Maximum likelihood and GLMs

"Never tell me the odds!" - *Han Solo*

Samuel Robinson, Ph.D.

November 26, 2020

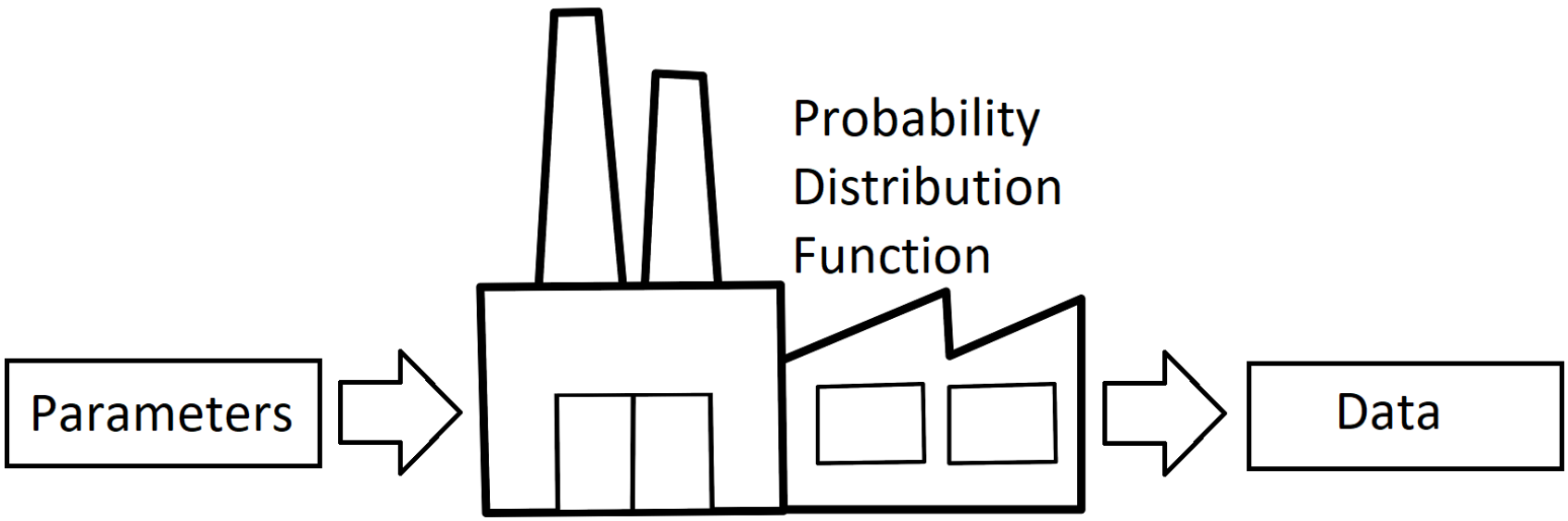
Outline

- Maximum likelihood
  - A way to think about data
  - Likelihood vs Probability
- Generalized linear models
  - Link functions
  - Predictors -> Linear model

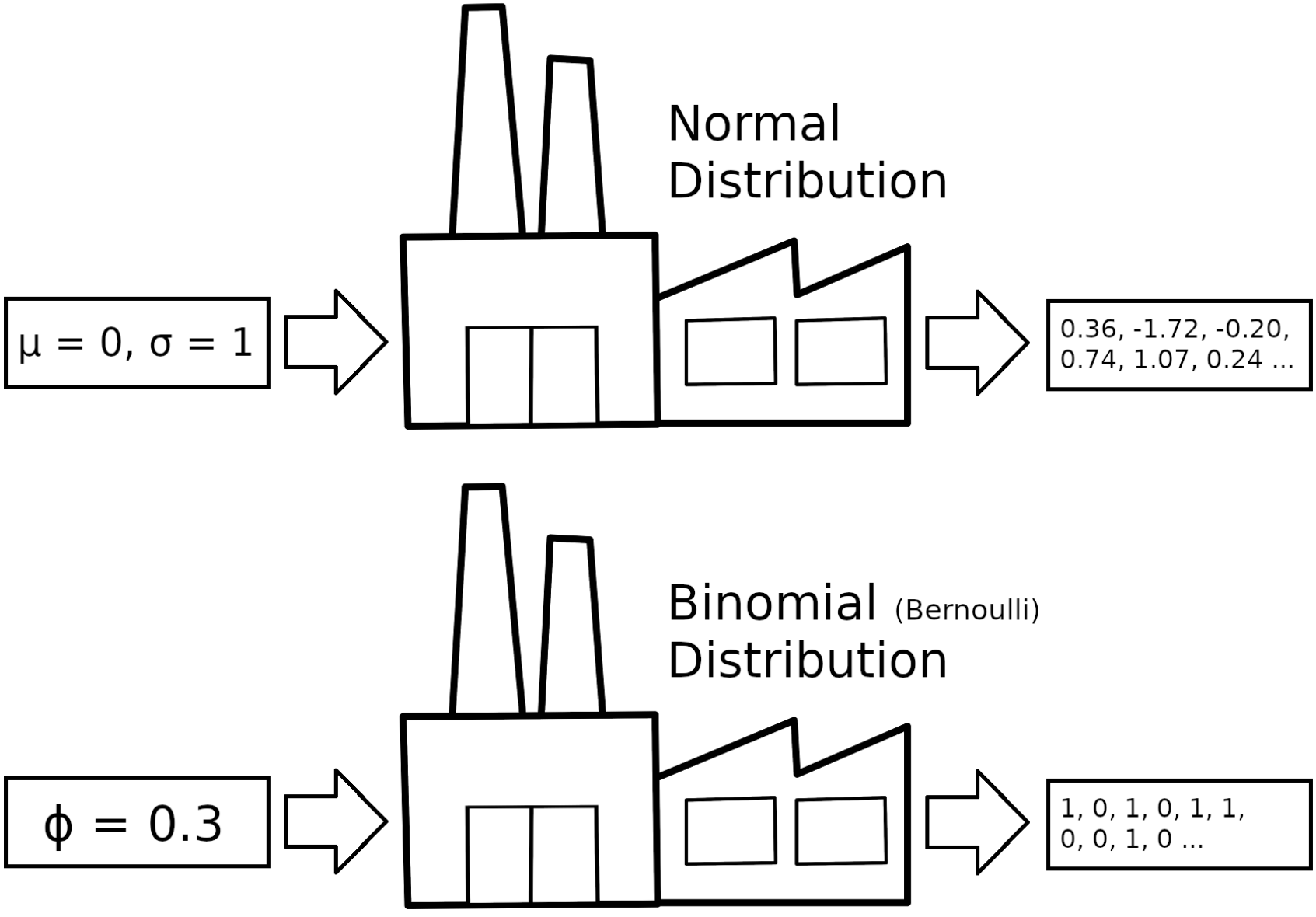
How is our data made?

Making data can be thought of as a *factory*

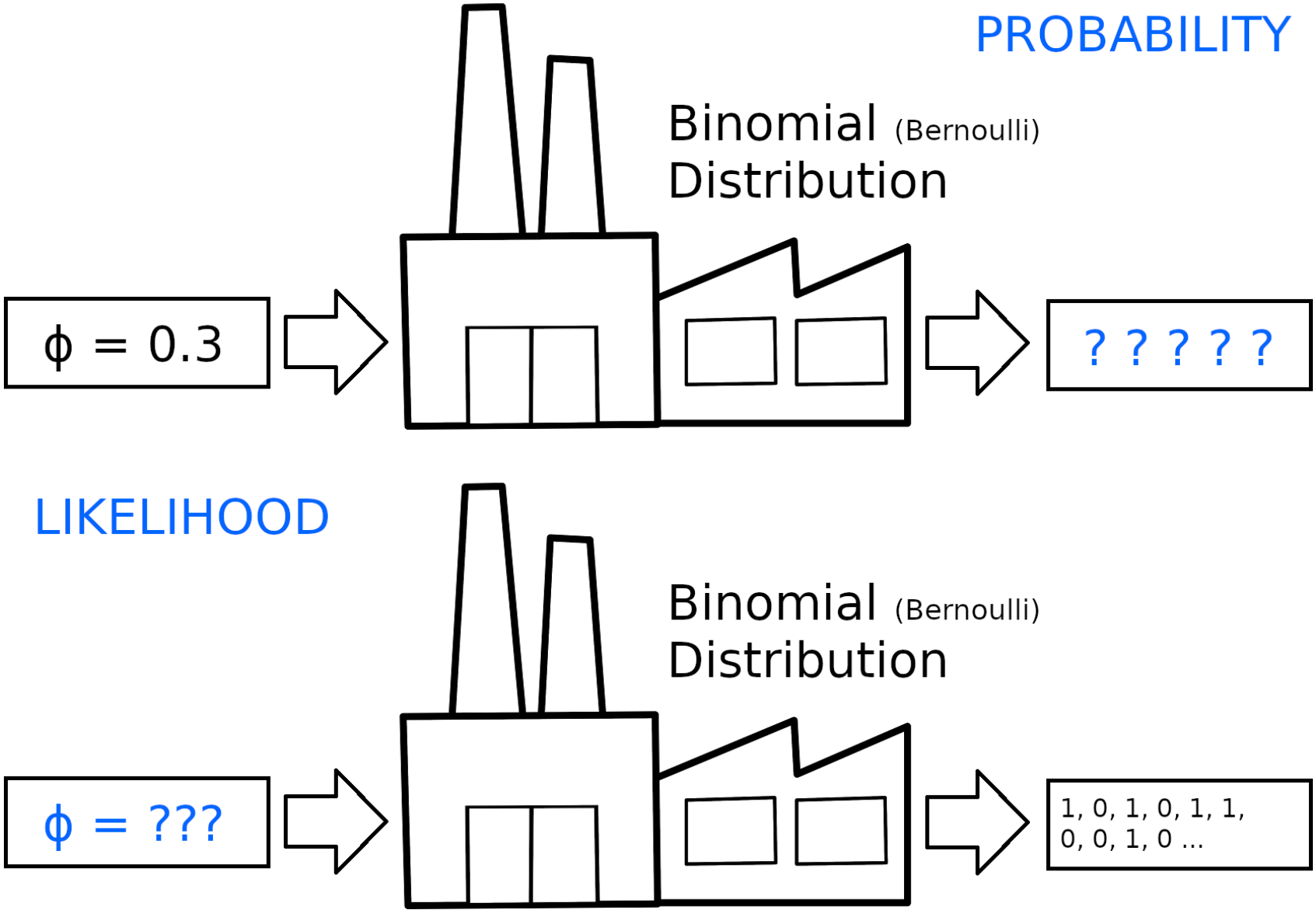
- Input: **parameters** (things that guide the process)
- Process: **probability function**
- Output: **data** (things made by the process)



Examples


Normal
Distribution
μ = 0, σ = 1
0.36, -1.72, -0.20,
0.74, 1.07, 0.24 ...
Binomial (Bernoulli)
Distribution
ϕ = 0.3
1, 0, 1, 0, 1, 1,
0, 0, 1, 0 ...

Likelihood vs Probability



Likelihood vs Probability (cont.)

Probability and likelihood both use the same PDF

- "I know that $\phi = 0.3$. What is the chance of getting 2 heads and a tail?"

```
dbinom(1,1,0.3)*dbinom(1,1,0.3)*dbinom(0,1,0.3)
```

```
## [1] 0.063
```

- "I got 2 heads and a tail. What is the likelihood that $\phi = 0.3$?"

```
dbinom(1,1,0.3)*dbinom(1,1,0.3)*dbinom(0,1,0.3)
```

```
## [1] 0.063
```

Likelihood vs Probability (cont.)

Let's see how *likelihood* changes with different values of ϕ:

```
#phi = 0.3
dbinom(1,1,0.3)*dbinom(1,1,0.3)*dbinom(0,1,0.3)
```

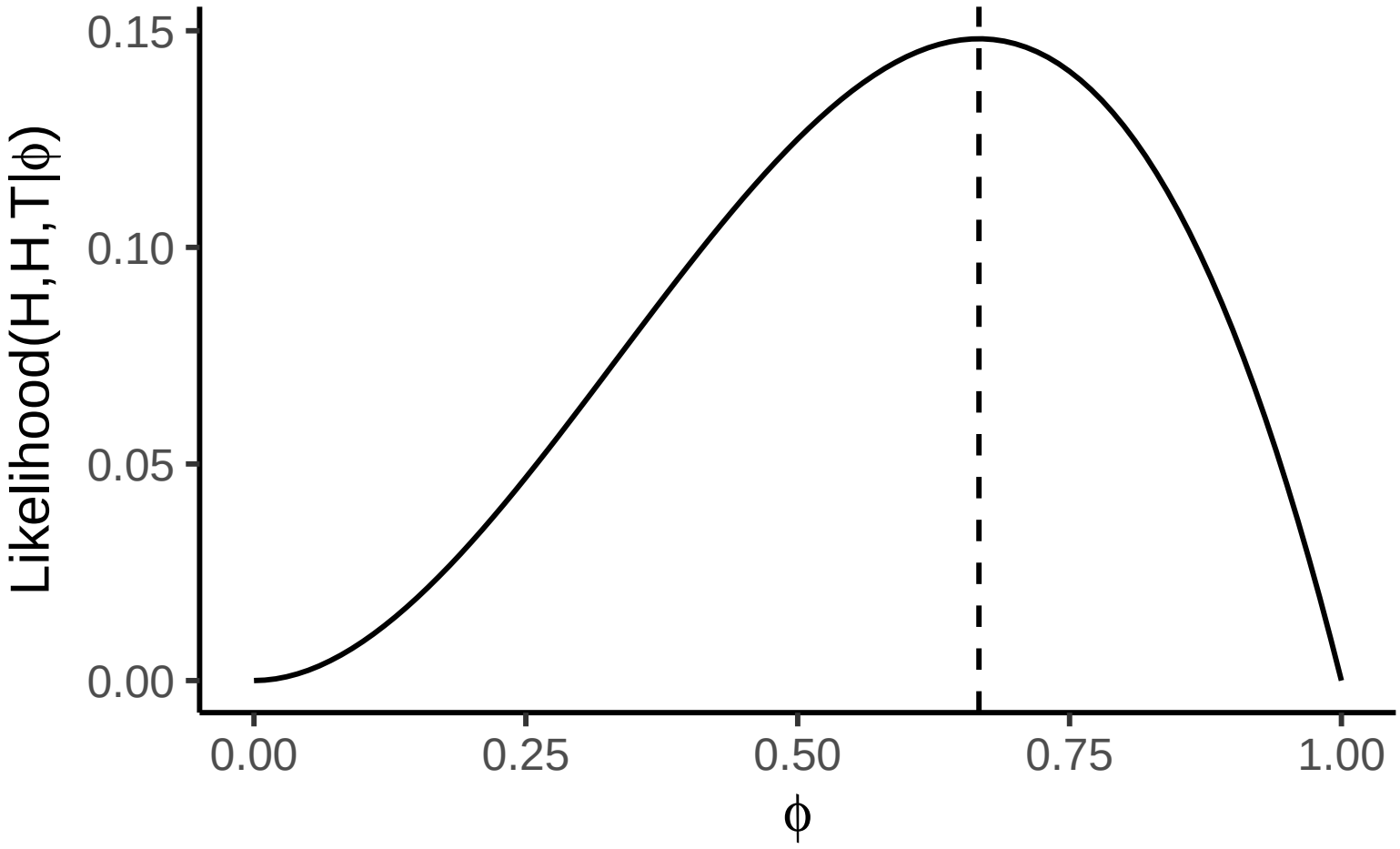
```
## [1] 0.063
```

```
#phi = 0.7
dbinom(1,1,0.7)*dbinom(1,1,0.7)*dbinom(0,1,0.7)
```

```
## [1] 0.147
```

Likelihood of $\phi = 0.7$ is higher, i.e. $\phi = 0.7$ matches our data *better*

Likelihood



The best match (maximum likelihood value) is at $\phi = 0.666$ (2 heads out of 3 flips)

Generalized Linear Models

`glm()` will fit a model like this, and find the ML solution

```
dat <- data.frame(flips=c(1,1,0)) #Data (2 heads, 1 tail)
mod1 <- glm(flips~1,data=dat,family='binomial') #Note family specification
summary(mod1)
```

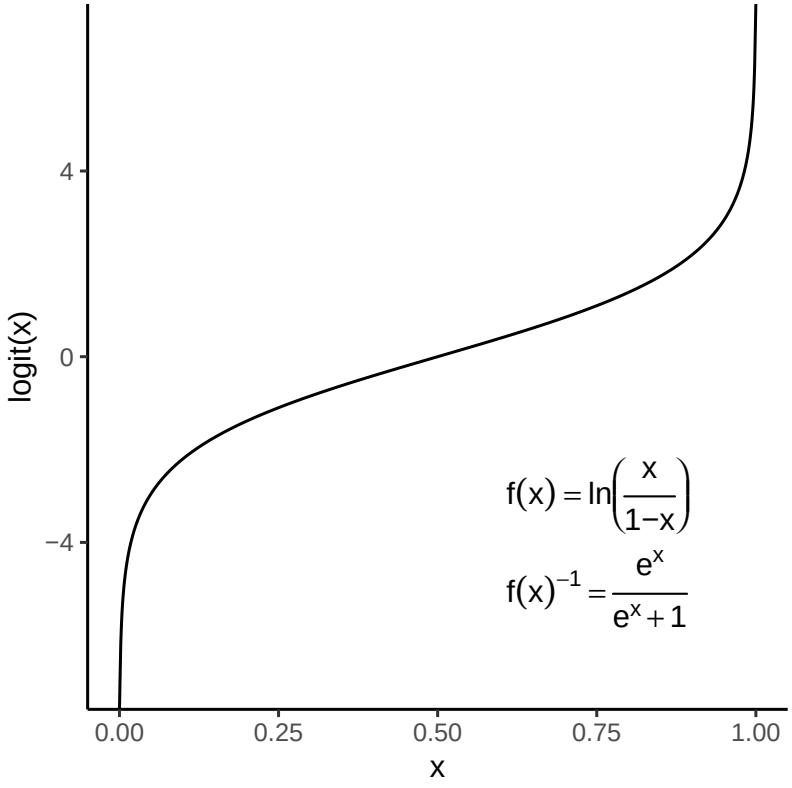
```
##
## Call:
## glm(formula = flips ~ 1, family = "binomial", data = dat)
##
## Deviance Residuals:
##       1        2        3
##  0.9005   0.9005  -1.4823
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   0.6931     1.2247   0.566    0.571
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3.8191  on 2  degrees of freedom
## Residual deviance: 3.8191  on 2  degrees of freedom
## AIC: 5.8191
##
## Number of Fisher Scoring iterations: 4
```

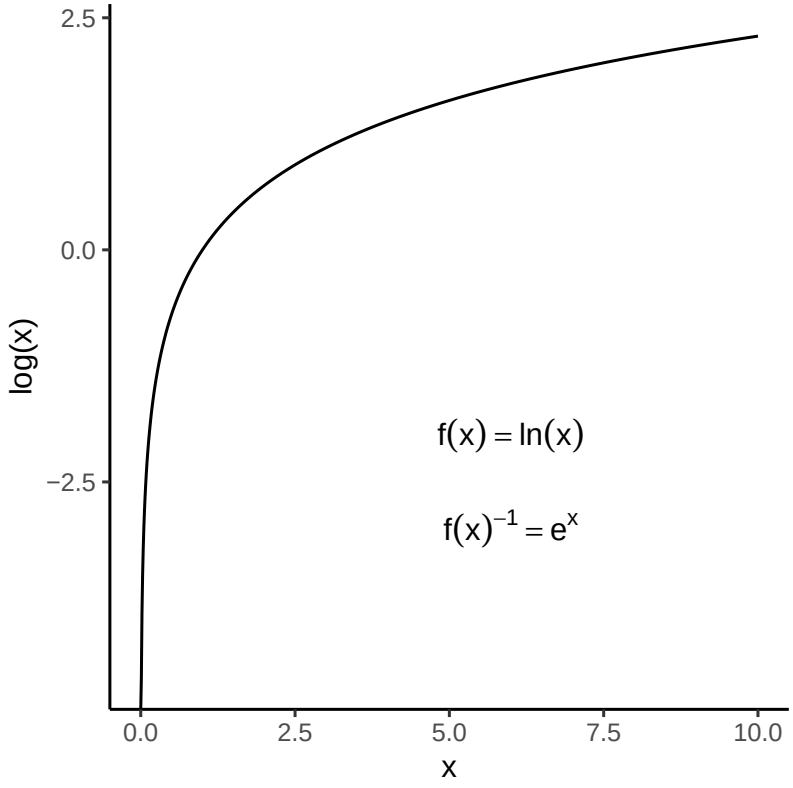
Wait. . . our estimate should be 0.666 (2/3), not 0.693!

Link functions

- Some parameters of PDFs have *limits*
  - Normal: $-\infty < \mu < \infty$, $0 < \sigma$
  - Binomial: $0 < \phi < 1$
  - Poisson: $0 < \lambda$
- GLMs use *link functions* to map values onto the appropriate parameter range
  - Normal: Identity (i.e. $\times 1$)
  - Binomial: Logit
  - Poisson/NB: Log
- $\mathit{logit}(0.693) = 0.666$, so the GLM actually got it right!

What do these functions look like?

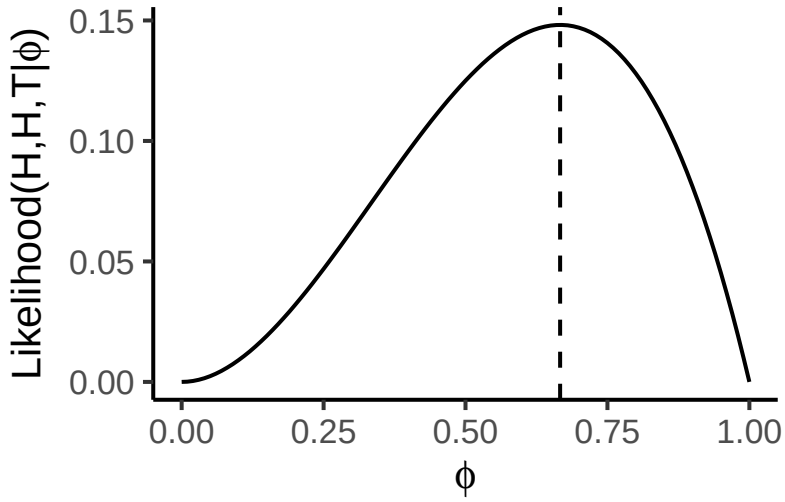


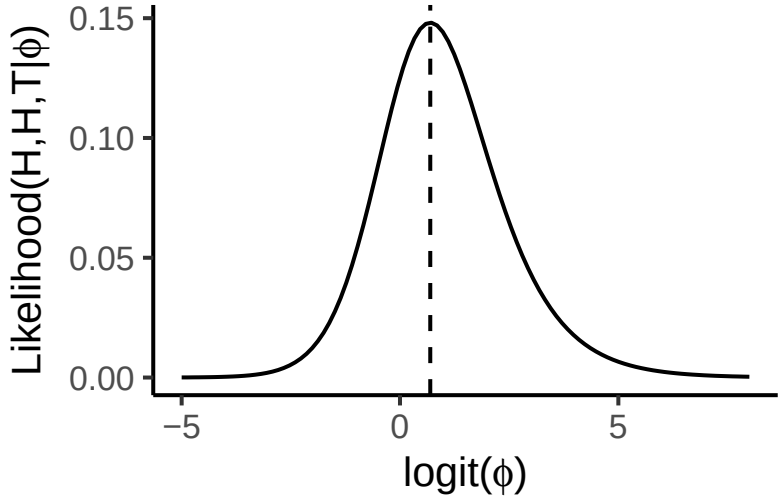


- These functions map parameter values from the appropriate range (0-1 or 0-∞) onto $-\infty$ to $+\infty$

Why do we bother with these link function?

- Likelihood functions are not symmetrical on the regular scale
- On the link-scale, they are closer to a normal distribution
- Makes it easier for R to find the ML estimate (and confidence intervals)





How do linear models fit into this?

- Usually we aren't interested in finding only a single parameter ϕ.
- Solution: ϕ becomes a *linear* function of the predictors
- Simple linear models take the form:

$$\hat{y} = b_0 + b_1 x_1 ... + b_i x_i$$
$$y \sim Normal(\hat{y}, \sigma)$$

- Generalized linear models are similar, except that:

1. Expected value (ϕ) fed through a link function
2. Data is fit to a non-normal probability function

$$logit(\hat{\phi}) = b_0 + b_1 x_1 ... + b_i x_i$$
$$flips \sim Binomial(\hat{\phi})$$

Instead of finding ϕ, **R finds the coefficients (b_0, b_1 ... b_i) that create ϕ**

How do I fit GLMs in R?

Syntax and model output is very similar to `lm`

```r
# y ~ x, where x is the predictor of y (~1 for just intercept)
mod_binomial <- glm(y2 ~ x1 + x2 , data = d1, family = 'binomial') #Fit a binomial GLM
summary(mod_binomial)
```

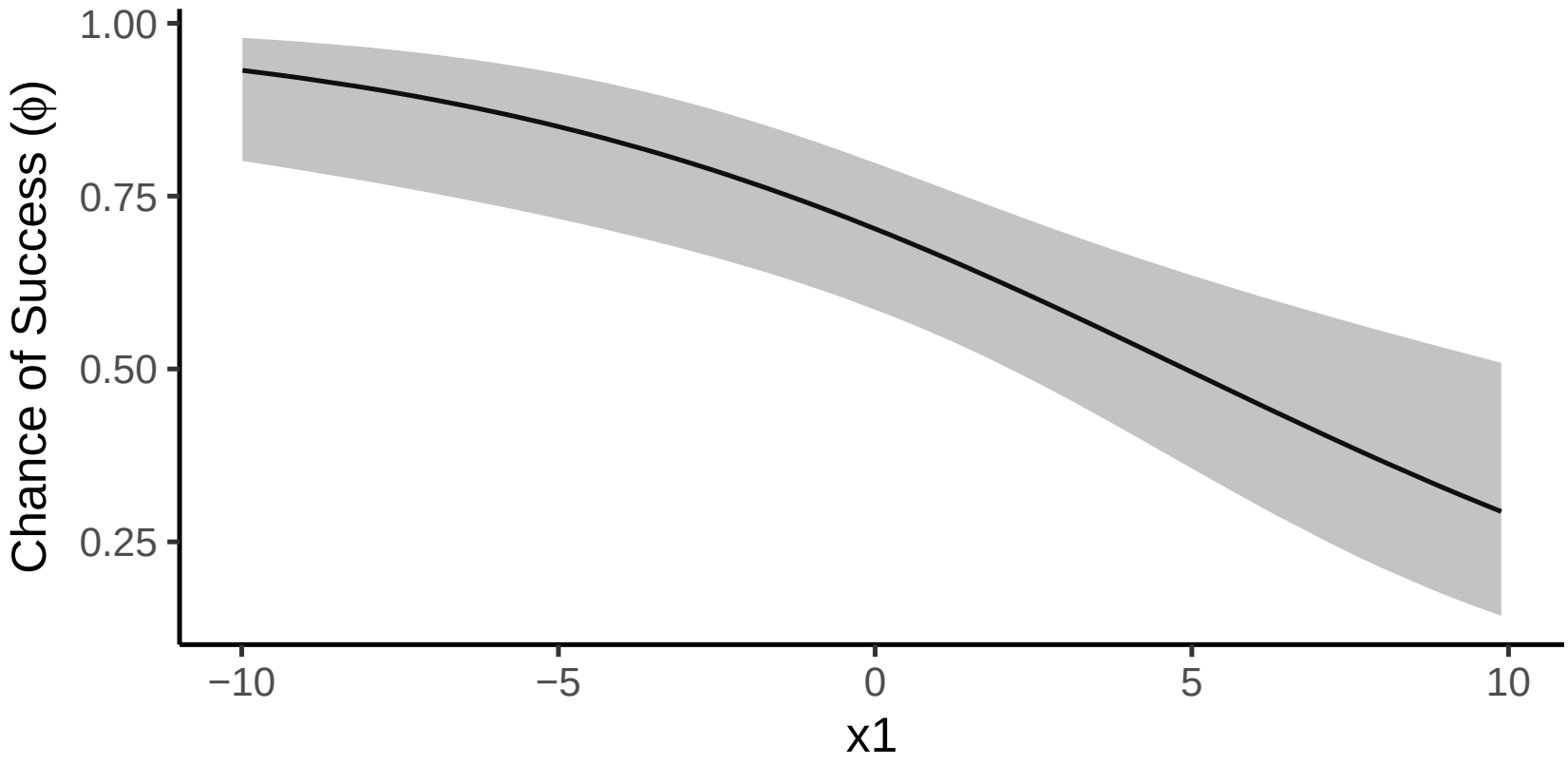
```
##
## Call:
## glm(formula = y2 ~ x1 + x2, family = "binomial", data = d1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0050  -0.9493   0.3924   0.8336   1.6806
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.81748    0.25851   3.162 0.001565 **
## x1          -0.17576    0.04871  -3.608 0.000309 ***
## x2           0.30193    0.09950   3.034 0.002410 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 129.49  on 99  degrees of freedom
## Residual deviance: 102.98  on 97  degrees of freedom
## AIC: 108.98
##
## Number of Fisher Scoring iterations: 4
```

Dispersion and deviance will be discussed later. . .

How do I get partial effects plots?

`crPlot` (from `car`) and `ggpredict` (`ggeffects`) work with fitted `glm` models

```
ggpredict(mod_binomial, terms='x1 [all]') %>% #Partial effect of x1 term
  ggplot(aes(x, predicted)) + geom_line() +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = 0.3) +
  labs(x = 'x1', y = expression(paste('Chance of Success (',phi,')')))
```



A challenger approaches!

- Dr. Roberto Darkley (Robert Barkley's evil nemesis) sent 2 people out to check out some bat roosts in Edmonton and Calgary. One of them dutifully counted bats at each roost, but the other one was really lazy, and just recorded "bats or no bats" (1 or 0).
- Fit a model to each of their data (found in `batDatGLM.csv`) using a GLM
  - `batCounts` should be modeled using a Poisson GLM, and `batPres` should use a Binomial GLM
  - Terms to include: `city` and `size` (no interaction)
- How do the models look? Compare the coefficients and see if they are different.
  - Bonus: make a partial regression plot of terms in the Poisson GLM

Model results