X}'\mathbf{W}\mathbf{X})^{-1}\mathbf{X}\mathbf{W}^{1/2},$$

  where $\mathbf{W}$ is a diagonal weight matrix "produced" in estimation

---

- In logistic regression, $\hat{\mathbf{y}} \neq \mathbf{H}\mathbf{y}$ no matrix can satisfy this requirement, as logistic regression does not produce linear estimates
- However, the diagonal elelments of $\mathbf{H}$, h_{ii}, are called *leverages*, and can be used to standardize the residuals:

$$r_{si} = r_i/\sqrt{1-h_{ii}}$$

$$d_{si} = d_i/\sqrt{1-h_{ii}}$$

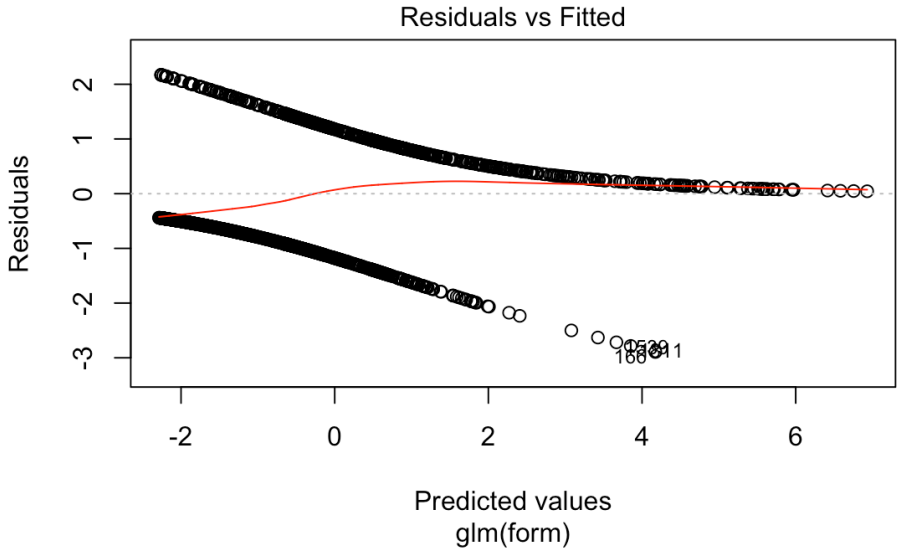
Leave-one-out diagnostics

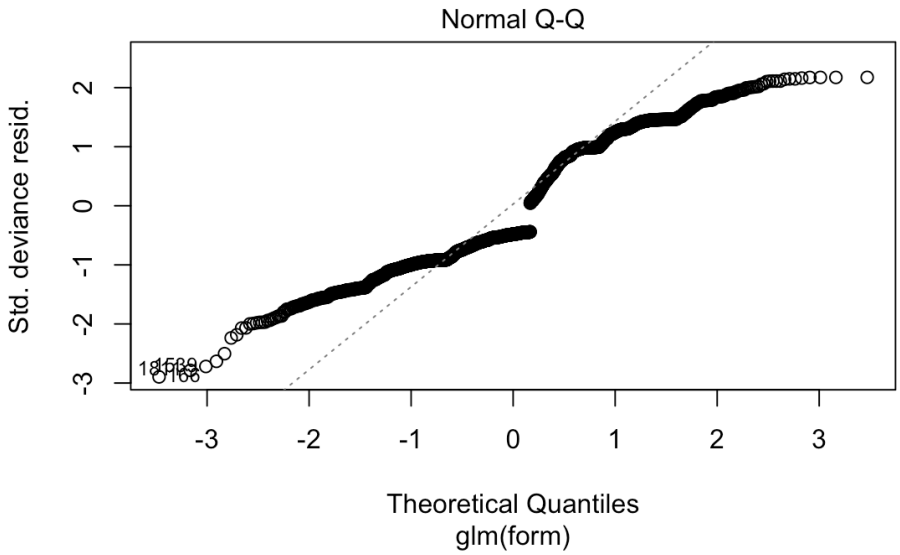
- You may recall that in linear regression there were a number of diagnostic measures based on the idea of leaving observation i out, refitting the model, and seeing how various things changed (residuals, coefficient estimates, fitted values)
- You may also recall that for linear regression, explicit shortcuts based on $\mathbf{H}$ were available.
- The same idea can be extended to generalized linear models, although we cannot take advantage of the explicit-solution shortcuts without making approximations.
- One-step approximations allow us to quickly calculate the following diagnostic statistics for GLMs:
  - Studentized deleted residuals (Jackknife deviance residual)
  - Cook's distance (for assessing overall influence over the model fit)

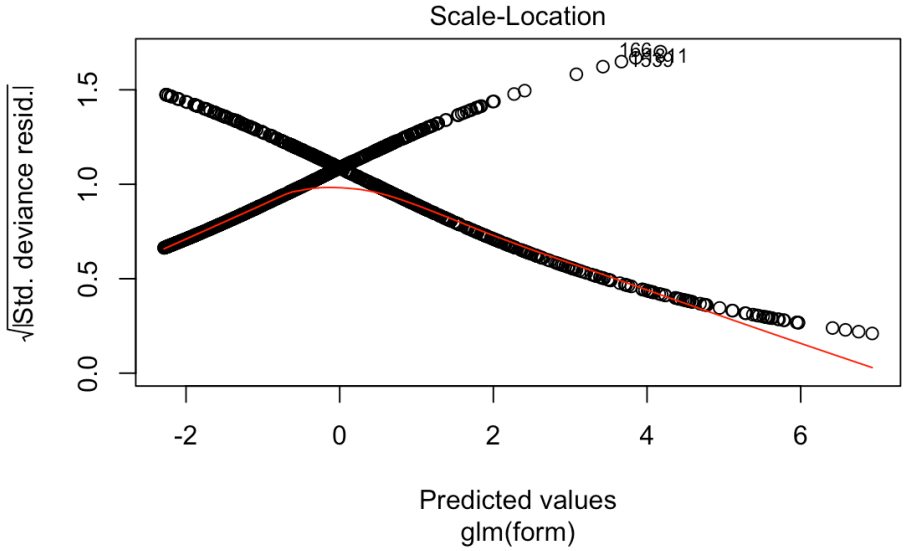
Model checking in R

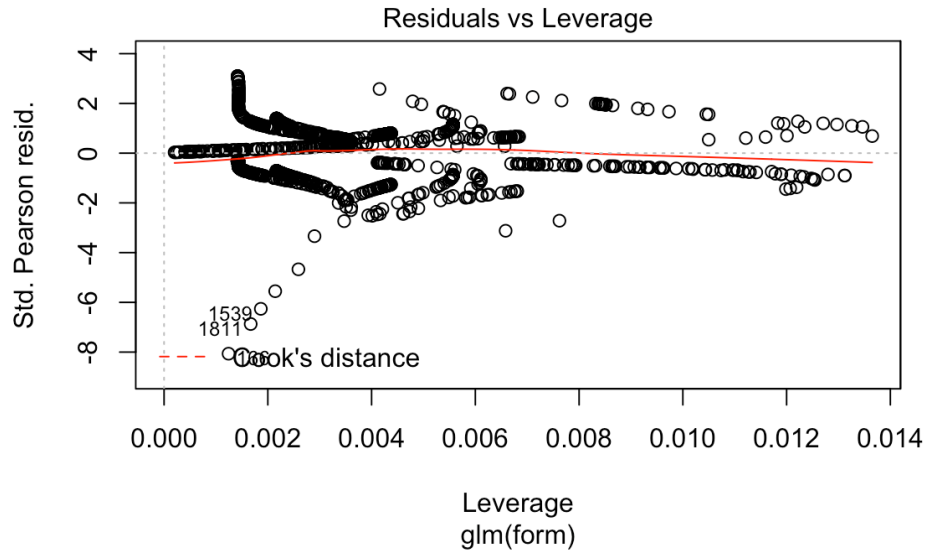
Using `glm.diag()` in package `boot`,

```
library(boot)
fit0.diag <- glm.diag(fit0)

# fit0.diag$res The vector of jackknife deviance residuals.
# fit0.diag$rd  The vector of standardized deviance residuals.
# fit0.diag$rp  The vector of standardized Pearson residuals.
# fit0.diag$cook    The vector of approximate Cook statistics.
# fit0.diag$h   The vector of leverages of the observations.

plot(fit0)
```









Model checking summary

- Residuals are certainly less informative for logistic regression than they are for linear regression: not only do yes/no outcomes inherently contain less information than continuous ones, but the fact that the adjusted response depends on the fit hampers our ability to use residuals as external checks on the model
- This is mitigated to some extent, however, by the fact that we are also making fewer distributional assumptions in logistic regression, so there is no need to inspect residuals for, say, skewness or heteroskedasticity
- Nevertheless, issues of outliers and influential observations are just as relevant for logistic regression as they are for linear regression
- If influential observations are present, it may or may not be appropriate to change the model, but you should at least understand why some observations are so influential

Null Deviance and Residual Deviance

- Deviance of an observation is computed as -2 times log likelihood of that observation. The importance of deviance can be further understood using its types: Null and Residual Deviance. Null deviance is calculated from the model with no features, i.e., only intercept. The null model predicts class via a constant probability.
- Residual deviance is calculated from the model having all the features. Compared to Linear Regression, think of residual deviance as residual sum of square (RSS) and null deviance as total sum of squares (TSS). The larger the difference between null and residual deviance, better the model.
- Residuals are rather used for “Analysis of Deviances” similar to ANOVA in linear regression:

```
anova(fit0)
```

```
## Analysis of Deviance Table
##
## Model: binomial, link: logit
##
## Response: winner
##
## Terms added sequentially (first to last)
##
##
##              Df Deviance Resid. Df Resid. Dev
## NULL                          1925     2635.9
## AFprop        1   276.17      1924     2359.7
## region        3   305.54      1921     2054.2
## pres04winner  1     6.30      1920     2047.9
```

Inference

- Standard inferential tools

```
confint(fit0)
```

```
## Waiting for profiling to be done...
```

```
##                         2.5 %     97.5 %
## (Intercept)       -0.86127574 -0.4758627
## AFprop             0.09533033  0.1211265
## regionNE          -1.57218602 -0.6433359
## regionS           -1.91916474 -1.3245300
## regionW            0.80885533  1.4449721
## pres04winnerkerry  0.08101960  0.6592810
```

- Backward variable selection

```
step(fit0, direction = "backward")
```

```
## Start:  AIC=2059.86
## winner ~ AFprop + region + pres04winner
##
##                Df Deviance    AIC
## <none>              2047.9 2059.9
## - pres04winner  1   2054.2 2064.2
## - region        3   2334.3 2340.3
## - AFprop        1   2493.2 2503.2
```

```
##
## Call:  glm(formula = winner ~ AFprop + region + pres04winner, family = "binomial",
##     data = train)
##
## Coefficients:
##       (Intercept)             AFprop           regionNE
##           -0.6673             0.1079             -1.0987
##           regionS            regionW  pres04winnerkerry
##           -1.6186             1.1242              0.3697
##
## Degrees of Freedom: 1925 Total (i.e. Null);  1920 Residual
## Null Deviance:       2636
## Residual Deviance: 2048  AIC: 2060
```

- Prediction and confidence interval for $p(X)$

```
predict(fit0, newdata = test[1,], type = "link", se.fit=TRUE) # Predict "log-odds" of p(X)
```

```
## $fit
##         1
## -1.696679
##
## $se.fit
## [1] 0.1049312
##
## $residual.scale
## [1] 1
```

```
predict(fit0, newdata = test[1,], type = "response", se.fit=TRUE) # Predict p(X)
```

```
## $fit
##         1
## 0.1548995
##
## $se.fit
##          1
## 0.01373609
##
## $residual.scale
## [1] 1
```

Fitting an ordinary logistic regression using `glmnet()`

```
library(glmnet)
predictors <- model.matrix(form, data = train)
fit1 <- glmnet(predictors, train$winner,
               family = "binomial",
               lambda = 0)
fit1$beta
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                            s0
## (Intercept)        .
## AFprop             0.1078684
## regionNE          -1.0985967
## regionS           -1.6185119
## regionW            1.1242079
## pres04winnerkerry  0.3697000
```

In the print-out above, ignore the first two lines. They are related to the numerical problem of efficiently handling the fitted values.

- We will discuss the argument `lambda = 0` shortly. Ignore for now.

Evaluation of the logistic regression models

1. Compare AIC
2. Compare residual deviance
3. Misclassification Error (viewed as a classification problem)
    - Our ordinary logistic model gives the testing misclassification rate of 0.2702703.

Regularized logistic regression

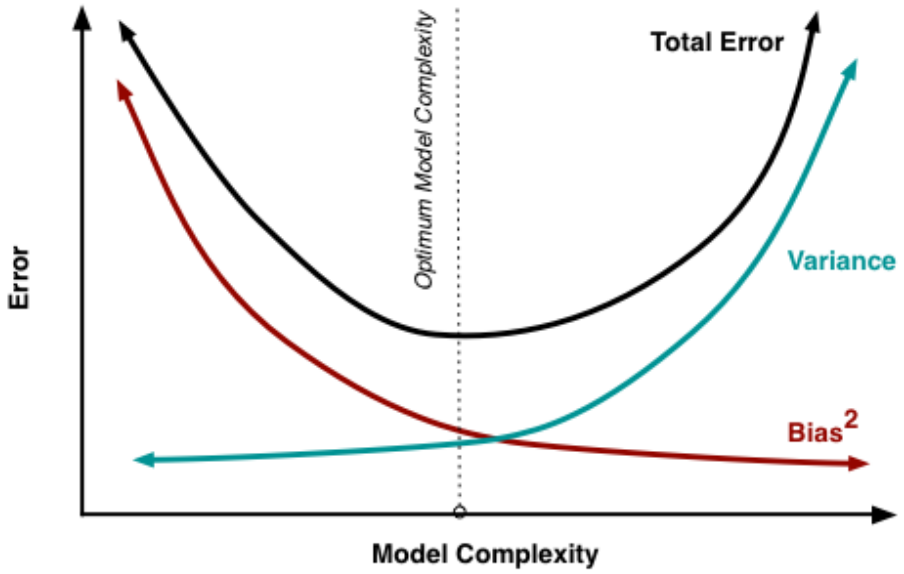
A regularized logistic regression finds the values of β's by minimizing the following:

$$\text{Variance} + \lambda \text{Bias}$$

This is often referred to as “Loss + Penalty”. The latter is correct, and the former is a bit misleading, but let us focus on the idea for now.

Variance–Bias trade-off

- Ordinary logistic regression minimizes the “variance”
- Regularized logistic regression introduced a “bias” in hope for minimizing the total error



Elastic Net logistic regression

- the “Variance” is in fact a *negative goodness-of-fit measure*:

$$GF(\beta) = -\left[\frac{1}{n}\sum_{i=1}^{n} y_i \cdot (\beta' x_i) - \log(1 + e^{\beta' x_i})\right]$$

- the “Bias” is in fact the size of β:

$$Size(\beta) = \left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1\right]$$

Elastic Net logistic regression amounts to the optimization of the form:

$$\min_{(\beta_0,\beta)\in\mathbb{R}^{p+1}} -GF(\beta) + \lambda Size(\beta)$$

What is λ?

- In the regularized regression, λ is called a **tuning parameter**:

> That is, the result of analysis is tunable by changing the value of λ.

- Large λ forces to set many β equal to 0:

```
fit2 <- glmnet(predictors, train$winner,
               family = "binomial", lambda = 0.05) # lambda > 0
fit2$beta
```

```
## 6 x 1 sparse Matrix of class "dgCMatrix"
##                             s0
## (Intercept)          .
## AFprop               0.05034915
## regionNE             .
## regionS             -0.42272871
## regionW              0.73860634
## pres04winnerkerry    .
```

Classification by regularized logistic regression

- The question of which variables to model is now replaced by how to choose a value of λ?

lambda	Testing Misclassification Rate
0 (Ordinary logistic regression)	0.2702703
0.05 (Regularized logistic regression)	0.2619543

Model evaluation: Cross-validation

- So far, we set aside 80% of the observations to use as a training set, but held back another 20% for testing.
- Another approach is cross-validation.
- The package `glmnet` provides a handy cross-validation function `cv.glmnet()`.
- In the argument of `cv.glmnet()`, the option `type = "class"` is used to specify that the misclassification error is used as a measure of comparison.

```
predictors <-
  model.matrix(form, data = primary) # use whole data
cv.fit <-
  cv.glmnet(predictors, primary$winner,
                    family = "binomial", type = "class")
```

```
plot(cv.fit)
```