

Statistics for Ecologists

Olivier Gimenez
October 2020

Who's that guy?!

- Senior scientist at CNRS, in Montpellier - France.
- Trained as a statistician
- Soon attracted by the bright side of ecology
- Working at the interface of animal demography, statistical modelling and social sciences
- More on <https://oliviergimenez.github.io/>
- Twitter @oaggimenez

Acknowledgments

Acknowledgments

- Sean Anderson, Ben Bolker, Jason Matthiopoulos, David Miller, Denis Réale and Francisco Rodriguez-Sánchez for sharing their courses material

This Class

Slides, R codes, data and practicals

- I used R, and RStudio is your friend
- I also used R Markdown to write reproducible documents (slides/exercises)
- All material is available on GitHub
<https://github.com/oliviergimenez/statistics-for-ecologists-Master-courses>
- Check out the files `gimenez_lectures.R` and `gimenez_practicals.R`
- You will need the following R packages: `arm`, `bbmle`, `broom`, `dplyr`, `effects`,
`lme4`, `mgcv`, `MuMIn`, `R2jags`, `tibble`, `visreg`

On our plate

- Distributions and likelihoods
- Hypothesis testing and multimodel inference
- Introduction to Bayesian inference
- Generalized Linear Models (GLMs)
- Generalized Additive Models (GAMs)
- Mixed Effect Models

On our plate

- Distributions and likelihoods
- Hypothesis testing and multimodel inference
- Introduction to Bayesian inference
- Generalized Linear Models (GLMs)
- Generalized Additive Models (GAMs)
- Mixed Effect Models

Distributions and likelihoods

Distributions

- What for?
- Conceptual models, bearing in mind that:
All models are wrong, but some are useful (G.E.P. Cox, 1976)
- Either represent how the world works
- Or capture the behavior of a statistic under some null hypothesis we'd like to test
- Discrete or continuous

Discrete distributions

Bernoulli distribution

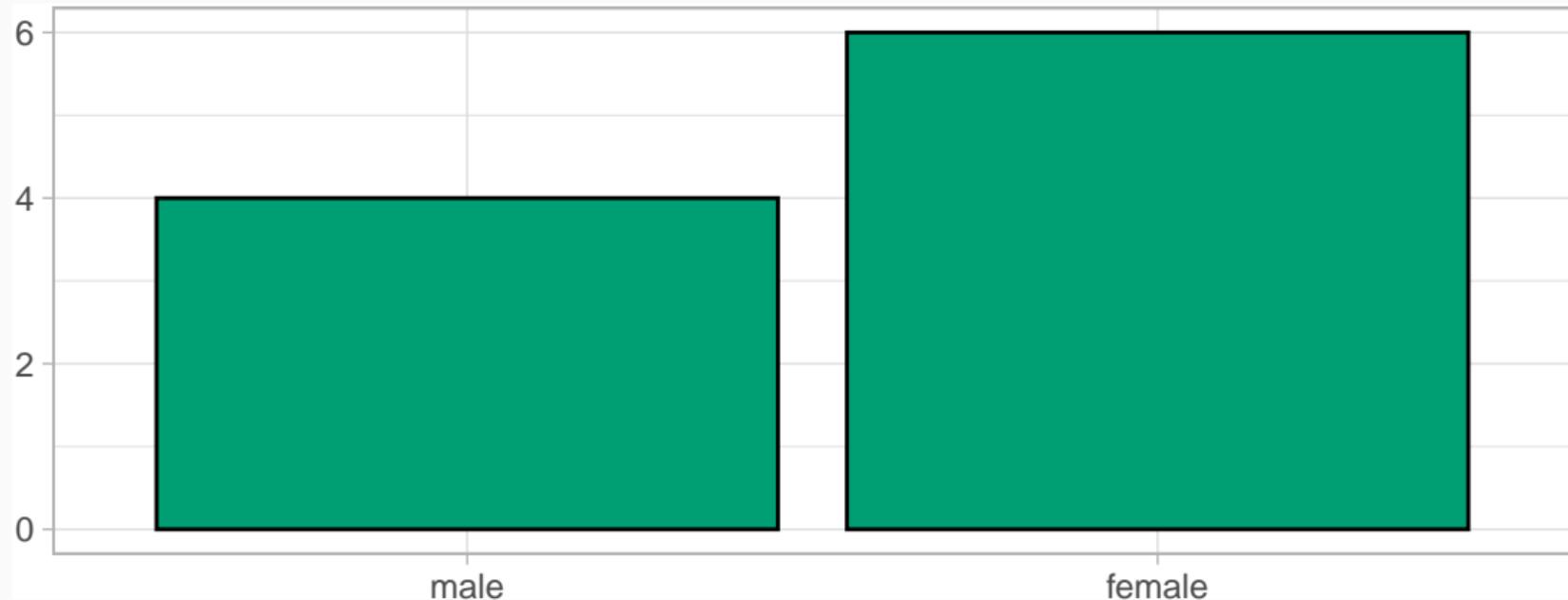
Context: A single trial with two outcomes, success/failure

$X \sim \text{Bern}(p)$ with p probability of having a success

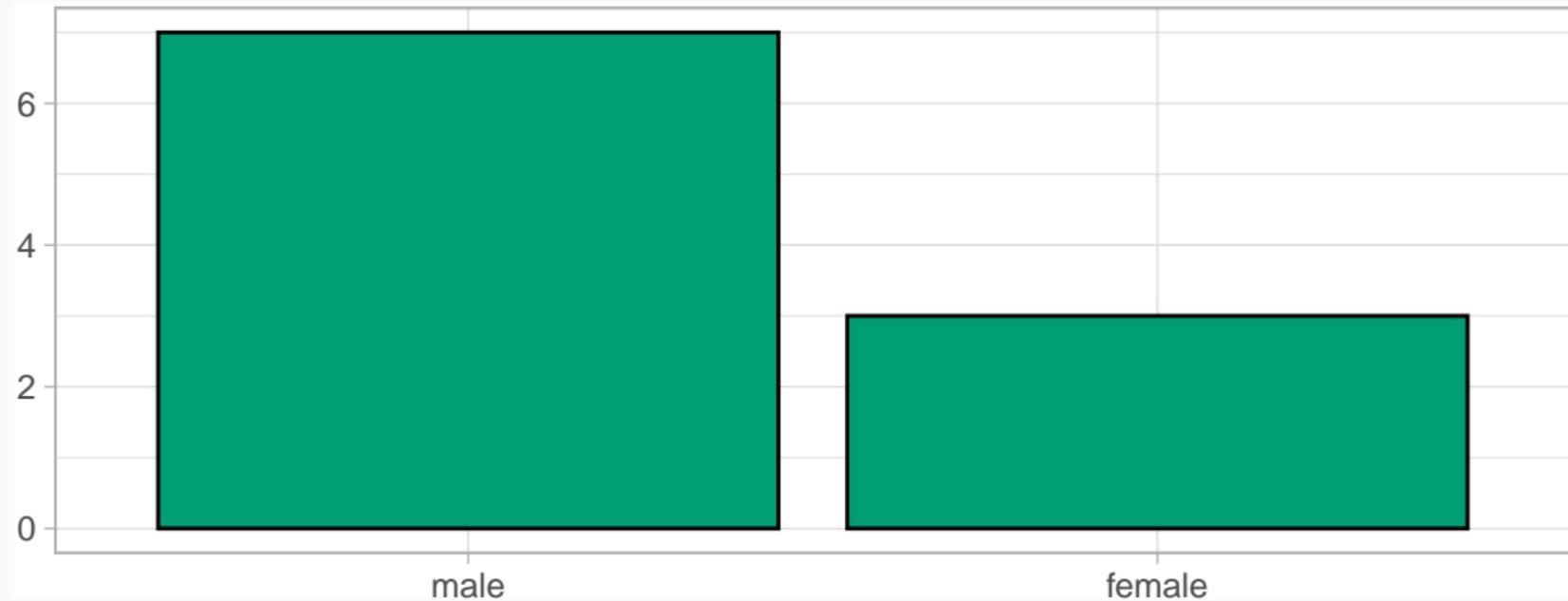
$$\begin{array}{c} \hline x & P(X = x) \\ \hline 1 & p \\ 0 & 1 - p \\ \hline \end{array}$$

Example: X is the random variable *being born a female*

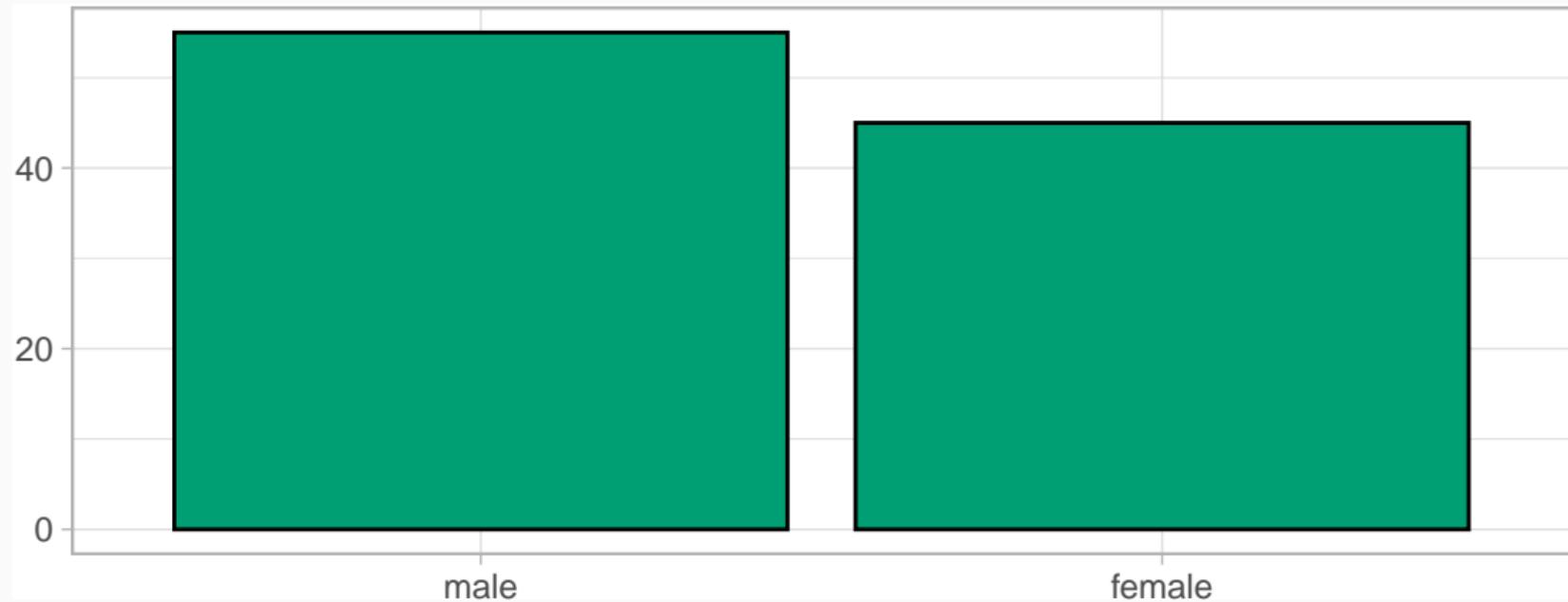
Ten Bernoulli trials with $p = 0.5$



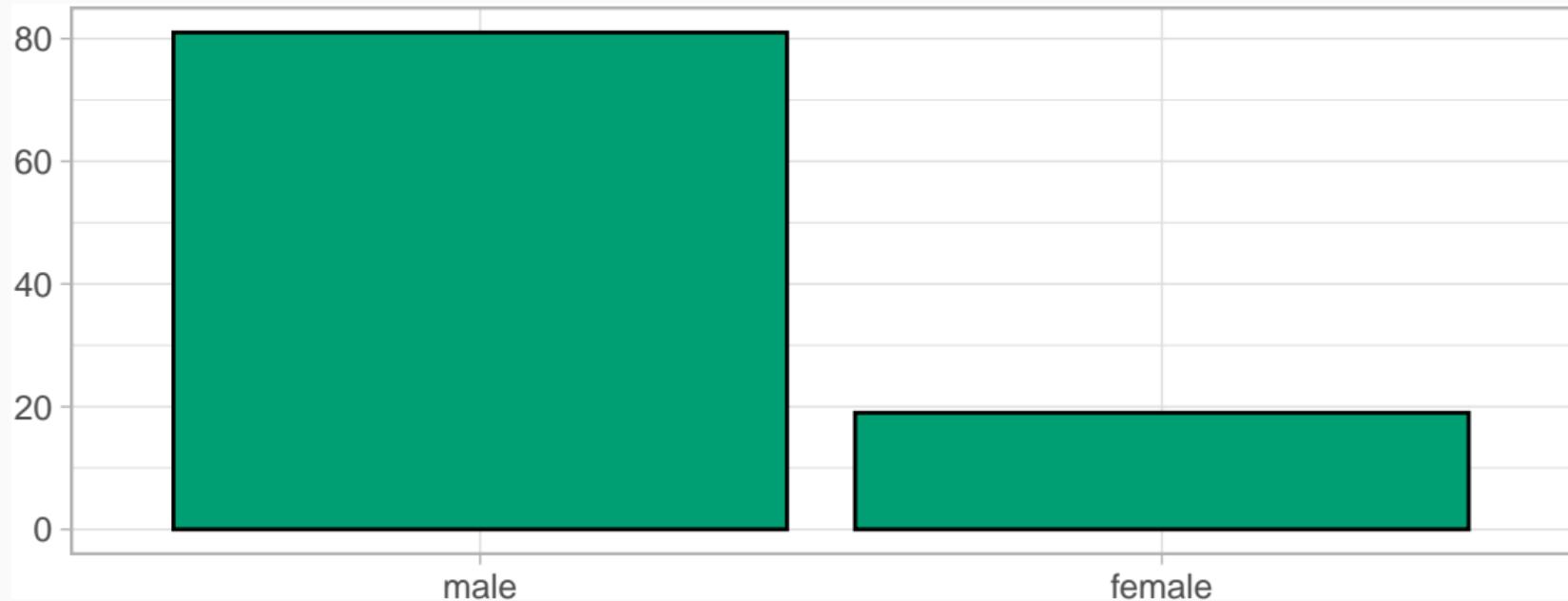
Ten Bernoulli trials with $p = 0.5$, again



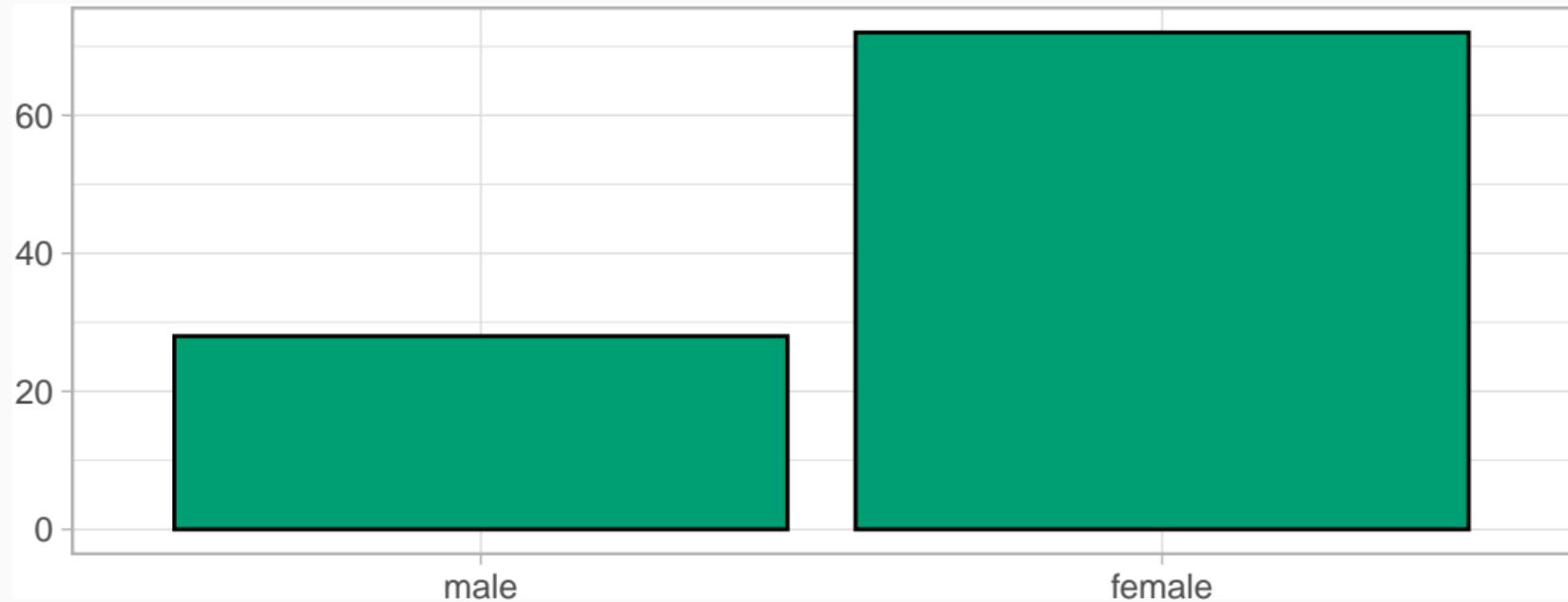
Hundred Bernoulli trials with $p = 0.5$



Hundred Bernoulli trials with $p = 0.2$



Hundred Bernoulli trials with $p = 0.8$



Summary: Bernoulli distribution

- **notation:** $X \sim \text{Bern}(p)$
- **range:** discrete, $x = 0, 1$
- **distribution:** $P(X = x) = p^x(1 - p)^{1-x}$
- **parameters:** p is the probability of success
- **mean:** p
- **variance:** $p(1 - p)$

Binomial distribution

Context: Total number of successes from a fixed number of independent Bernoulli trials, all with same probability of success

$X \sim \text{Bin}(N, p)$ with p probability of having a success and N number of trials

$$P(X = x) = \frac{N!}{x!(N-x)!} p^x (1-p)^{N-x} = \binom{N}{x} p^x (1-p)^{N-x}$$

Example: X is the random variable *number of heads in a series of coin flipping*

Binomial distribution

$$P(X = x) = \binom{N}{x} p^x (1-p)^{N-x}$$

x	$P(X = x)$
0	$(1-p)^N$
1	$Np(1-p)^{N-1}$
...	...
N	p^N

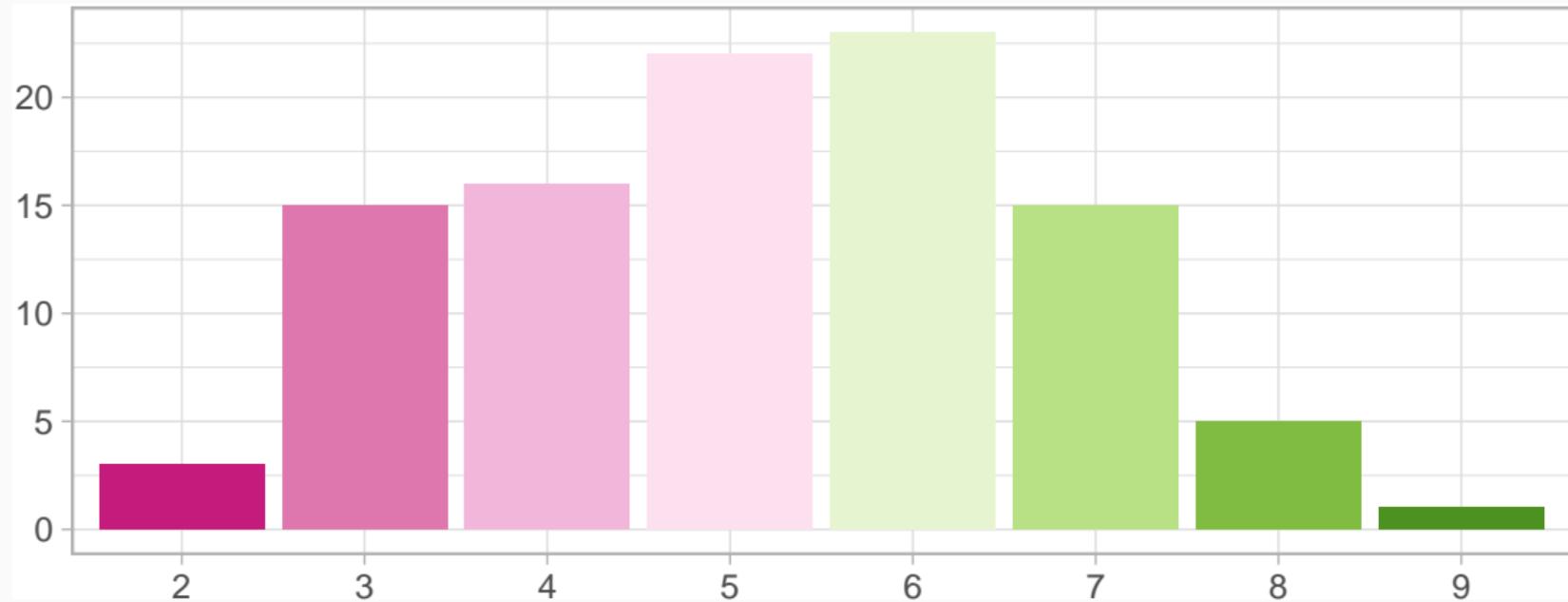
Binomial distribution

x	$P(X = x)$
0	$(1 - p)^N$
1	$Np(1 - p)^{N-1}$
...	...
N	p^N

Fortunately, R has this pre-programmed

```
dbinom(x = 1, size = 10, prob = 0.5) # equals 10*0.5*(1-0.5)^(10-1)
## [1] 0.009765625
```

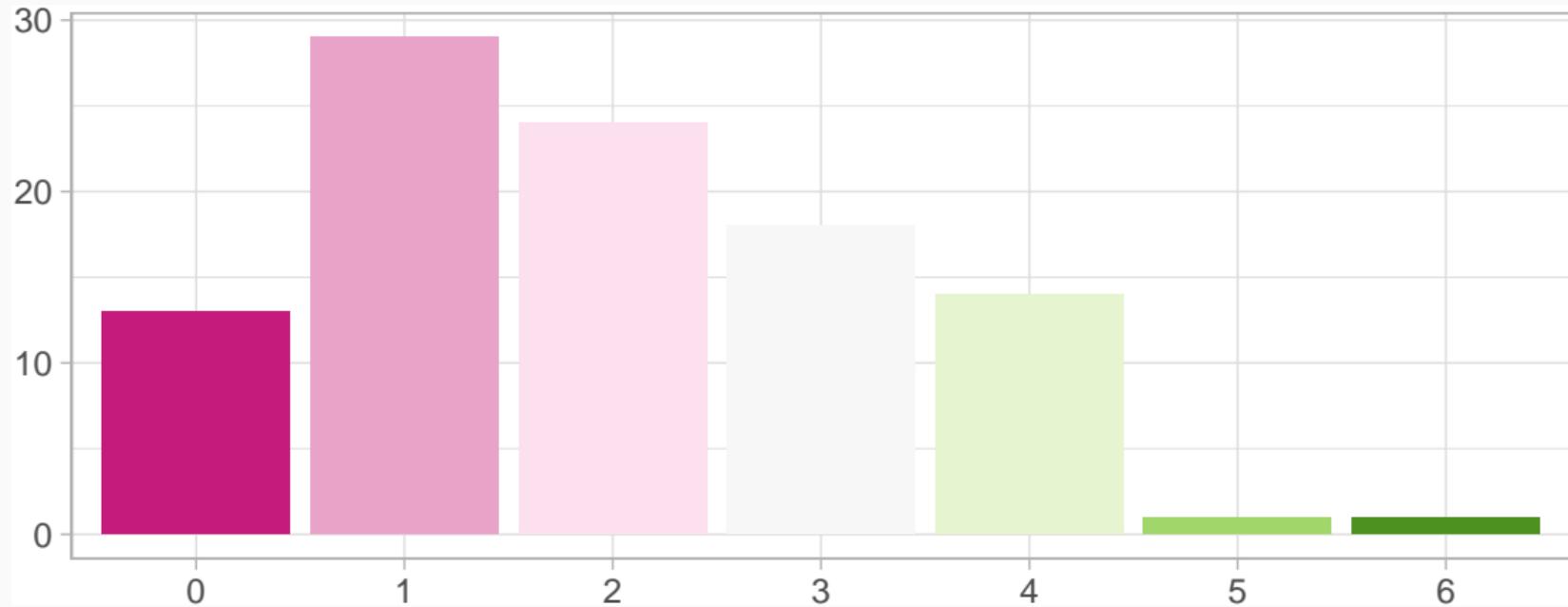
Hundred Binomial trials with $N = 10$ and $p = 0.5$



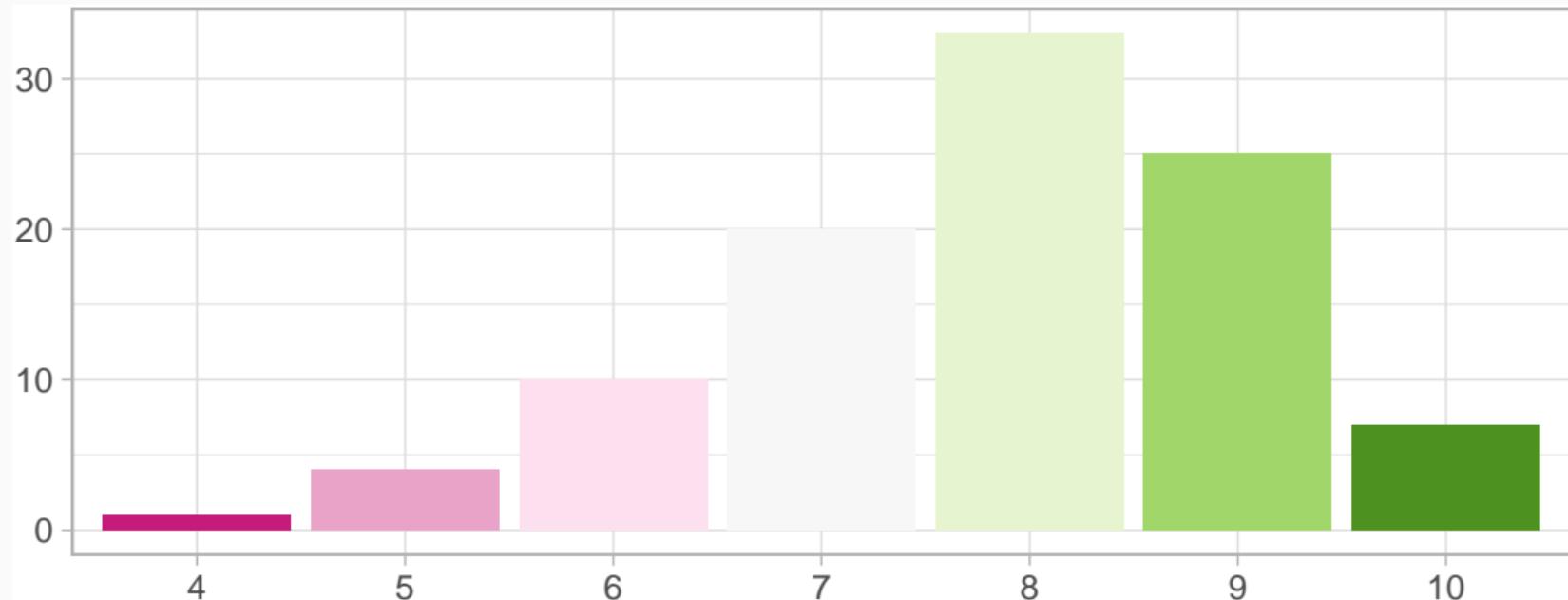
Hundred Binomial trials with $N = 10$ and $p = 0.5$, again



Hundred Binomial trials with $N = 10$ and $p = 0.2$



Hundred Binomial trials with $N = 10$ and $p = 0.8$



Playing around with probabilities

- Let's say $X \sim \text{Bin}(N = 10, p = 0.5)$ is a random variable counting the number of males

Playing around with probabilities

- Let's say $X \sim \text{Bin}(N = 10, p = 0.5)$ is a random variable counting the number of males
- What is the probability of having at most 2 males?

Playing around with probabilities

- Let's say $X \sim \text{Bin}(N = 10, p = 0.5)$ is a random variable counting the number of males
- What is the probability of having at most 2 males?
- $P(X \leq 2) = P(X = 0) + P(X = 1)$

Playing around with probabilities

- Let's say $X \sim \text{Bin}(N = 10, p = 0.5)$ is a random variable counting the number of males
- What is the probability of having at most 2 males?
- $P(X \leq 2) = P(X = 0) + P(X = 1)$
- How to compute this in R?

Playing around with probabilities

- Let's say $X \sim \text{Bin}(N = 10, p = 0.5)$ is a random variable counting the number of males
- What is the probability of having at most 2 males?
- $P(X \leq 2) = P(X = 0) + P(X = 1)$
- How to compute this in R?
- `dbinom(x=0,size=10,prob=0.5) + dbinom(x=1,size=10,prob=0.5)`

Summary: Binomial distribution

- **notation:** $X \sim \text{Bin}(N, p)$
 - **range:** discrete, $0 \leq x \leq N$
 - **distribution:** $P(X = x) = \binom{N}{x} p^x (1 - p)^{1-x}$
 - **parameters:** p the probability of success, and N the number of trials
- **mean:** Np
 - **variance:** $Np(1 - p)$
 - **in R:** `rbinom`, `dbinom`

Poisson distribution

Context: Number of occurrences of an event over a given unit of space or time.

$X \sim \text{Poisson}(\lambda)$ with λ expected number of occurrences

$$P(X = x) = \frac{e^{-\lambda} \lambda^x}{x!}$$

Example: X is the random variable *number of birds counted on a colony during the breeding season*

Poisson distribution

$$P(X = x) = \frac{e^{-\lambda} \lambda^x}{x!}$$

x	$P(X = x)$
0	$e^{-\lambda}$
1	$\lambda e^{-\lambda}$
...	...

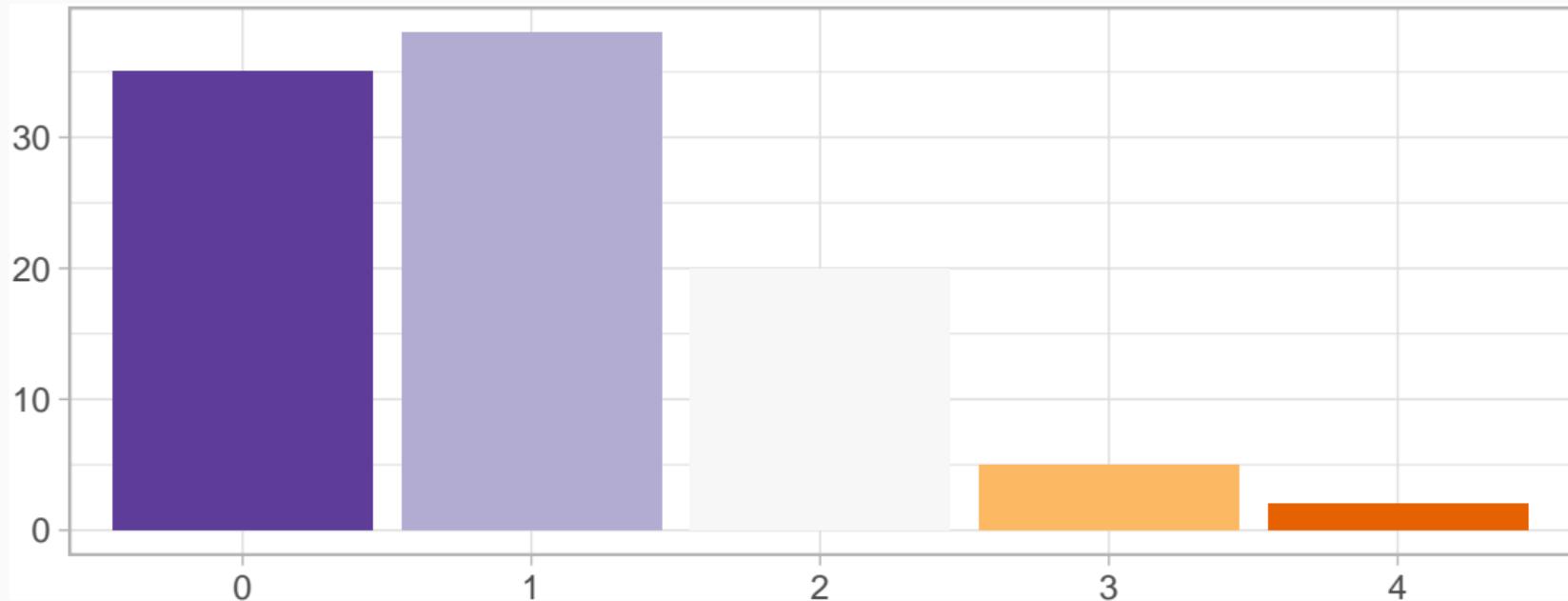
Poisson distribution

x	$P(X = x)$
0	$e^{-\lambda}$
1	$\lambda e^{-\lambda}$
...	...

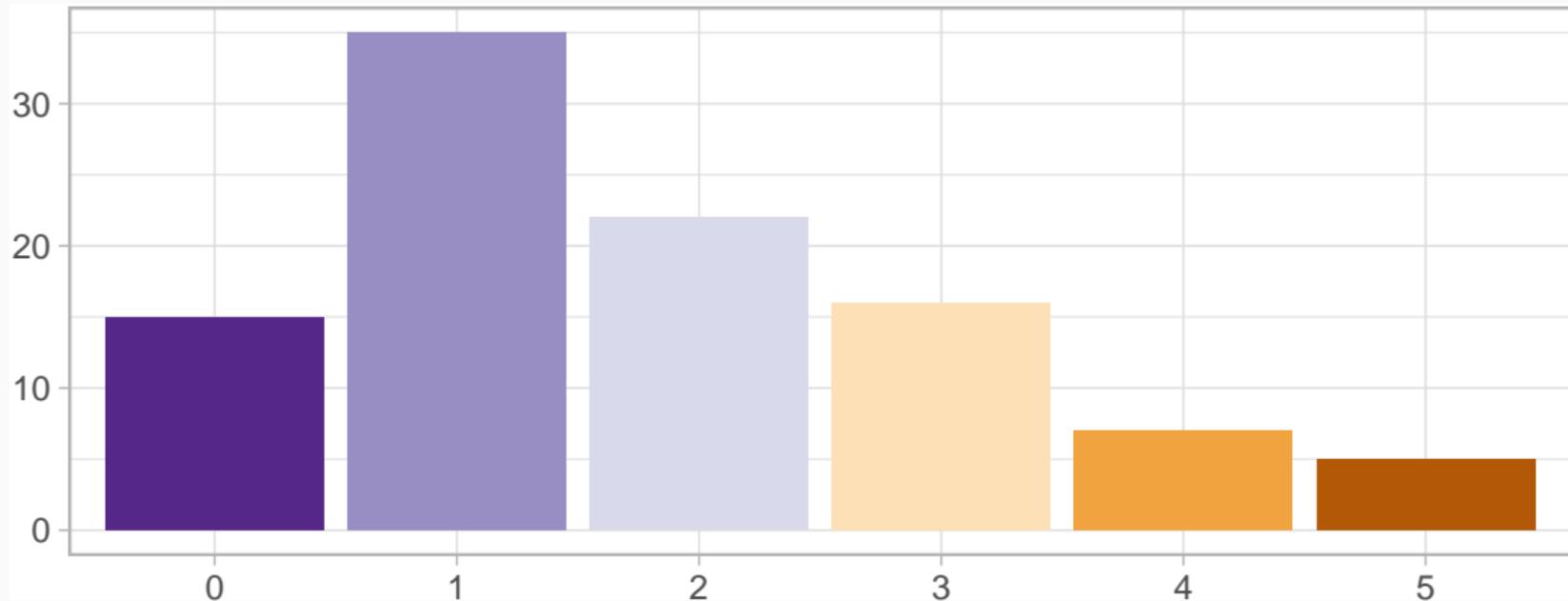
Fortunately, R has this pre-programmed

```
dpois(x=0,lambda=3) # equals exp(-3)
## [1] 0.04978707
```

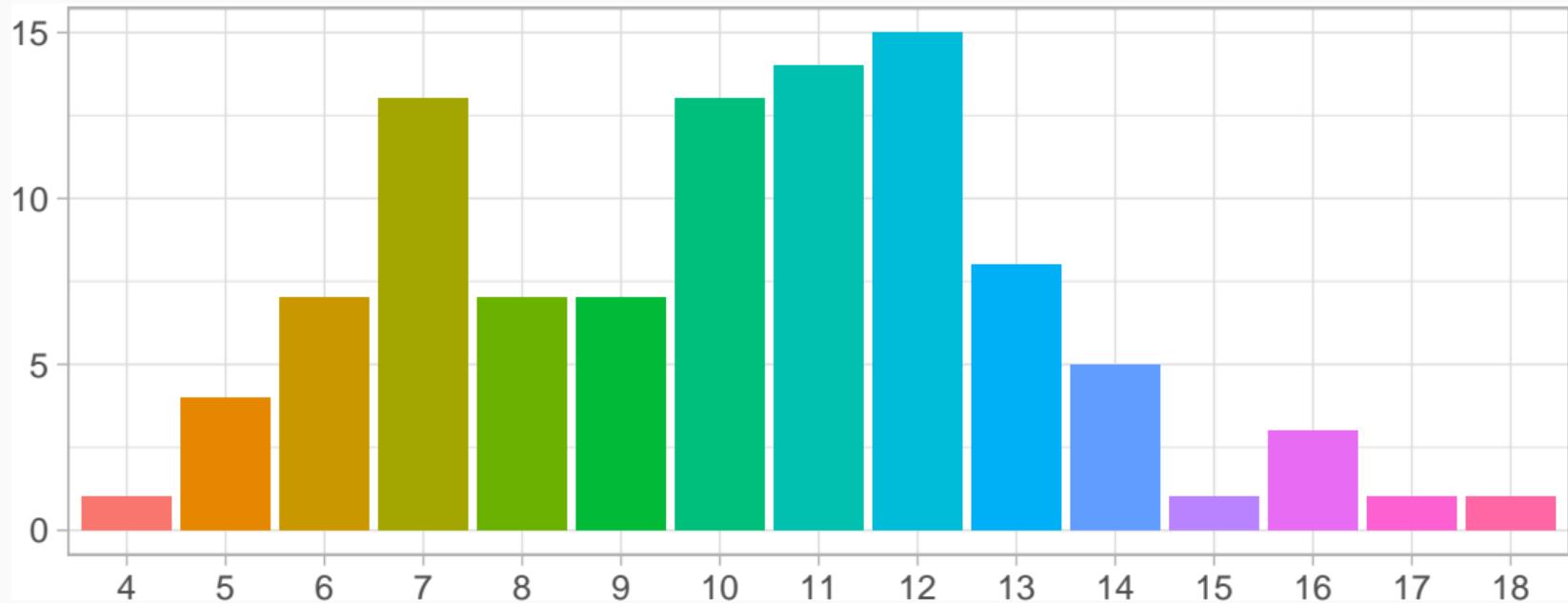
Hundred Poisson trials with $\lambda = 1$



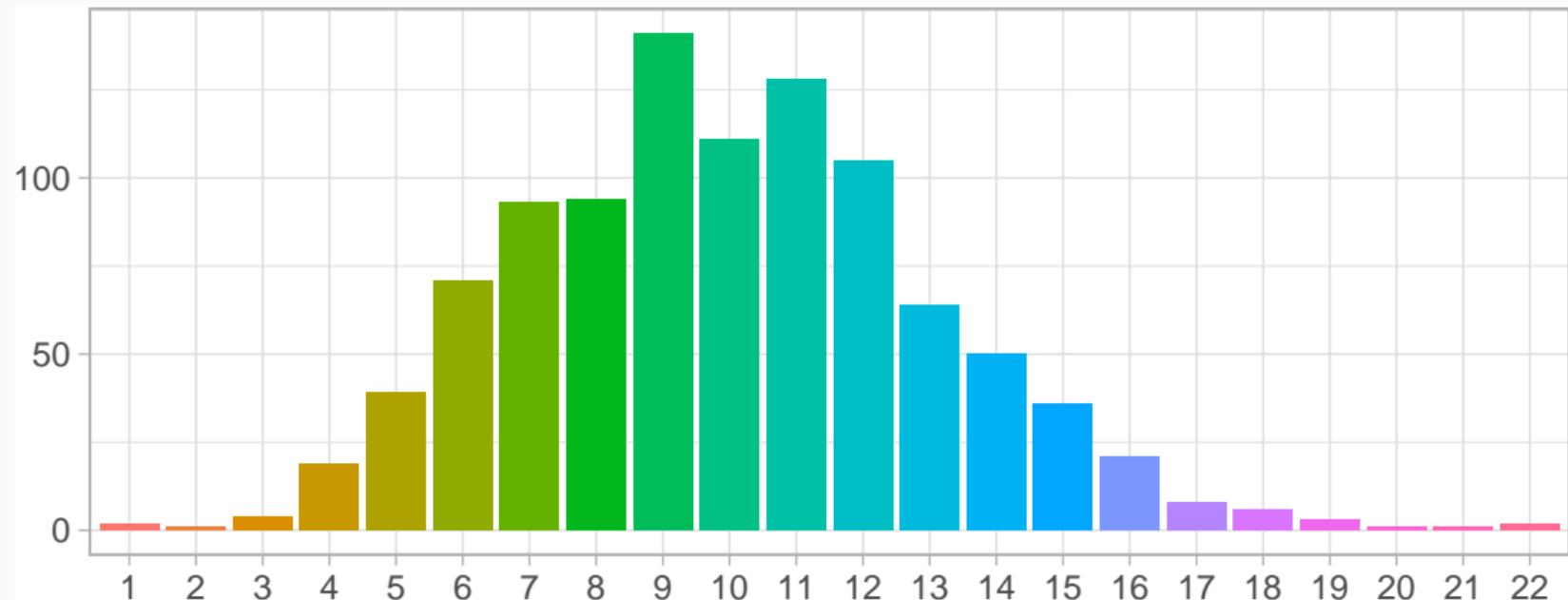
Hundred Poisson trials with $\lambda = 2$



Hundred Poisson trials with $\lambda = 10$



Thousand Poisson trials with $\lambda = 10$



Summary: Poisson distribution

- **notation:** $X \sim \text{Poisson}(\lambda)$
 - **range:** discrete, $x \geq 0$
 - **distribution:** $P(X = x) = \frac{e^{-\lambda} \lambda^x}{x!}$
 - **parameters:** λ the rate or expected number per sample
- **mean:** λ
 - **variance:** λ
 - **in R:** rpois, dpois

Continuous distribution

Normal (Gaussian) distribution

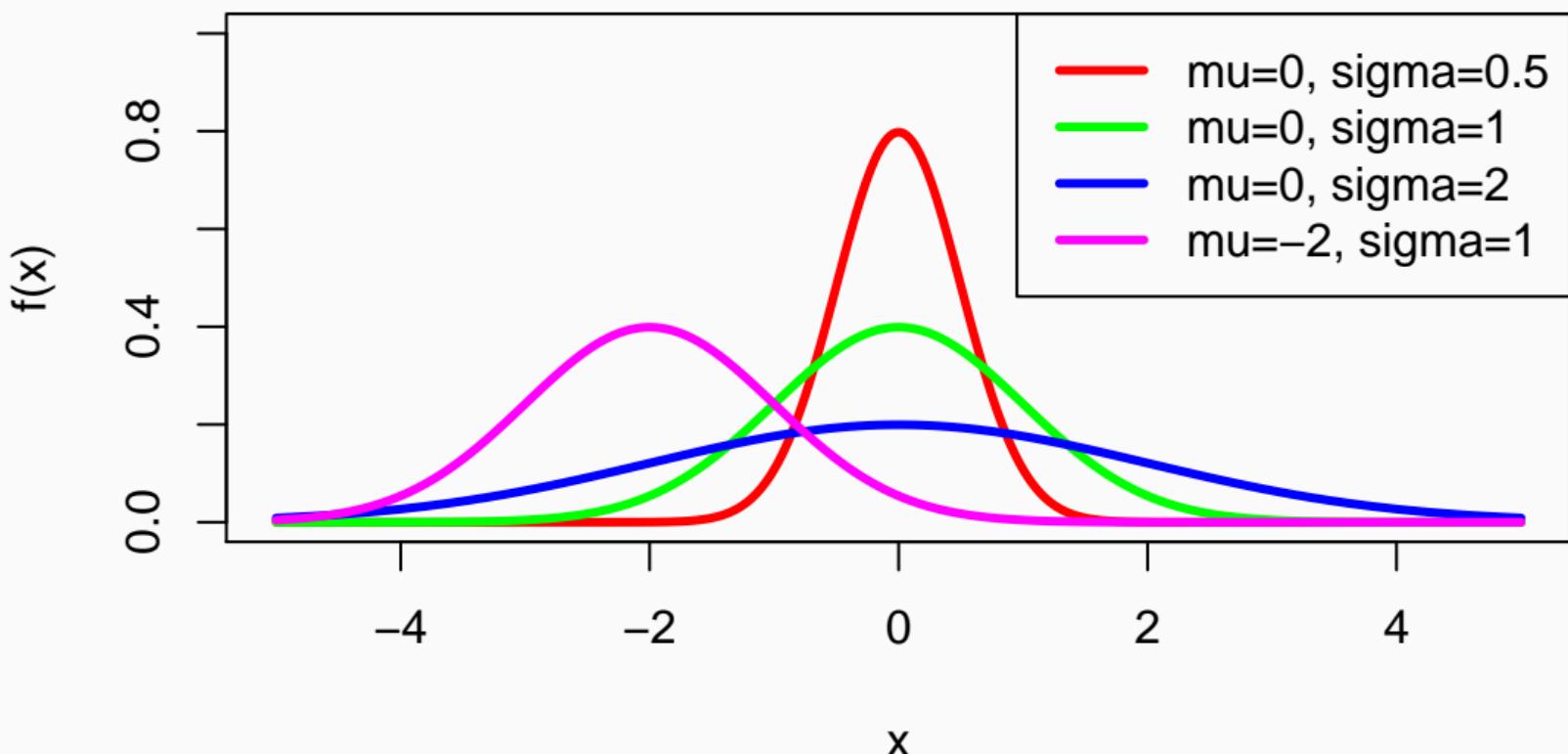
Context: Distribution of “adding lots of things together”. Derived from *Central Limit Theorem*, which says that if you add a large number of independent samples from the same distribution the distribution of the sum will be approximately normal.

$X \sim \text{Normal}(\mu, \sigma^2)$ where μ is the mean and σ^2 the variance

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

Example: Practically everything.

Normal probability density function



Summary: Normal distribution

- **notation:** $X \sim N(\mu, \sigma^2)$
 - **range:** continuous, all real values
- **distribution:** $f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$
 - **parameters:** μ the mean and σ the standard deviation
- **mean:** μ
 - **variance:** σ^2
 - **in R:** rnorm, dnorm

Why do we love the Normal distribution

- If has nice properties, such as: if $X \sim N(\mu, \sigma^2)$, then $Z = \frac{X - \mu}{\sigma} \sim N(0, 1)$
- It is a limiting distribution (*Central Limit Theorem*)
- It can be a good approximation for other distributions

Example: Approximating Binomial by Normal (1)

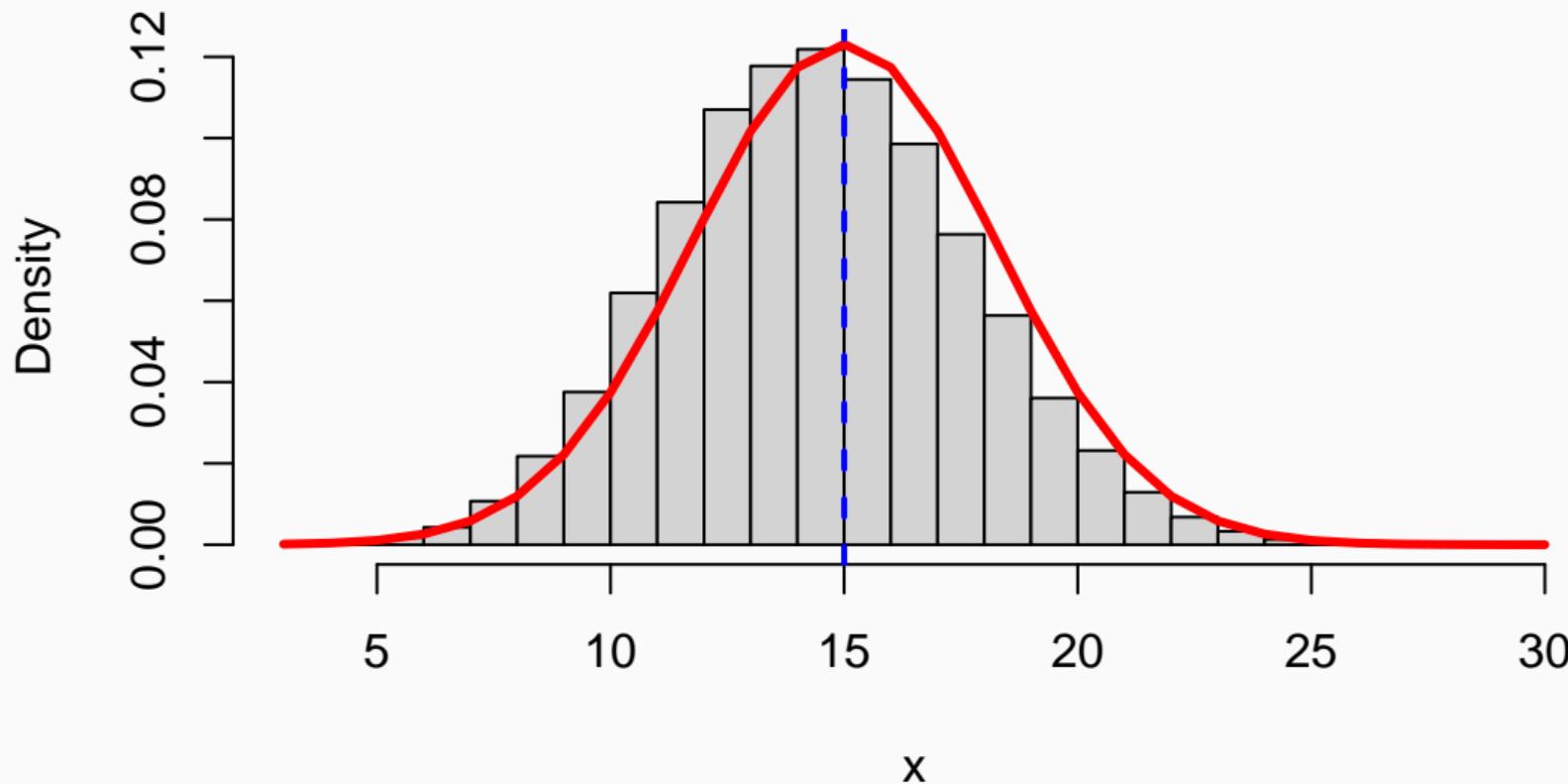
$X \sim \text{Bin}(N = 50, p = 0.3)$

Mean is $Np = 50 \times 0.3 = 15$

Variance is $Np(1 - p) = 50 \times 0.3 \times 0.7 = 10.5$

Therefore, X can be approximated by $Y \sim N(15, \sigma = \sqrt{10.5})$

Example: Approximating Binomial by Normal (2)



Conclusions about distributions

Common Distributions - Discrete

- When we have something that is dichotomous (either 0 or 1, negative/positive, false/true, male/female, present/absent):

Binomial(number of trials, probability)

- When we have something that is a discrete count, with no theoretical maximum, but with a common average:

Poisson(lambda)

Common Distributions - Discrete

- When we are recording the number of *failures* before a number of *successes*, or when we have something that is a discrete count with no theoretical maximum, and with more variation than Poisson:

NegativeBinomial(number of successes, probability of success)

NegativeBinomial(mean, overdispersion)

Common Distributions - Continuous

- When we have something that is continuous, symmetrical about the mean and unbounded:

Normal(mean, standard deviation)

- When we have something that is continuous, not symmetrical, and bounded at zero:

Exponential(rate)

Gamma(shape, rate)

Common Distributions - Continuous

- When we have something that is continuous, not symmetrical, and bounded at zero:

$\text{Lognormal}(\text{logmean}, \text{logstdev})$

- When we have something that is continuous, and bounded between 0 and 1:

$\text{Beta}(\text{alpha}, \text{beta})$

- Simple bounded distribution:

$\text{Uniform}(\text{min}, \text{max})$

More? Check out in R:

?Distributions

Likelihoods

Fitting distributions to data

- So far, when talking about probability distributions, we assumed that we knew the parameter values
- And we wanted to know what data we might get from these distributions
- In the real world, it is usually the other way around
- A more relevant question might be:

We have observed 3 births by a female during her 10 breeding attempts. What does this tell us about the true probability of getting a successful breeding attempt from this female? For the population?

Fitting distributions to data

We don't know what the probability of a birth is, but we can see what the probability of getting our data would be for different values:

```
dbinom(x = 3, size = 10, prob = 0.1)
## [1] 0.05739563
```

Fitting distributions to data

We don't know what the probability of a birth is, but we can see what the probability of getting our data would be for different values:

```
dbinom(x=3, size=10, prob=0.9)
## [1] 8.748e-06
```

Fitting distributions to data

We don't know what the probability of a birth is, but we can see what the probability of getting our data would be for different values:

```
dbinom(x=3,size=10,prob=0.25)
## [1] 0.2502823
```

So we would be more likely to observe 3 births if the probability is 0.25 than 0.1 or 0.9

The likelihood

- This reasoning is so common in statistics that it has a special name:
- **The likelihood** is the probability of observing the data under a certain model
- The data are known, we usually consider the likelihood as a function of the model parameters $\theta_1, \theta_2, \dots, \theta_p$

$$L = P(\theta_1, \theta_2, \dots, \theta_p \mid \text{data})$$

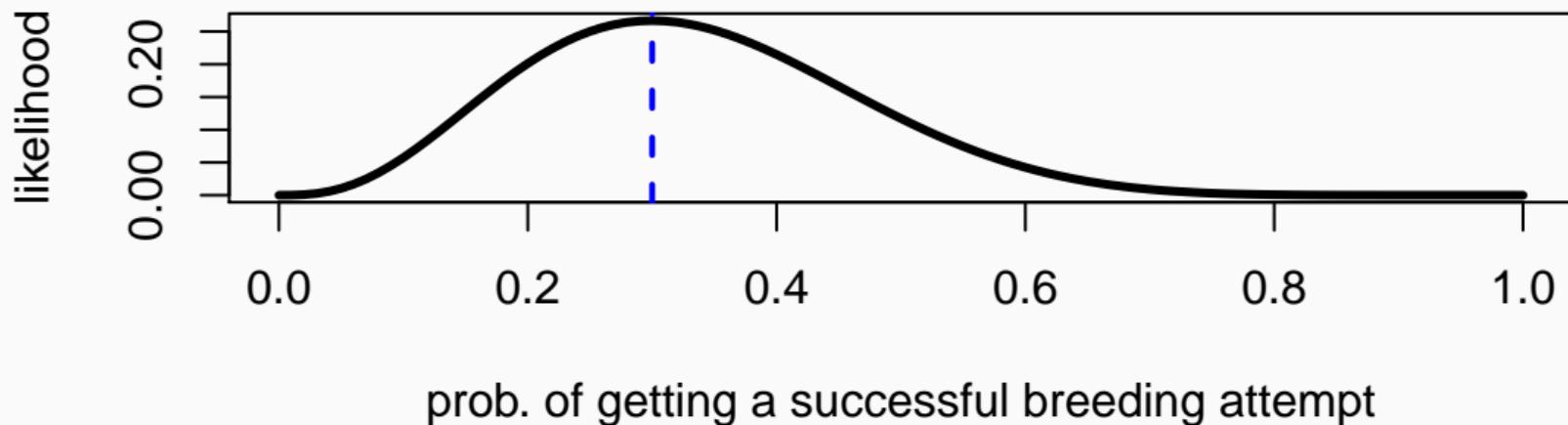
- This is a very important concept

Likelihood functions

We may create a function to calculate a likelihood e.g.:

```
lik.fun <- function(parameter){  
  ll <- dbinom(x=3, size=10, prob=parameter)  
  return(ll)  
}  
  
lik.fun(0.3)  
## [1] 0.2668279  
  
lik.fun(0.6)  
## [1] 0.04246733
```

Maximize the likelihood (3 successes out of 10 attempts)



The *maximum* of the likelihood is at value 0.3

The Maximum Likelihood

- There is always a set of parameters that gives you the highest likelihood of observing the data: the Maximum Likelihood Estimate(s) [MLEs]
- This can be calculated using:
- Trial and error (not efficient!)
- Compute the maximum of a function by hand (rarely doable in practice)
- An iterative optimization algorithm: `?optimize` (1 parameter) and `?optim` (> 1 parameter) in R

By hand: compute MLE of p from $Y \sim \text{Bin}(N = 10, p)$ with $k = 3$ successes

$$P(Y = k) = \binom{k}{N} p^k (1 - p)^{N - k} = L(p)$$

$$\log(L(p)) = \text{cte} + k \log(p) + (N - k) \log(1 - p)$$

We are searching for the maximum of L , or equivalently that of $\log(L)$

Compute derivate w.r.t. p : $\frac{d \log(L)}{dp} = \frac{k}{p} - \frac{(N - k)}{(1 - p)}$

Then solve $\frac{d \log(L)}{dp} = 0$; the MLE is $\hat{p} = \frac{k}{N} = \frac{3}{10} = 0.3$

Here, the MLE is the proportion of observed successes

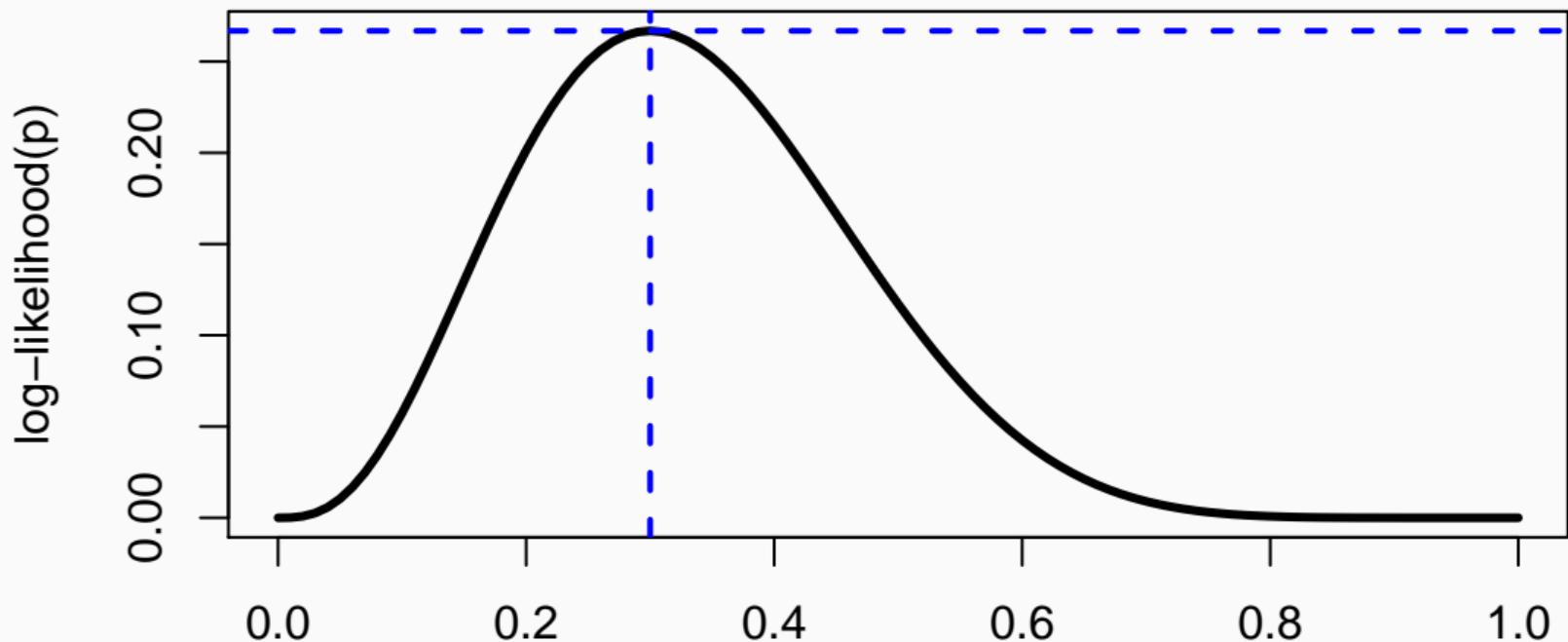
Using a computer: MLE of p from $Y \sim \text{Bin}(N = 10, p)$ with $k = 3$ successes

```
lik.fun <- function(parameter) dbinom(x=3, size=10, prob=parameter)
# ?optimize
optimize(lik.fun,c(0,1),maximum=TRUE)
## $maximum
## [1] 0.3000157
##
## $objective
## [1] 0.2668279
```

Use `optim` when the number of parameters is > 1 .

Using a computer: MLE of p from $Y \sim \text{Bin}(N = 10, p)$ with $k = 3$ successes

Binomial likelihood with 3 successes out of 10 attempts

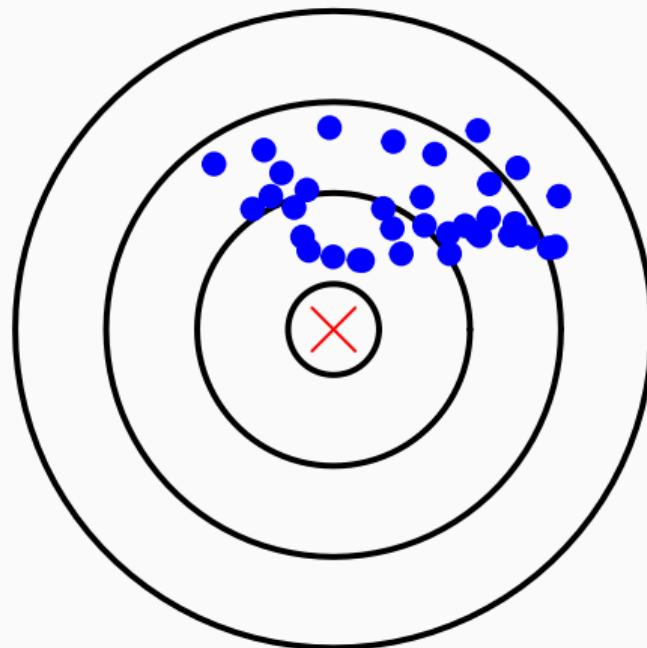


The Maximum Likelihood Estimate (MLE)

- The MLE is the best guess set of parameter values for our given data

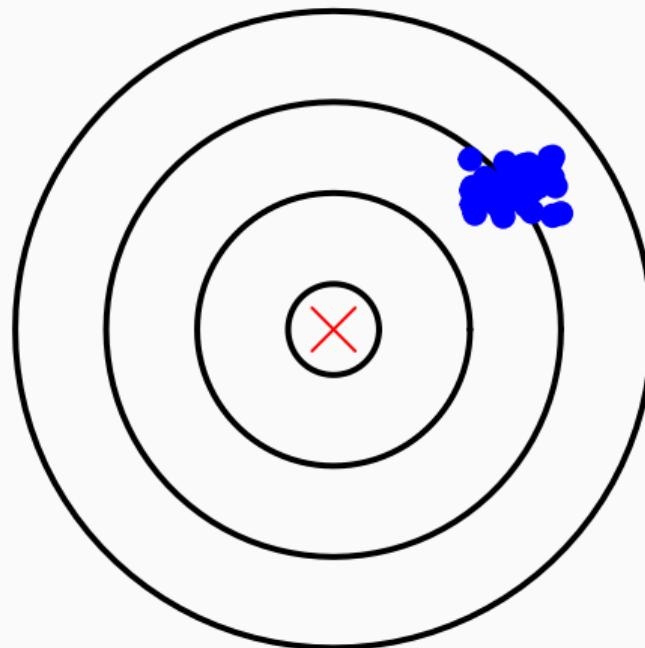
A dart target, with the red cross representing the true parameter value

Imprecise and biased



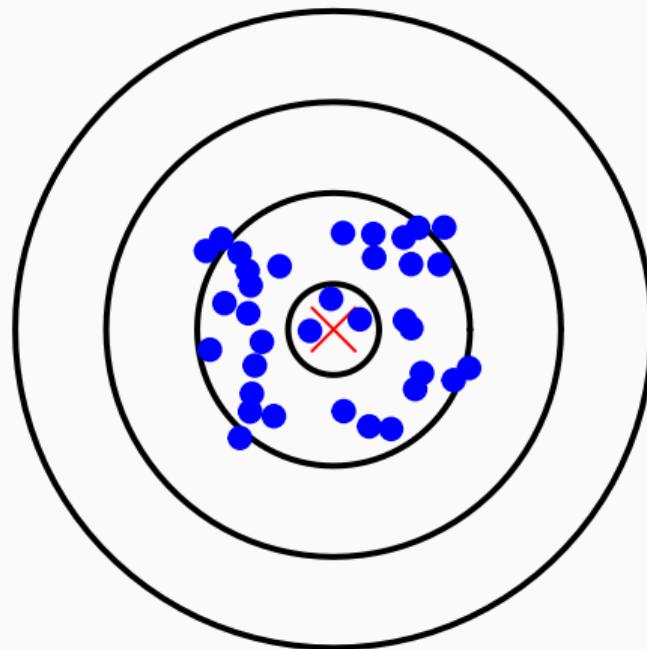
A dart target, with the red cross representing the true parameter value

Precise but biased



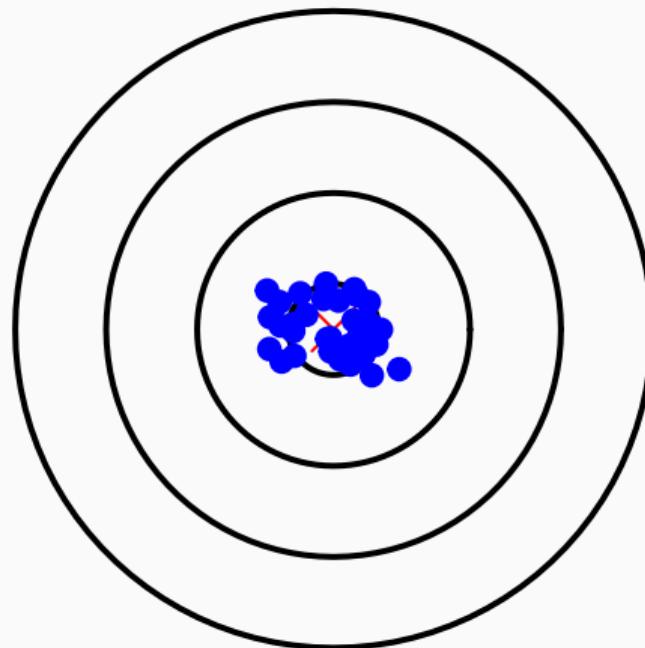
A dart target, with the red cross representing the true parameter value

Unbiased but imprecise



A dart target, with the red cross representing the true parameter value

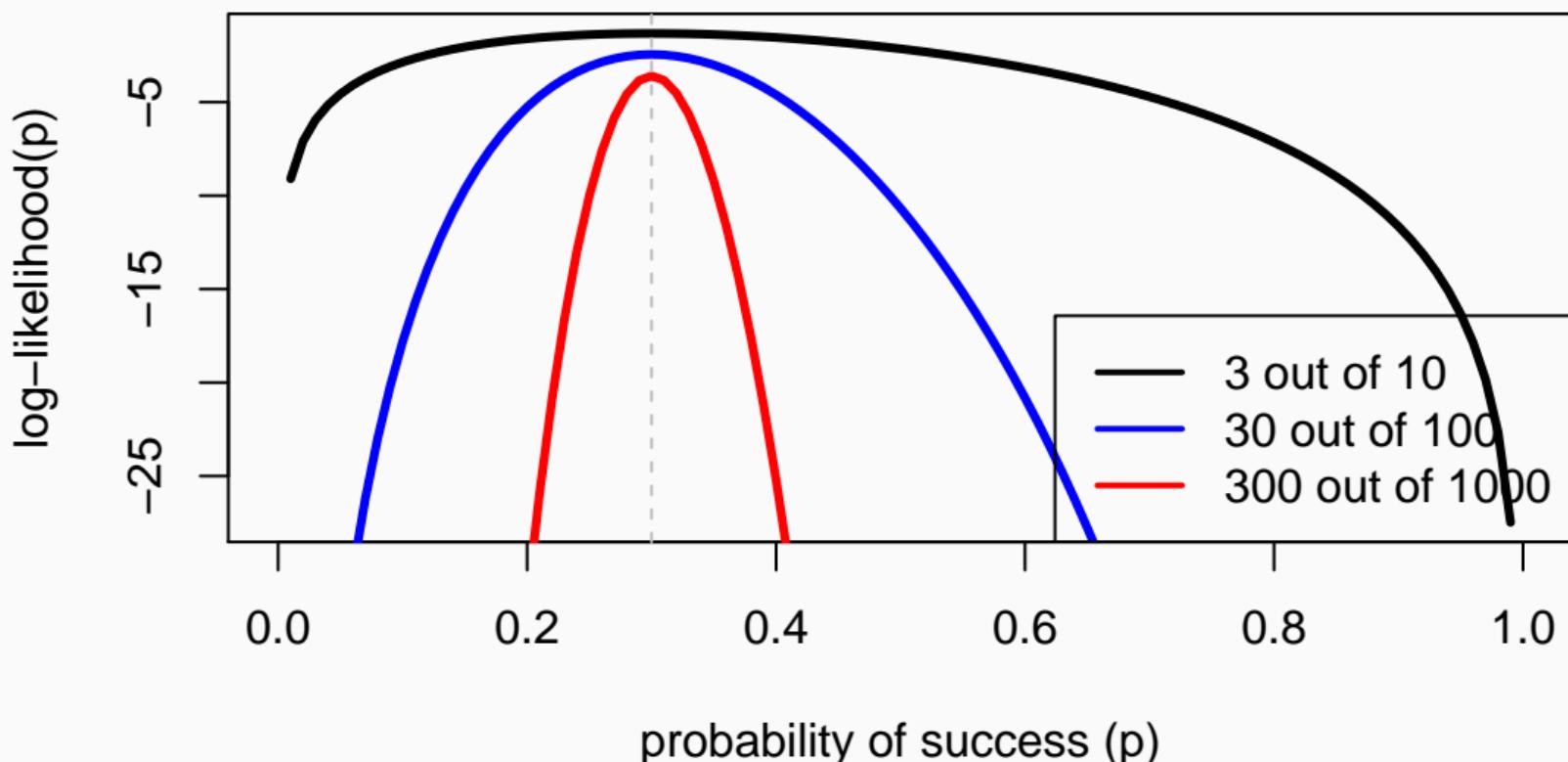
Unbiased and precise!



The Maximum Likelihood Estimate (MLE)

- The MLE is the best guess set of parameter values for our given data
- But the chances of the true parameter values being close to the MLE is dependent on the amount of information in the data!

Binomial likelihood with increasing sample size



Confidence intervals: A refresher

Let's approach confidence intervals through simulations

Imagine you are measuring the temperature of a cup of water 10 times but you have an old really bad thermometer. The true temperature is 3 degrees Celsius and the standard deviation on the sampling error is 5.

Simulate data:

```
mu <- 3  
sigma <- 5  
n <- 10  
y <- rnorm(n = n, mean = mu, sd = sigma)
```

```
y
```

```
## [1] 5.9276441 6.5473301 2.4534834 0.7325141 6.0294373 -6.0897798  
## [7] 6.1504928 1.6190795 1.5792013 -1.5966100
```

Apply linear regression

We will estimate a mean temperature by fitting an intercept only linear regression model:

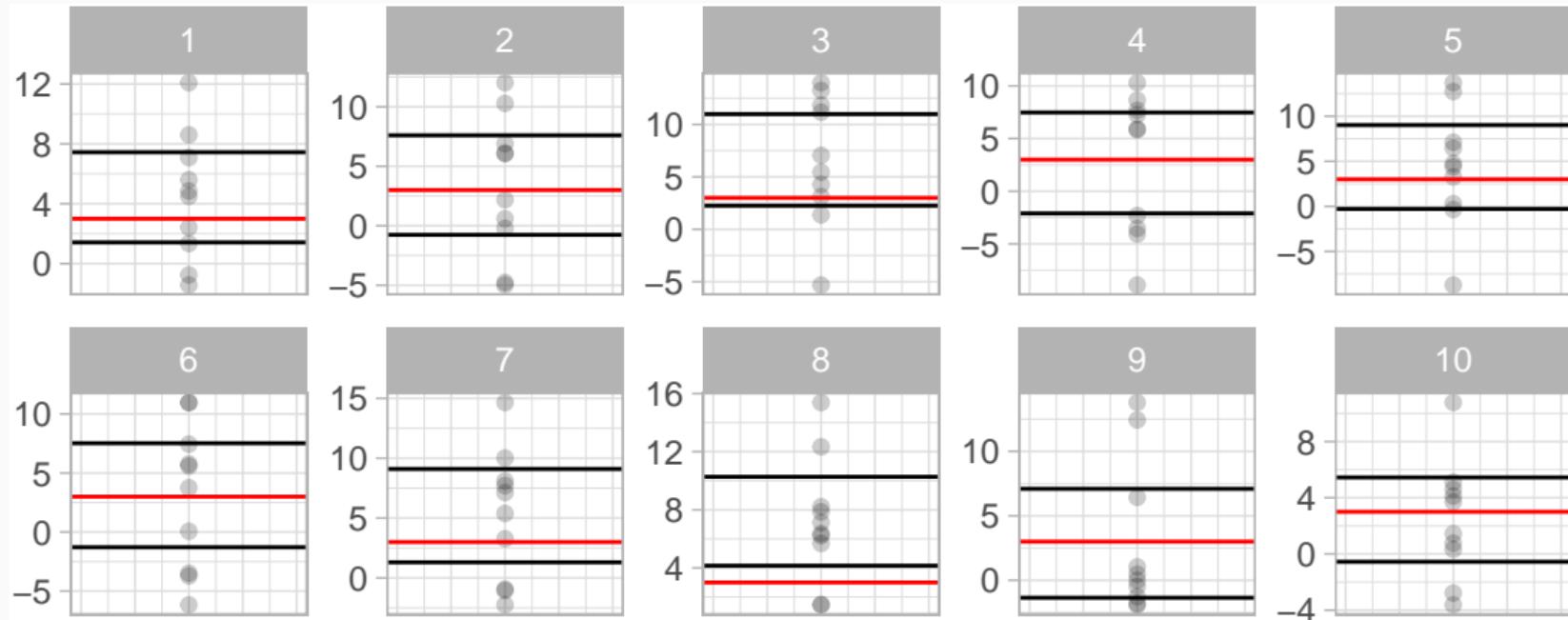
```
m <- lm(y~1)
broom::tidy(m)
## # A tibble: 1 x 5
##   term      estimate std.error statistic p.value
##   <chr>      <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  2.34      1.29      1.82    0.103

confint(m)
##                   2.5 %   97.5 %
## (Intercept) -0.5749909 5.245549
```

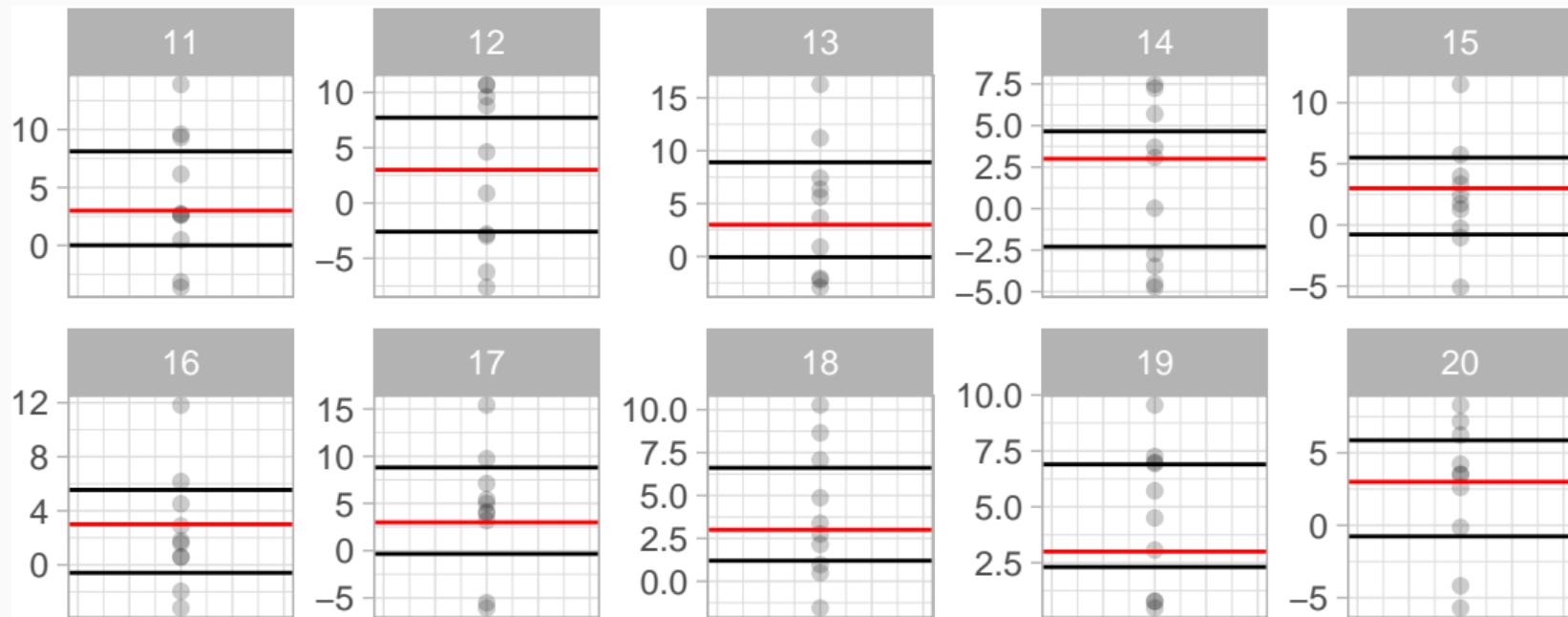
Let's illustrate what those confidence intervals really represent.

- Imagine you went outside 20 times and each time you measured the cup of water 10 times. Then you fitted a linear regression model with an intercept only each time and plotted the confidence intervals.
- 19 times out 20 (95%) the 95% confidence intervals should contain the true value.

Does that look approximately correct?



Does that look approximately correct?

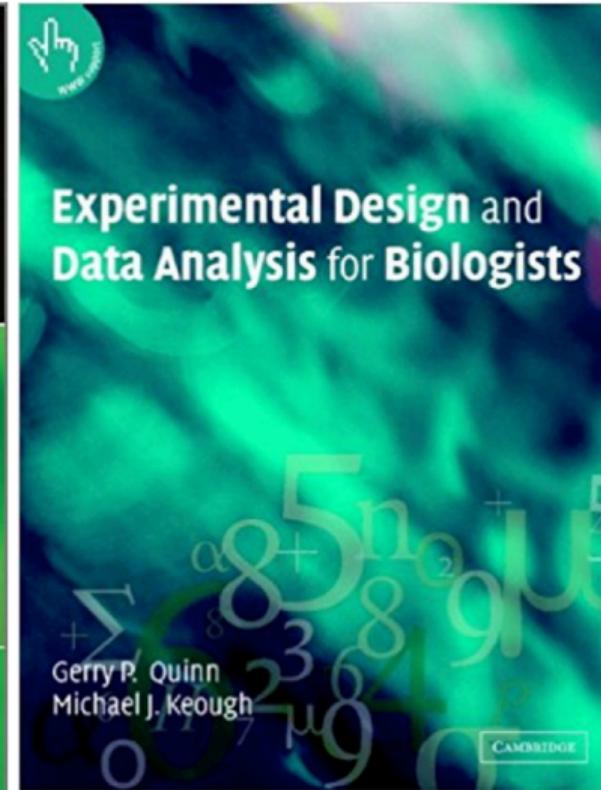
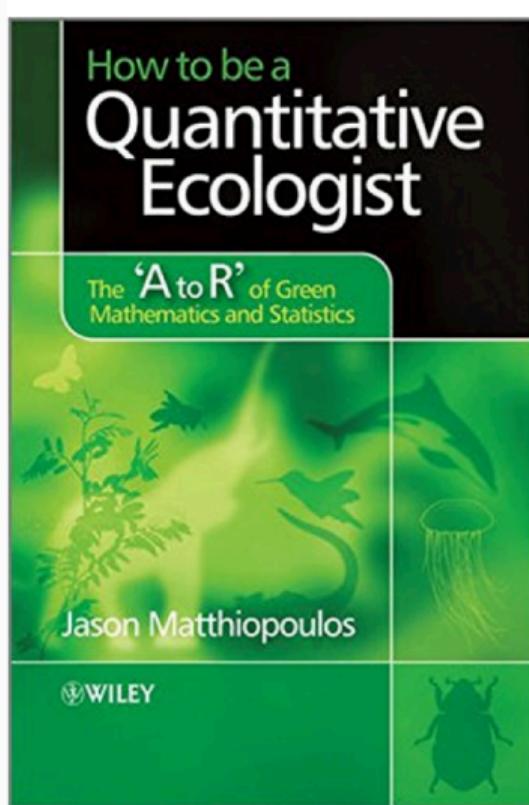


Confidence intervals are just **one realization** of a theoretically repeated experiment.

Likelihood key facts

- The likelihood is **the probability of observing a (fixed) dataset given a set of parameter values** (to be estimated)
- Maximum likelihood theory provides **estimates with optimal properties**: unbiased, minimal variance and normally distributed (asymptotically)
- The **rate of change of the likelihood** around the MLE is an indication of our confidence in the estimated parameter values
- Use **confidence intervals to capture uncertainty** surrounding MLEs
- Likelihood functions **can get very complicated!**

Textbooks



This Class

On our plate

- Distributions and likelihoods
- Hypothesis testing and multimodel inference
- Introduction to Bayesian inference
- Generalized Linear Models (GLMs)
- Generalized Additive Models (GAMs)
- Mixed Effect Models

Hypothesis testing: Rationale

The problem:

Suppose a coin toss turns up $k = 12$ heads out of $N = 20$ trials. Can we say that the coin toss is fair? Do we get more heads than expected (assuming the coin toss is fair)?

Hypothesis testing: Rationale

1. Define the null and alternative hypotheses. The null hypothesis is usually the one that represents the less complicated explanation of the real world
 - H_0 : the coin toss is fair
 - H_1 : the coin toss is unfair, we get more heads or tails than expected

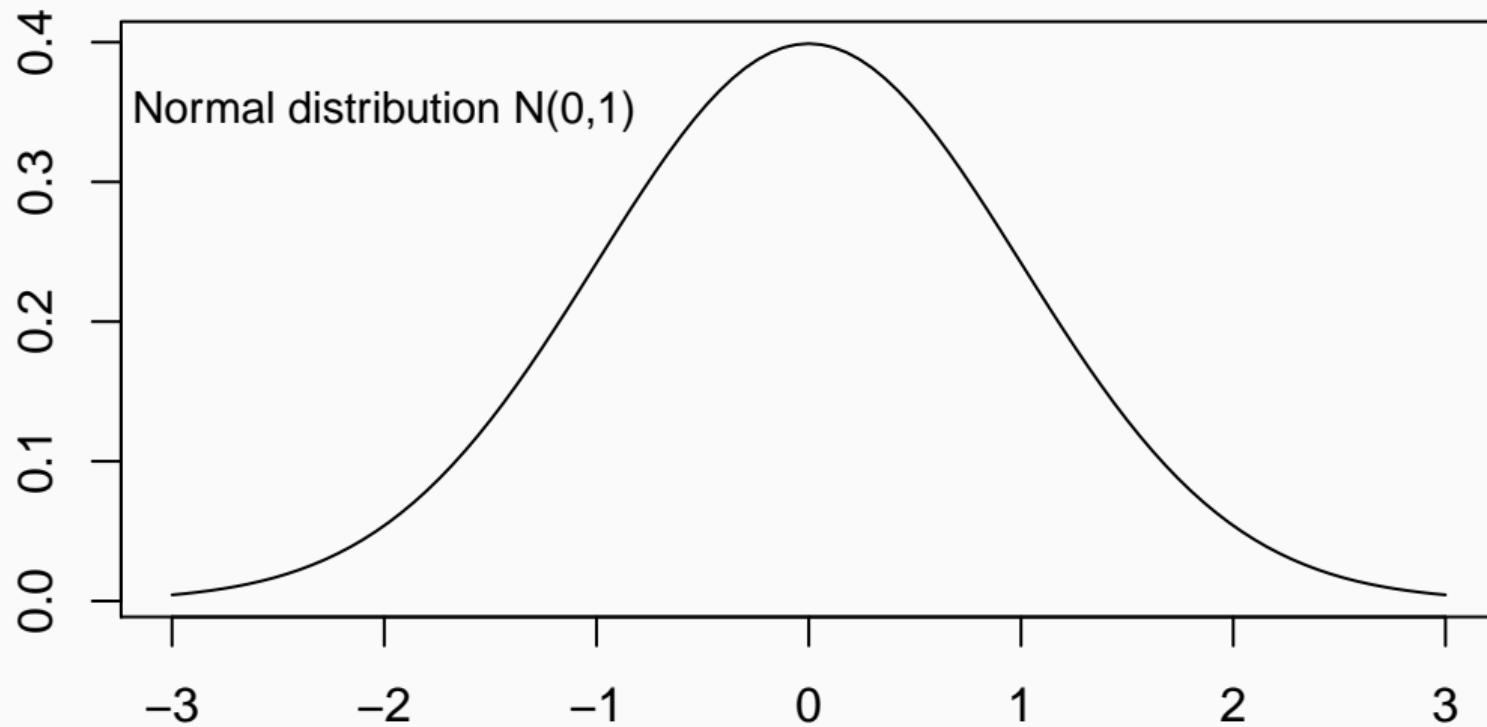
Hypothesis testing: Rationale

2. Construct a sampling distribution for the estimator under H_0
 - The number of heads X is a Binomial distribution with parameter p
 - Under H_0 , we have $p = p_0$ with $p_0 = 0.5$ if the coin toss is fair
 - Under H_1 , we have $p \neq p_0$
 - Remember that an estimator of p is the MLE $\hat{p} = k/N$, which is normally distributed with mean p and some variance; therefore, we have under H_0 :

$$\frac{\hat{p} - p_0}{\sqrt{p_0(1 - p_0)/N}} \sim \text{Normal}(0, 1)$$

Hypothesis testing: Rationale

2. Construct a sampling distribution for the estimator under H_0



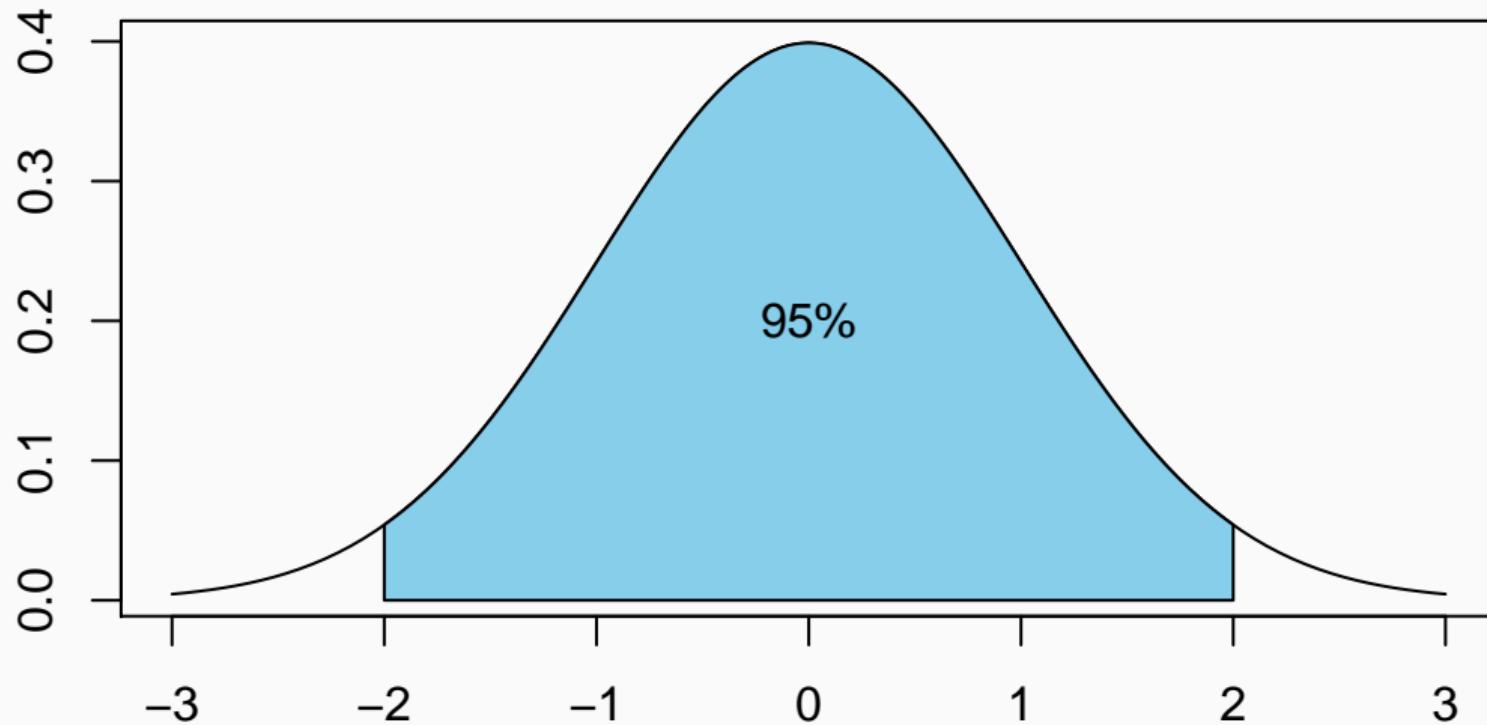
Hypothesis testing: Rationale

3. The sampling distribution will assign a likelihood to every possible value of the estimator. Very small values of likelihood – at the extremes of the sampling distribution – can be taken as evidence that the population generating the data has a parameter different to the one postulated by H_0 .

A probability value α is chosen to represent the level of significance required of the result. For example $\alpha = 0.05$ means that, under H_0 , the estimator will be found in the extreme regions of parameter space only five times every one hundred samples.

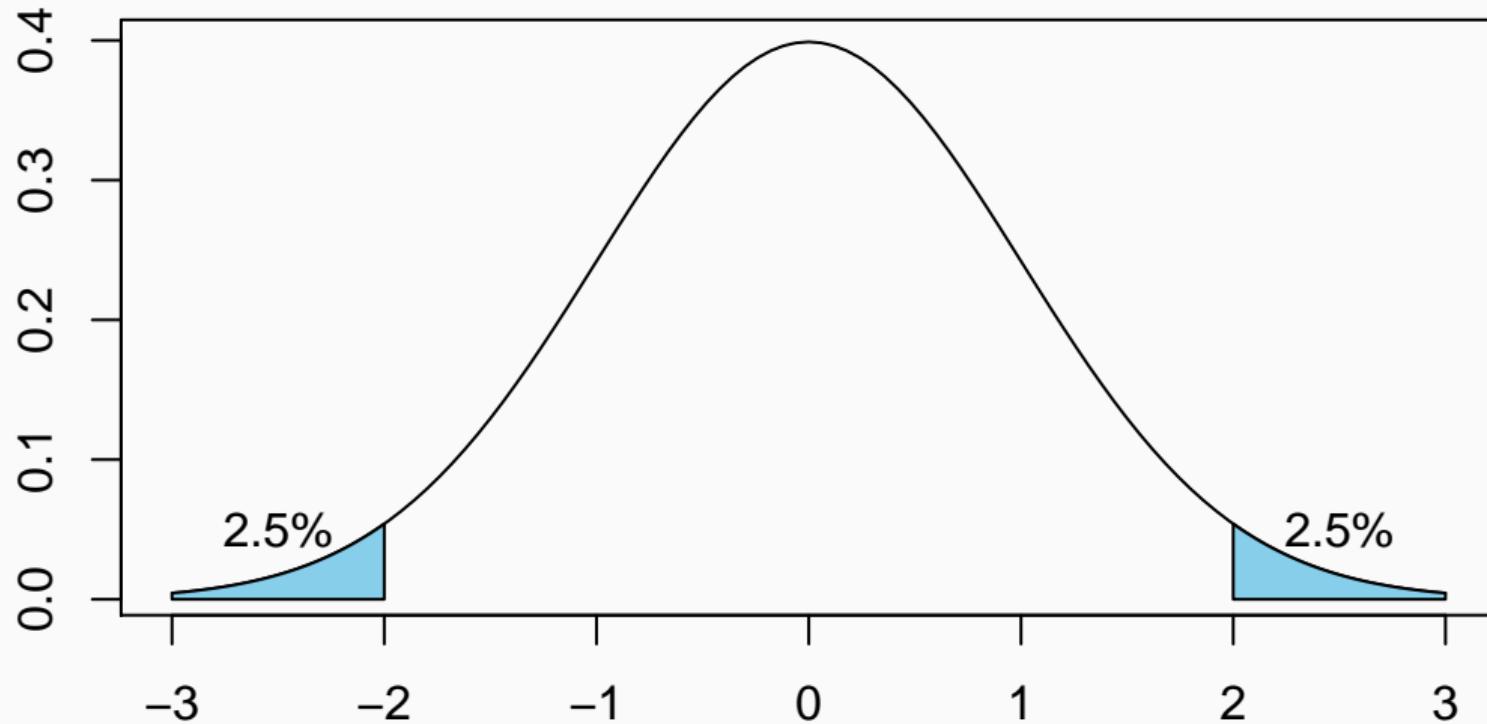
Hypothesis testing: Rationale

3. For example, say the level of significance is $\alpha = 0.05$



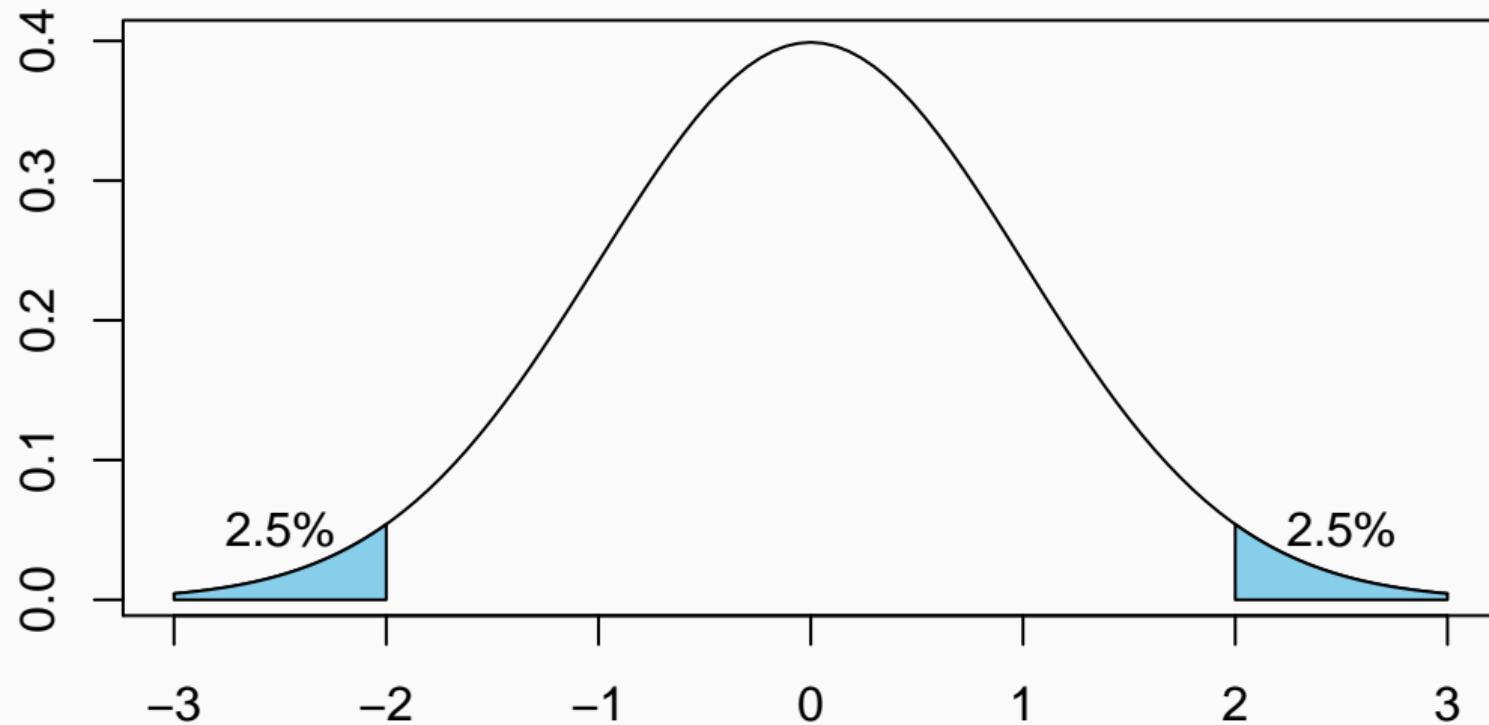
Hypothesis testing: Rationale

3. For example, say the level of significance is $\alpha = 0.05$



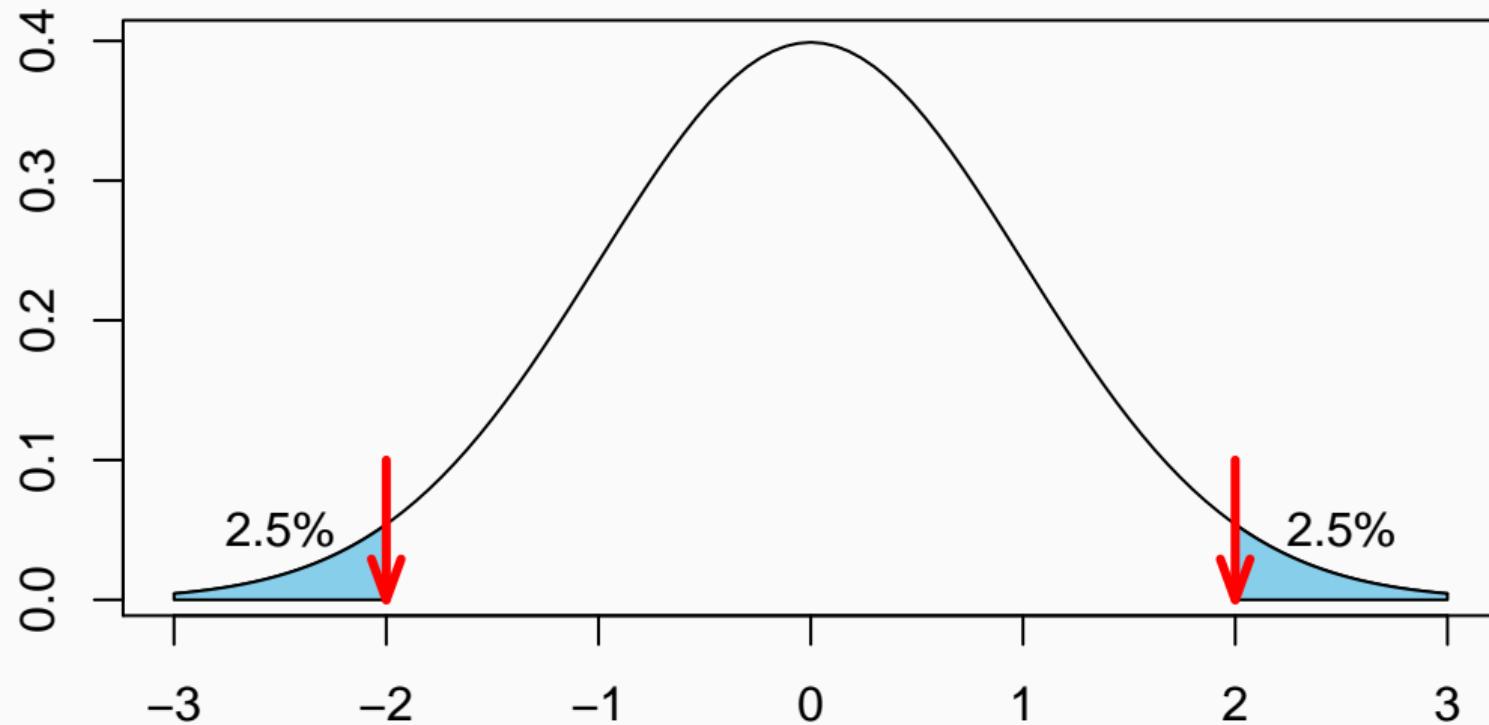
Hypothesis testing: Rationale

- Find the values that tell us if a particular estimate is extreme



Hypothesis testing: Rationale

- Find the values that tell us if a particular estimate is extreme



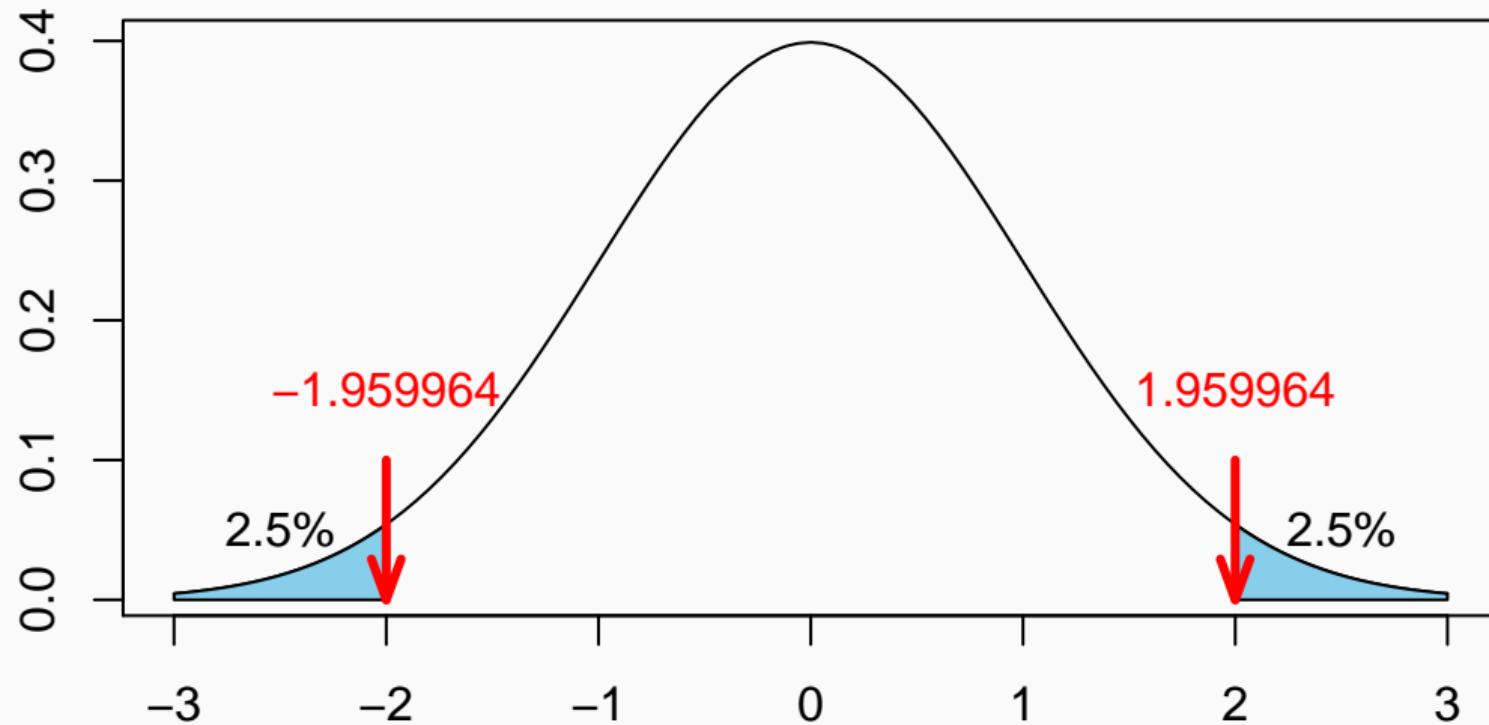
Hypothesis testing: Rationale

4. Find the values that tell us if a particular estimate is significantly different from what would be expected under H_0
 - Straightforward in R:

```
alpha <- 0.05
z.half.alpha <- qnorm(1 - alpha/2)
c(-z.half.alpha, z.half.alpha)
## [1] -1.959964  1.959964
```

Hypothesis testing: Rationale

- Find the values that tell us if a particular estimate is extreme



Hypothesis testing: Rationale

5. The estimate (\hat{q}) is calculated from the data and compared with the critical value(s)

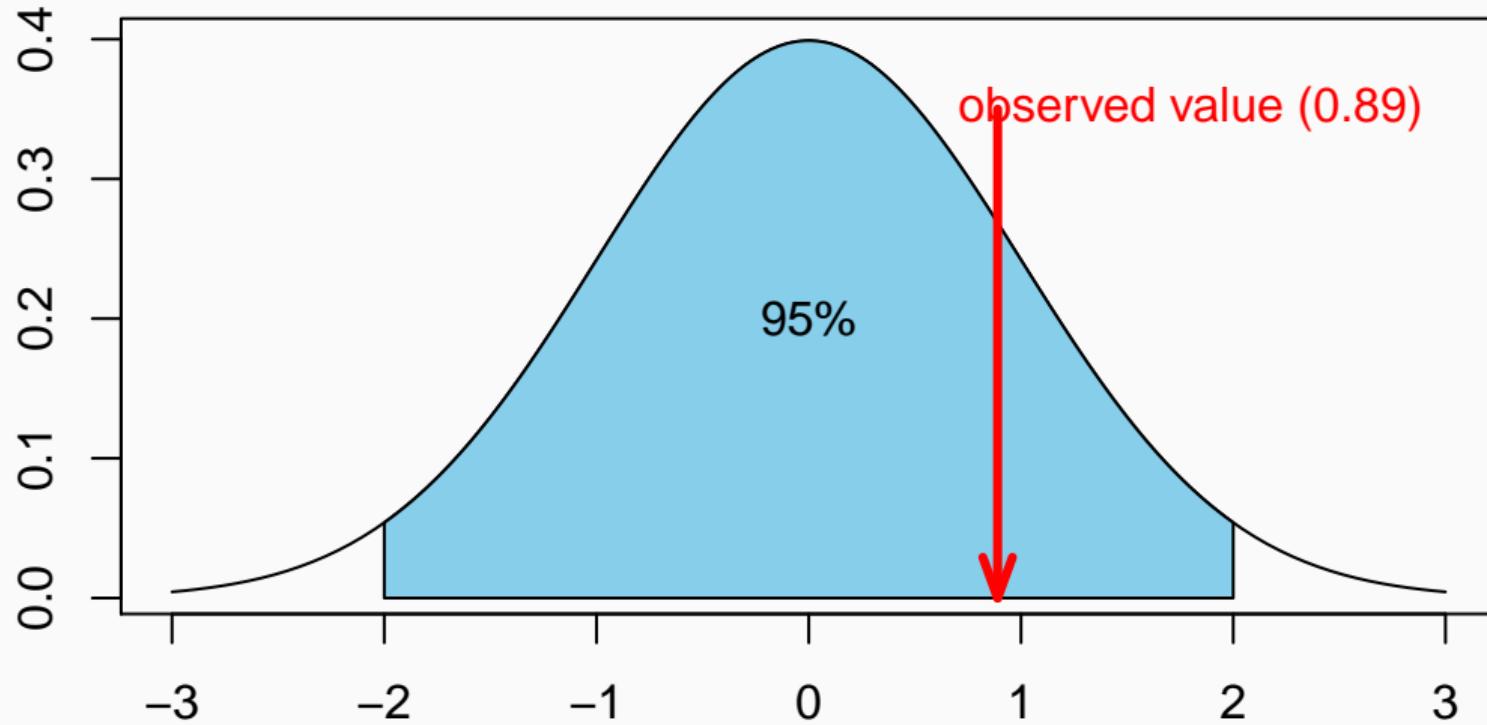
$$\frac{\hat{p} - p_0}{\sqrt{p_0(1 - p_0)/N}} = \frac{12/20 - 0.5}{\sqrt{0.5(1 - 0.5)/20}} = 0.89$$

Hypothesis testing: Rationale

6. If the estimate falls in the region of extreme values, then H_0 is rejected, otherwise we say that there is not enough evidence to reject it
- The test statistic 0.89 lies between the critical values -1.96 and 1.96. Hence, at $\alpha = 0.05$ significance level, we do not reject the null hypothesis that the coin toss is fair

Hypothesis testing: Rationale

6. If the estimate falls in the region of extreme values, then H_0 is rejected

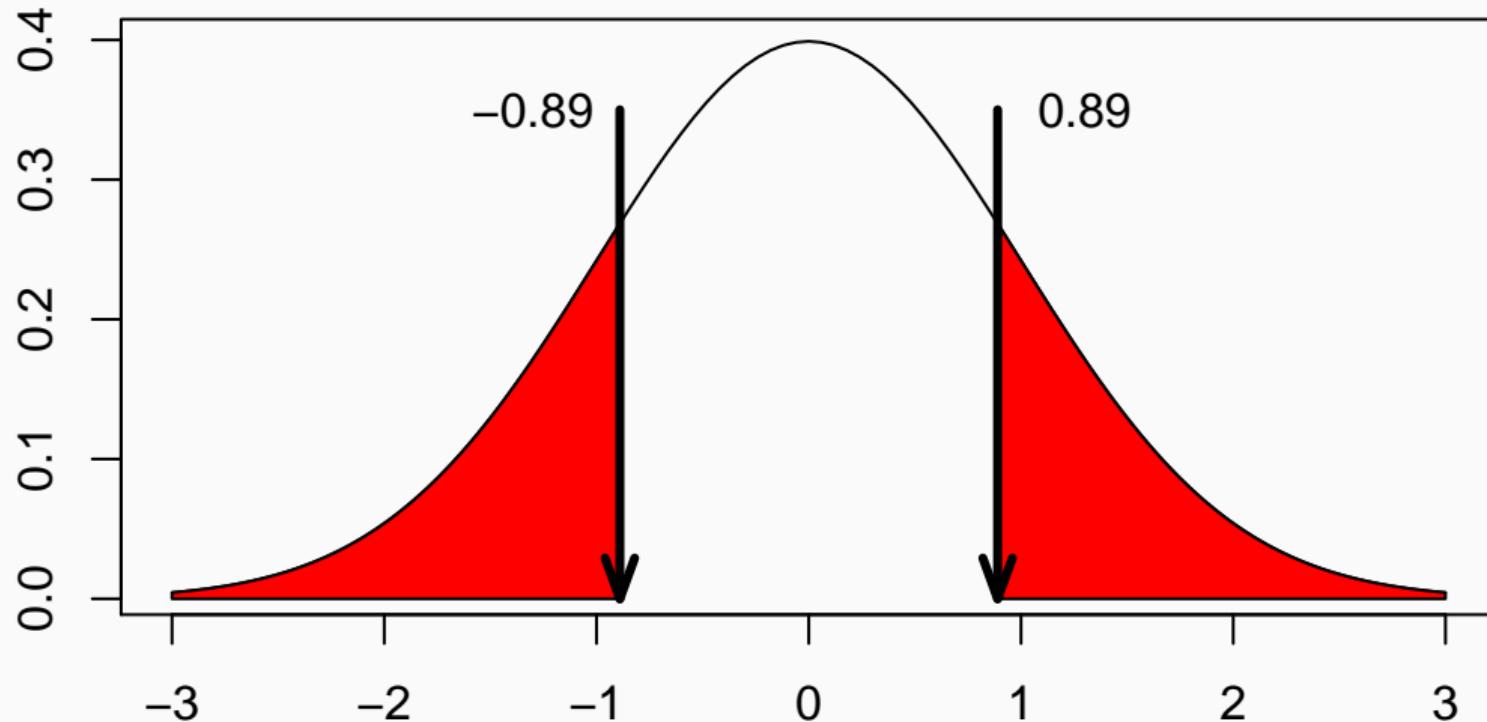


Hypothesis testing: Rationale

- Another way to test significance is to use the p-value
- Probability that, when H_0 is true, the value of the test statistics would be the same as or more extreme than the actual observed results
- If the p-value is $< \alpha$, then reject H_0

Hypothesis testing: Rationale

- The p-value is the red area

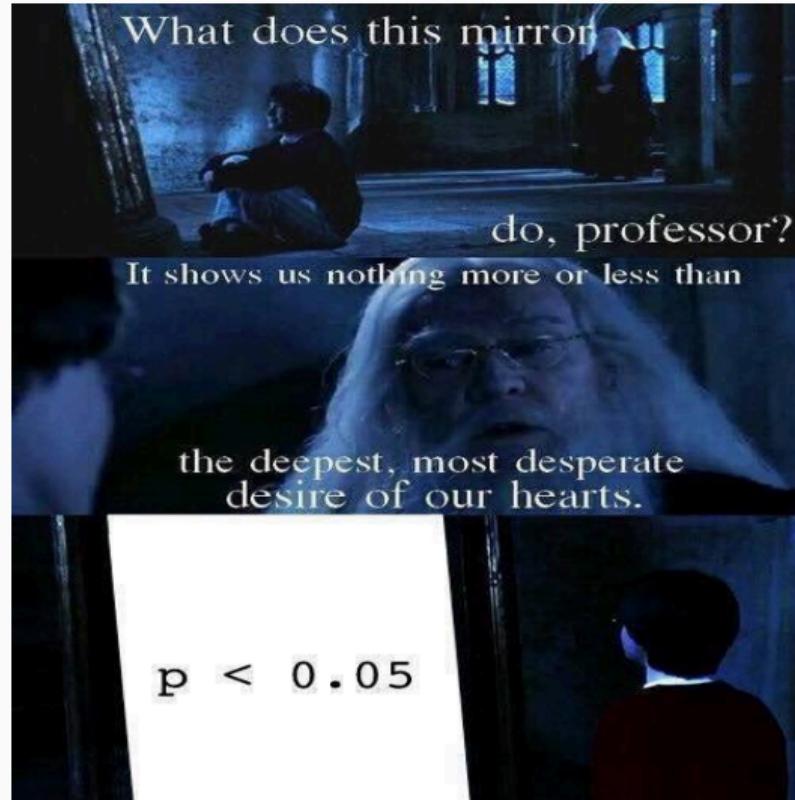


Hypothesis testing: Rationale

- To compute the p-value, we need $P(X \geq 0.89) + P(X \leq -0.89)$
- This can be obtained in R as follows:

```
pval <- 2 * (1 - pnorm(0.89)) # pnorm(x) = P(X <= x)
pval # two-tailed p-value
## [1] 0.3734659
```

Problems with hypothesis testing



Problems with hypothesis testing

- **Significance levels are arbitrary:** Changing α magically turns an ordinary result into something worth reporting.

Problems with hypothesis testing

- **Significance levels are arbitrary:** Changing α magically turns an ordinary result into something worth reporting.
- **Results are only qualitative:** We get an idea of whether H_0 is true but not how well supported it is by the data. The use of p-values as measures of evidence has also received criticism.

Problems with hypothesis testing

- **Significance levels are arbitrary:** Changing α magically turns an ordinary result into something worth reporting.
- **Results are only qualitative:** We get an idea of whether H_0 is true but not how well supported it is by the data. The use of p-values as measures of evidence has also received criticism.
- **Null hypotheses are guaranteed to be false:** In the sense that all models are wrong, no population parameter will ever be exactly the same as our expectations.

Problems with hypothesis testing

- **Significance levels are arbitrary:** Changing α magically turns an ordinary result into something worth reporting.
- **Results are only qualitative:** We get an idea of whether H_0 is true but not how well supported it is by the data. The use of p-values as measures of evidence has also received criticism.
- **Null hypotheses are guaranteed to be false:** In the sense that all models are wrong, no population parameter will ever be exactly the same as our expectations.
- **A significant result is guaranteed if the sample size is large enough:** α must be appropriately chosen in relation to sample size. There are methods for doing this, known as power analyses.

Problems with hypothesis testing

- **Significance levels are arbitrary:** Changing α magically turns an ordinary result into something worth reporting.
- **Results are only qualitative:** We get an idea of whether H_0 is true but not how well supported it is by the data. The use of p-values as measures of evidence has also received criticism.
- **Null hypotheses are guaranteed to be false:** In the sense that all models are wrong, no population parameter will ever be exactly the same as our expectations.
- **A significant result is guaranteed if the sample size is large enough:** α must be appropriately chosen in relation to sample size. There are methods for doing this, known as power analyses.
- **The dichotomy between a null/alternative hypotheses is limiting:** Why not look at several candidate values at the same time?

Multimodel inference

Linear regression example

Impact of climatic conditions on white stork breeding success



Linear regression example

Impact of climatic conditions on white stork breeding success

```
nb_young <- c(2.55,1.85,2.05,2.88,3.13,2.21,2.43,2.69,2.55,2.84,2.47,2.69,  
             2.52,2.31,2.07,2.35,2.98,1.98,2.53,2.21,2.62,1.78,2.30)  
temperature <- c(15.1,13.3,15.3,13.3,14.6,15.6,13.1,13.1,15.0,11.7,15.3,  
                 14.4,14.4,12.7,11.7,11.9,15.9,13.4,14.0,13.9,12.9,15.1,  
                 13.0)  
rainfall <- c(67,52,88,61,32,36,72,43,92,32,86,28,57,55,66,26,28,96,48,90,  
                86,78,87)  
lin_reg <- lm(nb_young ~ temperature + rainfall)
```

Linear regression example

Impact of climatic conditions on white stork breeding success

```
library(broom)

tidy(lin_reg) # elegant summary using broom package
## # A tibble: 3 x 5
##   term      estimate std.error statistic p.value
##   <chr>     <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  2.45      0.765     3.20    0.00446
## 2 temperature  0.0311    0.0547    0.568   0.576
## 3 rainfall     -0.00732   0.00290   -2.53   0.0201
```

How to select a best model?

- The proportion of explained variance R^2 is problematic, because the more variables you have, the bigger R^2 is
- Idea: **penalize models with too many parameters**

Akaike information criterion (AIC)

$$AIC = -2 \log(L(\hat{\theta}_1, \dots, \hat{\theta}_K)) + 2K$$

with L the likelihood and K the number of parameters θ_i

Akaike information criterion (AIC)

$$AIC = -2 \log(L(\hat{\theta}_1, \dots, \hat{\theta}_K)) + 2K$$

A measure of goodness-of-fit of the model to the data: the more parameters you have, the smaller the deviance is (or the bigger the likelihood is)

Akaike information criterion (AIC)

$$AIC = -2 \log(L(\hat{\theta}_1, \dots, \hat{\theta}_K)) + 2K$$

A **penalty**: twice the number of parameters K

Akaike information criterion (AIC)

- AIC makes the balance between *quality of fit* and *complexity* of a model
- Best model is the one with lowest AIC value
- Two models are difficult to distinguish if $\Delta AIC < 2$

Back to the linear regression example

Fit all candidate models on white stork data and get their AIC

```
linreg_temp_rain <- lm(nb_young ~ temperature + rainfall)
linreg_temp <- lm(nb_young ~ temperature)
linreg_rain <- lm(nb_young ~ rainfall)
linreg_null <- lm(nb_young ~ 1)
```

```
c(AIC(linreg_temp_rain),AIC(linreg_temp),AIC(linreg_rain),AIC(linreg_null))
## [1] 17.97668 22.34309 16.34487 20.44140
```

Looks as though model with rainfall has the lowest AIC

However, the model with both covariates has an AIC value within 2 units

Where to go from there?! **Multimodel inference**

Multimodel inference

AIC weights

- Let ΔAIC_i be the difference between AIC of model i and the lowest AIC (corresponding to the best model)
- Akaike weight w_i for model i gives the probability that model i is the best model

$$w_i = \frac{\exp\left(-\frac{1}{2}\Delta\text{AIC}_i\right)}{\sum_{n=1}^N \exp\left(-\frac{1}{2}\Delta\text{AIC}_i\right)}$$

AIC weights with R: Back to the stork example

Compute the weights:

```
library(bbmle)
AICtab(linreg_temp_rain,linreg_temp,linreg_rain,linreg_null,
       base = T, weights = T)
```

	AIC	dAIC	df	weight
## linreg_rain	16.3	0.0	3	0.617
## linreg_temp_rain	18.0	1.6	4	0.273
## linreg_null	20.4	4.1	2	0.080
## linreg_temp	22.3	6.0	3	0.031

Model averaging

- Model-averaged estimates are weighted averages (by the w_i) of the parameters from each of the models

$$\bar{\hat{\theta}}_j = \sum_{i=1}^K w_i \hat{\theta}_j(\text{model}_i)$$

Model-averaged estimate of rainfall effect, by hand (1)

```
tidy(linreg_temp_rain)

## # A tibble: 3 x 5
##   term      estimate std.error statistic p.value
##   <chr>     <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  2.45      0.765     3.20    0.00446
## 2 temperature  0.0311    0.0547    0.568   0.576
## 3 rainfall     -0.00732   0.00290   -2.53   0.0201

0.273 * (-0.007315652)
```

Model-averaged estimate of rainfall effect, by hand (2)

```
tidy(linreg_temp)
```

```
## # A tibble: 2 x 5
##   term      estimate std.error statistic p.value
##   <chr>      <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  2.18     0.849     2.57    0.0179
## 2 temperature  0.0183    0.0610    0.300    0.767
```

$$0.273 * (-0.007315652) + 0.031 * 0$$

Model-averaged estimate of rainfall effect, by hand (3)

```
tidy(linreg_rain)
```

```
## # A tibble: 2 x 5
##   term      estimate std.error statistic p.value
##   <chr>      <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  2.87     0.186     15.5  5.90e-13
## 2 rainfall     -0.00716   0.00284    -2.52 1.97e- 2
```

$$0.273 * (-0.007315652) + 0.031 * 0 + 0.617 * (-0.007163572)$$

Model-averaged estimate of rainfall effect, by hand (4)

```
tidy(linreg_null)
```

```
## # A tibble: 1 x 5
##   term      estimate std.error statistic p.value
##   <chr>     <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  2.43     0.0738    33.0  3.10e-20
```

$$0.273 * (-0.007315652) + 0.031 * 0 + 0.617 * (-0.007163572) + 0.080 * 0 \\ = -0.006417097$$

Model-averaging with R: Back to the stork example

Perform model averaging

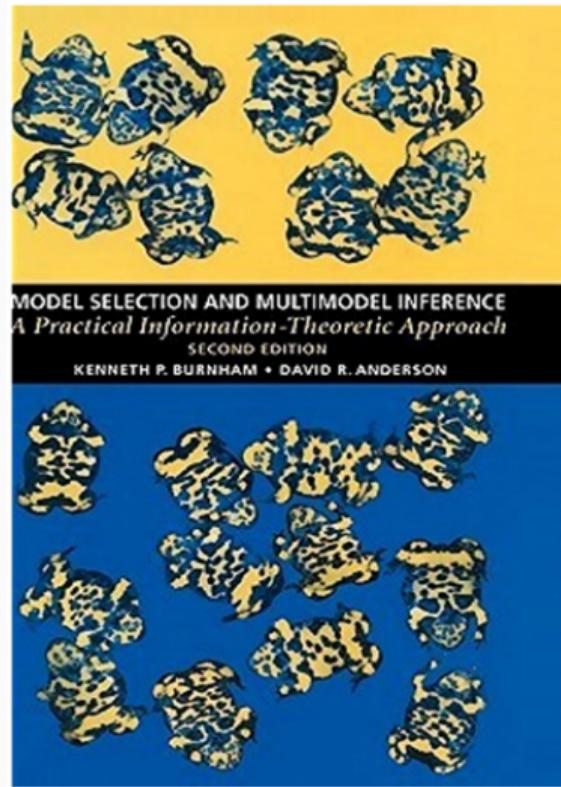
```
library(MuMIn)
m.ave <- model.avg(linreg_temp_rain,linreg_temp,linreg_rain,
                     linreg_null,rank = "AIC")
m.ave$coefficients
##           (Intercept)      rainfall temperature
## full      2.701251 -0.006414962 0.009038803
## subset    2.701251 -0.007210205 0.029776596
```

- The **full** average assumes that a variable is included in every model, but in some models the corresponding coefficient is set to zero.
- The **subset** (or **conditional**) average only averages over the models where the parameter appears.

Conclusions about multimodel inference

- Several models can be **ranked and weighted** to provide a quantitative measure of **relative support** for each competing hypothesis
- If there are two or more models with **similarly high levels of support**, **model averaging** of this 'top model set' provides a robust means of obtaining parameter estimates
- Acknowledge **uncertainty in the selection of a single best model**

Textbooks



This Class

On our plate

- Distributions and likelihoods
- Hypothesis testing and multimodel inference
- **Introduction to Bayesian inference**
- Generalized Linear Models (GLMs)
- Generalized Additive Models (GAMs)
- Mixed Effect Models

Bayesian inference

Introduction

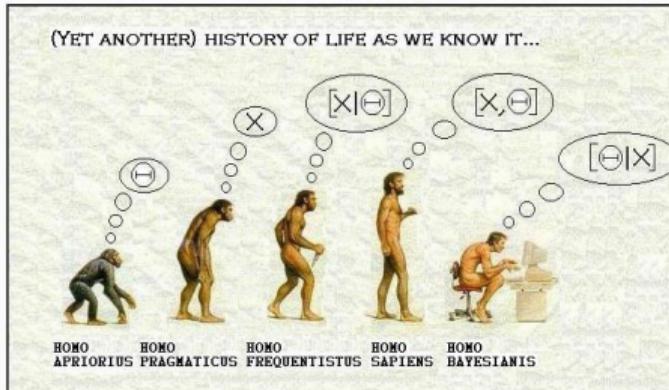
- The Bayesian approach dates back to 18th century to Reverend Thomas Bayes.



- However, due to practical problems of implementing the Bayesian approach, little advance was made for over two centuries.
- Recent advances in **computational power** coupled with the development of new methodology have led to a great increase in the application of Bayesian methods within the last two decades.

Classical versus Bayesian

- Typical stats problems involve estimating parameter θ with available data.
- The frequentist approach (maximum likelihood estimation – MLE) assumes that **the parameters are fixed, but have unknown values to be estimated.**
- Classical estimates generally provide a **point estimate** of the parameter of interest.
- The Bayesian approach assumes that **the parameters are not fixed but have some fixed unknown distribution** - a distribution for the parameter.



What is the Bayesian Approach?

- The approach is based upon the idea that the experimenter begins with **some prior beliefs** about the system.
- And then **updates** these beliefs on the basis of observed data.
- This updating procedure is based upon what is known as Bayes' Theorem:

$$\Pr(A | B) = \frac{\Pr(B | A) \Pr(A)}{\Pr(B)}$$

What is the Bayesian Approach?

- Schematically, if $A = \theta$ and $B = \text{data}$
- The Bayes' Theorem

$$\Pr(A | B) = \frac{\Pr(B | A) \Pr(A)}{\Pr(B)}$$

- Translates into:

$$\Pr(\theta | \text{data}) = \frac{\Pr(\text{data} | \theta) \ Pr(\theta)}{\Pr(\text{data})}$$

Bayes formula

$$\Pr(\theta \mid \text{data}) = \frac{\Pr(\text{data} \mid \theta) \Pr(\theta)}{\Pr(\text{data})}$$

- **Posterior distribution:** the basis for inference, a distribution, possibly multivariate if more than one parameter (θ)
- **Likelihood:** we know that guy from before, same as in the MLE approach
- **Prior distribution:** the source of much discussion about the Bayesian approach
- $\Pr(\text{data}) = \int L(\text{data} \mid \theta) \Pr(\theta) d\theta$: possibly high-dimensional integral, difficult if not impossible to calculate. This is one of the reasons why we need simulation (MCMC) methods - more soon.

A Simple Example

- Let us take a simple example to fix ideas
- 120 deer were radio-tracked over winter
- 61 close to a plant, 59 far from any human activity
- Question: is there a treatment effect on survival?

	Released	Alive	Dead	Other
treatment	61	19	38	4
control	59	21	38	0

A Simple Example

- So, $n = 57$ deer were assigned to the treatment group of which $k = 19$ survived the winter.
- Of interest is the probability of **over-winter survival**, call it θ , for the general population within the treatment area.
- The obvious estimate is simply to take the ratio $k/n = 19/57$.
- How would the classical statistician justify this estimate?

A Simple Example

- Our model is that we have a Binomial experiment (assuming independent and identically distributed draws from the population)
- K the number of alive individuals at the end of the winter, so that
$$P(K = k) = \binom{n}{k} \theta^k (1 - \theta)^{n-k}$$
- The classical approach is to maximise the corresponding likelihood with respect to θ to obtain the entirely plausible MLE:

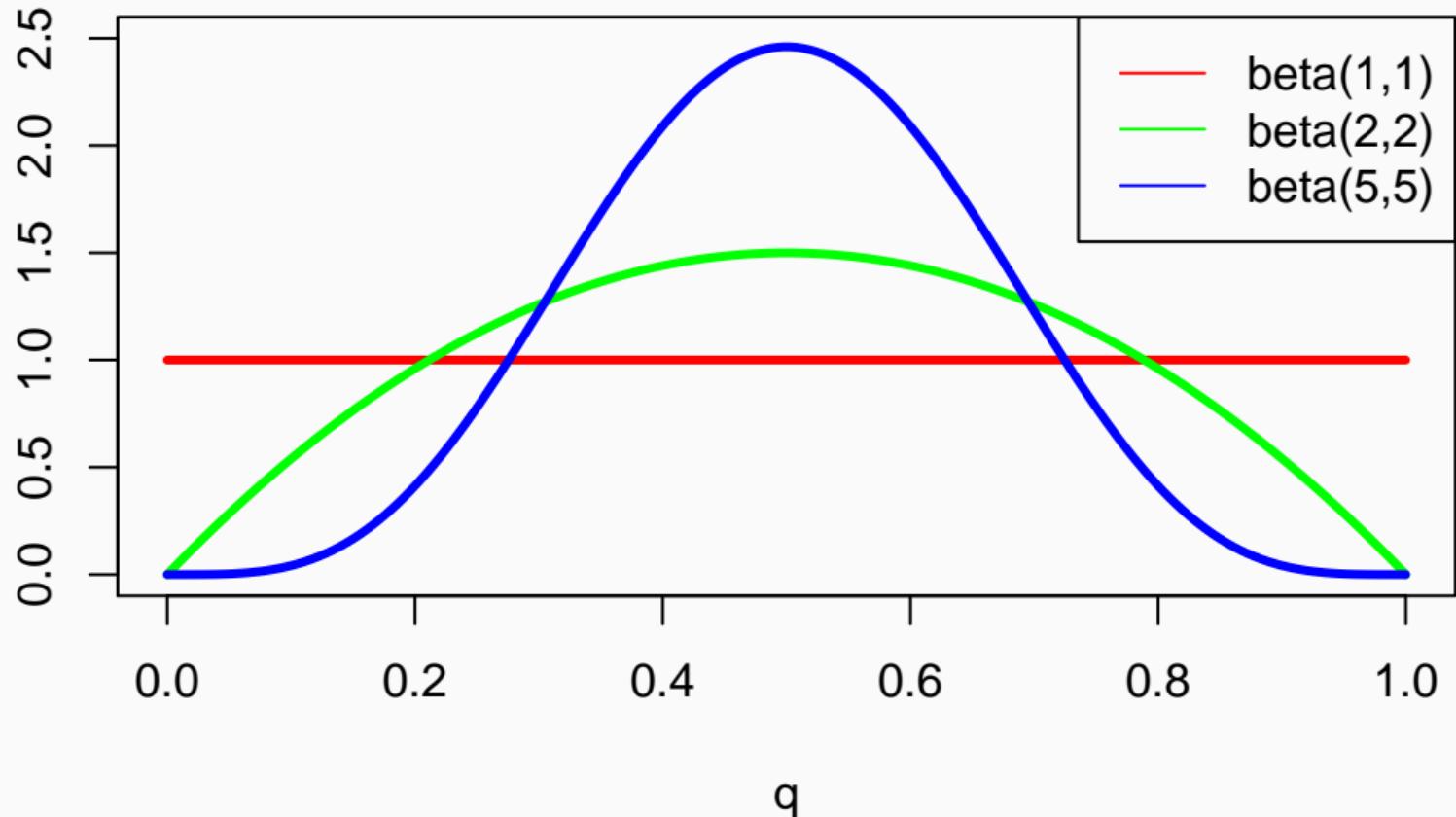
$$\hat{\theta} = k/n = 19/57$$

- Remember lecture on likelihoods

The Bayesian Approach

- The Bayesian starts off with **a prior**.
- Now, the one thing we know about θ is that it is a continuous random variable and that it lies between zero and one.
- Thus, a suitable prior distribution might be the Beta which is defined on this range $[0, 1]$.
- What is the Beta distribution?

What is the Beta distribution?



The Bayesian Approach

- Suppose we assume a priori that $\theta \sim Beta(a, b)$ so that $Pr(\theta) = \theta^{a-1}(1 - \theta)^{b-1}$
- Then we have:

$$\begin{aligned} Pr(\theta | k) &\propto \binom{n}{k} \theta^k (1 - \theta)^{n-k} \theta^{a-1} (1 - \theta)^{b-1} \\ &\propto \theta^{(a+k)-1} (1 - \theta)^{(b+n-k)-1} \end{aligned}$$

- That is:

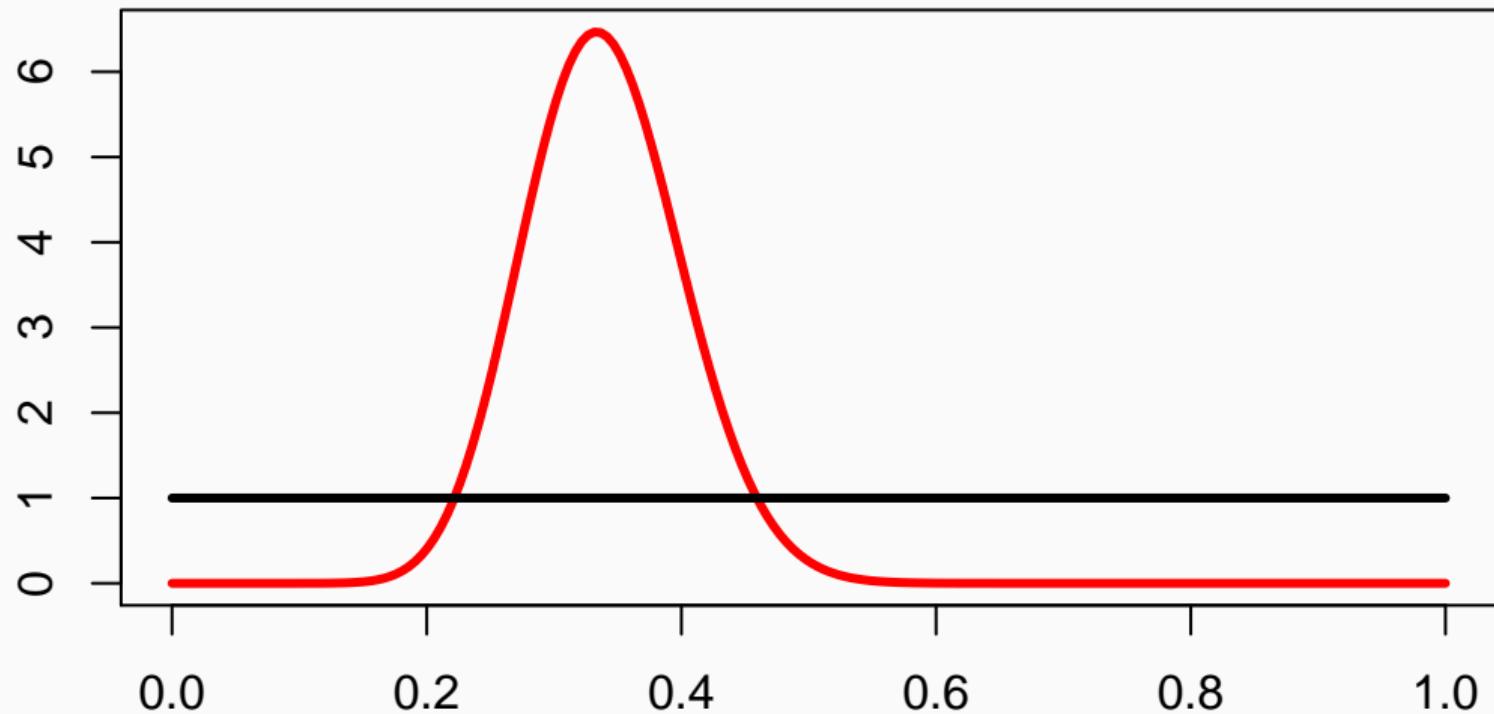
$$\theta | k \sim Beta(a + k, b + n - k)$$

- Take a Beta prior with a Binomial likelihood, you get a Beta posterior (conjugacy)

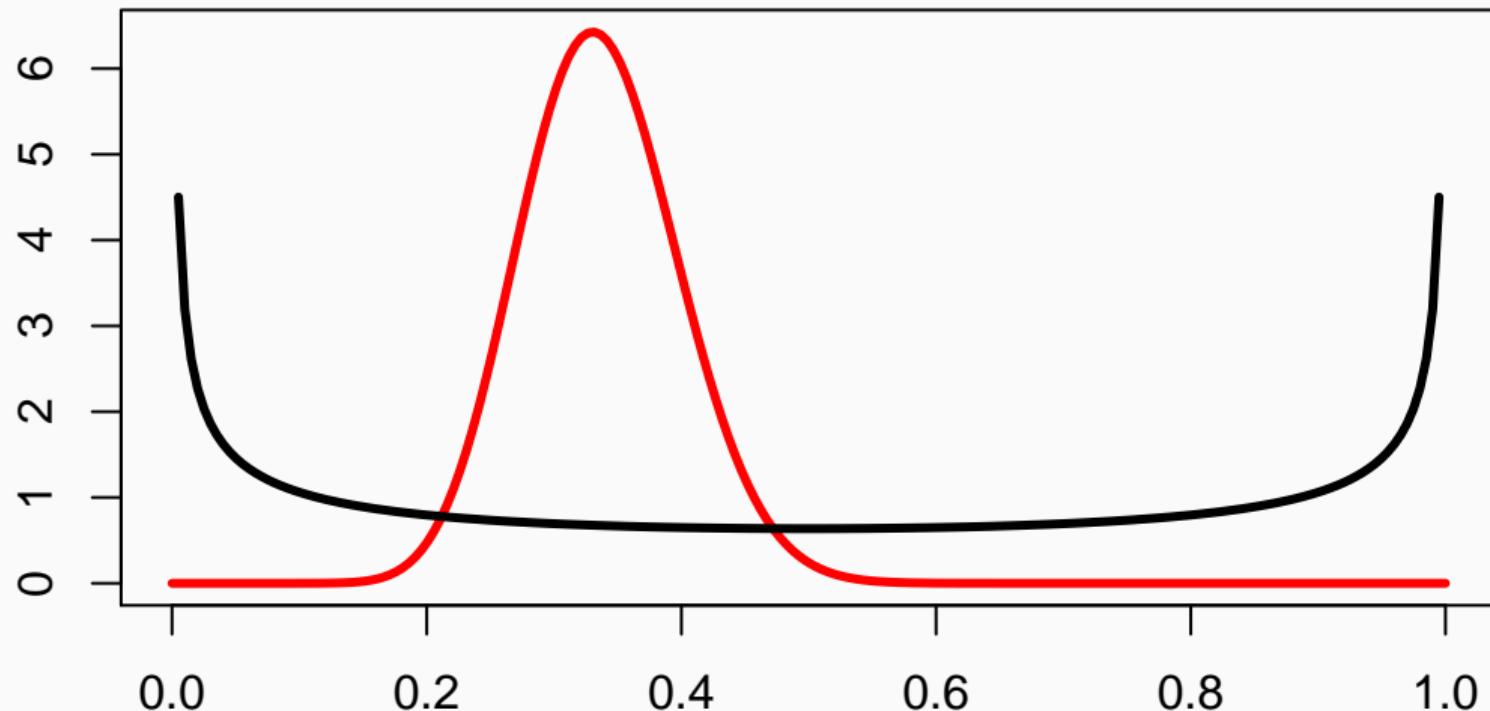
Application to the deer example

- We have that survival $\theta \mid k \sim Beta(a + k, b + n - k)$
- The posterior has an **explicit expression**, easy to manipulate
- $E(\theta \mid k) = \frac{a + k}{n + a + b}$
- $V(\theta \mid k) = \frac{(a + k)(b + n - k)}{(n + a + b)^2(n + a + b + 1)}$
- If we take a Uniform prior, i.e. $Beta(1, 1)$, then we have
- $\theta_{treatment} \sim Beta(1 + 19, 1 + 57 - 19) = Beta(20, 39)$
- $E(\theta_{treatment}) = 0.339$ and $V(\theta_{treatment}) = 0.061$

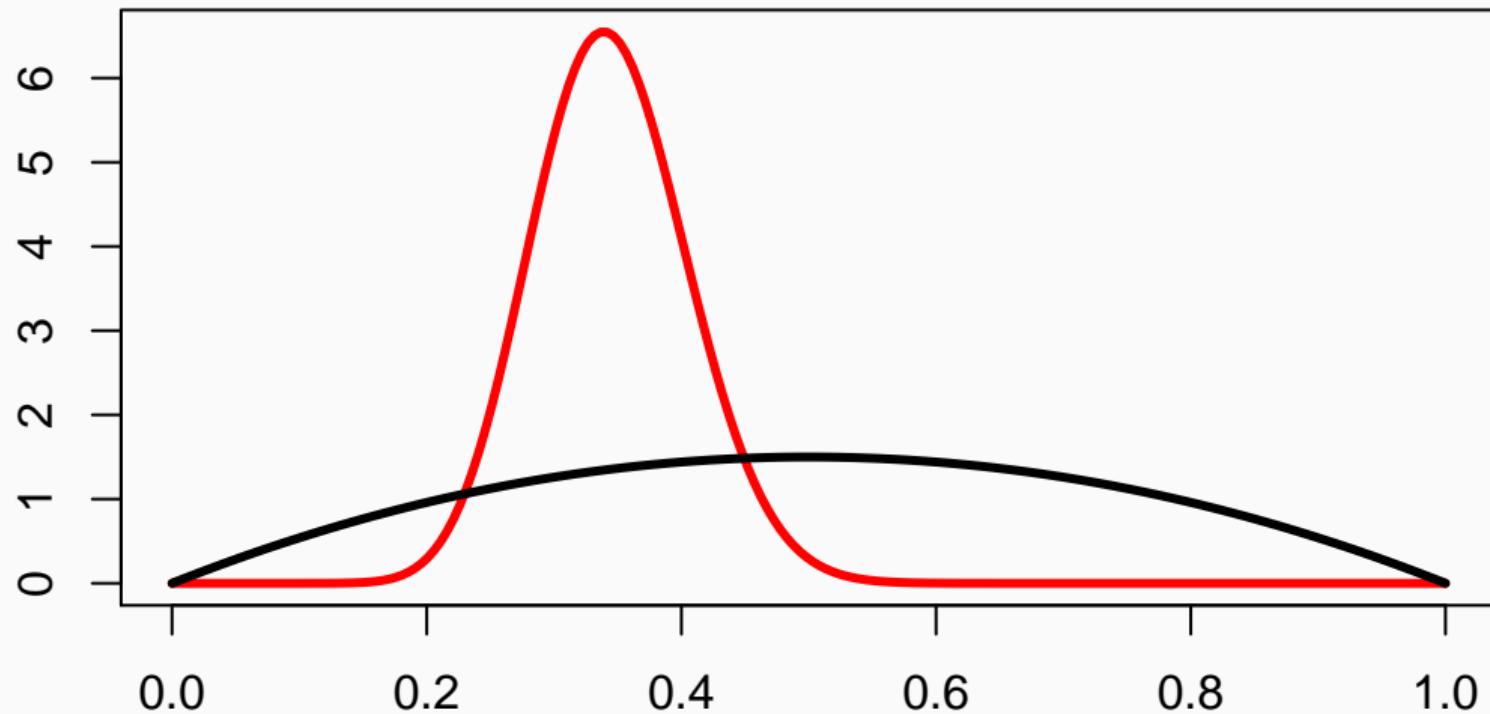
Prior $Beta(1, 1)$ and posterior survival $Beta(20, 39)$



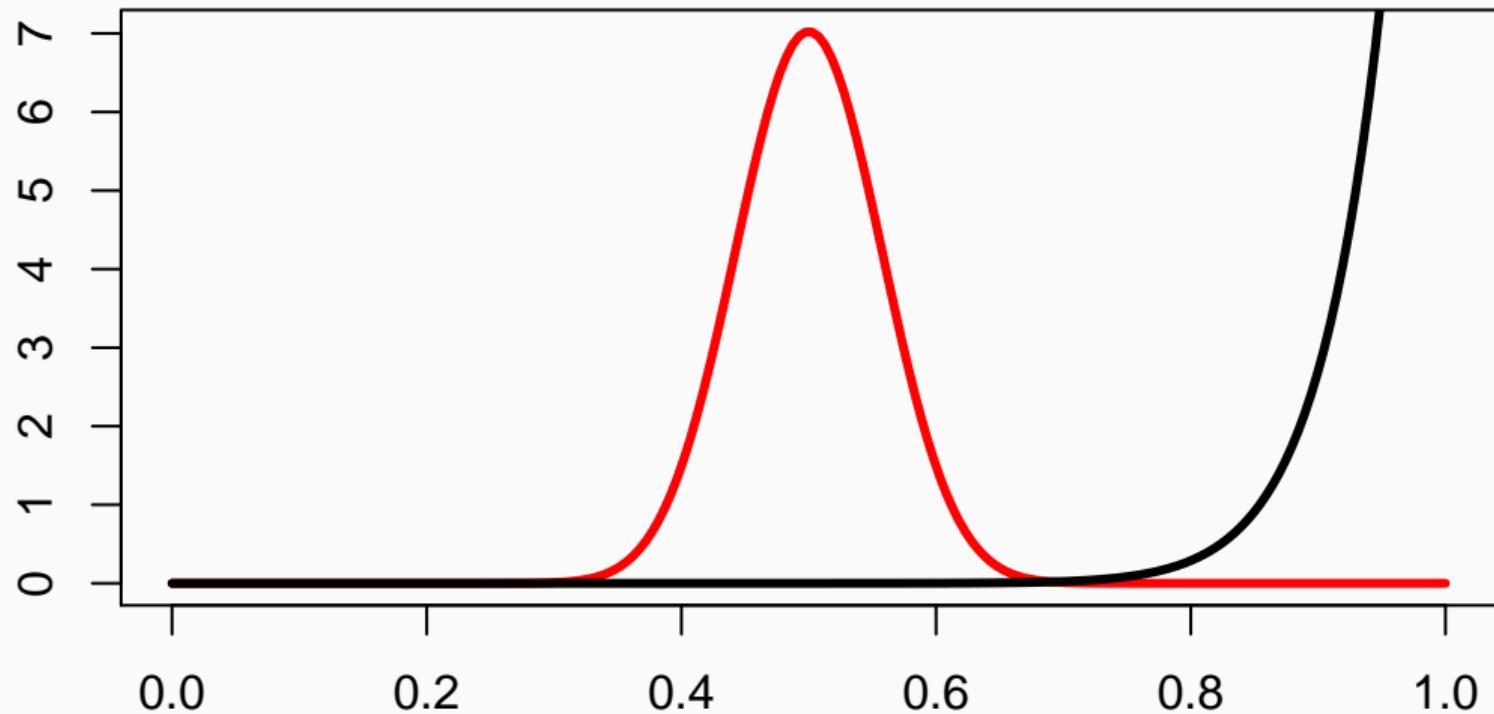
Prior $Beta(0.5, 0.5)$ and posterior survival $Beta(19.5, 38.5)$



Prior $Beta(2, 2)$ and posterior survival $Beta(21, 40)$



Prior $Beta(20, 1)$ and posterior survival $Beta(39, 49)$



The Role of the Prior

- In biological applications, the prior is a convenient means of **incorporating expert opinion or information from previous or related studies** that would otherwise need to be ignored.
- With sparse data, the role of the prior can be to enable inference on key parameters that would otherwise be impossible.
- With sufficiently large and informative datasets the prior typically has little effect on the results.
- **Always perform a sensitivity analysis!**

Informative Priors / No Information

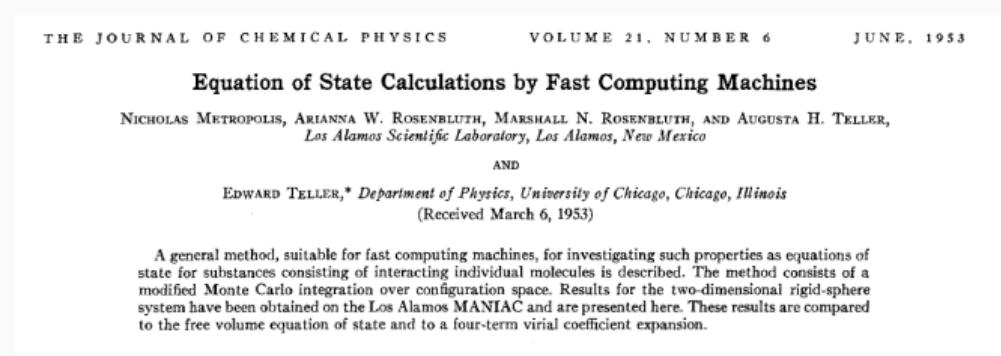
- Informative priors aim to reflect information available to the analyst that is gained independently of the data being studied.
- In the absence of any prior information on one or more model parameters we wish to ensure that this lack of knowledge is properly reflected in the prior.
- **Always perform a sensitivity analysis!**

Back to the Bayes formula

- Bayes inference is easy! Well, not so easy in real-life applications...
- The issue is in $\Pr(\theta | \text{data}) = \frac{\Pr(\text{data} | \theta) \Pr(\theta)}{\Pr(\text{data})}$
- $\Pr(\text{data}) = \int L(\text{data} | \theta) \Pr(\theta) d\theta$ is a N -dimensional integral if $\theta = \theta_1, \dots, \theta_N$
- Difficult, if not impossible to calculate!
- Until recently, Bayesian analysis of complex models not possible

Bayesian computation

- In the early 1990s, statisticians rediscovered work from the 1950's in physics



- Use **stochastic simulation** to draw samples from posterior distributions
- Avoid explicit calculation of integrals in Bayes formula
- Instead, approximate posterior to arbitrary degree of precision by drawing large sample
- Markov chain Monte Carlo = MCMC**; boost to Bayesian statistics!

MANIAC: Mathematical Analyzer, Numerical Integrator, and Computer



MANIAC:
1000 pounds
5 kilobytes of memory
70k multiplications/sec

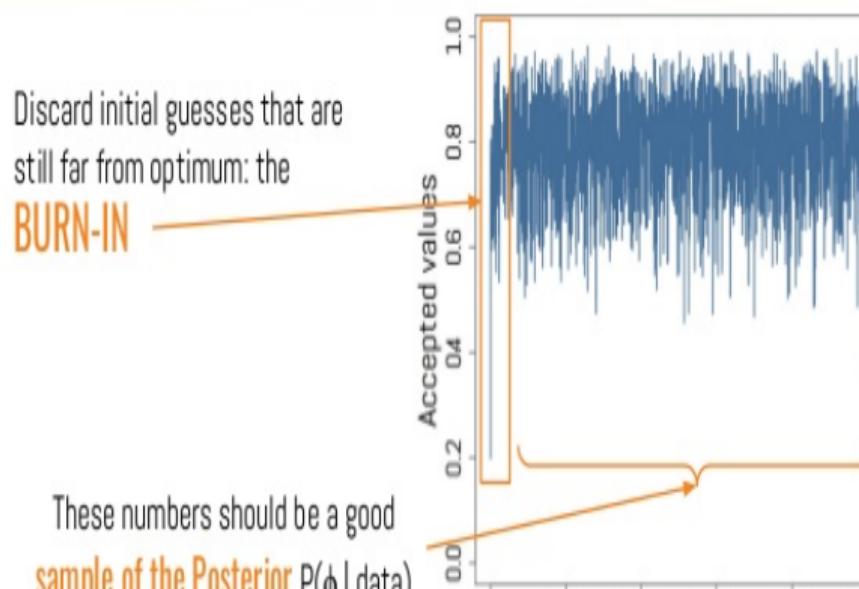
Your laptop:
4–7 pounds
2–8 million kilobytes
Billions of multiplications/sec

Why are MCMC methods so useful?

- MCMC: stochastic algorithm to produce sequence of dependent random numbers (from Markov chain)
- Converge to equilibrium (aka stationary) distribution
- Equilibrium distribution is the desired posterior distribution
- Several ways of constructing these chains: Metropolis-Hastings, Gibbs sampler, Metropolis-within-Gibbs, ...
- How to implement them in practice?!

In practice, when is equilibrium attained?

- Run multiple chains from arbitrary starting places (inits)
- Assume convergence when all chains reach same regime
- Discard initial burn-in phase
- Summarize posterior distribution with mean, sd and credible intervals



Introduction to JAGS (Just Another Gibbs Sampler)

Martyn Plummer



Let's redo the logistic regression with the White stork data

- We'll need data
- We'll need to build a model - write down the likelihood
- We'll need to specify priors for parameters

Let us read in the data

```
nbsuccess = c(151,105,73,107,113,87,77,108,118,122,112,120,122,89,69,71,  
            53,41,53,31,35,14,18)  
nbpairs = c(173,164,103,113,122,112,98,121,132,136,133,137,145,117,90,80,  
           67,54,58,39,42,23,23)  
temp = c(15.1,13.3,15.3,13.3,14.6,15.6,13.1,13.1,15.0,11.7,15.3,14.4,14.4,  
        12.7,11.7,11.9,15.9,13.4,14.0,13.9,12.9,15.1,13.0)  
rain = c(67,52,88,61,32,36,72,43,92,32,86,28,57,55,66,26,28,96,48,90,86,  
        78,87)  
datax = list(N=23,nbsuccess = nbsuccess,nbpairs = nbpairs,  
            temp = temp,rain = rain)
```

What is the model?

$$\text{nbchicks}_i \sim \text{Binomial}(\text{nbpairs}_i, p_i)$$

$$\text{logit}(p_i) = a + b_{temp} \text{ temp}_i + b_{rain} \text{ rainfall}_i$$

Let us build the model

```
{  
# Likelihood  
for( i in 1 : N){  
    nbsuccess[i] ~ dbin(p[i],nbpairs[i])  
    logit(p[i]) <- a + b.temp * temp[i] + b.rain * rain[i]  
}  
# ...
```

Let us specify priors

```
{  
# Likelihood  
  for( i in 1 : N){  
    nbsuccess[i] ~ dbin(p[i],nbpairs[i])  
    logit(p[i]) <- a + b.temp * temp[i] + b.rain * rain[i]  
  }  
  
# Priors  
a ~ dnorm(0,0.001)  
b.temp ~ dnorm(0,0.01)  
b.rain ~ dnorm(0,0.01)  
}
```

Warning: Jags uses precision for Normal distributions (1 / variance)

Let us specify a few additional things

```
# list of lists of initial values (one for each MCMC chain)
init1 <- list(a=-.5)
init2 <- list(a=.5)
inits <- list(init1,init2)

# specify parameters that need to be estimated
parameters <- c("a","b.temp","b.rain")

# specify nb iterations for burn-in and final inference
nb.burnin <- 1000
nb.iterations <- 2500
```

Let us run Jags!

```
# load R2jags to run Jags through R
library(R2jags)
reglogcig.sample <- jags(datax,inits,parameters,n.iter=nb.iterations,
                           model.file="reglogistique.txt",
                           n.chains=2,n.burnin=nb.burnin)
```

```
## Compiling model graph
##      Resolving undeclared variables
##      Allocating nodes
## Graph information:
##      Observed stochastic nodes: 23
##      Unobserved stochastic nodes: 3
##      Total graph size: 181
##
```

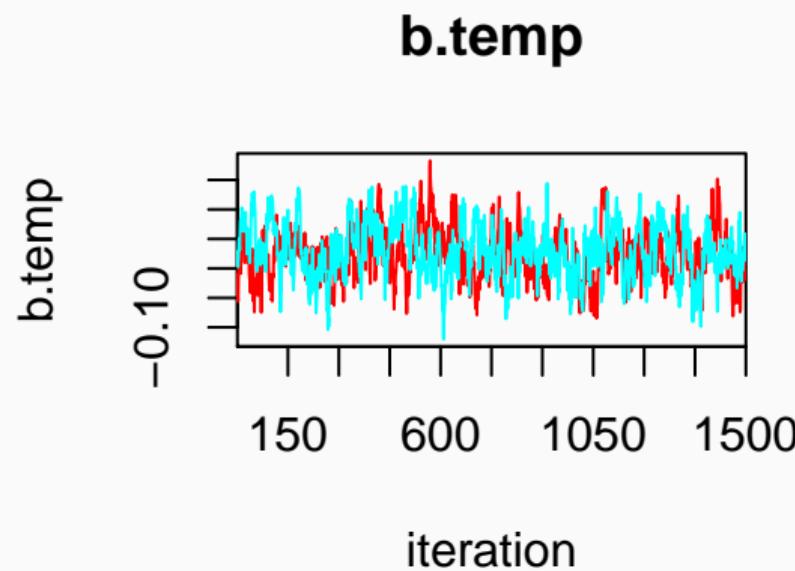
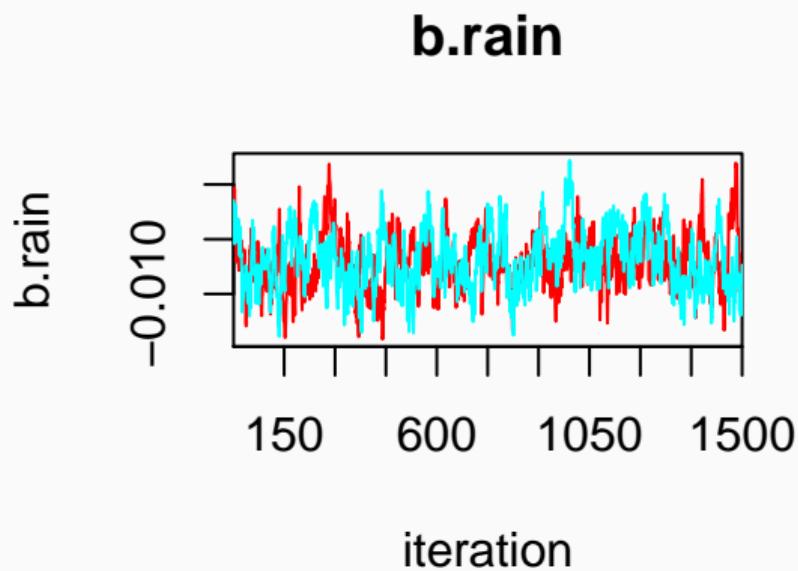
Display parameter estimates

reglogcig.sample

```
## Inference for Bugs model at "reglogistique.txt", fit using jags,
## 2 chains, each with 2500 iterations (first 1000 discarded)
## n.sims = 3000 iterations saved
##          mu.vect sd.vect    2.5%    25%    50%    75%   97.5% Rhat n
## a          1.620   0.614   0.373   1.217   1.653   2.034   2.776 1.014
## b.rain     -0.007   0.003  -0.012  -0.009  -0.007  -0.005  -0.002 1.007
## b.temp      0.026   0.045  -0.060  -0.004   0.025   0.055   0.117 1.010
## deviance 205.730  22.894 201.798 202.833 203.962 205.724 211.479 1.120
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1)
##
```

Let us assess convergence

```
traceplot(reglogcig.sample, mfrow = c(1, 2),  
         varname=c('b.rain','b.temp'), ask = FALSE)
```



Let us explore the results

```
res <- as.mcmc(reglogcig.sample) # convert outputs in a list
res <- rbind(res[[1]],res[[2]]) # put two MCMC lists on top of each other
head(res)

##           a      b.rain      b.temp deviance
## [1,] 0.3250383 -0.0017822462 -0.010819894 1168.0554
## [2,] 1.0819934 -0.0002896054 -0.045386385  733.6820
## [3,] 1.5275917 -0.0012412205 -0.056259011  522.3072
## [4,] 1.1585552 -0.0004168704 -0.022488124  420.0934
## [5,] 1.0370266 -0.0027828682  0.005601119  344.2514
## [6,] 1.2534198 -0.0032150507 -0.002007754  305.9672
```

Compute a posteriori $\Pr(\text{rain} < 0)$

```
# probability that the effect of rainfall is negative  
mean(res[, 'b.rain'] < 0)  
  
## [1] 0.993
```

Compute a posteriori $\Pr(\text{temp} < 0)$

```
# probability that the effect of temperature is negative  
mean(res[, 'b.temp'] < 0)  
  
## [1] 0.2826667
```

Get credible interval for the rain effect

```
quantile(res[, 'b.rain'], c(0.025, 0.975))
```

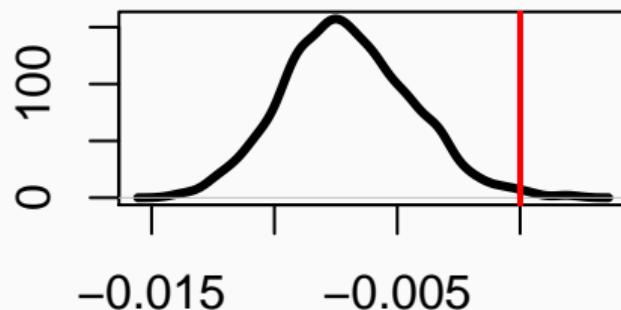
```
##           2.5%         97.5%
## -0.011871062 -0.001714676
```

Get credible interval for the temperature effect

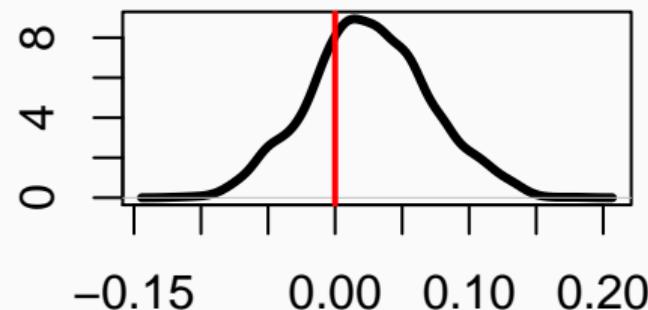
```
quantile(res[, 'b.temp'], c(0.025, 0.975))
```

```
##           2.5%         97.5%
## -0.05987025  0.11665989
```

a posteriori density of the rainfall effect



a posteriori density of the temperature effect



There is an influence of rainfall, but not temperature (credible interval does not and does contain 0)

```
## Compiling model graph
## Resolving undeclared variables
## Allocating nodes
```

How to incorporate prior information? A capture-recapture example

- Estimating survival using capture-recapture data
- E.g. 101 i.e. captured, missed and recaptured
- Simplest model: constant survival ϕ and detection p probabilities

$$\Pr(101) = \phi(1 - p)\phi p$$

- Assuming a vague prior

$$\phi_{prior} \sim \text{Uniform}(0, 1)$$

Case study

- European dippers in Eastern France (1981-1987)



- Mean posterior is $\phi_{posterior} = 0.56$ with credible interval [0.51, 0.61]

How to incorporate prior information?

- Using information on body mass and annual survival of 27 European passernines, we can predict survival of European dippers using only body mass
- For dippers, body mass is 59.8g, therefore $\phi = 0.57$ with $sd = 0.073$
- Assuming an **informative prior** $\phi_{prior} \sim \text{Normal}(0.57, 0.073)$
- Mean posterior $\phi_{posterior} = 0.56$ with credible interval $[0.52, 0.60]$
- No increase of precision in posterior inference

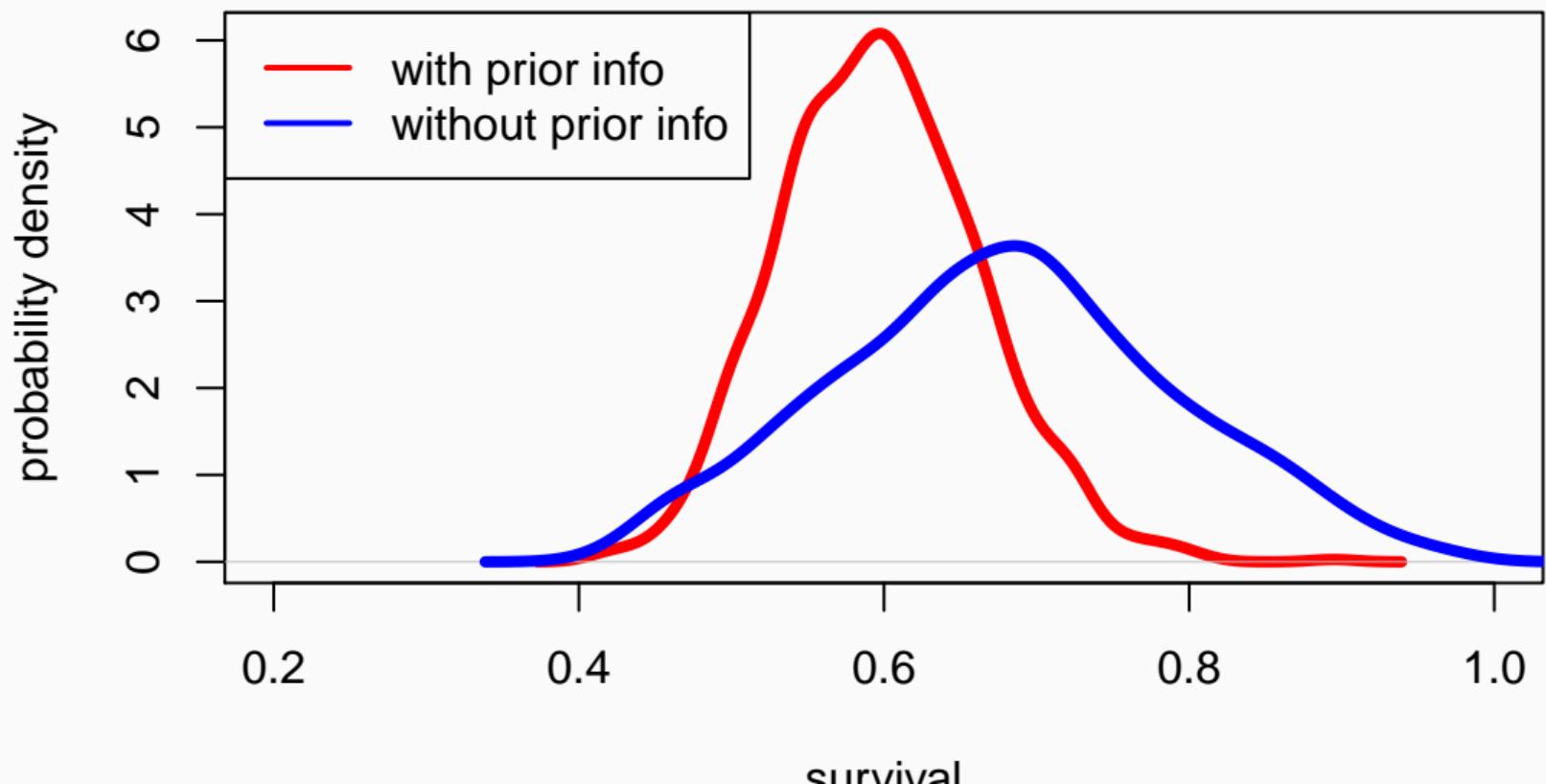
A general result

This is a general result, the Bayesian and frequentist estimates will always agree if there is sufficient data, so long as the likelihood is not explicitly ruled out by the prior

How to incorporate prior information?

- Using information on body mass and annual survival of 27 European passerines, we can predict survival of European dippers using only body mass
- For dippers, body mass is 59.8g, therefore $\phi = 0.57$ with $sd = 0.073$
- Assuming an informative prior $\phi_{prior} \sim \text{Normal}(0.57, 0.073)$
- **With 3 first years only**
- Width of credible interval is 0.47 (vague prior) vs. 0.30 (informative prior)
- Huge increase of precision in posterior inference (40% gain)!

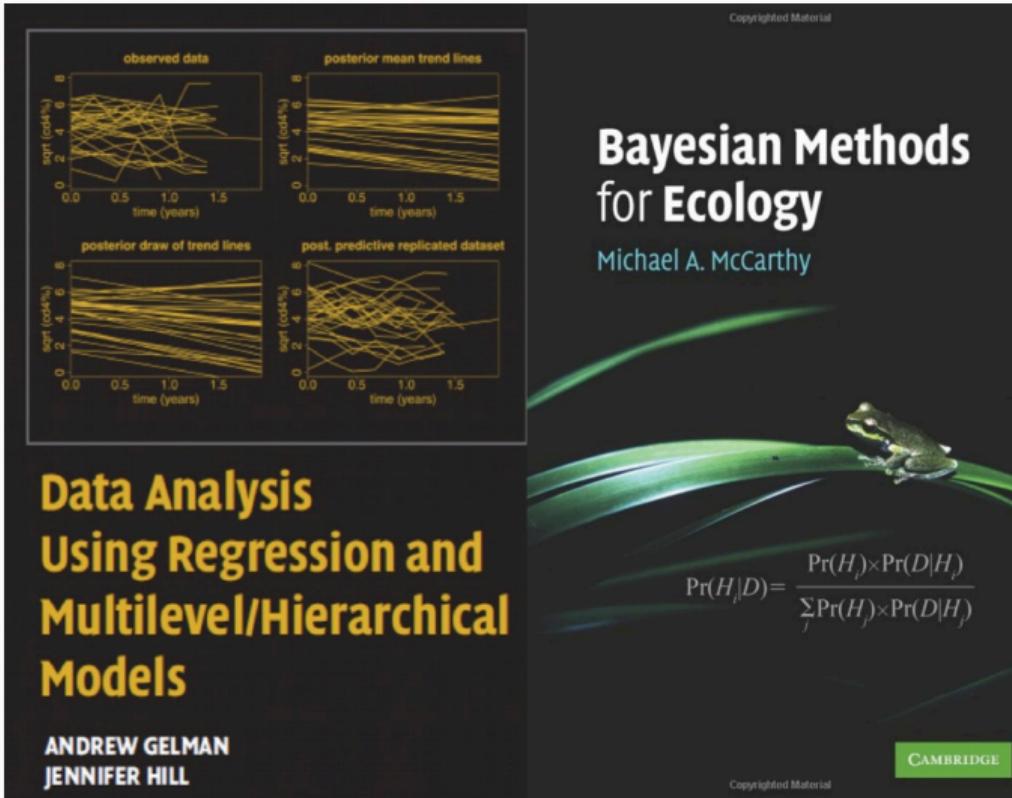
Compare **vague** vs. **informative** prior



Take-home message: shall I go for frequentist or Bayes?

- Pros
 - allows formal use of prior information
 - error propagation made easy
 - with same MCMC algorithms, complex (hierarchical) models can be implemented
- Cons
 - computational burden can be high
 - model selection is still difficult to perform
 - checking convergence is painful
 - is Jags too flexible?
- So what?
 - make an informed and pragmatic choice
 - are you after complexity, speed, uncertainties, etc?
 - talk to colleagues

Textbooks



On our plate

- Distributions and likelihoods
- Hypothesis testing and multimodel inference
- Introduction to Bayesian inference
- Generalized Linear Models (GLMs)
- Generalized Additive Models (GAMs)
- Mixed Effect Models

On our plate

- Distributions and likelihoods
- Hypothesis testing and multimodel inference
- Introduction to Bayesian inference
- Generalized Linear Models (GLMs)
- Generalized Additive Models (GAMs)
- Mixed Effect Models

Generalized Linear Models (GLMs)

Survival of passengers on the Titanic ~ Class

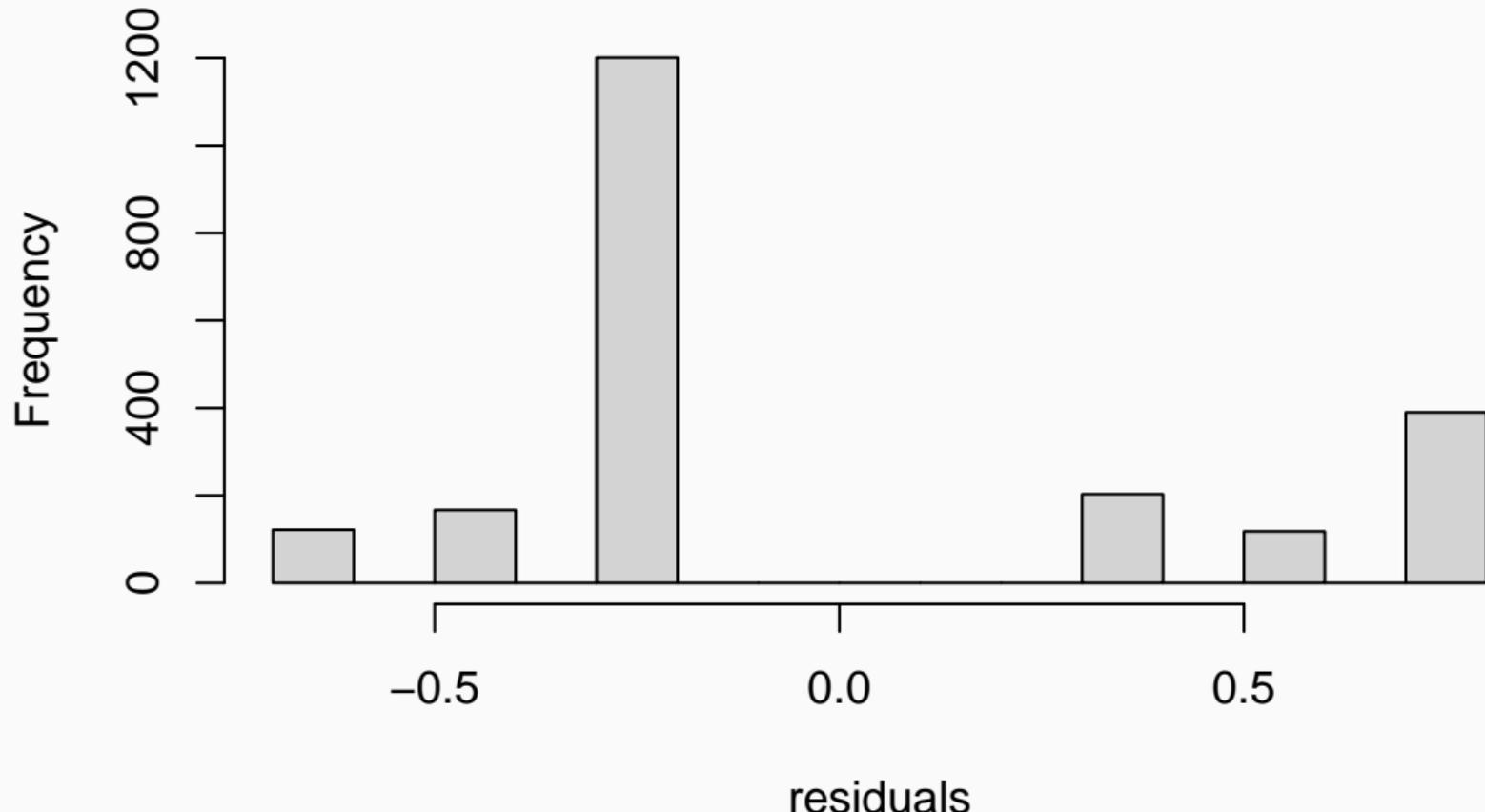
Read titanic_long.csv dataset.

```
titanic <- read.csv("dat/titanic_long.csv") %>%  
  mutate(across(where(is.character), as_factor))  
head(titanic)  
##   class    age   sex survived  
## 1 first adult male      1  
## 2 first adult male      1  
## 3 first adult male      1  
## 4 first adult male      1  
## 5 first adult male      1  
## 6 first adult male      1
```

Let's fit a linear model

```
titanic.lm <- lm(survived ~ class, data = titanic)
```

Clearly, the residuals are not normal!



Generalized linear models (GLMs)

- GLMs extend the linear model to scenarios that involve **non-normal error distributions**, hence the term **generalized**
- The **mean response** is expressed as a **linear function of the predictors** using a **link function**, hence the term **linear**

Generalized Linear Models

1. Response variable

- Bernoulli/Binomial: Binary variables 0/1
- Poisson: Counts 0, 1, 2, ...
- Normal: Real values
- etc

2. Predictors (continuous or categorical)

3. Link function

- Gaussian: identity
- Binomial: logit
- Poisson: log
- Type in ?family

Bernoulli/Binomial distribution (logistic regression)

- Response variable: Yes/No (e.g. dead/alive, male/female, presence/absence)
- Link function: logit

$$\text{logit}(p) = \ln \left(\frac{p}{1-p} \right)$$

- Then, if predictor is x

Response \sim Distribution(Mean Response)

$$Y_i \sim \text{Bernoulli}(p_i)$$

$$\text{logit}(p_i) = a + b x_i$$

$$p_i = \text{logit}^{-1}(a + b x_i) = \frac{e^{a+b x_i}}{1 + e^{a+b x_i}}$$

Back to survival of Titanic passengers

How many passengers travelled in each class?

```
tapply(titanic$survived, titanic$class, length)
##   first  second   third     crew
##     325     285     706    885
```

Back to survival of Titanic passengers

How many survived?

```
tapply(titanic$survived, titanic$class, sum)
##   first  second  third    crew
##     203     118     178     212
```

Back to survival of Titanic passengers

What proportion survived in each class?

```
as.numeric(tapply(titanic$survived, titanic$class, mean))  
## [1] 0.6246154 0.4140351 0.2521246 0.2395480
```

Back to survival of Titanic passengers (package dplyr)

Arrange passenger survival according to class

```
library(dplyr)  
summarise(group_by(titanic, class, survived), count = n())`
```

Back to survival of Titanic passengers (package dplyr)

Same manipulation using the pipe operator %>%

```
titanic %>%  
  group_by(class, survived) %>%  
  summarise(count = n())
```

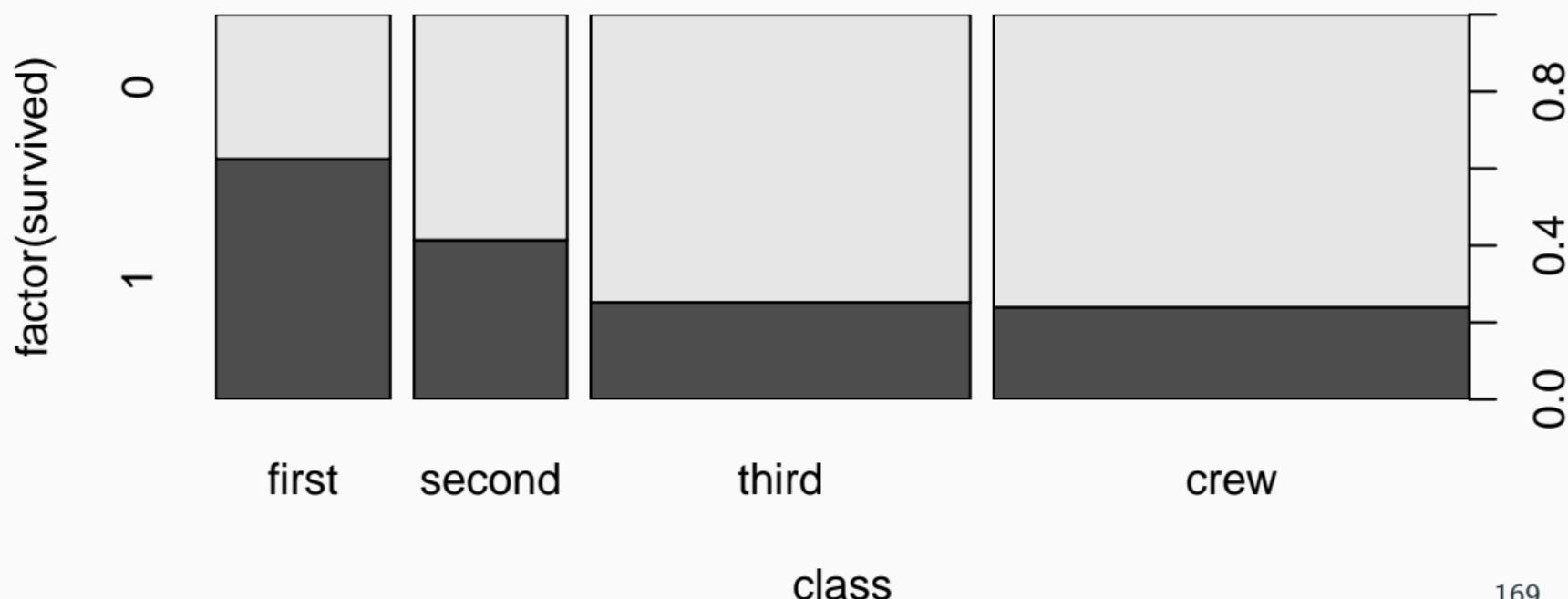
Back to survival of Titanic passengers (package dplyr)

Arrange passenger survival according to class

```
## # A tibble: 8 x 3
## # Groups:   class [4]
##   class   survived count
##   <chr>     <int> <int>
## 1 crew         0    673
## 2 crew         1    212
## 3 first        0    122
## 4 first        1    203
## 5 second       0    167
## 6 second       1    118
## 7 third        0    528
## 8 third        1    178
```

Or graphically...

```
plot(factor(survived) ~ class, data = titanic)
```



Fitting GLMs in R: `glm` function

```
titanic.glm <- glm(survived ~ class, data=titanic, family=binomial)

## # A tibble: 4 x 5
##   term       estimate std.error statistic p.value
##   <chr>     <dbl>    <dbl>     <dbl>    <dbl>
## 1 (Intercept)  0.509    0.115     4.44 8.79e- 6
## 2 classsecond -0.856    0.166    -5.16 2.51e- 7
## 3 classthird  -1.60     0.144    -11.1 1.07e-28
## 4 classcrew   -1.66     0.139    -12.0 4.97e-33
```

These estimates are on the logit scale!

Interpreting logistic regression outputs

Parameter estimates on the logit scale:

```
## (Intercept) classsecond classthird classcrew  
## 0.5091849 -0.8564941 -1.5964977 -1.6643440
```

We need to back-transform using the inverse logit function:

```
plogis(coef(titanic.glm)[1]) # crew survival probability  
## (Intercept)  
## 0.6246154
```

Looking at the data, the proportion of crew who survived is:

```
sum(titanic$survived[titanic$class == "crew"]) /  
nrow(titanic[titanic$class == "crew", ])  
## [1] 0.239548
```

Probability of survival for 1st class passengers?

Needs to add intercept (baseline) to the parameter estimate:

```
plogis(coef(titanic.glm)[1] + coef(titanic.glm)[2])  
## (Intercept)  
## 0.4140351
```

Again this value matches the data:

```
sum(titanic$survived[titanic$class == "first"]) /  
nrow(titanic[titanic$class == "first", ])  
## [1] 0.6246154
```

Model interpretation using effects package

```
library(effects)
allEffects(titanic.glm)

## model: survived ~ class
##
## class effect
## class
##      first     second      third      crew
## 0.6246154 0.4140351 0.2521246 0.2395480
```

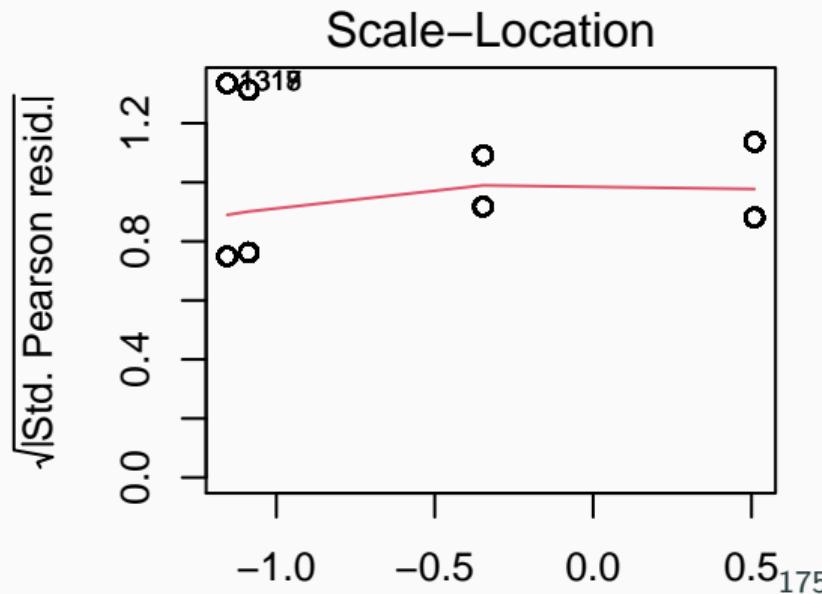
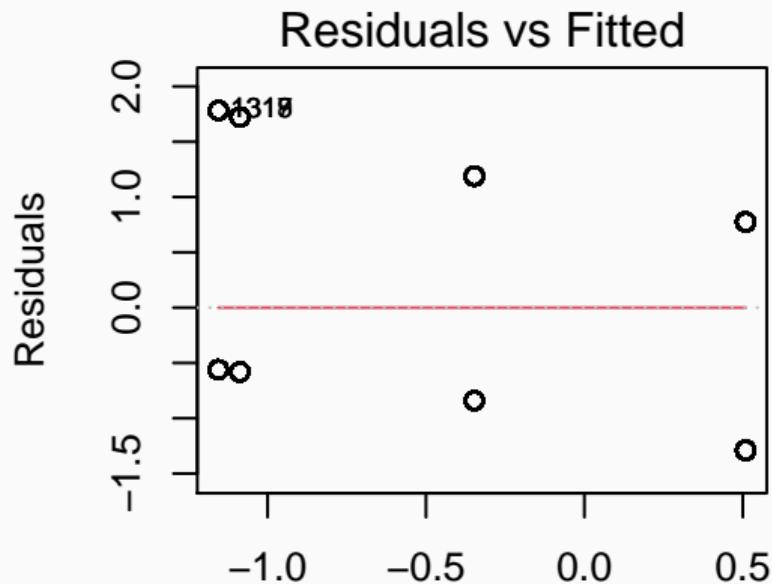
Effects plot

```
plot(allEffects(titanic.glm))
```



Logistic regression: model checking

```
layout(matrix(1:4, nrow=2))  
plot(titanic.glm)
```

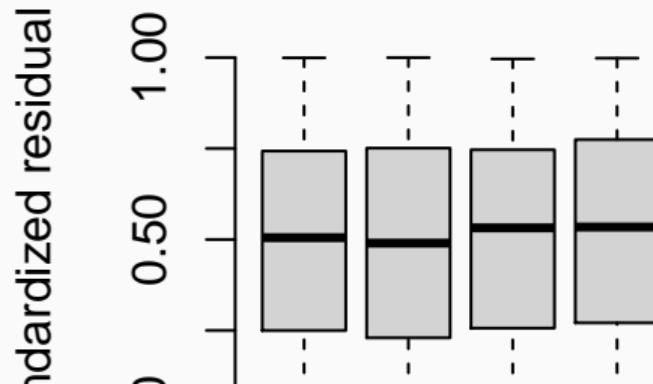
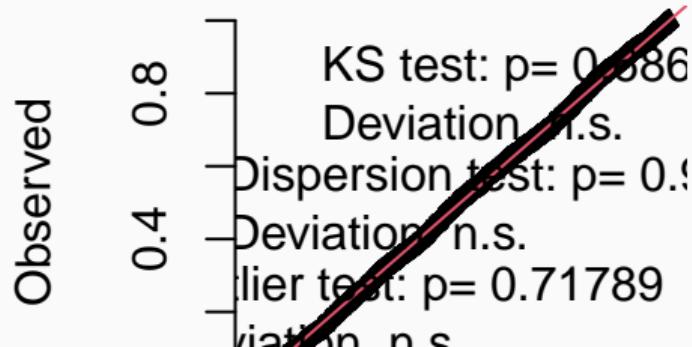


Residual diagnostics with R package DHARMA

```
library(DHARMA)  
simulateResiduals(titanic.glm, plot = TRUE)
```

DHARMA residual diagnostics

QQ plot residuals



Recapitulating

1. Import data: `read.table` or `read.csv`.
2. Check data: `summary`.
3. Plot data: `plot`.
4. Fit model: `glm`. Don't forget to specify `family`.
5. Examine models: `summary` or `tidy`.
6. Use `plogis` to apply back-transformation (*invlogit*) to parameter estimates (`coef`). Alternatively, use `allEffects` from `effects` package.
7. Plot model: `plot(allEffects(model))`. Alternatively, use package `visreg` (examples below).
8. Examine residuals: use `DHARMA::simulateResiduals`.

**Did men have higher survival than
women?**

Plot first

```
plot(factor(survived) ~ sex, data = titanic)
```



Fit model

```
titanic.sex <- glm(survived ~ sex, data = titanic, family = binomial)

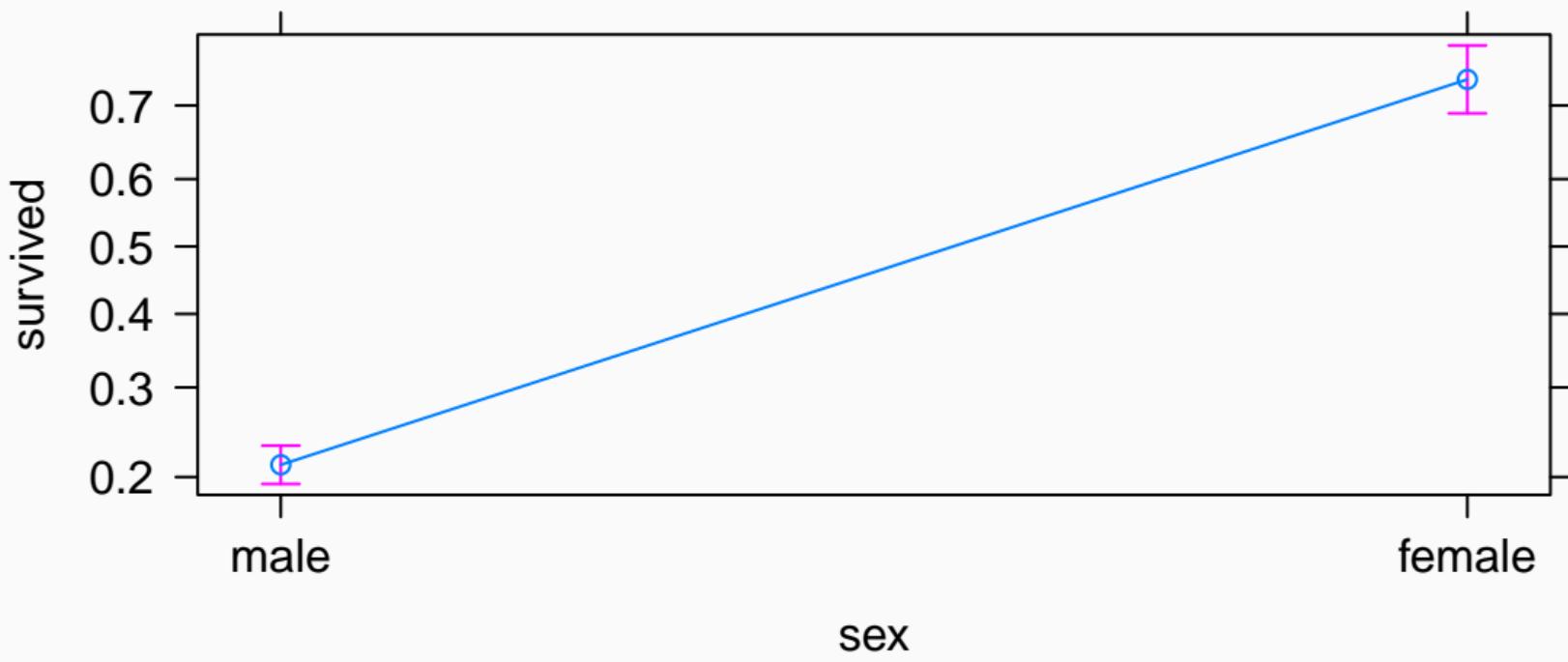
## # A tibble: 2 x 5
##   term      estimate std.error statistic p.value
##   <chr>     <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept) -1.31     0.0588    -22.3 2.10e-110
## 2 sexfemale    2.32     0.120      19.4 1.22e- 83
```

Effects

```
allEffects(titanic.sex)

##  model: survived ~ sex
##
##  sex effect
##  sex
##      male    female
##  0.2120162 0.7319149
```

sex effect plot



Did women have higher survival and travelled more in first class?

Let's look at the data

```
tapply(titanic$survived, list(titanic$class, titanic$sex), sum)
```

```
##           male female
## first      62    141
## second     25     93
## third      88     90
## crew      192     20
```

Mmmm...

Fit model with both factors (interactions)

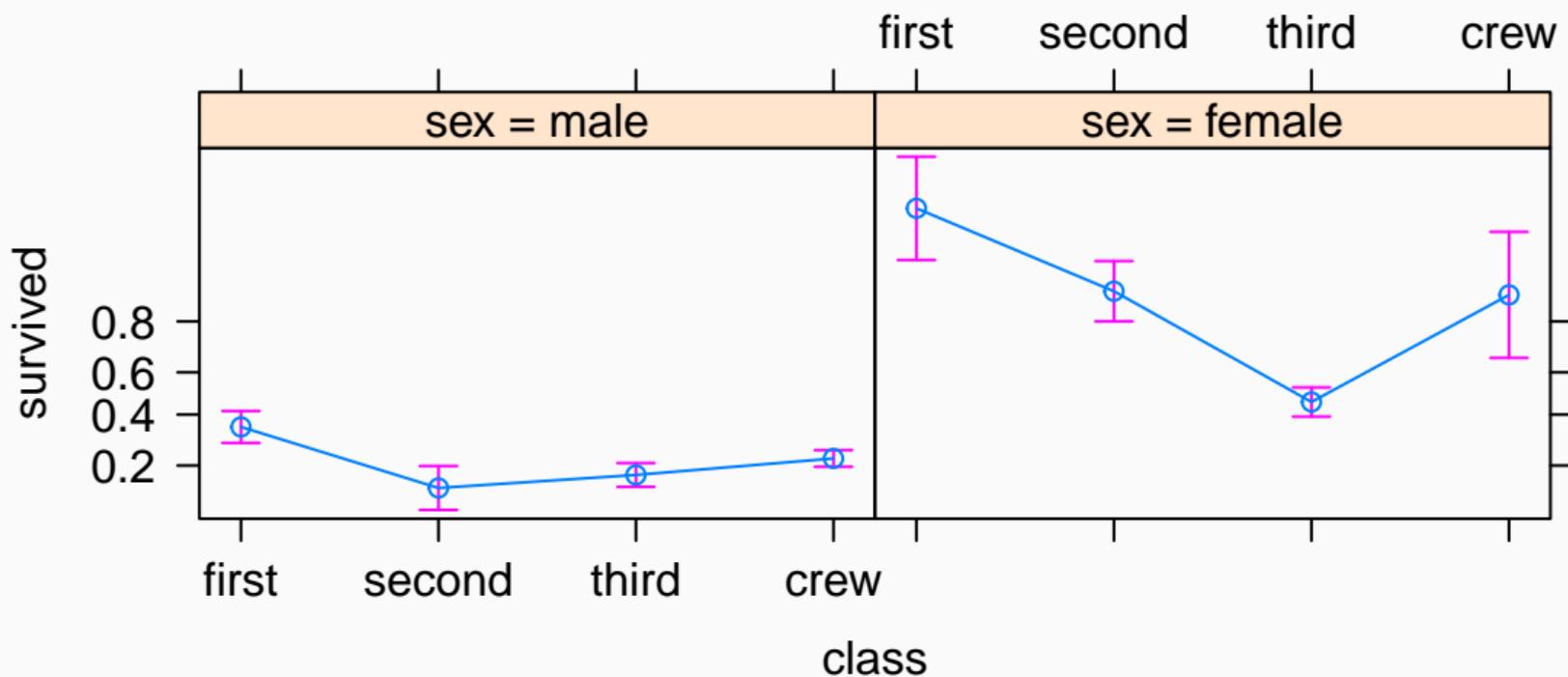
```
titanic.s.cl <- glm(survived ~ class * sex, data=titanic, family=binomial)

## # A tibble: 8 x 5
##   term            estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>      <dbl>    <dbl>
## 1 (Intercept) -0.644      0.157     -4.10  4.08e- 5
## 2 classecond   -1.17       0.267     -4.40  1.06e- 5
## 3 classthird   -0.924      0.196     -4.72  2.36e- 6
## 4 classcrew    -0.606      0.177     -3.43  6.12e- 4
## 5 sexfemale     4.21        0.531      7.92  2.29e-15
## 6 classecond:sexfemale -0.420      0.645     -0.652 5.15e- 1
## 7 classthird:sexfemale -2.80        0.562     -4.98  6.21e- 7
## 8 classcrew:sexfemale -1.06        0.820     -1.29  1.96e- 1
```

Effects

```
allEffects(titanic.s.cl)

##  model: survived ~ class * sex
##
##  class*sex effect
##          sex
##  class      male    female
##  first   0.3444444 0.9724138
##  second  0.1396648 0.8773585
##  third   0.1725490 0.4591837
##  crew    0.2227378 0.8695652
```

class*sex effect plot

Conclusions

Use AIC to test the effect formally:

```
AIC(glm(survived ~ 1, data = titanic, family = binomial)) # null model
## [1] 2771.457
AIC(titanic.glm) # class effect
## [1] 2596.555
AIC(titanic.sex) # sex effect
## [1] 2338.988
AIC(titanic.s.cl) # interaction of sex and class
## [1] 2179.733
```

So, women had higher probability of survival than men, irrespective of the class.

Logistic regression for proportions

Read Titanic data in different format

```
titanic.prop <- read.csv("dat/Titanic_prop.csv") %>%  
  mutate(across(where(is.character), as_factor))  
  
head(titanic.prop)  
##   X Class     Sex   Age  No Yes  
## 1 1 1st Female Adult    4 140  
## 2 2 1st Female Child    0   1  
## 3 3 1st Male   Adult 118  57  
## 4 4 1st Male   Child    0   5  
## 5 5 2nd Female Adult  13  80  
## 6 6 2nd Female Child    0  13
```

These are the same data, but compacted.

Bernoulli becomes a Binomial

Response \sim Distribution(Mean Response)

$$Y_i \sim \text{Binomial}(N_i, p_i)$$

$$\text{logit}(p_i) = a + b x_i$$

$$p_i = \text{logit}^{-1}(a + b x_i) = \frac{e^{a+b x_i}}{1 + e^{a+b x_i}}$$

Use `cbind(n.success, n.failures)` as response

```
prop.glm <- glm(cbind(Yes, No) ~ Class, data = titanic.prop,
                  family = binomial)

## # A tibble: 4 x 5
##   term      estimate std.error statistic p.value
##   <chr>     <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) 0.509     0.115     4.44 8.79e- 6
## 2 Class2nd    -0.856     0.166    -5.16 2.51e- 7
## 3 Class3rd    -1.60      0.144    -11.1 1.07e-28
## 4 ClassCrew   -1.66      0.139    -12.0 4.97e-33
```

Effects

```
allEffects(prop.glm)
## model: cbind(Yes, No) ~ Class
##
## Class effect
## Class
##      1st       2nd       3rd      Crew
## 0.6246154 0.4140351 0.2521246 0.2395480
```

Compare with former model based on raw data. Same results!

Logistic regression with continuous predictors

Read in GDP and infant mortality data (1998)

```
un <- read.csv("dat/UN_GDP_infantmortality.csv") %>%  
  mutate(X = as_factor(X))  
names(un) <- c("country", "mortality", "gdp")
```

- mortality: Infant mortality rate, infant deaths per 1000 live births.
- gdp: GDP per capita, in US dollars.

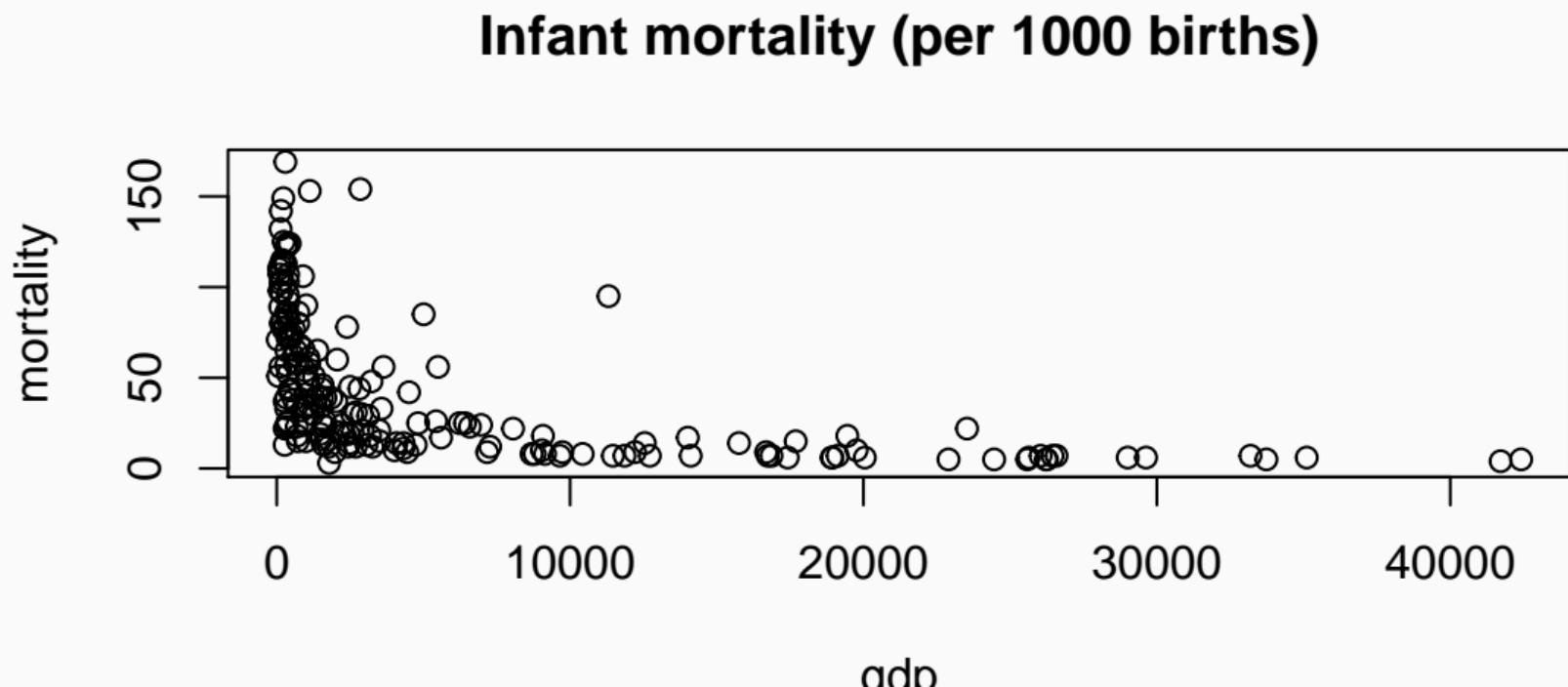
Explore the data

```
head(un)
```

```
##          country mortality   gdp
## 1      Afghanistan      154 2848
## 2         Albania        32  863
## 3         Algeria       44 1531
## 4 American.Samoa       11    NA
## 5        Andorra        NA    NA
## 6         Angola       124  355
```

Explore the data

```
plot(mortality~gdp,data=un,main="Infant mortality (per 1000 births)")
```



Fit model

```
gdp.glm <- glm(cbind(mortality, 1000 - mortality) ~ gdp,
                 data = un, family = binomial)

## # A tibble: 2 x 5
##   term       estimate std.error statistic p.value
##   <chr>     <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept) -2.66     0.0131    -203.    0.
## 2 gdp        -0.000128  0.00000346    -37.0  1.96e-299
```

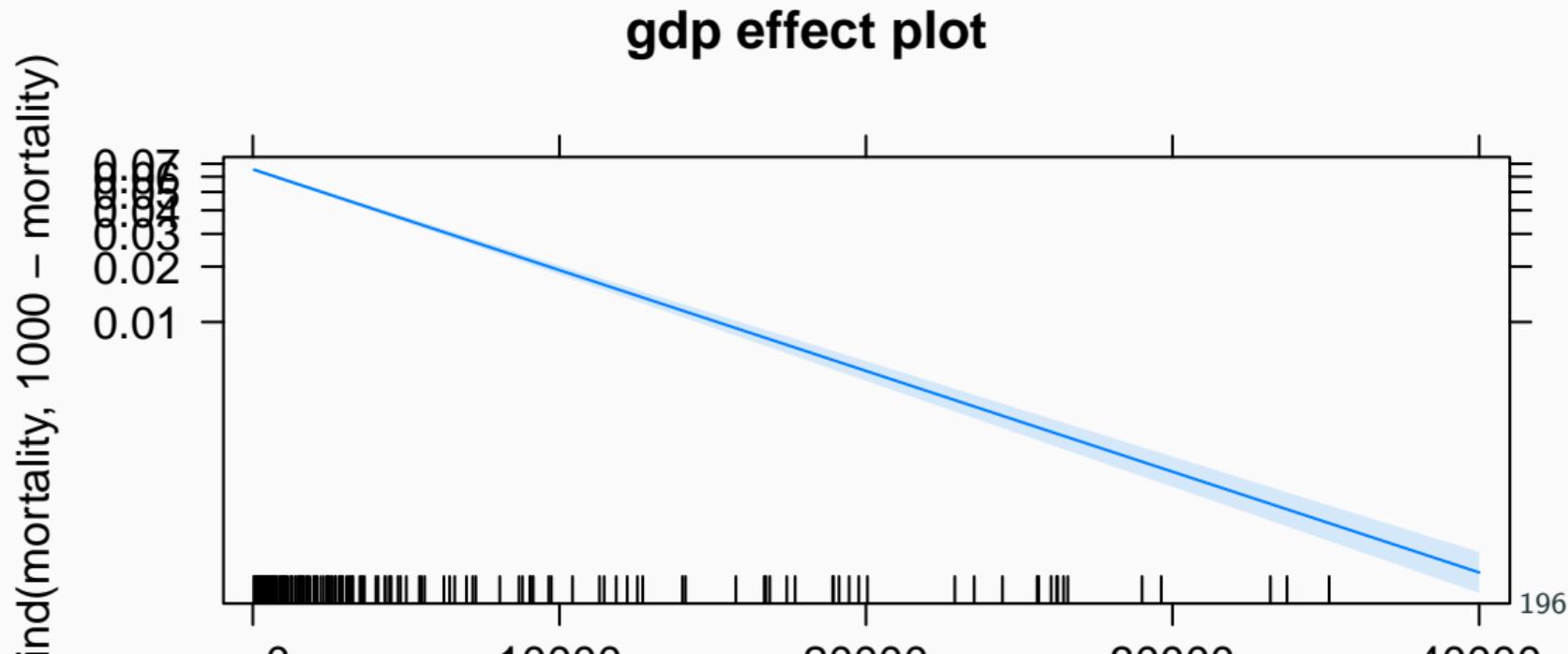
Effects

```
allEffects(gdp.glm)

##  model: cbind(mortality, 1000 - mortality) ~ gdp
##
##  gdp effect
##  gdp
##        40          10000         20000         30000         40000
## 0.0652177296 0.0191438829 0.0054028095 0.0015096074 0.0004206154
```

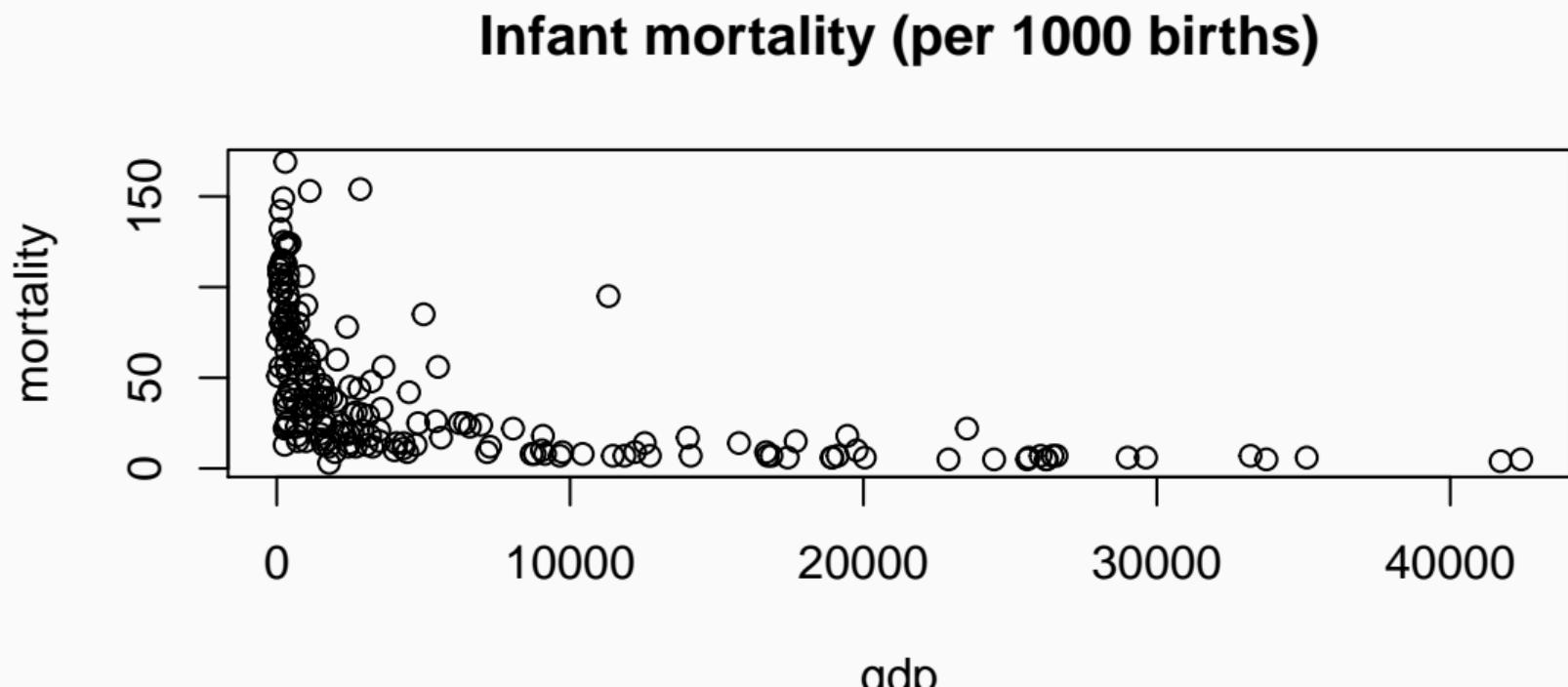
Effects plot

```
plot(allEffects(gdp.glm))
```



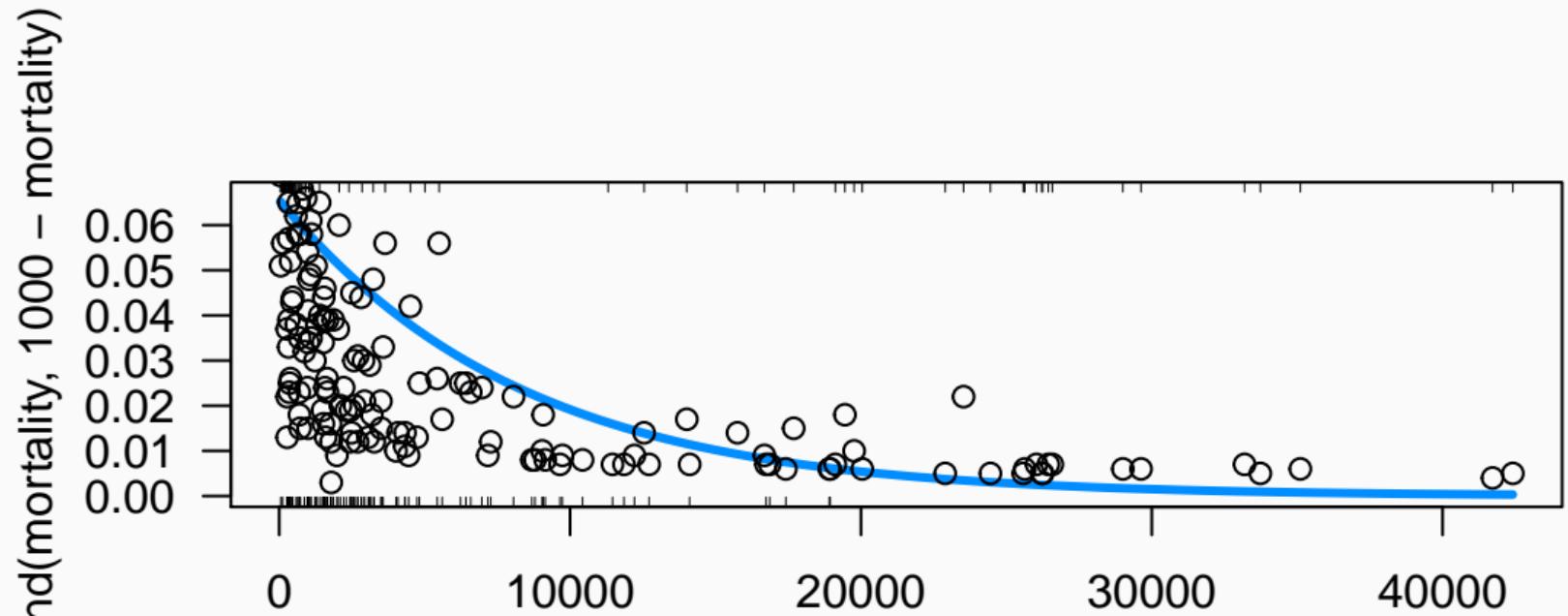
Plot model and data

```
plot(mortality~gdp,data=un,main="Infant mortality (per 1000 births)")
```



Plot model using visreg package

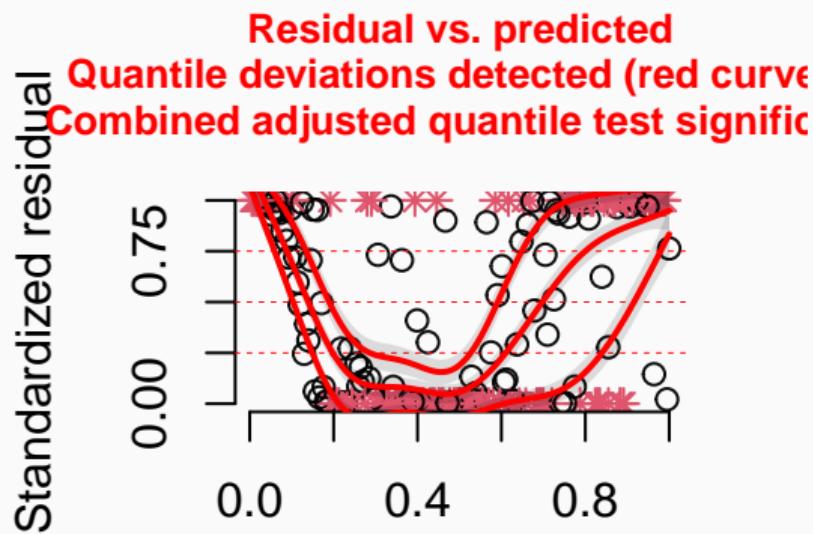
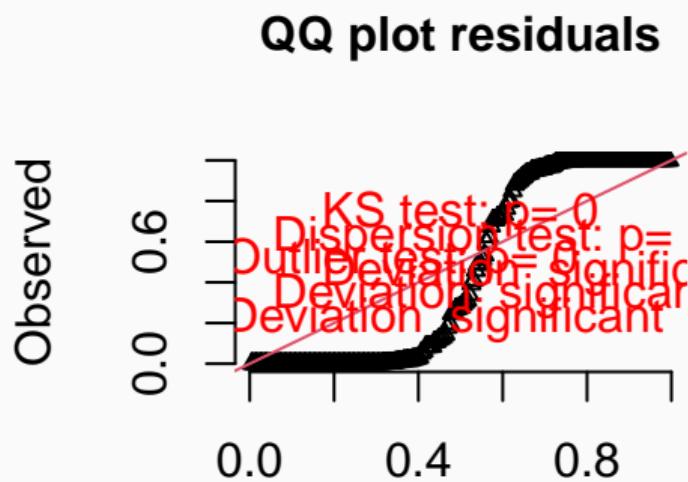
```
visreg::visreg(gdp.glm, scale = "response")
points(mortality/1000 ~ gdp, data = un)
```



Residuals diagnostics with DHARMA package

```
simulateResiduals(gdp.glm, plot = TRUE)
```

DHARMA residual diagnostics



Overdispersion

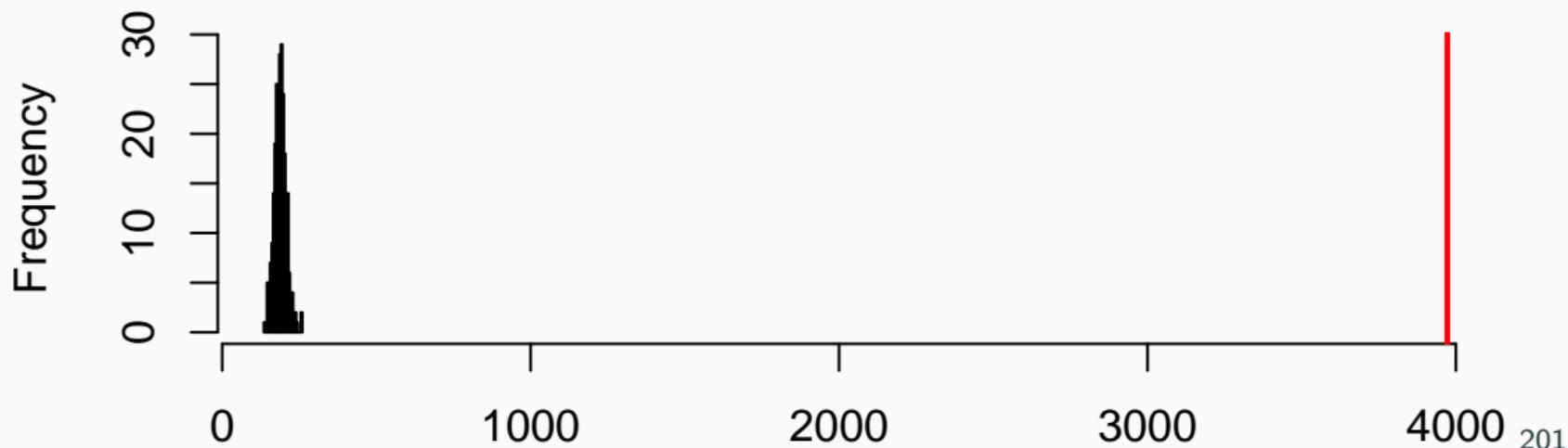
What is overdispersion?

- The variance is higher than expected from a Poisson or Binomial process
- Often due to pseudoreplication, dependence among statistical units
- To account for these cases, the trick is to consider ‘quasi’ functions that use the parameter ϕ to increase the expected variance (check out practical #1)

Testing for overdispersion with DHARMa package

```
simres <- simulateResiduals(gdp.glm, refit = TRUE)  
testOverdispersion(simres)
```

Dispersion test significant



Overdispersion in logistic regression with proportions

```
gdp.overdisp <- glm(cbind(mortality, 1000 - mortality) ~ gdp,  
                     data = un, family = quasibinomial)
```

```
## # A tibble: 2 x 5  
##   term      estimate std.error statistic p.value  
##   <chr>      <dbl>     <dbl>     <dbl>    <dbl>  
## 1 (Intercept) -2.66     0.0598    -44.5  1.06e-102  
## 2 gdp        -0.000128  0.0000158   -8.11  5.96e- 14
```

Mean estimates do not change after accounting for overdispersion

```
allEffects(gdp.overdisp)

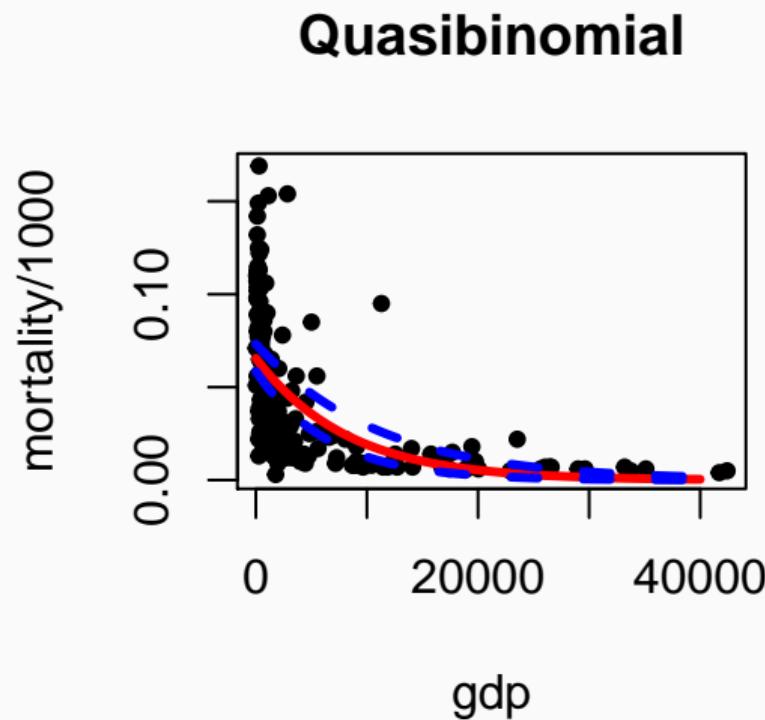
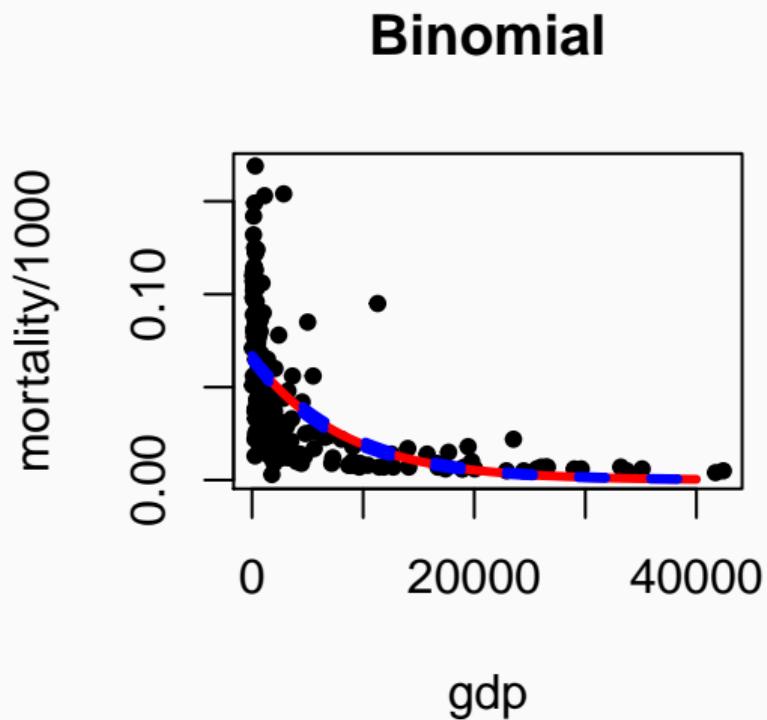
##  model: cbind(mortality, 1000 - mortality) ~ gdp
##
##  gdp effect
##  gdp
##        40          10000         20000         30000         40000
## 0.0652177296 0.0191438829 0.0054028095 0.0015096074 0.0004206154
```

Mean estimates do not change after accounting for overdispersion

```
allEffects(gdp.glm)

##  model: cbind(mortality, 1000 - mortality) ~ gdp
##
##  gdp effect
##  gdp
##        40          10000         20000         30000         40000
## 0.0652177296 0.0191438829 0.0054028095 0.0015096074 0.0004206154
```

But standard errors do!



GLMs for counts: Poisson regression

Types of response variable

- Gaussian: `lm`
- Bernoulli/Binomial: `glm` (family `binomial/quasibinomial`)
- Counts: `glm` (family `poisson/quasipoisson`)

Poisson regression

- **Discrete** response variable: Counts (0, 1, 2, 3...)
- Link function: \log

Response \sim Distribution(Mean Response)

$$Y_i \sim \text{Poisson}(\lambda_i)$$

$$\log(\lambda_i) = a + b x_i$$

$$\lambda_i = e^{a+b x_i}$$

Example dataset: Seedling counts in $0.5m^2$ quadrats

```
seedl <- read.csv("dat/seedlings.csv")
names(seedl)

## [1] "X"      "count"   "row"     "col"     "light"
```

- Light is the proportion of global solar radiation (GSF Global Site Factor)

Explore data

```
head(seed1)
```

```
##   X count row col    light
## 1 1     0   1   1 70.71854
## 2 2     1   1   2 88.26021
## 3 3     2   1   3 67.35133
## 4 4     3   1   4 67.57850
## 5 5     4   1   5 26.63098
## 6 6     3   1   6 15.79433
```

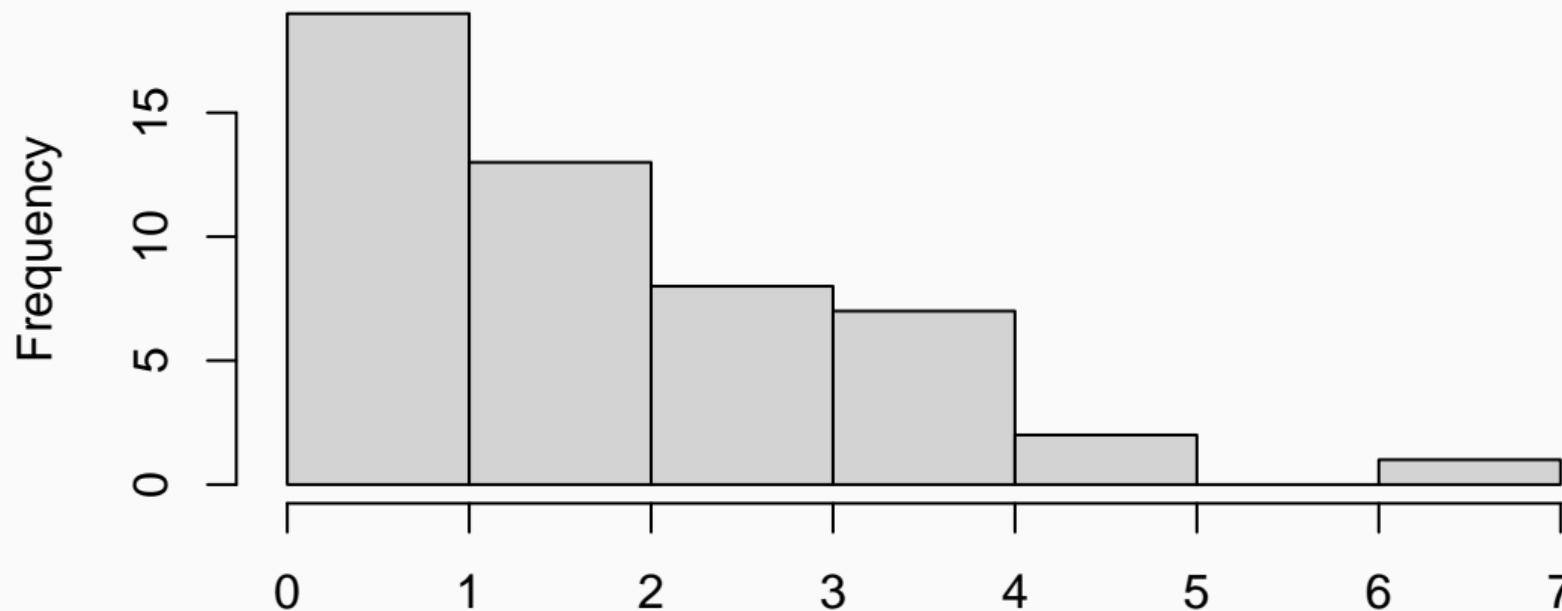
Explore data

```
table(seed1$count)
##
##   0   1   2   3   4   5   7
##  7 12 13   8   7   2   1
```

Explore data

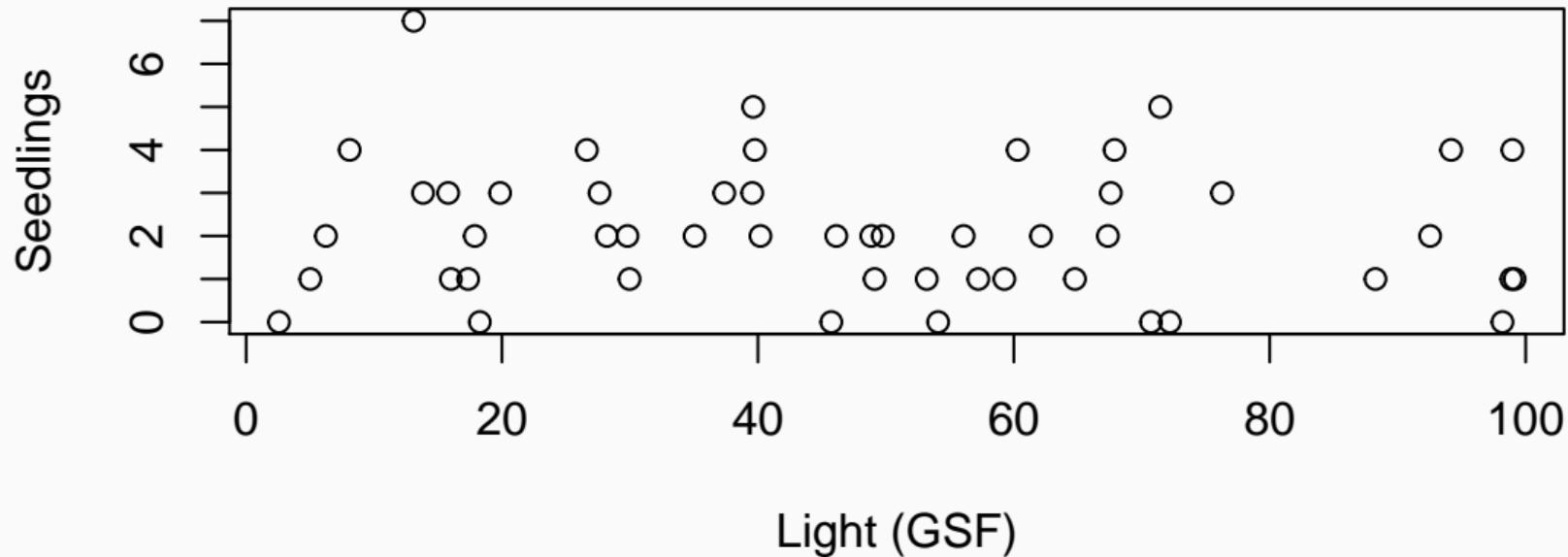
```
hist(seed$count)
```

Histogram of seed\$count



Relationship between Nseedlings and light?

```
plot(seedl$light, seedl$count, xlab = "Light (GSF)", ylab = "Seedlings")
```



Let's fit a GLM (Poisson regression)

```
seedl.glm <- glm(count ~ light, data = seedl, family = poisson)
tidy(seedl.glm)

## # A tibble: 2 x 5
##   term      estimate std.error statistic p.value
##   <chr>     <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  0.882     0.189     4.67  0.00000304
## 2 light       -0.00258   0.00353   -0.730  0.465
```

Does light explain variation in counts

```
AIC(seedl.glm) # model with light  
## [1] 182.0335  
AIC(glm(count ~ 1, data = seedl, family = poisson)) # null model  
## [1] 180.5706
```

Should consider multimodel inference...

Interpreting Poisson regression output

Parameter estimates (log scale):

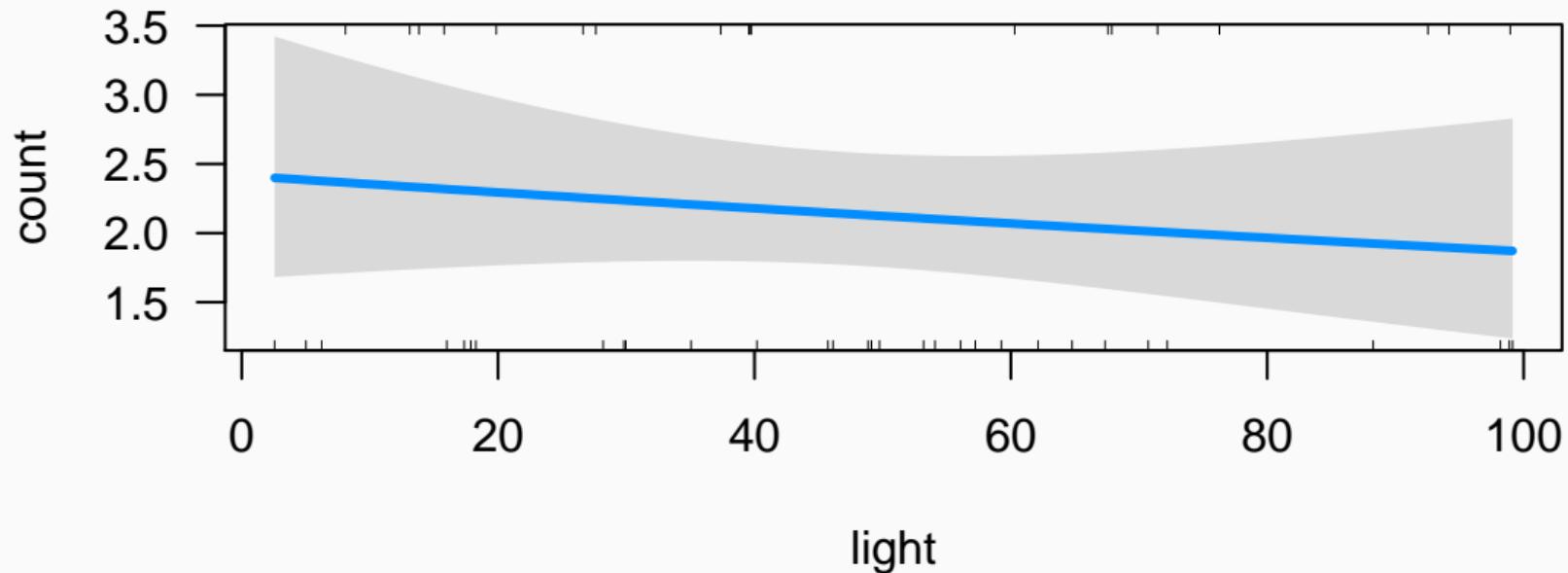
```
coef(seed1.glm)
## (Intercept)      light
## 0.881805022 -0.002575656
```

Let's back-transform the intercept for $light = 0$ to get corresponding number of seedlings

```
exp(coef(seed1.glm)[1])
## (Intercept)
##     2.415255
```

What is the relationship between Nseedlings and light? Use visreg package

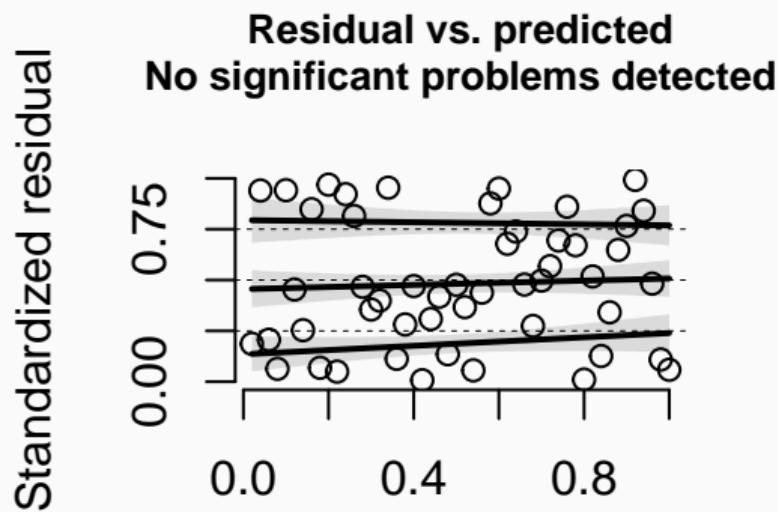
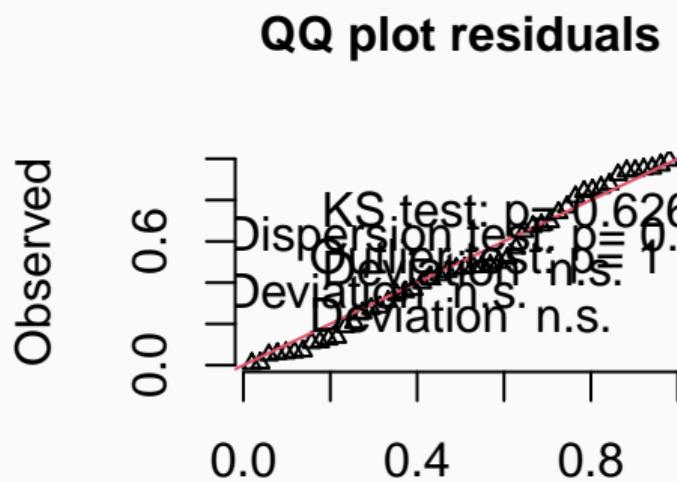
```
visreg::visreg(seedl.glm, scale = "response")
```



Residuals diagnostics with DHARMA package

```
simulateResiduals(seed1.glm, plot = TRUE)
```

DHARMA residual diagnostics

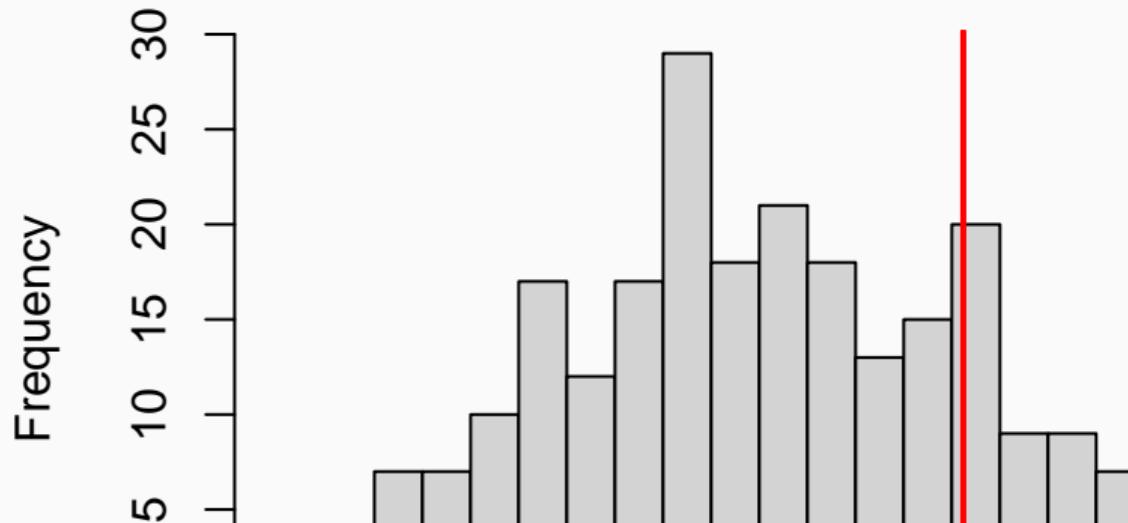


Poisson regression: Overdispersion

Always check overdispersion with count data

```
simres <- simulateResiduals(seed1.glm, refit = TRUE)  
testOverdispersion(simres)
```

Dispersion test n.s.



Accounting for overdispersion in count data

Use family quasipoisson

```
seedl.overdisp <- glm(count ~ light, data = seedl, family = quasipoisson)
tidy(seedl.overdisp)
```

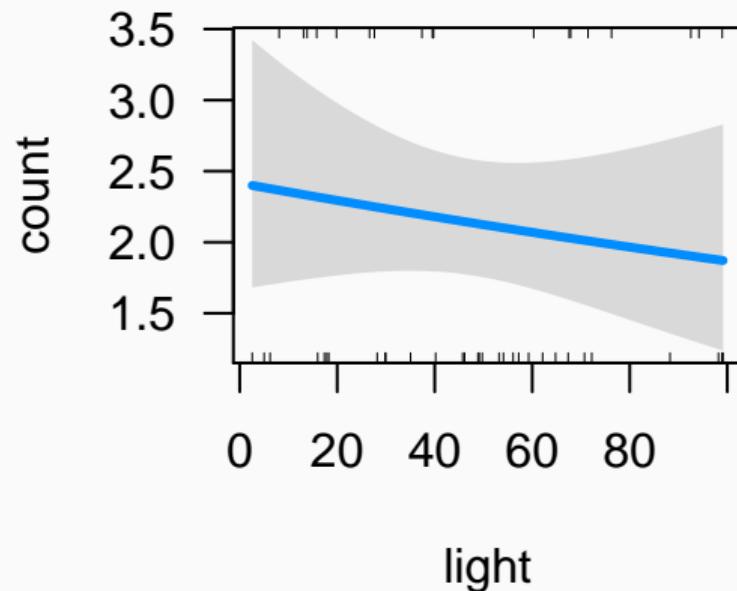
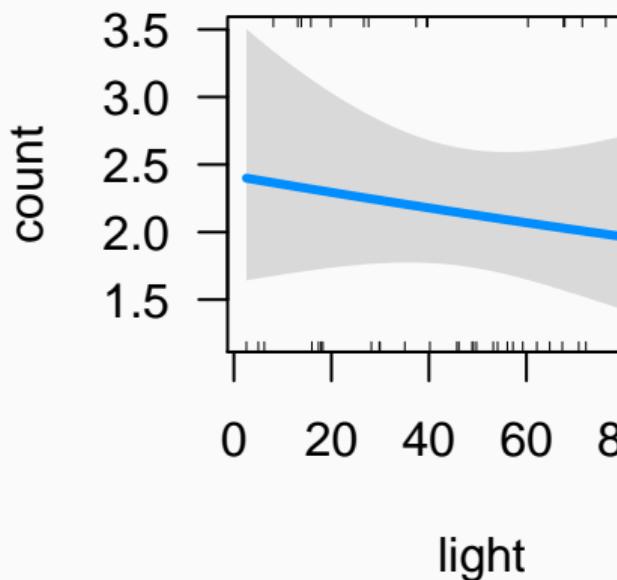
```
## # A tibble: 2 x 5
##   term      estimate std.error statistic p.value
##   <chr>      <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  0.882     0.201     4.38  0.0000637
## 2 light       -0.00258   0.00376   -0.685  0.496
```

Mean estimates do not change after accounting for overdispersion

```
allEffects(seed1.overdisp)
## model: count ~ light
##
## light effect
## light
##      3       30       50       70       100
## 2.396665 2.235657 2.123408 2.016794 1.866826

allEffects(seed1.glm)
## model: count ~ light
##
## light effect
## light
##      3       30       50       70       100
## 2.396665 2.235657 2.123408 2.016794 1.866826
```

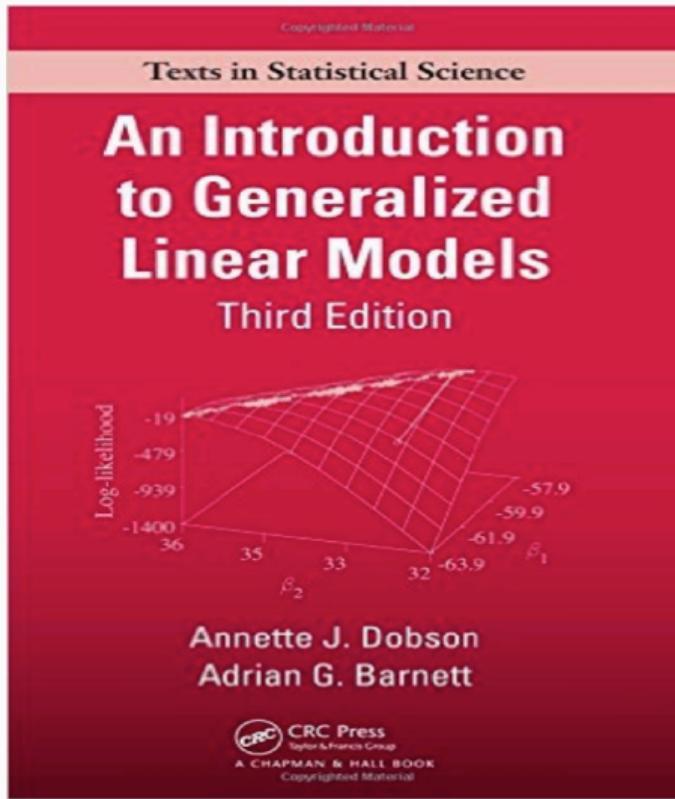
Standard errors do not change here



GLMs in a nutshell

Distribution	Link	Link ⁻¹	Use for	R syntax
normal	identity	1	real values	<code>lm()</code>
poisson	log	exp	counts	<code>glm(,family=poisson)</code>
binomial	logit	$1/(1 + \exp(-x))$	binary, proportions	<code>glm(,family=binomial)</code>

Textbooks



Practical #1

This Class

On our plate

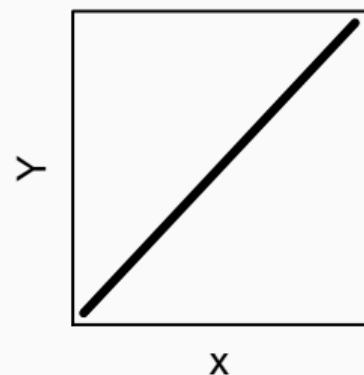
- Distributions and likelihoods
- Hypothesis testing and multimodel inference
- Introduction to Bayesian inference
- Generalized Linear Models (GLMs)
- Generalized Additive Models (GAMs)
- Mixed Effect Models

Generalized Additive Models (GAMs)

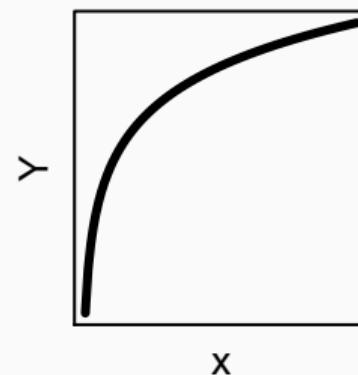
Motivation

In GLMs, we have **monotonic** link functions only

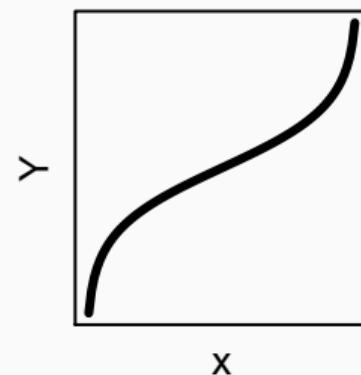
linear model



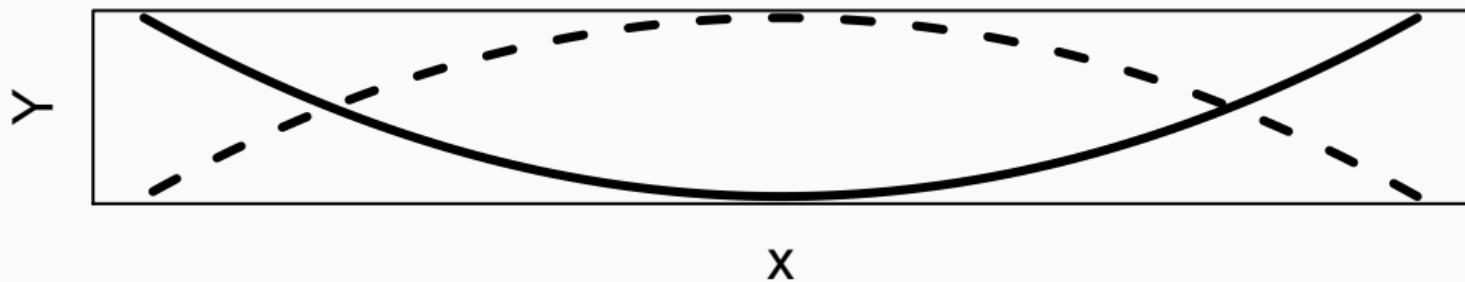
log model



logistic model



What about **non-monotonic** functions?



GAAAAAAAAAAAAAMs

- What is a GAM?
- What is smoothing?
- How do GAMs work? (*Roughly*)
- Fitting and plotting simple models

Generalized additive models

- Generalized: many response distributions
- Additive: terms **add** together
- Models: well, it's a model...

(Generalized) Linear Models

Models that look like:

$$Y_i \sim \text{Distribution}(\text{Mean Response})$$

$$\text{Mean Response} = \beta_0 + x_1 \beta_1 + \dots + x_P \beta_P$$

Describe the response, Y_i , as linear combination of the covariates, x_j for $j = 1, \dots, P$

For Distribution, we may pick a Normal, Poisson, Binomial, etc.

**Why bother with anything more
complicated?!**

Is this linear?

```
set.seed(2) ## simulate some data...
dat <- gamSim(1, n=400, dist="normal", scale=1, verbose=FALSE)
dat <- dat[,c("y", "x0", "x1", "x2", "x3")]
p <- ggplot(dat,aes(y=y,x=x1)) +
    geom_point() +
    theme_minimal()
print(p)
```



Is this linear? Maybe?

```
lm(y ~ x1, data=dat)
```

```
p <- ggplot(dat, aes(y=y, x=x1)) + geom_point() +  
  theme_minimal()  
print(p + geom_smooth(method="lm"))
```

15



231

What can we do?

Adding a quadratic term?

```
lm(y ~ x1 + poly(x1, 2), data=dat)

p <- ggplot(dat, aes(y=y, x=x1)) + geom_point() +
  theme_minimal()
print(p + geom_smooth(method="lm", formula=y~x+poly(x, 2)))
```

15



232

Is this sustainable?

- Adding in quadratic (and higher terms) *can* make sense
- This feels a bit *ad hoc*
- Better if we had a **framework** to deal with these issues?

```
p <- ggplot(dat, aes(y=y, x=x2)) + geom_point() +  
  theme_minimal()  
print(p + geom_smooth(method="lm", formula=y~x+poly(x, 2)))
```

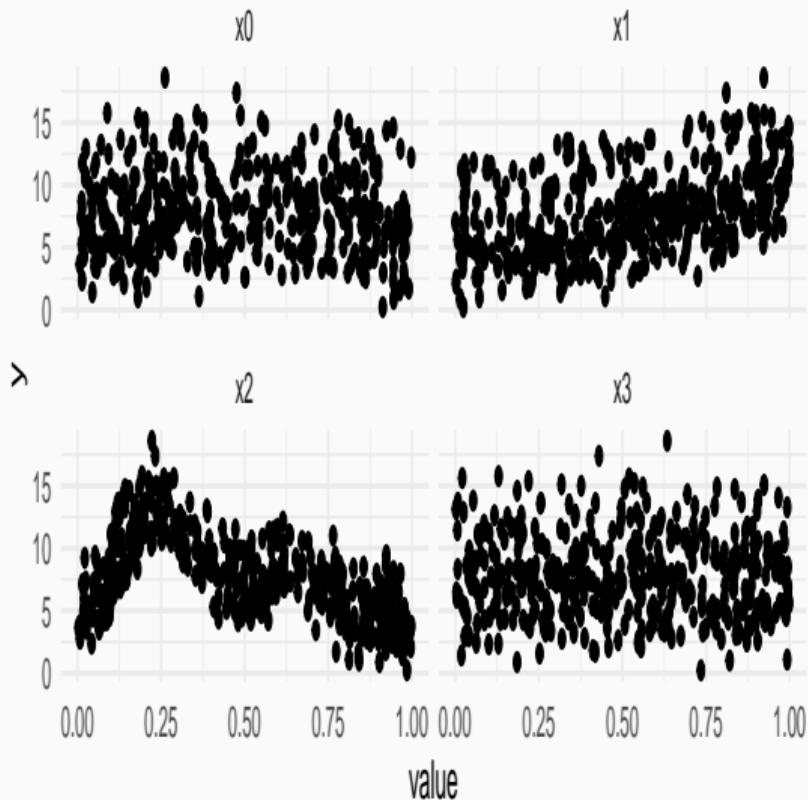


What does a model look like?

$$Y_i \sim \text{Distribution}(\text{Mean Response})$$

$$\text{Mean Response} = \beta_0 + s(x_1) + \dots + s(x_P)$$

Okay, but what about these “s” things?



- Think **s** for **smooth**
- Want to model the covariates flexibly
- Covariates and response not necessarily linearly related!
- Want some "wiggles"

Okay, but what about these “s” things?

- Think **s** for **smooth**
- Want to model the covariates flexibly
- Covariates and response not necessarily linearly related!
- Want some "wiggles"

Splines

Measuring wigglyness

- Visually:
 - * Lots of wiggles: Not smooth enough
 - * Straight line: Too smooth

Measuring wigglyness

```
# make three plots, w. estimated smooth, truth and data on each
par(mfrow=c(1,3), cex.main=3.5)

plot(b.justright, se=FALSE, ylim=ylim, main='Smoothing is fine',lwd=3)
points(dat$x2, dat$y-mean(dat$y),col='red')
#curve(f2,0,1, col="blue", add=TRUE)

plot(b.sp0, se=FALSE, ylim=ylim, main='Too many wiggles',lwd=3)
points(dat$x2, dat$y-mean(dat$y),col='red')
#curve(f2,0,1, col="blue", add=TRUE)

plot(b.spinf, se=FALSE, ylim=ylim, main='Straight line',lwd=3)
points(dat$x2, dat$y-mean(dat$y),col='red')
```

Is there a trade-off between lots of wiggles and a straight line?

- Yes, but:
 - Complex machinery involving derivatives
 - Calculate a **smoothing parameter**

Is there a trade-off between lots of wiggles and a straight line?

```
# make three plots, w. estimated smooth, truth and data on each
par(mfrow=c(1,3), cex.main=3.5)

plot(b.justright, se=FALSE, ylim=ylim, main=expression(lambda*plain(" is j"))
points(dat$x2, dat$y-mean(dat$y), col='red')
curve(f2,0,1, col="blue", add=TRUE)

plot(b.sp0, se=FALSE, ylim=ylim, main=expression(lambda*plain(" is ")*0),lu
points(dat$x2, dat$y-mean(dat$y), col='red')
curve(f2,0,1, col="blue", add=TRUE)

plot(b.spinf, se=FALSE, ylim=ylim, main=expression(lambda*plain(" is ")*infin
points(dat$x2, dat$y-mean(dat$y), col='red')
curve(f2,0,1, col="blue", add=TRUE)
```

Smoothing parameter λ selection

- Many methods
 - AIC
 - Mallow's C_p
 - cross-validation
 - Maximum-likelihood, restricted maximum-likelihood (REML)
- Based on simulation and practice, use REML

- Straight lines are boring — we want **wiggles**
- Use little functions (**basis functions**) to make big functions (**smooths**)
- Need to make sure your smooths are **wiggly enough**
- Use a **penalty** to trade off wigginess/generality

Fitting GAMs in practice

Translating maths into R

A simple example:

$$Y_i \sim \text{Normal}(\text{Mean Response}, \sigma^2)$$

$$\text{Mean Response} = \beta_0 + s(x) + s(w)$$

- linear predictor: `formula = y ~ s(x) + s(w)`
- response distribution: `family=gaussian()`
- data: `data=some_data_frame`

Putting that together

```
my_model <- gam(y ~ s(x) + s(w),  
                 family = gaussian(),  
                 data = some_data_frame,  
                 method = "REML")
```

- method="REML" uses REML for smoothness selection (default is "GCV.Cp")

What about an example?

Pantropical spotted dolphins

- Example taken from Miller et al. (2013). Spatial models for distance sampling data: recent developments and future directions. *Methods in Ecology and Evolution* 4: 1001-1010.
- See a proper analysis at
<http://distancesampling.org/R/vignettes/mexico-analysis.html>
- Simple example here, ignoring all kinds of important stuff!

```
knitr:::include_graphics('img/spotteddolphin_swfsc.jpg')
```



Inferential aims

```
load("dat/mexdolphins.RData")

library(rgdal)
library(rgeos)
library(maptools)
library(plyr)
# fill must be in the same order as the polygon data
grid_plot_obj <- function(fill, name, sp){

  # what was the data supplied?
  names(fill) <- NULL
  row.names(fill) <- NULL
  data <- data.frame(fill)
  names(data) <- name
```

A simple dolphin model

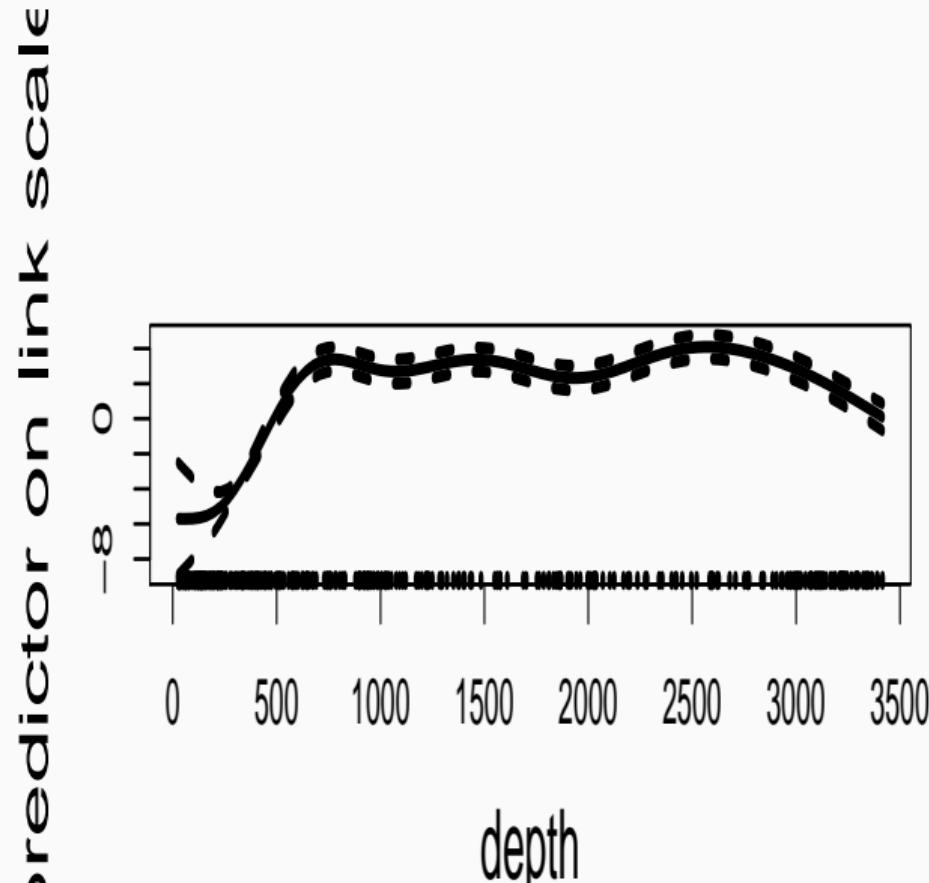
```
library(mgcv)
dolphins_depth <- gam(count ~ s(depth) + offset(off.set),
                      data = mexdolphins,
                      family = poisson(),
                      method = "REML")
```

- count is a function of depth
- off.set is the effort expended
- we have count data, hence we use a Poisson distribution

What did that do?

```
broom::tidy(dolphins_depth)
## # A tibble: 1 x 5
##   term      edf ref.df statistic p.value
##   <chr>    <dbl>  <dbl>     <dbl>    <dbl>
## 1 s(depth)  8.79   8.97     2283.      0
```

Plotting with `plot(dolphins_depth)`



Do we need the GAM?

```
AIC(dolphins_depth) # GAM
## [1] 17824.04

AIC(glm(count ~ depth + offset(off.set),
         data = mexdolphins,
         family = poisson()))) # GLM
## [1] 23349.98
```

Looks as though we definitely need it!

Bivariate terms on geographic locations

- Assumed an additive structure, with no interaction
- We can go a bit further and specify $s(x, y)$

```
dolphins_depth_xy <- gam(count ~ s(x, y) + offset(offset),  
                           data = mexdolphins,  
                           family=poisson(), method="REML")  
  
par(mfrow=c(1,3))  
  
vis.gam(dolphins_depth_xy, view=c("x", "y"), phi=45, theta=20, asp=1)  
vis.gam(dolphins_depth_xy, view=c("x", "y"), phi=45, theta=60, asp=1)  
vis.gam(dolphins_depth_xy, view=c("x", "y"), phi=45, theta=160, asp=1)
```



Adding a term

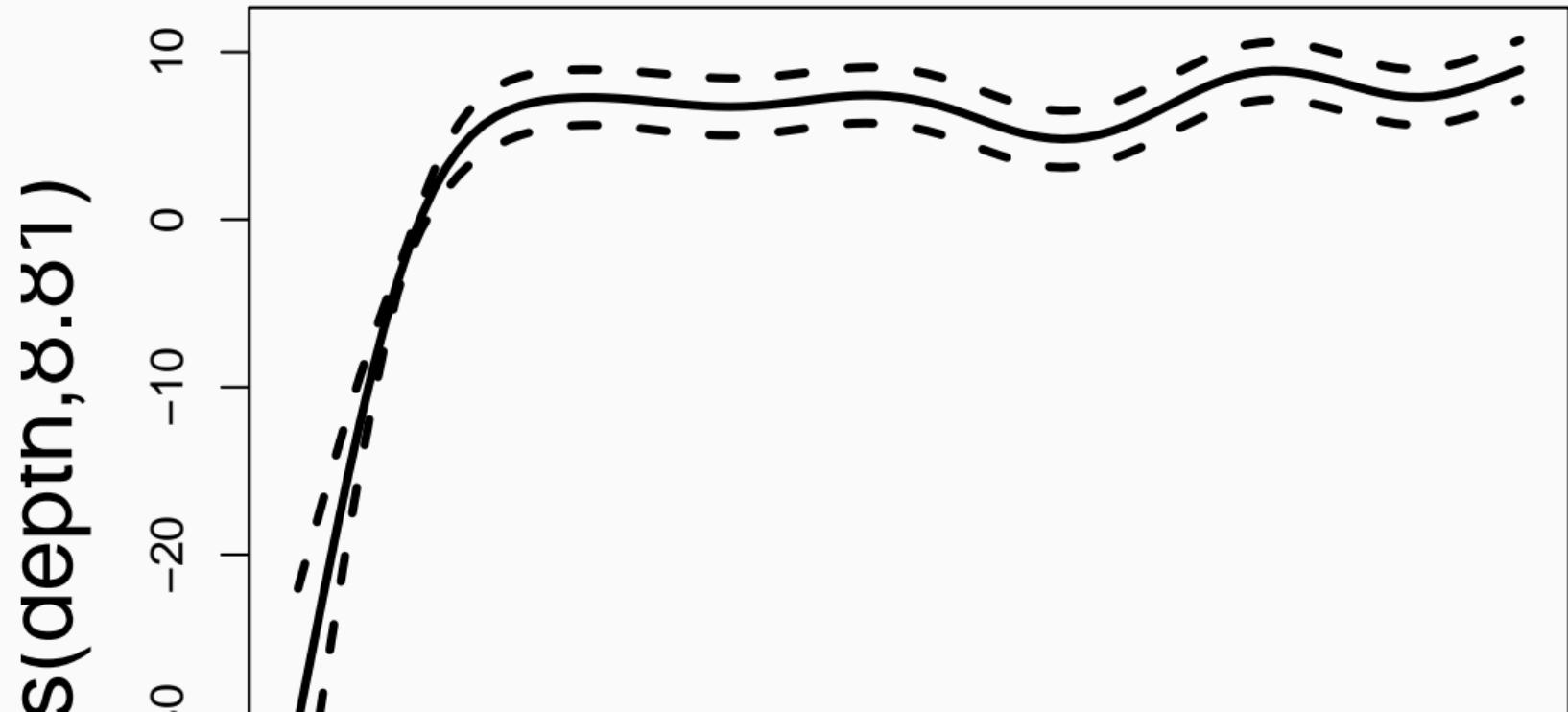
- Add a **surface** for location (x and y)
- Just use the + sign for an extra term

```
dolphins_depth_xy <- gam(count ~ s(depth) + s(x, y) + offset(offset),  
                           data = mexdolphins,  
                           family=poisson(), method="REML")
```

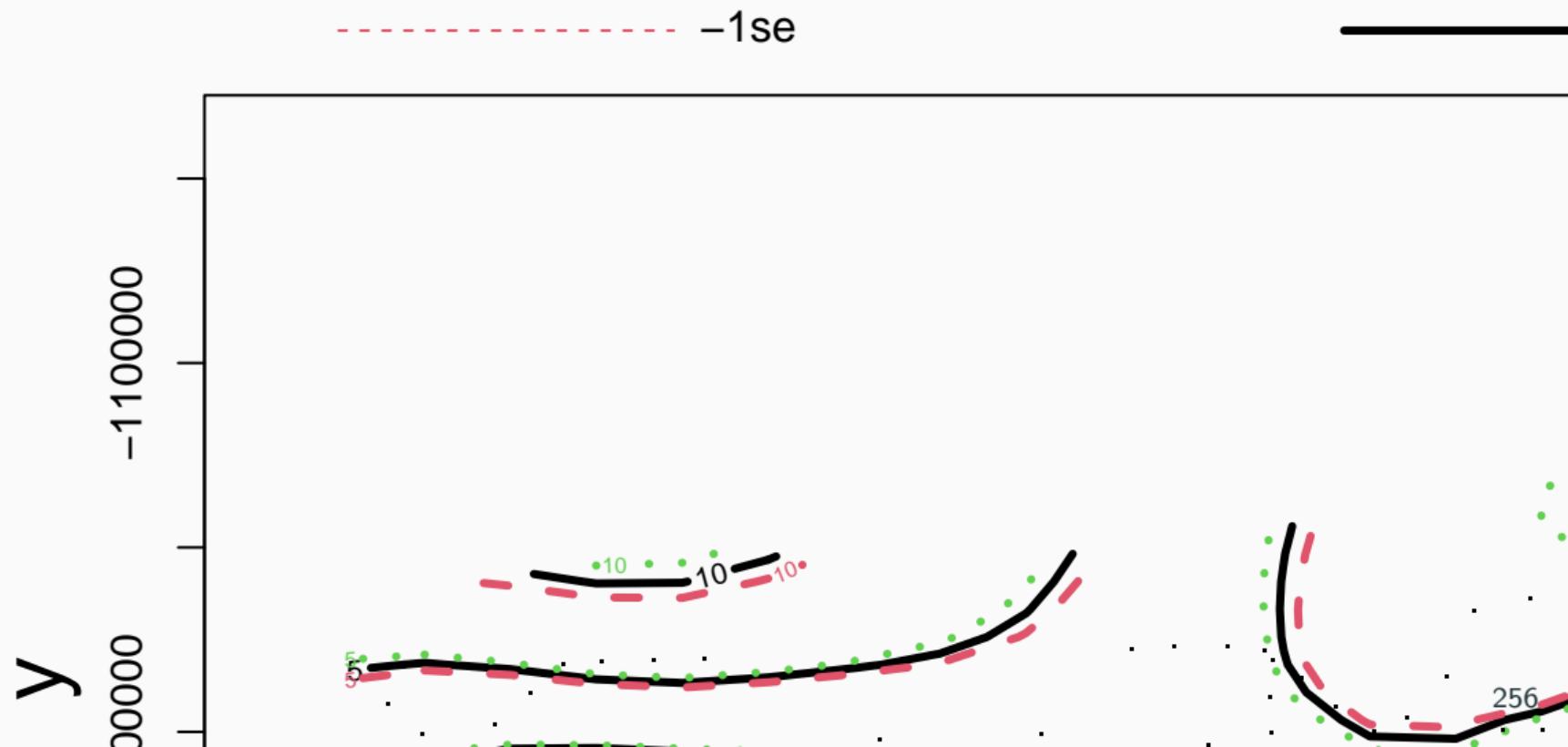
Summary

```
broom::tidy(dolphins_depth_xy)
## # A tibble: 2 x 5
##   term      edf ref.df statistic  p.value
##   <chr>    <dbl>  <dbl>     <dbl>    <dbl>
## 1 s(depth)  8.81   8.97     1140. 1.64e-236
## 2 s(x,y)   28.8   29.0     3574. 0.
```

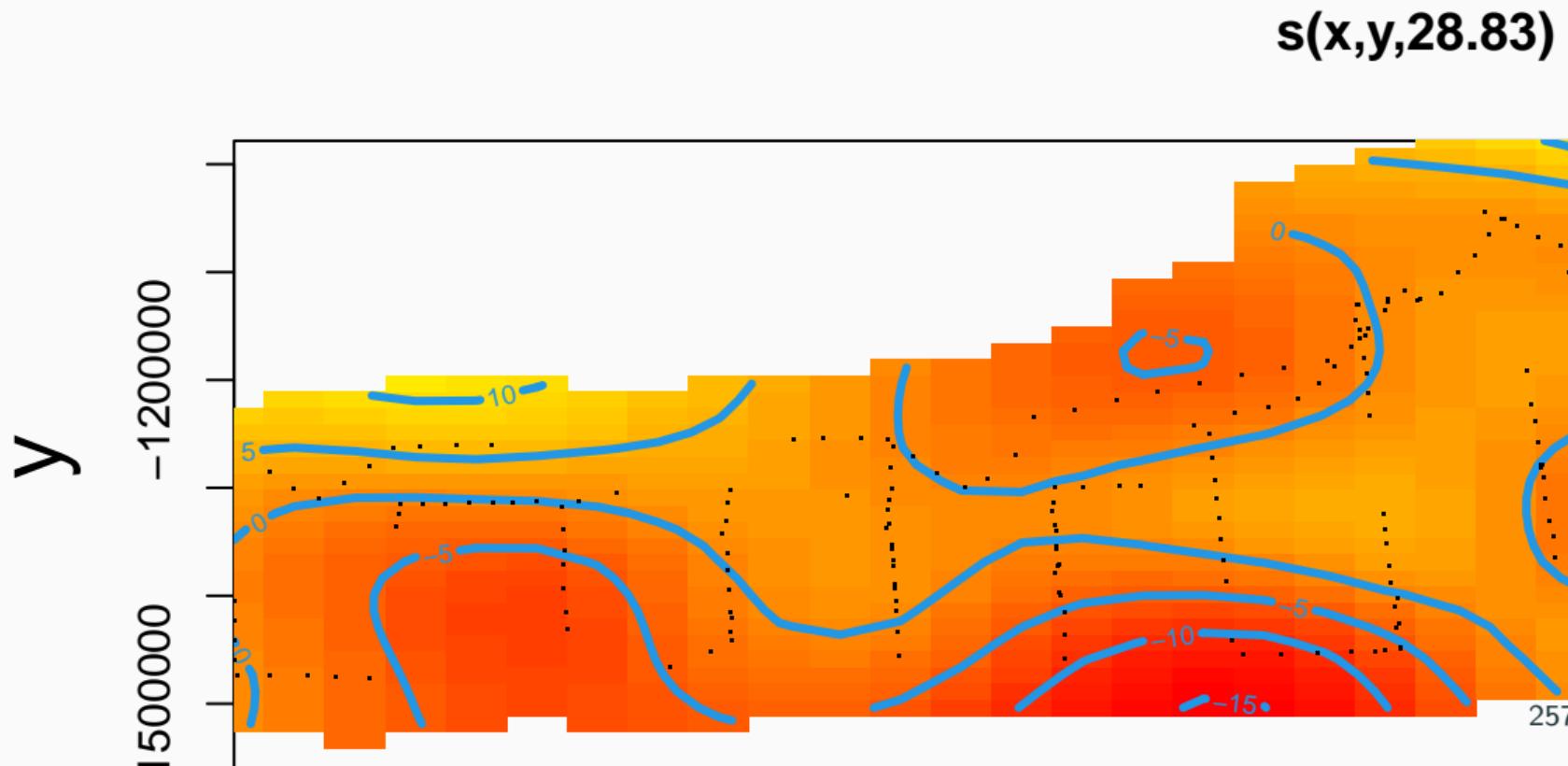
Plotting with `plot(dolphins_depth_xy, scale=0, pages=1)`



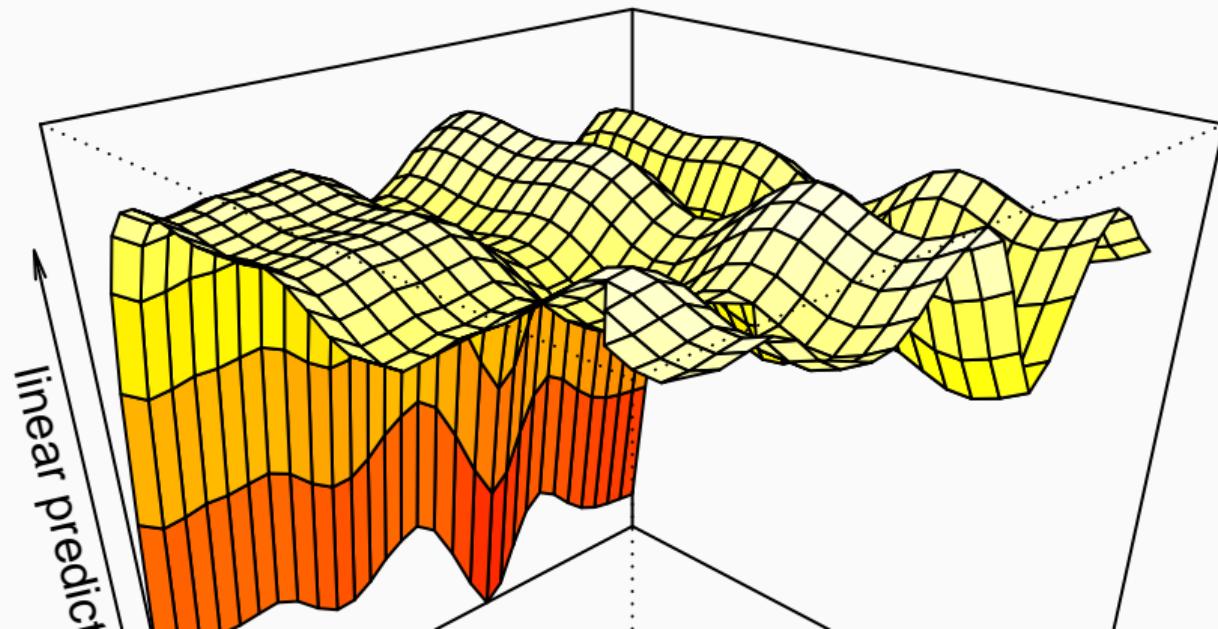
Plotting 2d terms with `plot(dolphins_depth_xy, select=2)`



Instead, let's try `plot(dolphins_depth_xy, select=2, scheme=2)`



More complex plots with vis.gam package



Fitting/plotting GAMs summary

- `gam` does all the work
- very similar to `glm`
- `s` indicates a smooth term
- `plot` can give simple plots
- `vis.gam` for more advanced stuff

Prediction

What is a prediction?

- Evaluate the model, at a particular covariate combination
- For example, answering the question “at a given depth, how many dolphins?”
- Steps: 1. evaluate the $s(\dots)$ terms 2. move to the response scale (exponentiate? Do nothing?) 3. if needed, multiply any offset

Example of prediction

- In maths:
 - * Model: $\text{count}_i = A_i \exp(\beta_0 + s(x_i, y_i) + s(\text{Depth}_i))$
 - * Drop in the values of x, y, Depth (and A)
- In R:
 - * Build a `data.frame` with x, y, Depth, A
 - * Use `predict()`

```
preds <- predict(my_model, newdat=my_data, type="response")
```

(`se.fit=TRUE` gives a standard error for each prediction)

Back to the dolphins...

Where are the dolphins?

```
d_preds <- predict(dolphins_depth_xy, newdata=preddata, type="response")  
  
p <- ggplot() +  
  grid_plot_obj(d_preds, "N", pred.polys) +  
  geom_line(aes(x, y, group=Transect.Label), data=mexdolphins) +  
  geom_polygon(aes(x=x, y=y, group = group), fill = "#1A9850", data=mapping) +  
  geom_point(aes(x, y, size=count),  
             data=mexdolphins[mexdolphins$count>0],  
             colour="red", alpha=I(0.7)) +  
  coord_fixed(ratio=1, ylim = range(mexdolphins$y), xlim = range(mexdolphins$x)) +  
  scale_fill_viridis() +  
  labs(fill="Predicted\\ndensity", x="x", y="y", size="Count") +  
  theme_minimal()  
  
print(p)
```

Where are the dolphins?

```
d_preds <- predict(dolphins_depth_xy, newdata=preddata,type="response")  
  
p <- ggplot() +  
  grid_plot_obj(d_preds, "N", pred.polys) +  
  geom_line(aes(x, y, group=Transect.Label), data=mexdolphins) +  
  geom_polygon(aes(x=x, y=y, group = group), fill = "#1A9850", data=mapping) +  
  geom_point(aes(x, y, size=count),  
             data=mexdolphins[mexdolphins$count>0],  
             colour="red", alpha=I(0.7)) +  
  coord_fixed(ratio=1, ylim = range(mexdolphins$y), xlim = range(mexdolphins$x)) +  
  scale_fill_viridis() +  
  labs(fill="Predicted\\ndensity", x="x", y="y", size="Count") +  
  theme_minimal()  
  
print(p)
```

Overdispersion, again! Use family=quasipoisson() in GAM

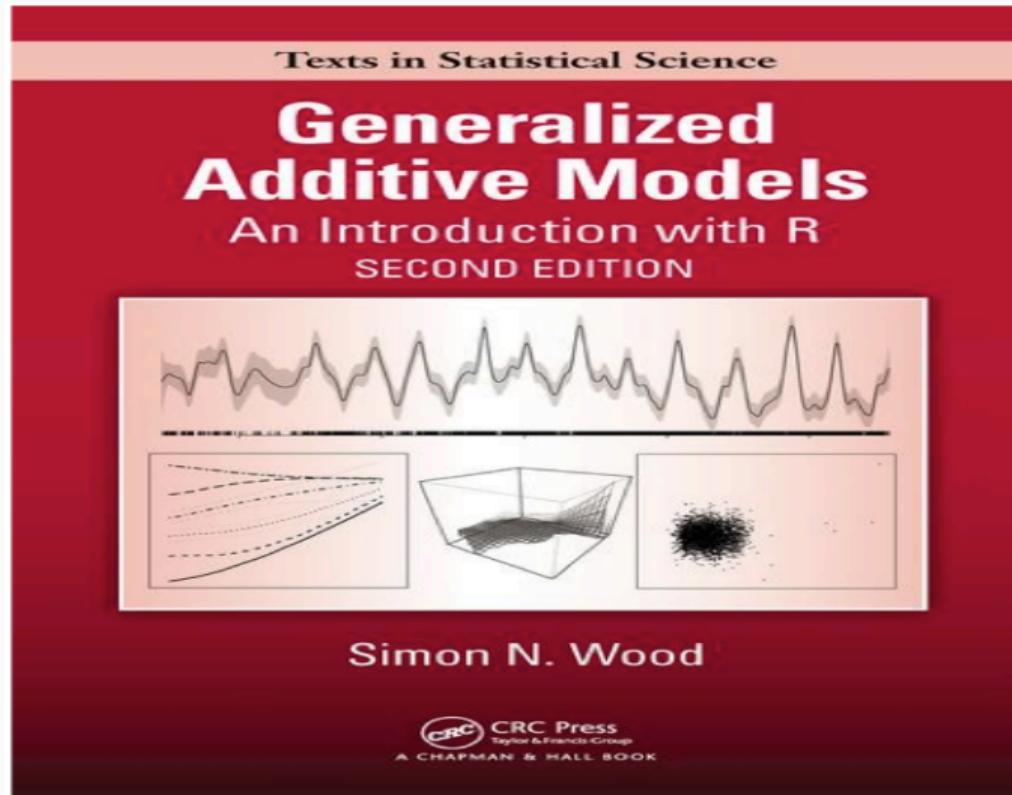
```
dolphins_depth_xy <- gam(count ~ s(depth) + s(x, y) + offset(off.set),  
                           data = mexdolphins,  
                           family=quasipoisson(), method="REML")  
dolphin_preds <- predict(dolphins_depth_xy, newdata=preddata,  
                           type="response")  
p <- ggplot() +  
  grid_plot_obj(dolphin_preds, "N", pred.polys) +  
  geom_line(aes(x, y, group=Transect.Label), data=mexdolphins) +  
  geom_polygon(aes(x=x, y=y, group = group), fill = "#1A9850", data=map) +  
  geom_point(aes(x, y, size=count),  
             data=mexdolphins[mexdolphins$count>0,],  
             colour="red", alpha=I(0.7)) +  
  coord_fixed(ratio=1, ylim = range(mexdolphins$y), xlim = range(mexdo  
264  
scale_fill_viridis() +
```

Summary

GAMs in a nutshell

- GAMs are GLMs plus some extra wiggles
- Need to make sure things are **just wiggly enough** - Basis + penalty is the way to do this
- Fitting looks like `glm` with extra `s()` terms
- You can mix `s()` terms with linear terms

Textbooks



Practical #2

This Class

On our plate

- Distributions and likelihoods
- Hypothesis testing and multimodel inference
- Introduction to Bayesian inference
- Generalized Linear Models (GLMs)
- Generalized Additive Models (GAMs)
- Mixed Effect Models

Mixed effect models

What are random effects?

- Mixed models include both fixed and random effects
- Random effects are statistical parameters that attempt to explain noise caused by sub-populations of the population you are trying to model
- A random-effect model assumes that the dataset being analysed consists of a hierarchy of different populations whose differences relate to that hierarchy
- Measurement that come in groups