# Discrete Response Model
# Lecture 2

datascience@berkeley

# An Example

# Example

Goal: Estimate the probability of success for a placekick (based on 1,425 placekicks from the 1995 NFL season.

The response variable is referred to as "Good" in the dataset. It is a 1 for successful placekicks and a 0 for failed placekicks.

Explanatory variables in the dataset:

- Week: week of the season
- Distance: Distance of the placekick in yards
- Change: Binary variable denoting lead-change (1) vs. non-lead-change (0) placekicks; successful lead-change placekicks are those that change which team is winning the game.
- Elap30: Number of minutes remaining before the end of the half with overtime placekicks receiving a value of 0
- PAT: Binary variable denoting the type of placekick where a point after touchdown (PAT) is a 1 and a field goal is a 0
- Type: Binary variable denoting dome (0) vs. outdoor (1) placekicks
- Field: Binary variable denoting grass (1) vs. artificial turf (0) placekicks
- Wind: Binary variable for placekicks attempted in windy conditions (1) vs. non-windy conditions (0); I define windy as a wind stronger than 15 miles per hour at kickoff in an outdoor stadium

# Example

There are 1,425 placekick observations from the 1995 NFL season that are within this dataset.

```
setwd("/Users/jeffrey/Documents/JStuff/AdvStat/pgms/CatData/Chapter2")
list.files("/Users/jeffrey/Documents/JStuff/AdvStat/pgms/CatData/Chapter2")

df<-read.table(file = "placekick.csv", header = TRUE, sep = ",")
str(df)
```

```
'data.frame':	1425 obs. of  9 variables:
 $ week    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ distance: int  21 21 20 28 20 25 20 27 44 32 ...
 $ change  : int  1 0 0 0 0 0 0 1 1 0 ...
 $ elap30  : num  24.72 15.85 0.45 13.55 21.87 ...
 $ PAT     : int  0 0 1 0 1 0 1 0 0 0 ...
 $ type    : int  1 1 1 1 0 0 0 0 0 0 ...
 $ field   : int  1 1 1 1 0 0 0 0 0 0 ...
 $ wind    : int  0 0 0 0 0 0 0 0 0 0 ...
 $ good    : int  1 1 1 1 1 1 1 1 1 1 ...
```

```
> head(df)
  week distance change  elap30 PAT type field wind good
1    1       21      1 24.7167   0    1     1    0    1
2    1       21      0 15.8500   0    1     1    0    1
3    1       20      0  0.4500   1    1     1    0    1
4    1       28      0 13.5500   0    1     1    0    1
5    1       20      0 21.8667   1    0     0    0    1
6    1       25      0 17.6833   0    0     0    0    1
```

Frequency table of the dependent variable of interest:

```
> table(df$good)

   0    1
 163 1262
> prop.table(table(df$good))

       0        1
0.114386 0.885614
```

# Example

For this particular example, we are only going to use the distance explanatory variable to estimate the probability of a successful placekick. Thus, our logistic regression model is

$$\boxed{logit(\pi) = \beta_0 + \beta_1 x_1}$$

where $Y$ is the good response variable and $x_1$ denotes the for the ~~placekick.~~

```
> mod.fit<-glm(formula = good ~ distance, family =binomial(link = logit), data = df)
> mod.fit

Call:  glm(formula = good ~ distance, family = binomial(link = logit),
    data = df)

Coefficients:
(Intercept)     distance
      5.812       -0.115

Degrees of Freedom: 1424 Total (i.e. Null);  1423 Residual
Null Deviance:      1013
Residual Deviance: 775.7        AIC: 779.7
```

The estimated logistic regression model is

$$logit(\pi) = 5.812 - 0.115 distance$$

# Example

There is actually much more information stored within the mod.fit object than showed so far. Through the use of the names() function, we obtain the following list of items:

```
> names(mod.fit)
 [1] "coefficients"      "residuals"         "fitted.values"     "effects"           "R"
 [6] "rank"              "qr"                "family"            "linear.predictors" "deviance"
[11] "aic"               "null.deviance"     "iter"              "weights"           "prior.weights"
[16] "df.residual"       "df.null"           "y"                 "converged"         "boundary"
[21] "model"             "call"              "formula"           "terms"             "data"
[26] "offset"            "control"           "method"            "contrasts"         "xlevels"
>
```

```
> length(mod.fit$coefficients)
[1] 2
> mod.fit$coefficients
(Intercept)    distance
  5.8120798  -0.1150267
> mod.fit$coefficients[1]
(Intercept)
    5.81208
> mod.fit$coefficients[1]
(Intercept)
    5.81208
```

# Example

To see a summary of all the information in mod.fit, we can use the summary() function.

```
> summary(object = mod.fit)

Call:
glm(formula = good ~ distance, family = binomial(link = logit),
    data = df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7441   0.2425   0.2425   0.3801   1.6092

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.812080   0.326277   17.81   <2e-16 ***
distance    -0.115027   0.008339  -13.79   <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: 1013.43  on 1424  degrees of freedom
Residual deviance:  775.75  on 1423  degrees of freedom
AIC: 779.75

Number of Fisher Scoring iterations: 6
```

# Example

Remember that R is an "object-oriented language" in the sense that every object in R has a class associated with it. The classes for mod.fit are:

```
> class(mod.fit)
[1] "glm" "lm"
```

Associated with each class, there are a number of "method" functions.

```
> methods(class = glm)
 [1] add1           anova          coerce         confint        cooks.distance deviance
 [7] drop1          effects        extractAIC     family         formula        influence
[13] initialize     logLik         model.frame    nobs           predict        print
[19] residuals      rstandard      rstudent       show           slotsFromS3    summary
[25] vcov           weights
see '?methods' for accessing help and source code
```

# Example

Recall that the effect of an explanatory variable on the response probability depends on the specific value taken by the the explanatory variable.

In this particular example, the estimated probability of success for a particular distance using:

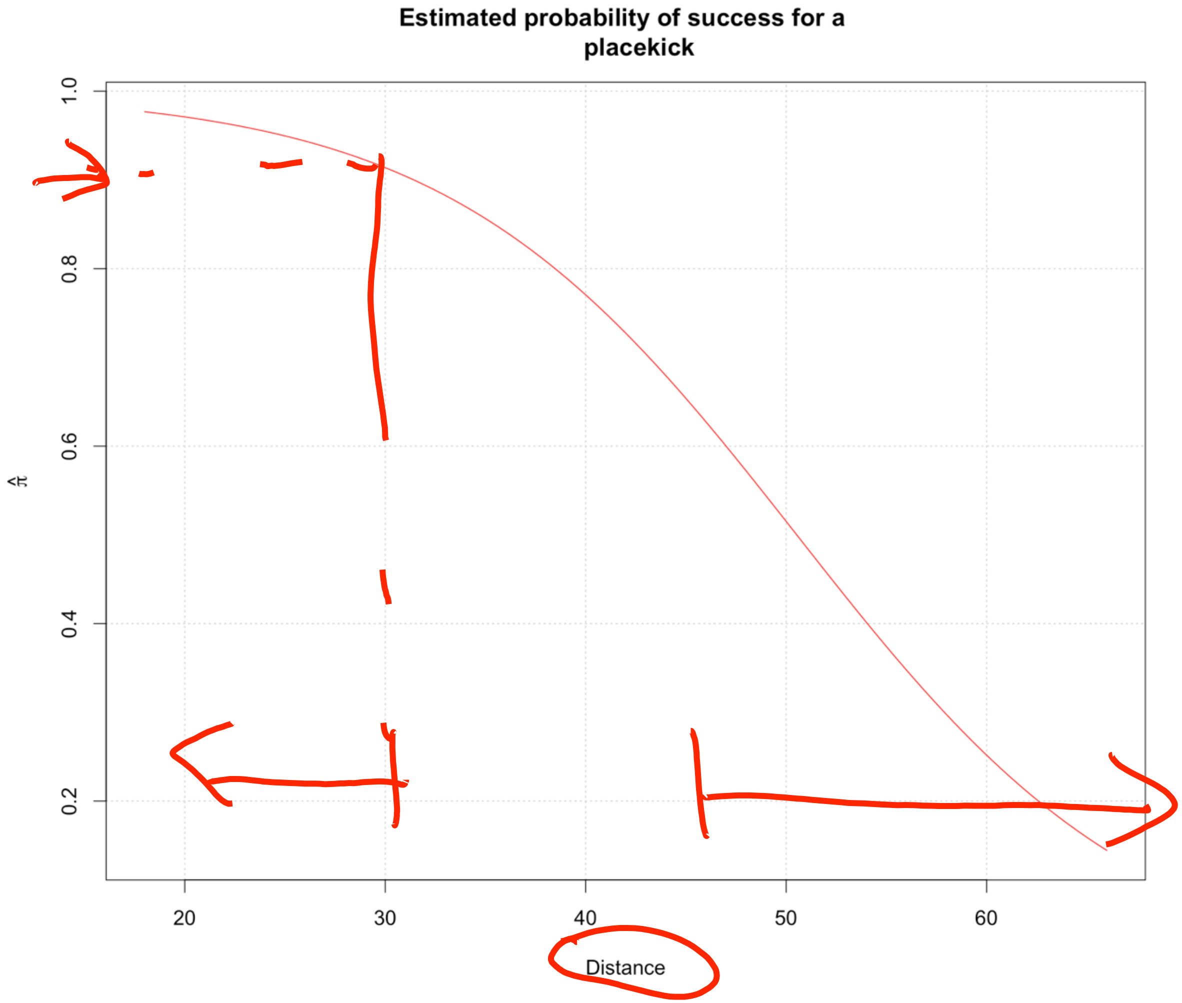
$$\boxed{\hat{\pi} = \frac{e^{5.812 - 0.115 distance}}{1 + e^{5.812 - 0.115 distance}}}$$

For example, the probability of success at a distance of 20 is 0.97. Likewise, the estimated probability of success for a distance of 50 yards is 0.52.

To get a perspective the above numbers, let's examine the distribution of distance in our sample:

```
summary(df$distance)
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
18.00   20.00   20.00   27.55   36.00   66.00
```

# Example




```
curve(expr = exp(mod.fit$coefficients[1] + mod.fit$coefficients[2]*x) /
        (1 + exp(mod.fit$coefficients[1] + mod.fit$coefficients[2]*x)),
      col = "red", xlim = c(18, 66), ylab = expression(hat(pi)), xlab = "Distance",
    main = "Estimated probability of success for a
    placekick", panel.first = grid())
```

# Example

If more than one explanatory variable is included in the model, the variable names can be separated by "+" symbols in the formula argument.

For example, suppose we include the change variable in addition to distance in the model:

```
mod.fit2<-glm(formula = good ~ change + distance, family= binomial(link = logit), data = df)
summary(mod.fit2)
```

```
Call:
glm(formula = good ~ change + distance, family = binomial(link = logit),
    data = df)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.7061   0.2282   0.2282   0.3750   1.5649

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept)  5.893181   0.333184  17.687   <2e-16 ***
change      -0.447783   0.193673  -2.312   0.0208 *
distance    -0.112889   0.008444 -13.370   <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: 1013.4  on 1424  degrees of freedom
Residual deviance:  770.5  on 1422  degrees of freedom
AIC: 776.5

Number of Fisher Scoring iterations: 6
```

The estimated logistic regression model is

$$logit(\hat{\pi}) = 5.8932 - 0.4478 change - 0.1129 distance$$

# Example

While we will use the glm() function extensively to find MLEs in this course, and this function is widely used in the industry, one could also find these estimates by programming the log likelihood function and maximizing it using another R function, optim().

This can come in handy for other "nonstandard" optimization problems that may occur in practice. We will not get into the details of programming the likelihood function manually. Please refer to the text for a simple illustration of how to program an MLE and optimize it manually.

Berkeley SCHOOL OF INFORMATION