

Methods 3: Multilevel Statistical Modeling and Machine Learning

Week 4: *Explanation and prediction*
October 5, 2021

by: Lau Møller Andersen

These slides are distributed according to the CC
BY 4.0 licence:

<https://creativecommons.org/licenses/by/4.0/>



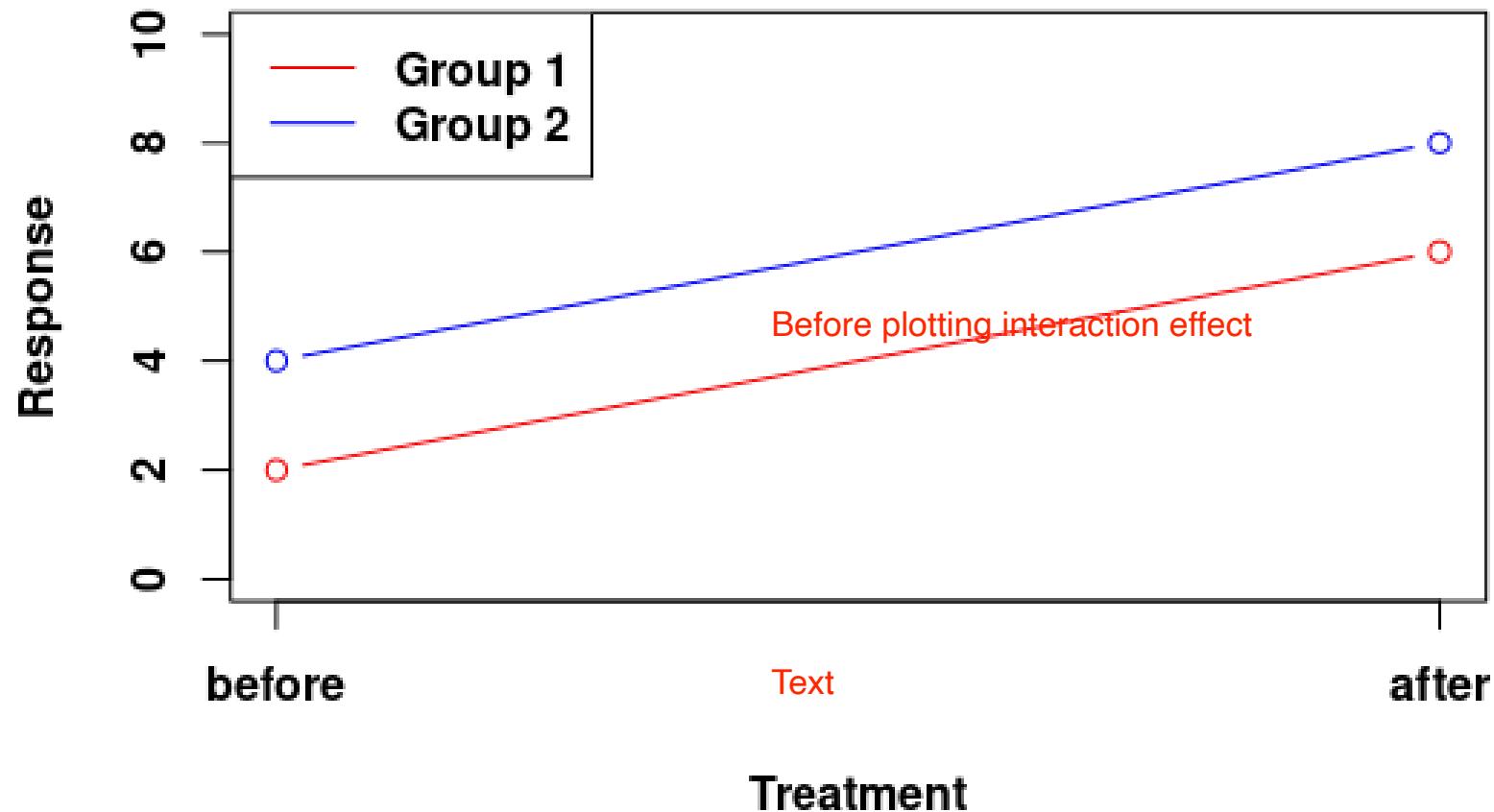
Attribution 4.0 International (CC BY 4.0)

Questions since last time

From CryptPad

Can you explain what it means that significance of main effects is uninterpretable in case of a significant interaction? In our case, the interaction wasn't significant. But what would we have done, had it in fact been significant?

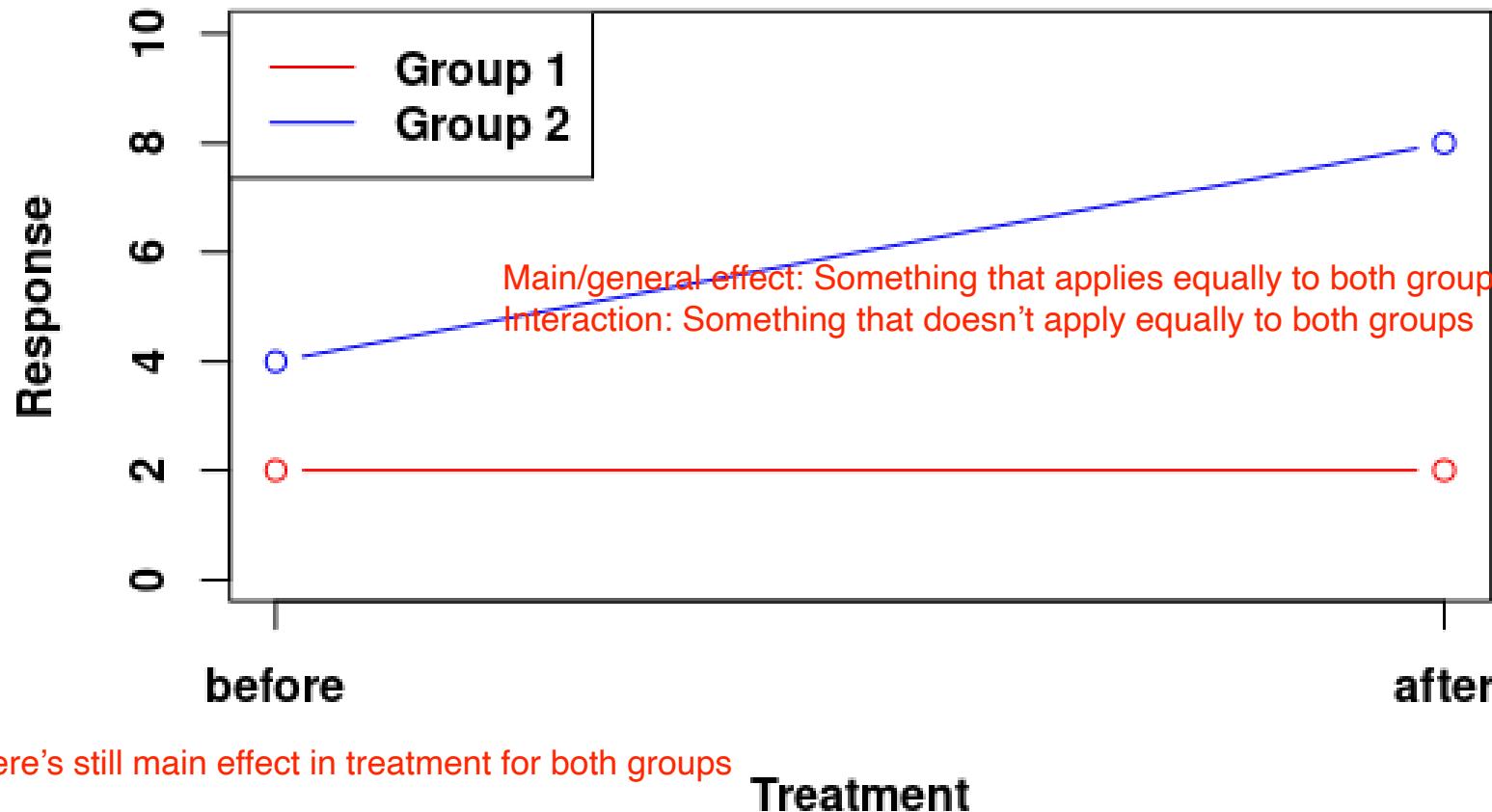
Main effect of 4 (interaction term = 0)



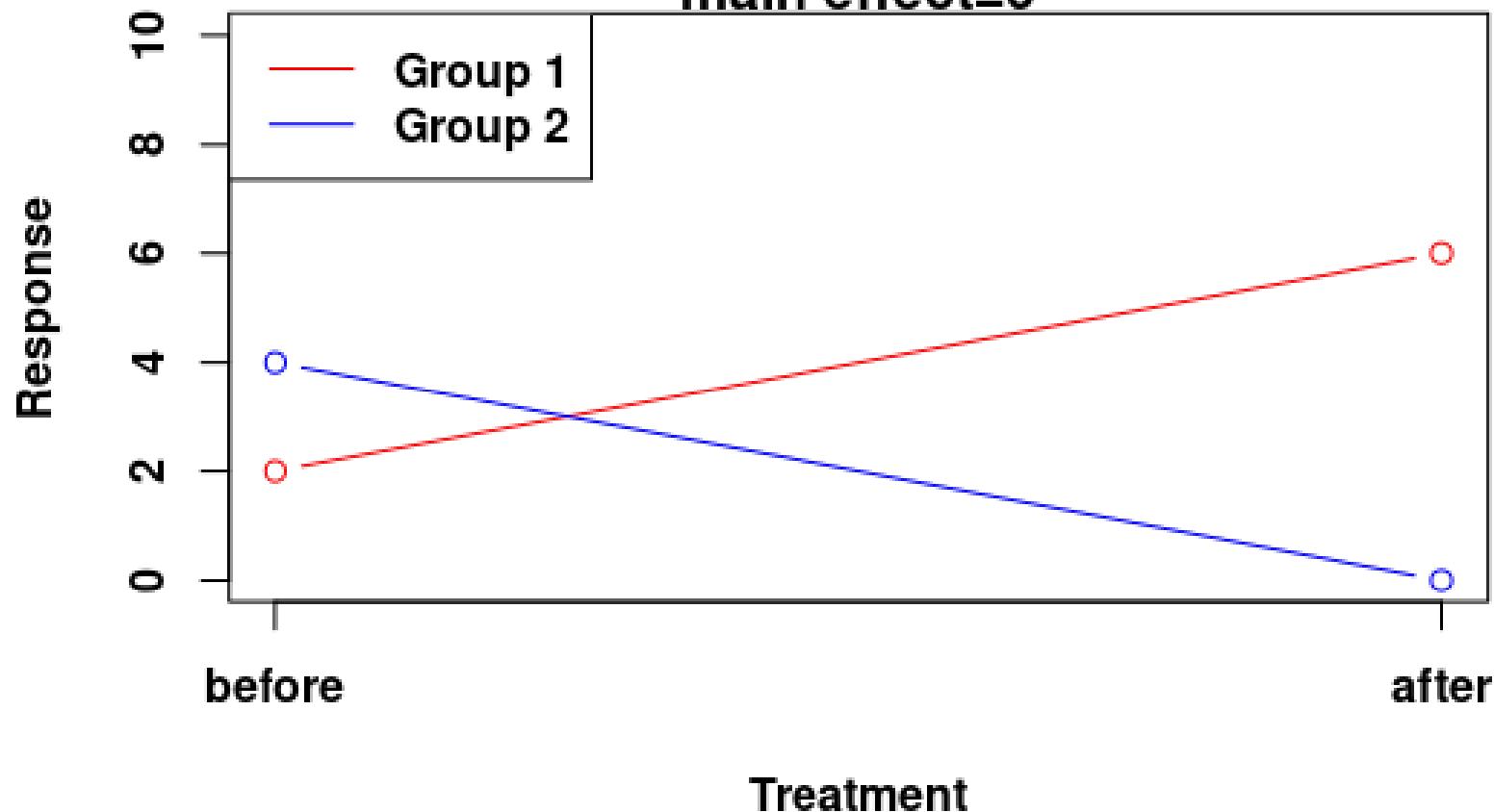
After modelling interaction effect:

PLOT
interaction.plot()

Interaction - (main effect = 2, not interpretable)



Interaction - cross-interaction; opposite effects main effect=0



```
mtcars$int <- interaction(mtcars$gear, mtcars$wt) # int = a new variable
```

this will save the four level interaction values

Instead of adding the coefficient numbers together, you can instead run

```
lm(resp ~ int + o)
```

More questions?

Learning goals^{from last time}

Generalized Linear Mixed Effects Models (GLMM)
multi level modelling including non normal data

- 1) Understanding that we can extend the scope of our multilevel modelling by using appropriate link functions and data distributions
- 2) Understanding the multilevel equivalent of the GLM

Text

First some remarks on Exercises

- Assignments (those that go in the portfolio are marked in the syllabus)
- I'll make an effort to write what I expect of you more succinctly
- Deadlines:
 - My suggestion: we move deadline to Wednesday (23.59)
- This week:
 - Practical exercise will be code review of each other's assignments so far

At least four ingredients needed

- 1) A data vector: $y = (y_1, \dots, y_n)$
- 2) Predictors: X and coefficients β , forming a linear predictor $X\beta$
- 3) A *link function* g : yielding a vector of transformed data $\hat{y} = g^{-1}(X\beta)$
that are used to model the data
Link function transforms the linear function
of which the inverse will bring the data
back to normal.
- 4) A data distribution: $p(y|\hat{y})$
if data's binary; how will i behave
or if count; -ll-

$$(X\beta = \beta_0 + X_1\beta_1 + \dots + X_k\beta_k)$$

(Gelman and Hill, 2006,
Chapter 6)



Breaking all
promises and
going back to
mtcars

1) A data vector: $y = (y_1, \dots, y_n)$

```
print(y <- mtcars$am)
```

```
## [1] 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1
```

2) Predictors: X and coefficients β , forming a linear predictor $X\beta$

```
logistic.model <- glm(am ~ wt + 1, data=mtcars, family='binomial')
X <- model.matrix(logistic.model)
print(head(X))
```

```
##                               (Intercept)      wt
## Mazda RX4                      1 2.620
## Mazda RX4 Wag                   1 2.875
## Datsun 710                      1 2.320
## Hornet 4 Drive                  1 3.215
## Hornet Sportabout                1 3.440
## Valiant                          1 3.460
```

```
print(beta.hat <- logistic.model$coefficients)
```

```
## (Intercept)          wt
##    12.04037     -4.02397
```

2) Predictors: X and coefficients β , forming a linear predictor $X\beta$

```
linear.predictor <- X %*% beta.hat  
print(head(linear.predictor))
```

```
## [1]  
## Mazda RX4      1.4975684  
## Mazda RX4 Wag  0.4714561  
## Datsun 710     2.7047594  
## Hornet 4 Drive -0.8966937  
## Hornet Sportabout -1.8020869  
## Valiant        -1.8825663
```

This lives on a continuous scale spanning all the real numbers

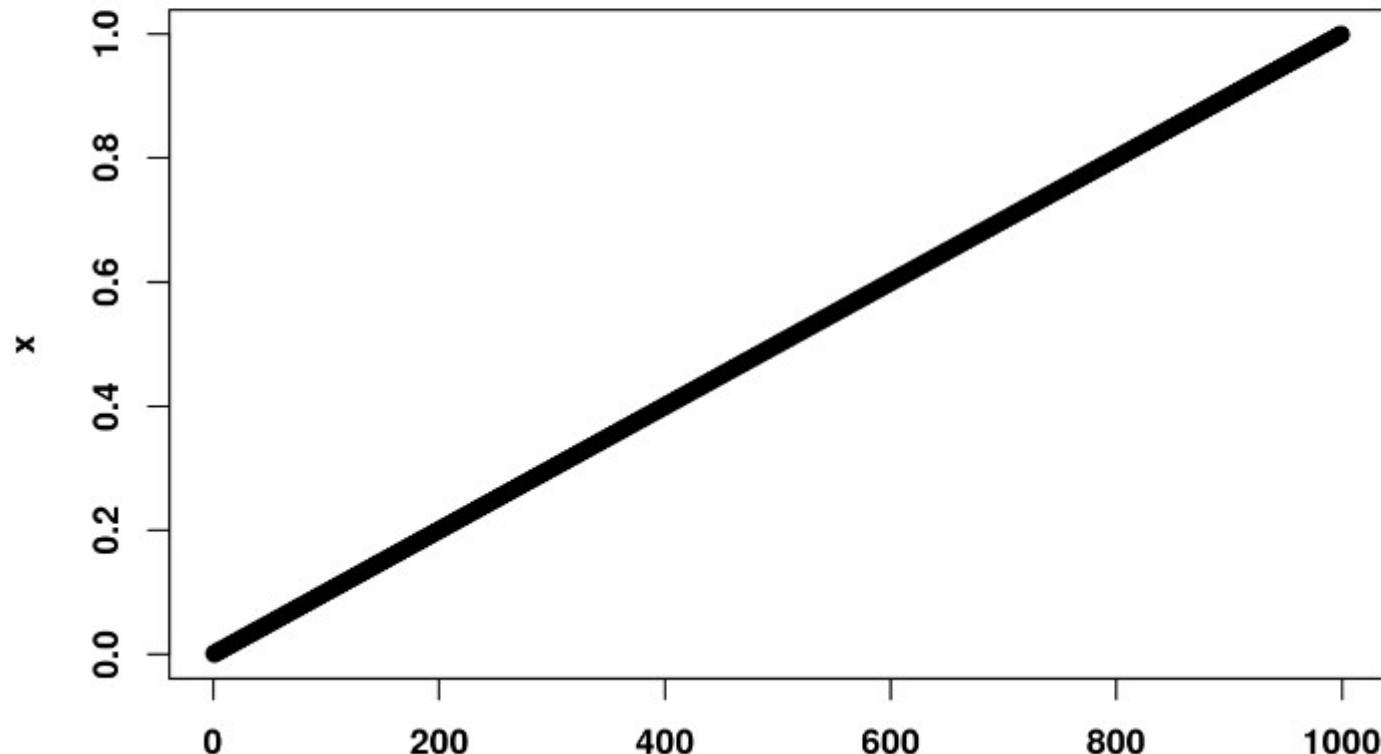
3) A *link function* g : yielding a vector of transformed data $\hat{y} = g^{-1}(X\beta)$ that are used to model the data

```
g <- function(x) log(x / (1 - x)) ## logit
inv.g <- function(x) exp(x) / (1 + exp(x)) ##logit--1
```

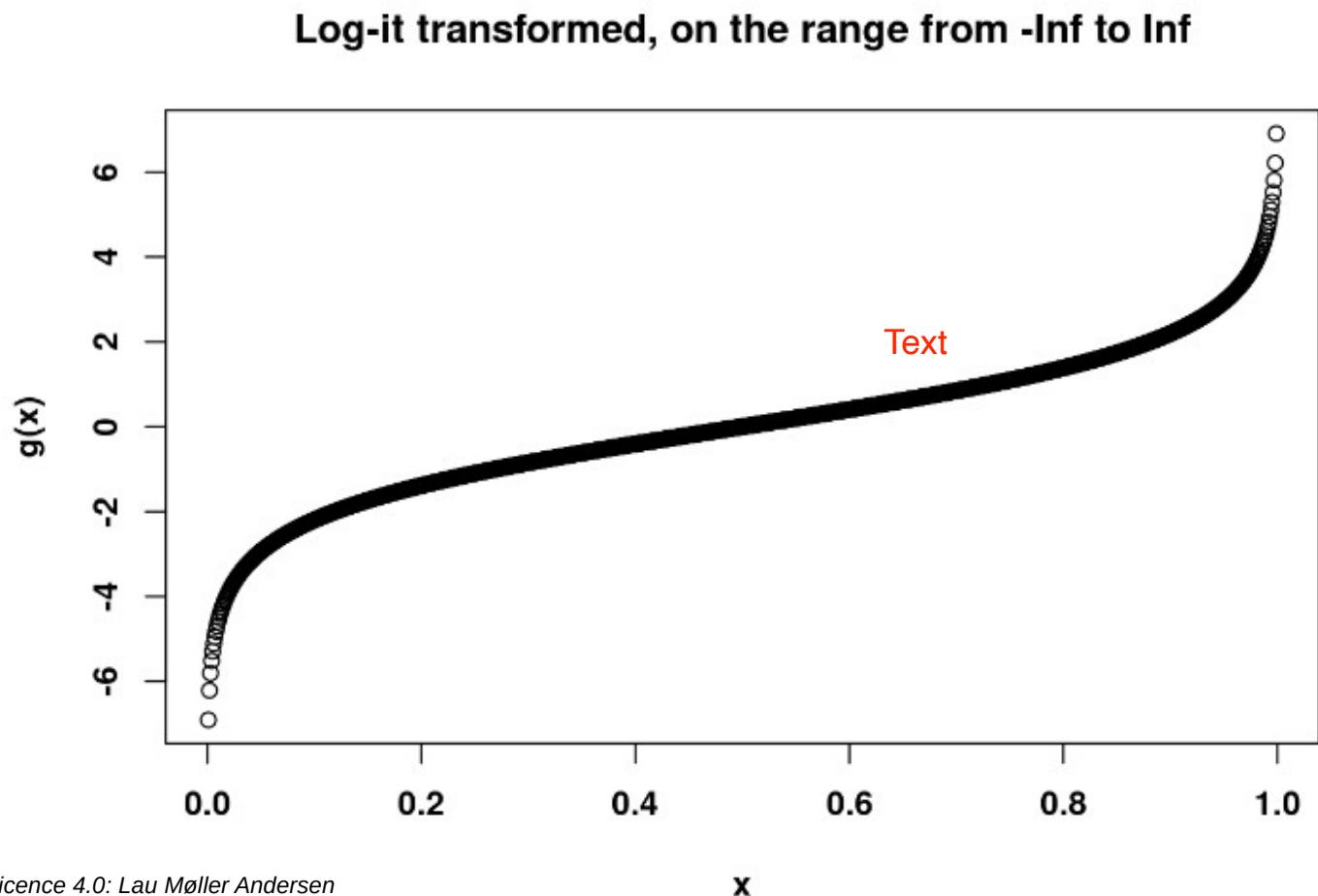
takes probabilityies, and brings them unto probability scale
this brings back to probability scale

```
x <- seq(0.001, 0.999, 0.001)
plot(x, main='Original probability data (on the range from 0-1)')
```

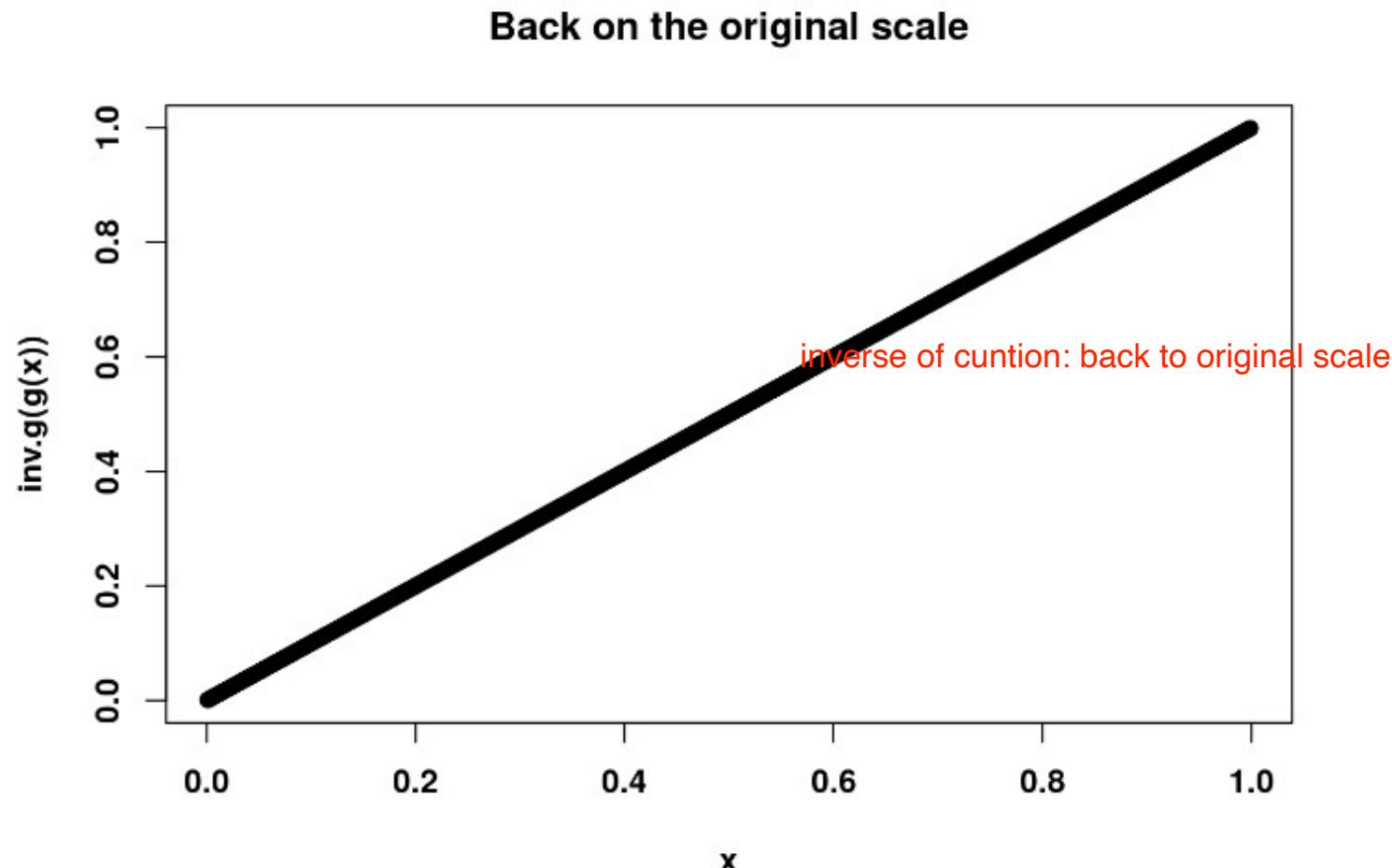
Original probability data (on the range from 0-1)



```
plot(x, g(x), main='Log-it transformed, on the range from -Inf to Inf')
```



```
plot(x, inv.g(g(x)), main='Back on the original scale')
```



These are the fitted values

```
y.hat <- inv.g(X %*% beta.hat)  
print(head(y.hat))
```

```
## [,1]  
## Mazda RX4      0.8172115  
## Mazda RX4 Wag   0.6157283  
## Datsun 710     0.9373069  
## Hornet 4 Drive  0.2897304  
## Hornet Sportabout 0.1415972  
## Valiant        0.1320944
```

```
print(head(y.hat - logistic.model$fitted.values))
```

Text

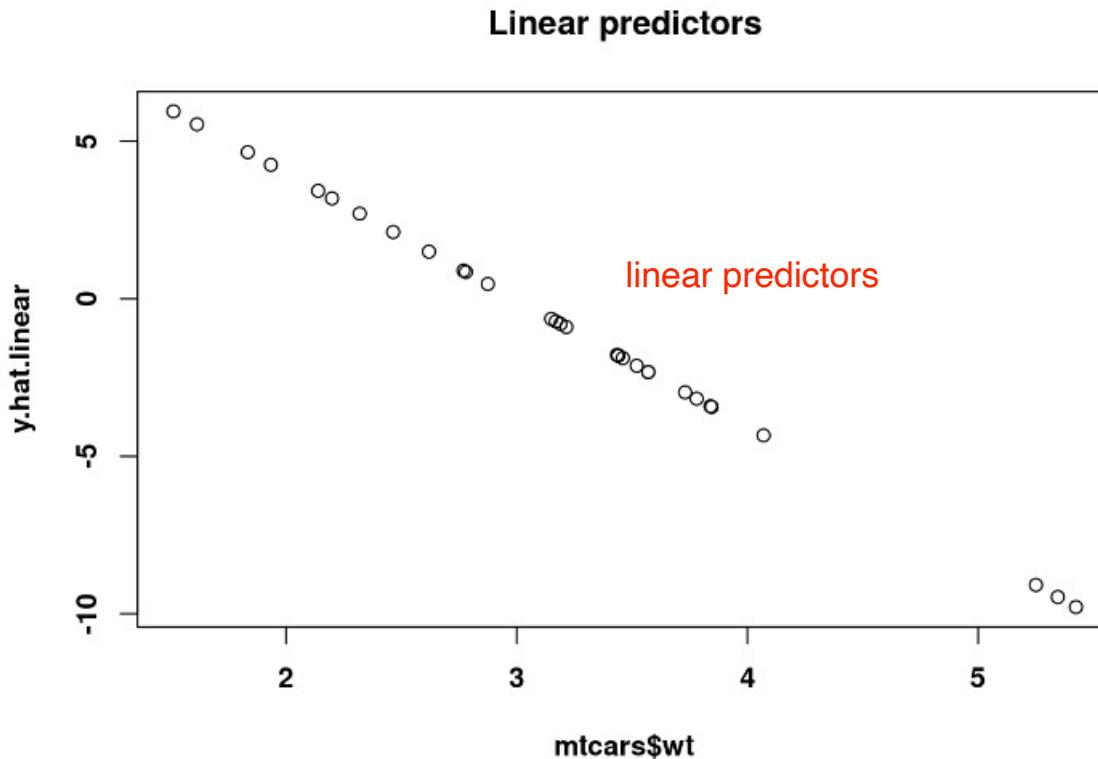
```
## [,1]  
## Mazda RX4      0  
## Mazda RX4 Wag   0  
## Datsun 710     0  
## Hornet 4 Drive  0  
## Hornet Sportabout 0  
## Valiant        0
```

These are the linear predictors

```
y.hat.linear <- X %*% beta.hat  
print(head(y.hat.linear - logistic.model$linear.predictors))
```

```
##                                     [,1]  
## Mazda RX4                  0  
## Mazda RX4 Wag               0  
## Datsun 710                  0  
## Hornet 4 Drive              0  
## Hornet Sportabout            0  
## Valiant                     0
```

Looks like a “normal” linear regression



4) A data distribution: $p(y|\hat{y})$

Binomial distribution $B(n, p)$

Probability mass function (PMF): $\binom{n}{k} p^k q^{(n-k)}$

$n \in \{0, 1, 2, \dots\}$ - number of trials

$p \in [0, 1]$ - success probability for each trial

$$q = 1 - p$$

$k \in \{0, 1, \dots, n\}$ - number of successes

special case of binomial distribution:

PMF:

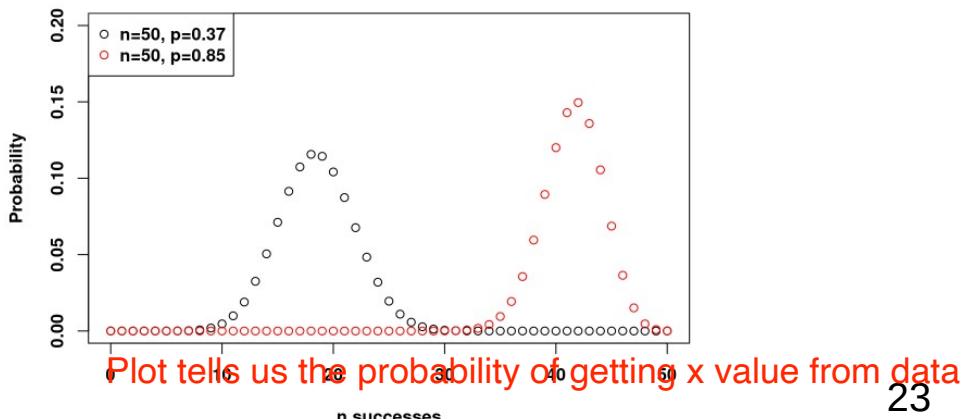
for the normal distribution, it is the normal bell-curve.

pmf function that tells in how many way u can get x succes.

```
x <- 0:50
n <- 50
p <- 0.37
pmf <- dbinom(x=x, size=n, prob=p)

par(font.lab=2, font.axis=2)
plot(x, pmf, xlab='n successes', ylab='Probability', ylim=c(0, 0.2))

new.p <- 0.83
new.pmf <- dbinom(x=x, size=n, prob=new.p)
points(x, new.pmf, col='red')
legend('topleft', col=c('black', 'red'),
       legend=c('n=50, p=0.37', 'n=50, p=0.85'), pch=1,
       text.font=2)
```



4) A data distribution: $p(y|\hat{y})$

But we are looking at the special case of n=1

in our case, our n is 1 (special case)

$$\Pr(y=1)=\hat{y}$$

The *Bernoulli* distribution :

$$0 \leq p \leq 1$$

$$q = 1 - p$$

$$k \in \{0,1\}$$

$$\text{PMF}_{\text{Bernoulli}} = p^k q^{1-k}$$

only one way, with one flip, to get one success (if throwing dices).

$$\text{PMF}_{\text{Bernoulli}} = \binom{n}{k} p^k q^{(n-k)} \quad (n=1 \text{ and } k \in \{0,1\})$$

byt om

4) A data distribution: $p(y|\hat{y})$

$$\text{PMF}_{\text{Bernoulli}} = p^k q^{1-k}$$

```
dbernoul <- function(p, k) p^k * (1 - p)^(1 - k)

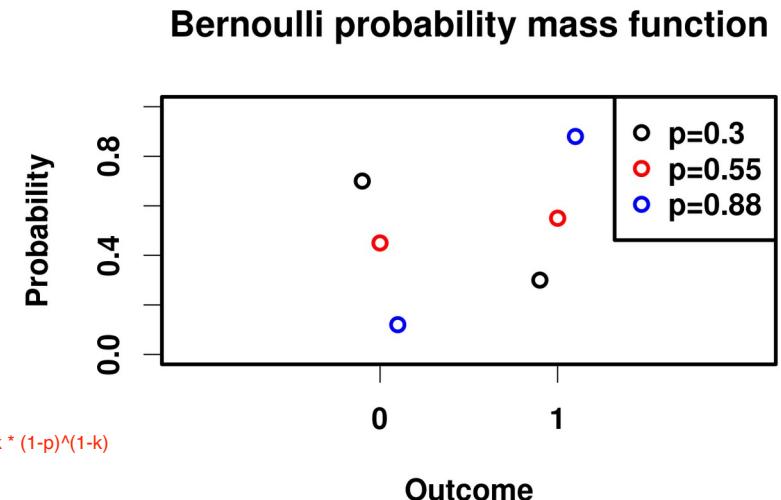
x <- c(0, 1)
pmf <- dbernoul(p=0.3, x)
par(font.lab=2, font.axis=2)
plot(x -0.1, pmf, xaxt='n', xlab='Outcome', ylab='Probability', ylim=c(0,
1),
     xlim=c(-1.1, 2.1), main='Bernoulli probability mass function')
axis(side=1, at=c(0, 1), labels=c(0, 1))

pmf <- dbernoul(p=0.55, x)
points(x, pmf, col='red')

pmf <- dbernoul(p=0.88, x)
points(x + 0.1, pmf, col='blue')

legend('topright', pch=1, col=c('black', 'red', 'blue'),
       legend=c('p=0.3', 'p=0.55', 'p=0.88'), text.font=2)
```

Text



Some link functions

Usage

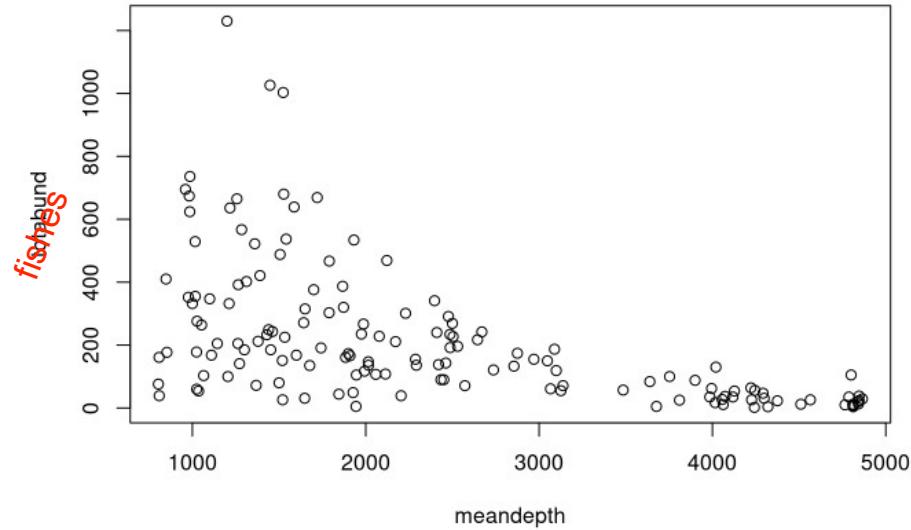
```
family(object, ...)

binomial(link = "logit")
gaussian(link = "identity")
Gamma(link = "inverse")
inverse.gaussian(link = "1/mu^2")
poisson(link = "log")
quasi(link = "identity", variance = "constant")
quasibinomial(link = "logit")
quasipoisson(link = "log")
```

Have a look at Poisson as well

library(COUNT)

```
data(fishing)  
plot(totabund ~ meandepth, data=fishing)
```



1) A data vector: $y = (y_1, \dots, y_n)$

```
print(y <- fishing$totabund)

## [1]  76 161  39 410 177 695 352 674 624 736 332 529 355
178  60
## [16] 276  54 264 103 347 168 205 1230 100 332 636 665 205
392 141
## [31] 567 185 402 522  72 212 421 233 249 1026 185 243  80
488 151
## [46] 26 1003 680 225 537 639 168 271  31 315 135 376 669
191 303
## [61] 467  44 387 320 161 173 166  49 534   5 105 235 267
117 147
## [76] 136 107 228 108 469 211  39 301 155 136 341 240 138
90  90
## [91] 143 291 234 192 269 227 196  71 217 242 121 133 174
155 150
## [106] 61 187 119  55  71  57  84   5 100  25  88  35  62
17  130
## [121] 27  11  37  35  54  64  26   2  56  47  31   4  23
12  26
## [136] 10  35 105  12   7   4  10  20  13  24  38  29
## attr(),"label")
## [1] "total number fish/site"
## attr(),"class")
## [1] "labelled" "integer"
## attr(),"format")
## [1] "%9.0g"
```

2) Predictors: X and coefficients β , forming a linear predictor $X\beta$

```
poisson.model <- glm(totabund ~ meandepth, data=fishing, family='poisson')
X <- model.matrix(poisson.model)
print(head(X))

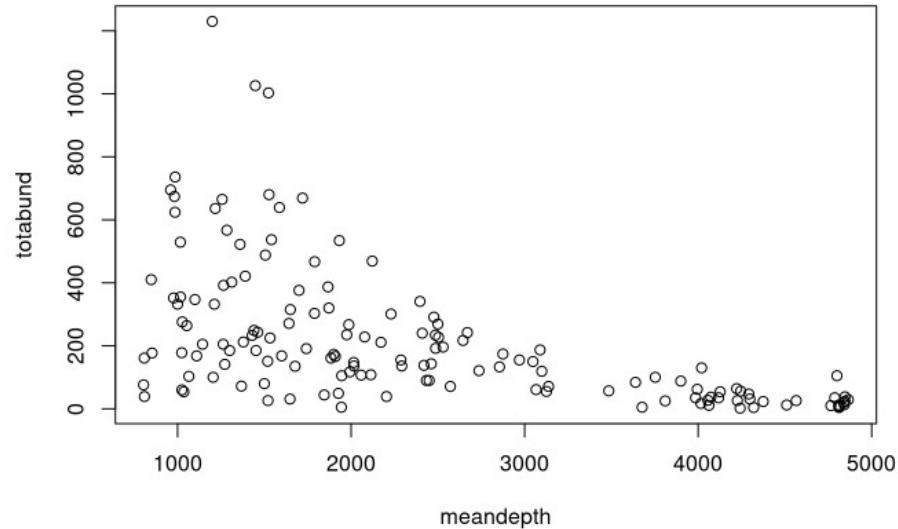
##      (Intercept) meandepth
## 1            1     804
## 2            1     808
## 3            1     809
## 4            1     848
## 5            1     853
## 6            1     960

print(beta.hat <- poisson.model$coefficients)

##      (Intercept)    meandepth
## 6.6465755329 -0.0006309148
```

Data is not continuous...

```
data(fishing)  
plot(totabund ~ meandepth, data=fishing)
```



... but the linear predictor is

$X\beta \in R_{>0}$

```
linear.predictor <- X %*% beta.hat  
print(head(linear.predictor))
```

```
##      [,1]  
## 1 6.139320  
## 2 6.136796  
## 3 6.136165  
## 4 6.111560  
## 5 6.108405  
## 6 6.040897
```

```
print(head(poission.model$linear.predictors))
```

```
##      1      2      3      4      5      6  
## 6.139320 6.136796 6.136165 6.111560 6.108405 6.040897
```

3) A *link function* g : yielding a vector of transformed data $\hat{y} = g^{-1}(X\beta)$ that are used to model the data

```
g <- log ## logarithm (using a given base, default: Euler's number); will  
bring it on a continuous scale from (0 to Inf)  
inv.g <- exp ## exponential (based on Euler's number) (will bring it back  
to a positive number (not an integer though))
```

3) A *link function* g : yielding a vector of transformed data $\hat{y} = g^{-1}(X\beta)$ that are used to model the data

```
y.hat <- inv.g(X %*% beta.hat)
print(head(y.hat))
```

```
##      [,1]
## 1 463.7382
## 2 462.5693
## 3 462.2776
## 4 451.0417
## 5 449.6211
## 6 420.2700
```

```
print(head(poisson.model$fitted.values))
```

```
##      1      2      3      4      5      6
## 463.7382 462.5693 462.2776 451.0417 449.6211 420.2700
```

4) A data distribution: $p(y|\hat{y})$

Poisson(λ)

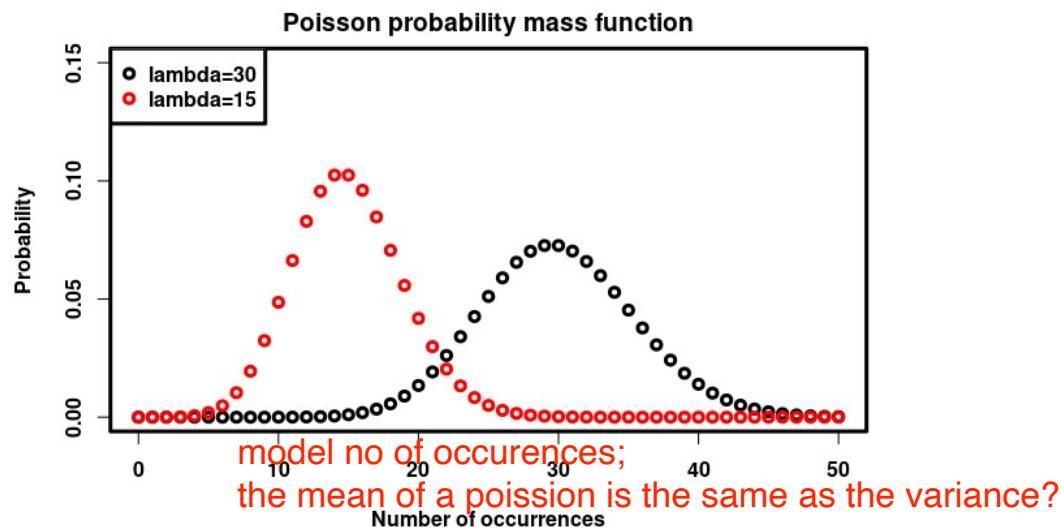
$\lambda \in (0, \infty)$: the Expected value (rate)

$k \in N_0$

$$PMF = \frac{\lambda^k e^{-\lambda}}{k!}$$

one parameter: anything from 0-infinite (in integers).

k = natural numbers (including 0); means how many fish we see (no half fishes).



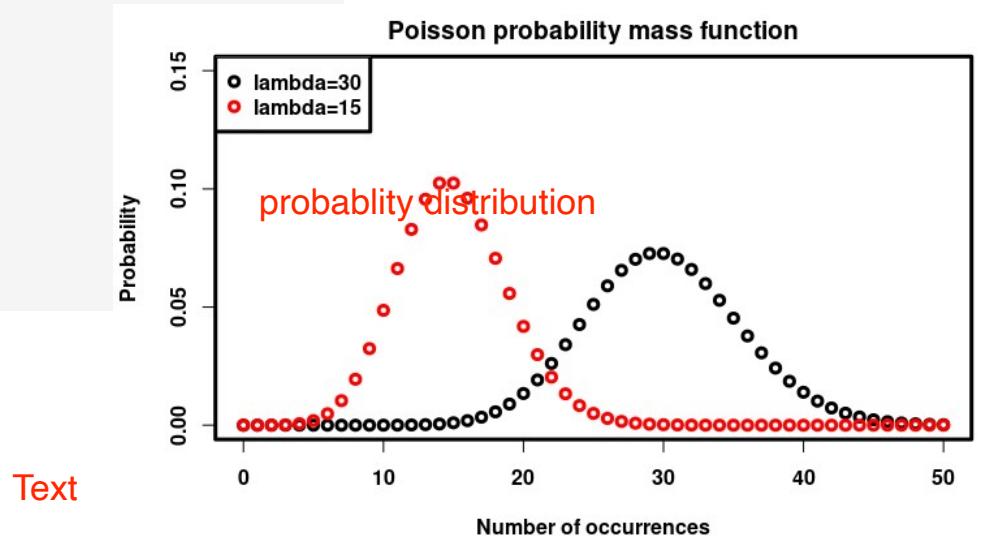
4) A data distribution: $p(y|\hat{y})$

```
x <- 0:50 # counts, thus integers
lambda <- 30
pmf <- dpois(x, lambda)

par(font.lab=2, font.axis=2, cex=1, lwd=3)
plot(x, pmf, xlab='Number of occurrences', ylab='Probability',
     main='Poisson probability mass function', ylim=c(0, 0.15))

lambda <- 15
pmf <- dpois(x, lambda)
points(x, pmf, col='red')

legend('topleft',
       legend=c(expression(paste(lambda, '=' , 30)),
               expression(paste(lambda, '=' , 15))),
       pch=1, col=c('black', 'red'), text.font=2)
```



... the identity link

Text

```
g <- identity  
inv.g <- identity ## identity-1 = identity
```

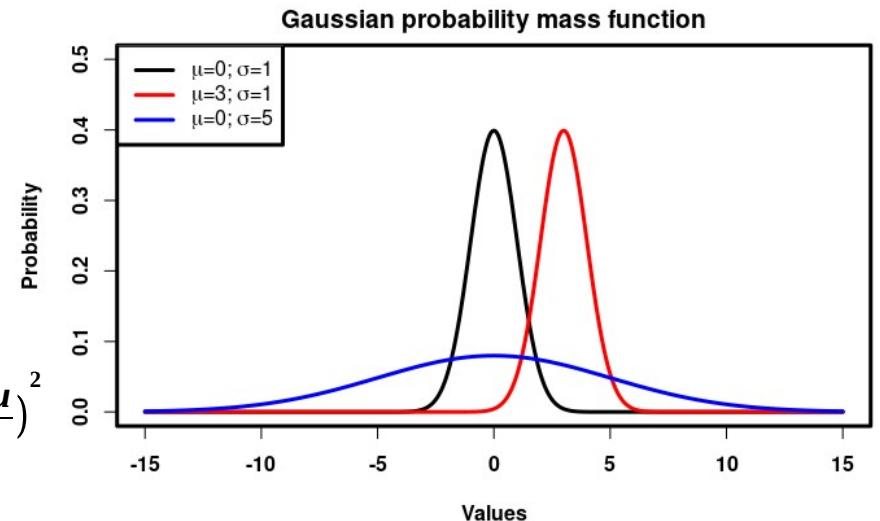
$$N(\mu, \sigma^2)$$

$$\mu \in R$$

$$\sigma^2 \in R_{>0}$$

$$x \in R$$

$$\text{PDF} = (2\pi\sigma^2)^{-1/2} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



Nomenclature

- General Linear Model (single level modelling)
 - $y = X\beta + \epsilon$
- Generalized Linear Model (single level modelling)
 - $y = X\beta + \epsilon$ – but now with link functions and other data distributions besides the Gaussian one
- General Linear Mixed Model (multilevel modelling)
 - $y = X\beta + Zu + \epsilon$
- Generalized Linear Mixed Model (multilevel modelling)
 - $y = X\beta + Zu + \epsilon$ – but now with link functions and other data distributions besides the Gaussian one

Specifying random (second-level) effects

```
(1 | subject) # intercept only  
(slope | subject) # slope and intercept  
Text  
(1 + slope | subject) # slope and intercept (explicit)  
(0 + slope | subject) # slope only  
(1 + slope || subject) # slope and intercept but force covariance to 0
```

The general linear mixed model (GLMM)

$$y = X\beta + Zu + \epsilon$$

y : $N \times 1$ column vector

X : $N \times p$ matrix of p predictor variables

β : unknown $p \times 1$ column vector of the first level regression coefficients

Z : $N \times q$ design matrix for the q random effects

u : unknown $q \times 1$ column vector of the second-level effects

ϵ : $N \times 1$ column vector of the residuals

Sleep study

partial pooling

```
model <- lmer(Reaction ~ days_deprived + (days_deprived | Subject), data=sleepstudy)
```

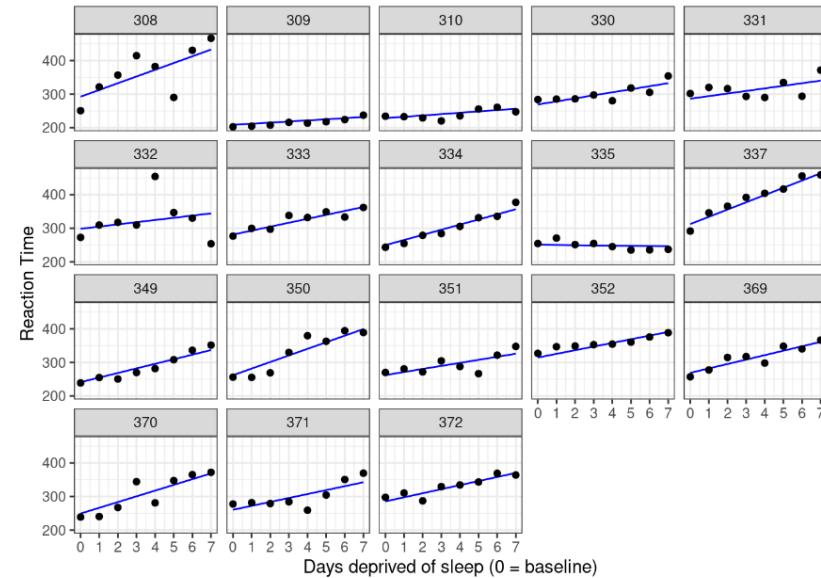


Figure 5.6: Data plotted against predictions from a partial pooling approach.

$$y = X\beta + Zu + \epsilon$$

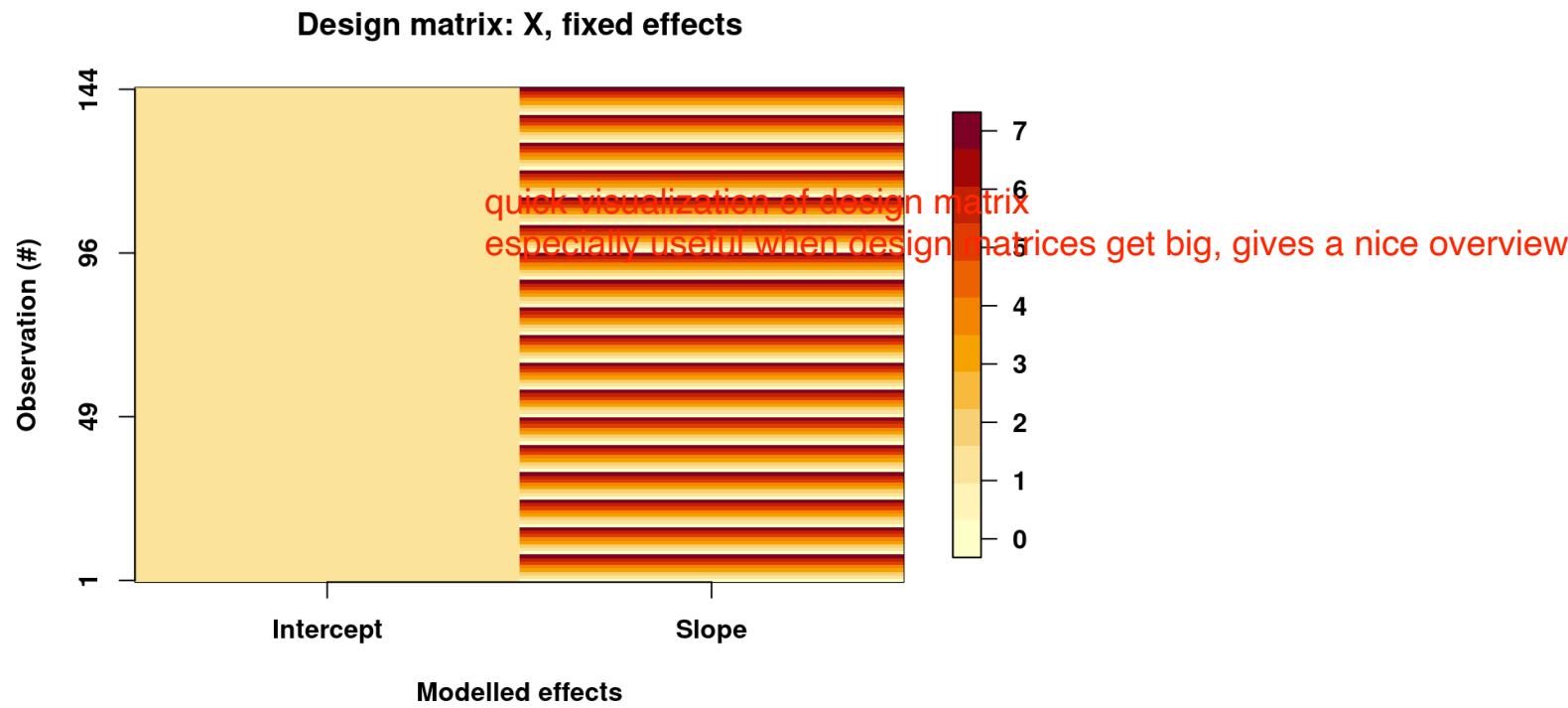
observations

```
head(sleepstudy$Reaction, 30)
```

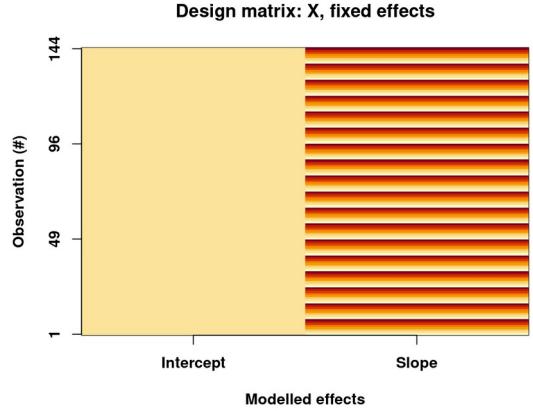
```
## [1] 250.8006 321.4398 356.8519 414.6901 382.2038 290.1486 430.5853 46  
6.3535  
## [9] 202.9778 204.7070 207.7161 215.9618 213.6303 217.7272 224.2957 23  
7.3142  
## [17] 234.3200 232.8416 229.3074 220.4579 235.4208 255.7511 261.0125 24  
7.5153  
## [25] 283.8565 285.1330 285.7973 297.5855 280.2396 318.2613
```

y =

$$y = \boxed{X} \beta + Z u + \epsilon$$



$$y = X \beta + Z u + \epsilon$$



$$\hat{\beta} =$$

```
fixef(model)
```

```
##   (Intercept) days_deprived
##      267.96742      11.43543
```

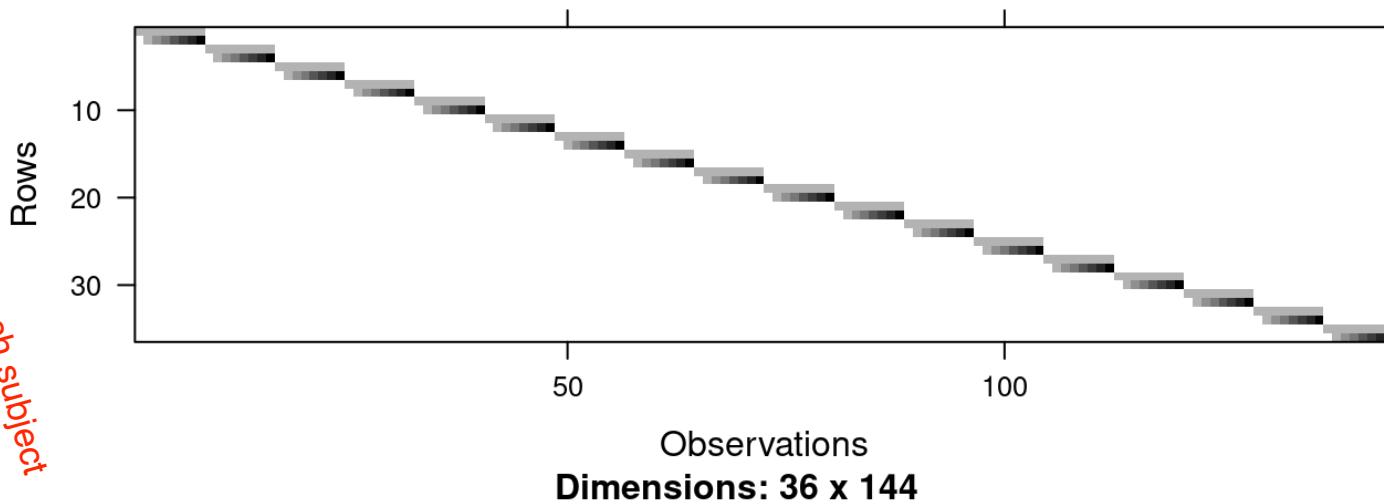
```
Xt <- t(getME(model, 'X'))
```

apply to model, and get information about model.

$$y = X\beta + \boxed{Zu} + \epsilon$$

modelling two effects for each subject

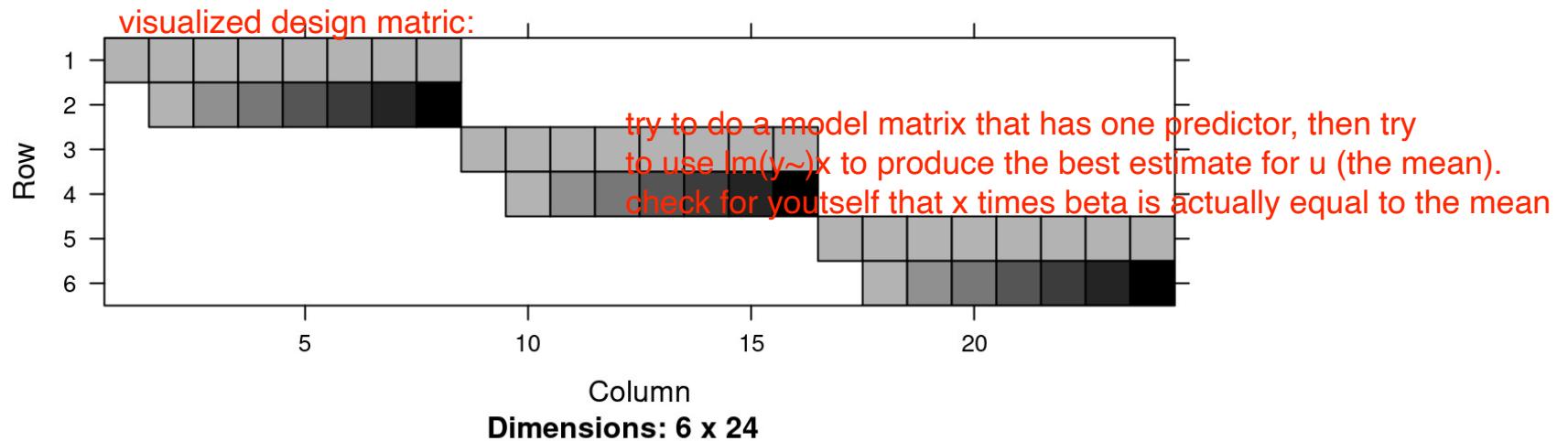
Design matrix: Z, random effects



```
Zt <- getME(model, 'Zt')
```

$$y = X\beta + Zu + \epsilon$$

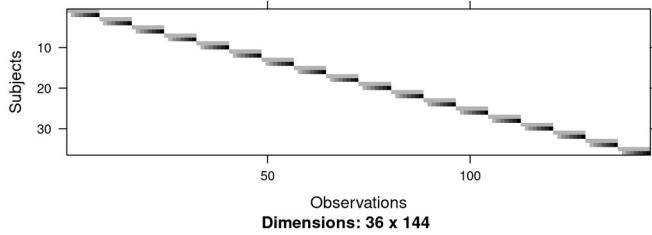
First 3 subjects



$$y = X\beta + Zu + \epsilon$$



Design matrix: Z, random effects



$$\hat{u} =$$

ranef(model)

```
## $Subject
##   (Intercept) days_deprived
## 308  24.4991541   8.6020120
## 309 -10.07875    8.1280623
## 310 -39.4754543  -7.4293973
## 330  1.3497667  -2.3845341
## 331  18.4565538  -3.7474999
## 332  30.5253684  -4.8933314
## 333  13.3677150   0.2889674
## 334 -18.1572414   3.8434349
## 335 -16.9742694 -12.0701072
## 337  44.5843235  10.1762260
## 349 -26.6826682   2.1944025
## 350 -5.9647975   8.1756325
## 351 -5.5710438  -2.3718429
## 352  46.6328765  -0.5612423
## 369   0.9617659   1.7384827
## 370 -18.5204352   5.6314774
## 371 -7.3428235   0.2728619
## 372  17.6819971   0.6625203
```

u hat = random effects (comes out of ranef())

$$y = X\beta + Zu + \epsilon$$

Text

```
head(round(residuals(model), 2), 6)
```

$\epsilon =$

##	1	2	3	4	5	6
##	-41.67	8.94	24.31	62.11	9.59	-102.51

Bonus question:

Why do the random effects have a mean of 0?

The mean of all the random effects is 0, because if they were not, the grand mean would be different.

The random effects are just extra information, that is still distributed around the model mean. They can never differ from the mean.

Did you learn?

Generalized Linear Mixed Effects Models (GLMM)

- 1) Understanding that we can extend the scope of our multilevel modelling by using appropriate link functions and data distributions
- 2) Understanding the multilevel equivalent of the GLM

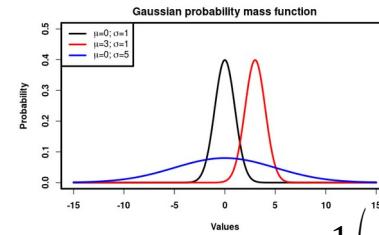
Learning goals (will not finish this today)

Explanation and prediction

- 1) Understanding that fitting (explaining) often leads to overfitting
- 2) Learning methods to prevent overfitting, e.g. by introducing *bias* and using *cross-validation*
- 3) Understanding how the error can be decomposed into *bias* and *variance*

Maximum likelihood estimation

$$\text{PDF} = (2\pi\sigma^2)^{-1/2} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$



$$\text{Conditional PDF } (y_i|X) = (2\pi\sigma^2)^{-1/2} e^{-\frac{1}{2}\left(\frac{y_i-x_i\beta}{\sigma_0}\right)^2}$$

$$\text{Likelihood function : } L(\sigma^2, \epsilon) = (2\pi\sigma^2)^{-N/2} e^{-\left(\frac{1}{2\sigma^2}\sum_{i=1}^N (y_i-x_i\beta)^2\right)}$$

Log-Likelihood function : $\log(L)$

y : dependent variable

$X\beta$: linear predictor

$y - X\beta = \epsilon$: residuals

N : number of observations

We have so far been fitting
explanatory models, i.e. by
maximising log-likelihood

```
model <- lmer(Reaction ~ days_deprived + (days_deprived | Subject),  
             data=sleepstudy)  
model.ranint <- lmer(Reaction ~ days_deprived + (1 | Subject),  
                      data=sleepstudy)
```

```
print(ll.m <- logLik(model))
```

```
## 'log Lik.' -702.0472 (df=6)
```

```
exp(ll.m)
```

```
## 'log Lik.' 1.272865e-305 (df=6)
```

```
print(ll.mr <- logLik(model.ranint))
```

```
## 'log Lik.' -715.01 (df=4)
```

```
exp(ll.mr)
```

```
## 'log Lik.' 2.986092e-311 (df=4)
```

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

maximises the likelihood of

$$Y = X \beta + \epsilon$$

for which link function?

To fit is to overfit

(Yarkoni and Westfall, 2017)

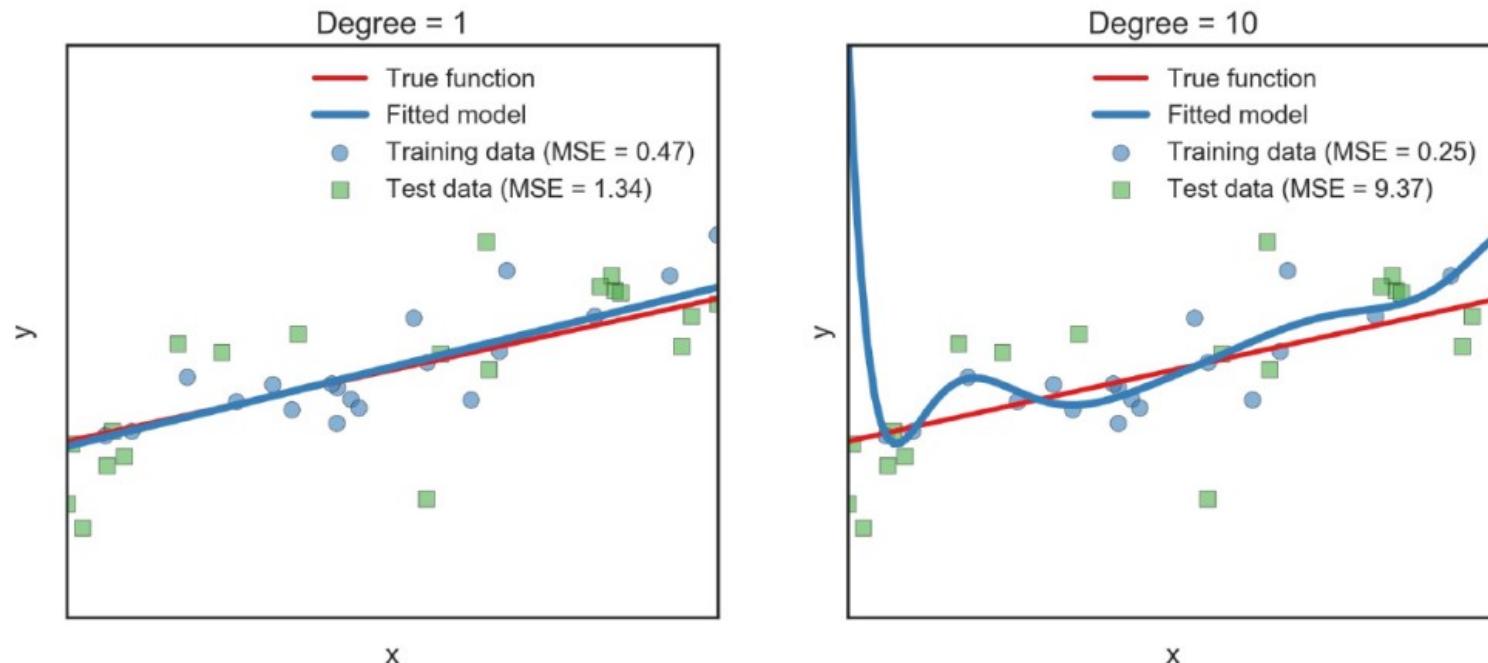
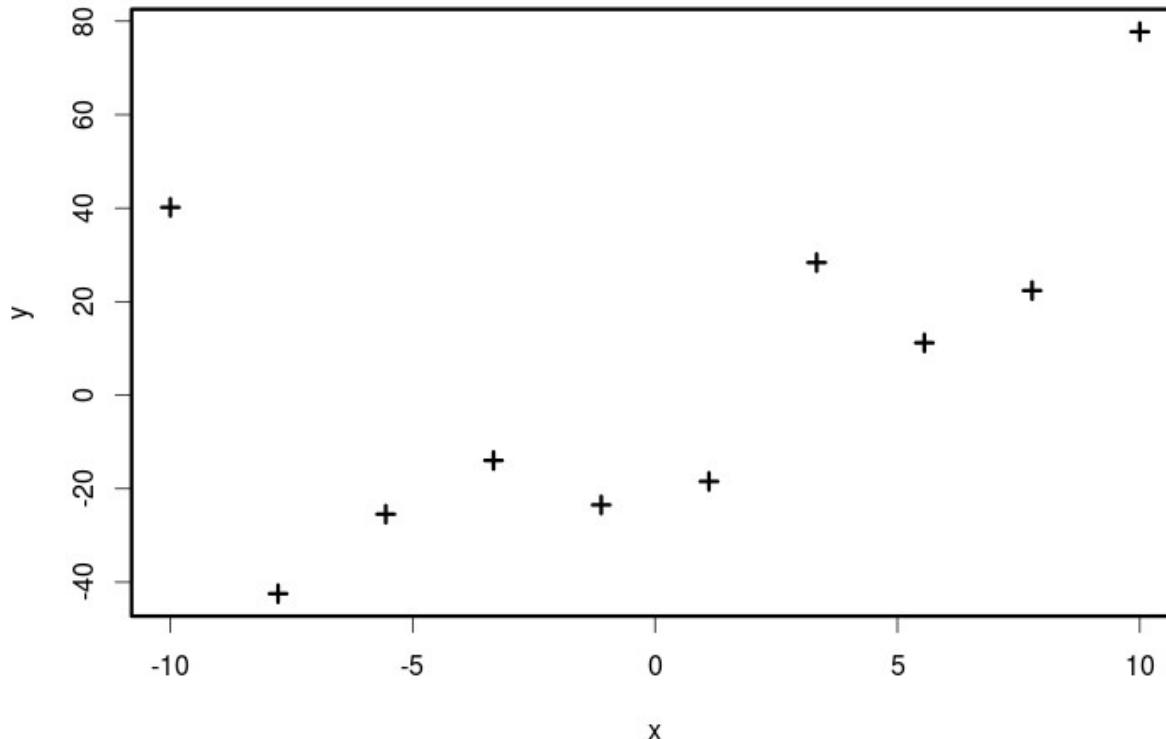


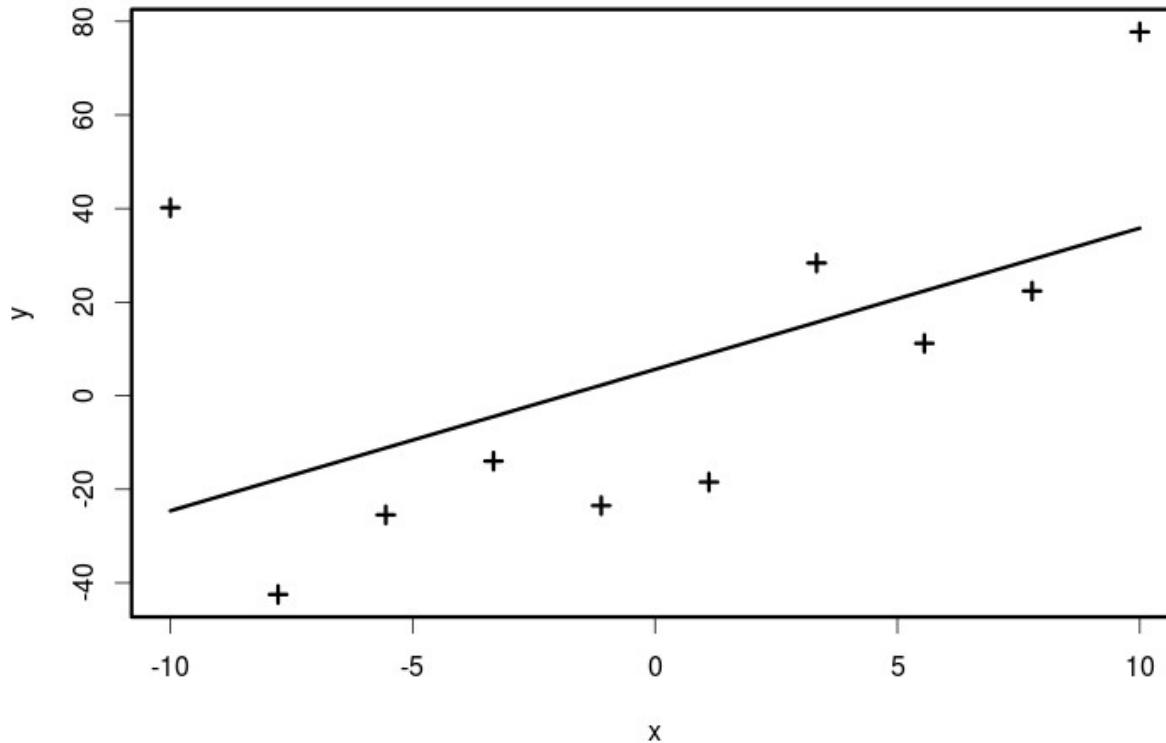
Fig. 1. Training and test error produced by fitting either a linear regression (left) or a 10th-order polynomial regression (right) when the true relationship in the population (red line) is linear. In both cases, the test data (green) deviate more from the model's predictions (blue line) than the training data (blue). However, the flexibility of the 10th-order polynomial model facilitates much greater overfitting, resulting in lower training error but much higher test error than the linear model. MSE = mean squared error.

(Yarkoni and Westfall, 2017)

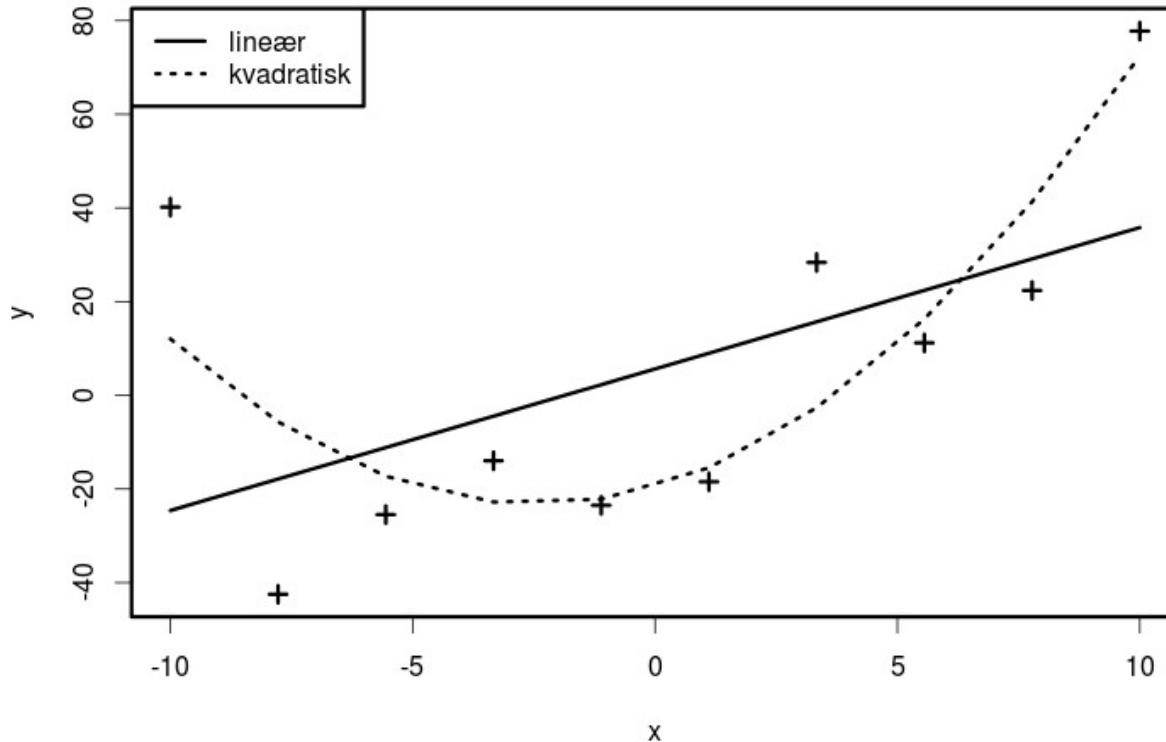
A sample of 10 linear or quadratic?



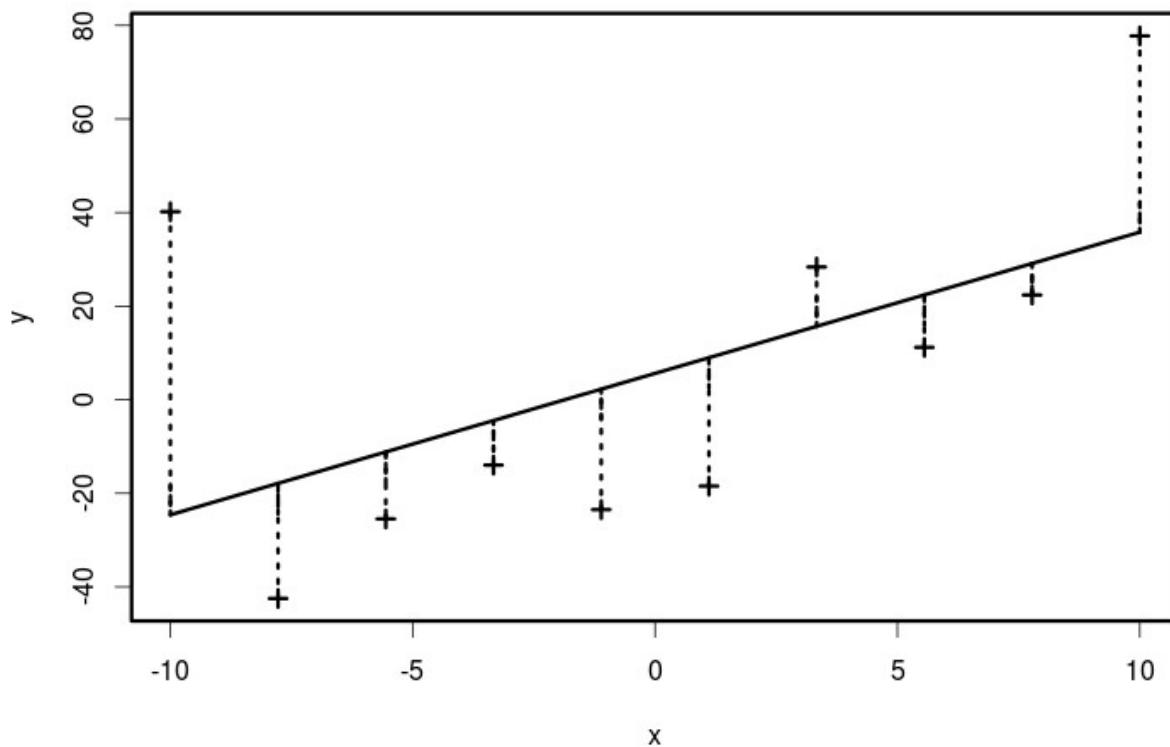
A sample of 10 linear or quadratic?



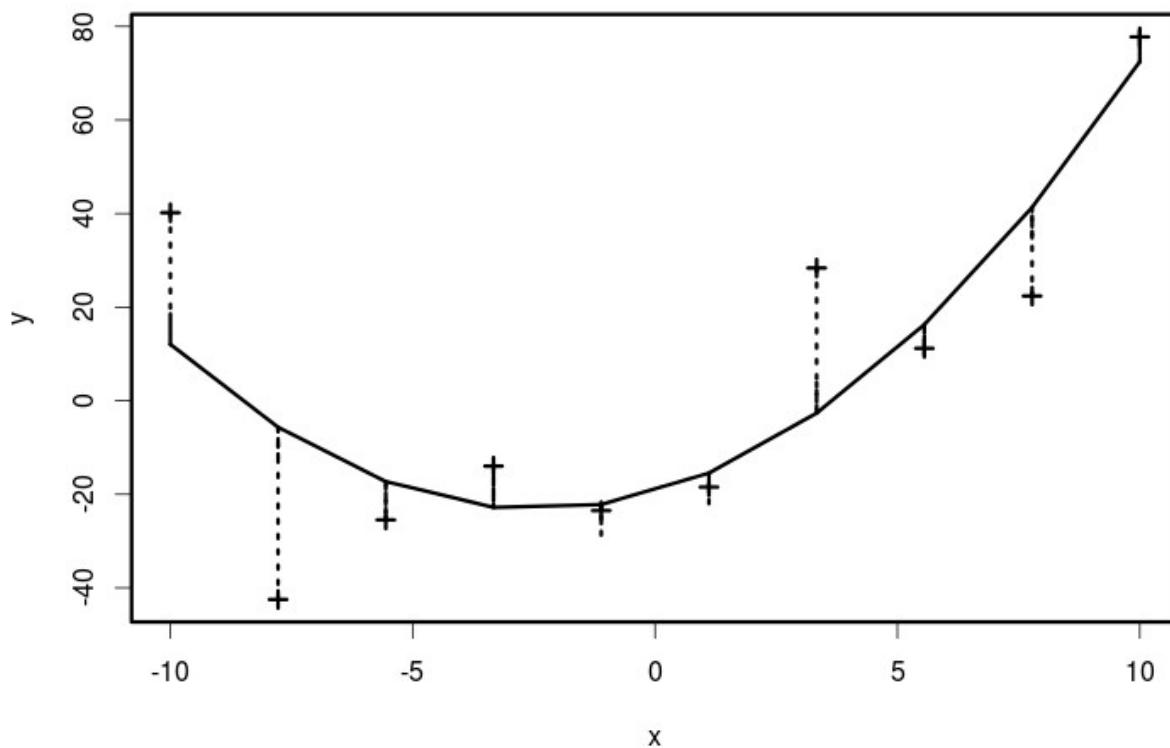
A sample of 10 linear or quadratic?



Residuals (linear)



Residuals (quadratic)



Quadratic:

$$ax^2 + bx + c$$

Linear:

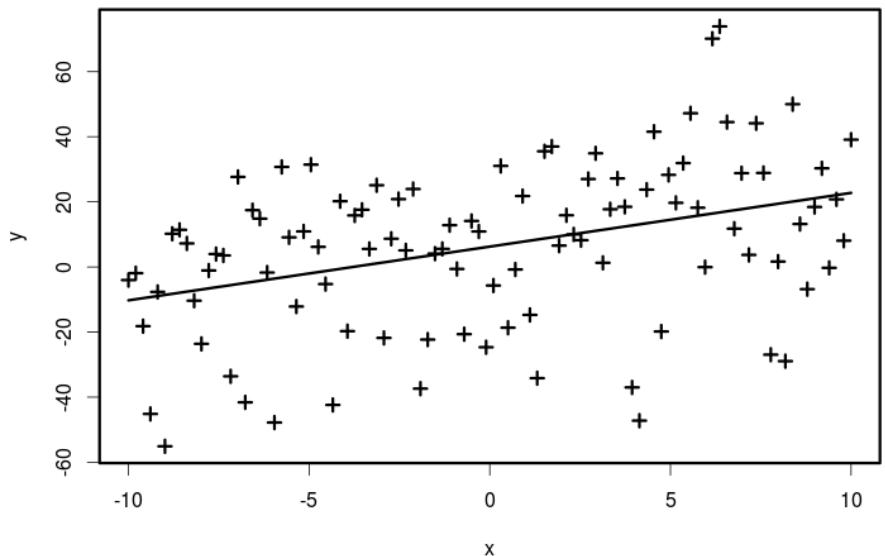
$$bx + c$$

Estimates

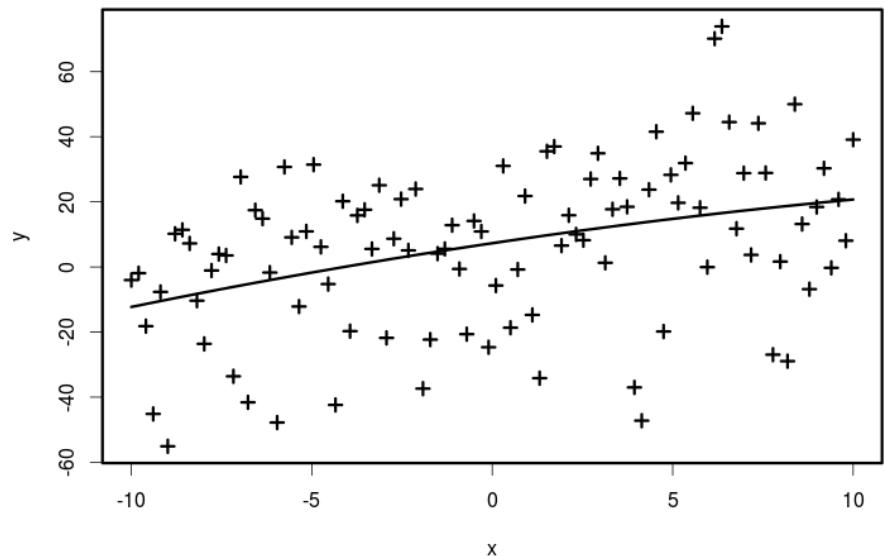
$$a = 0,6184; b = 3,0201$$

$$b = 3,020$$

Now a sample of 100



$$b = 1,650$$



$$a = -0,03074 \approx 0; b = 1,650$$

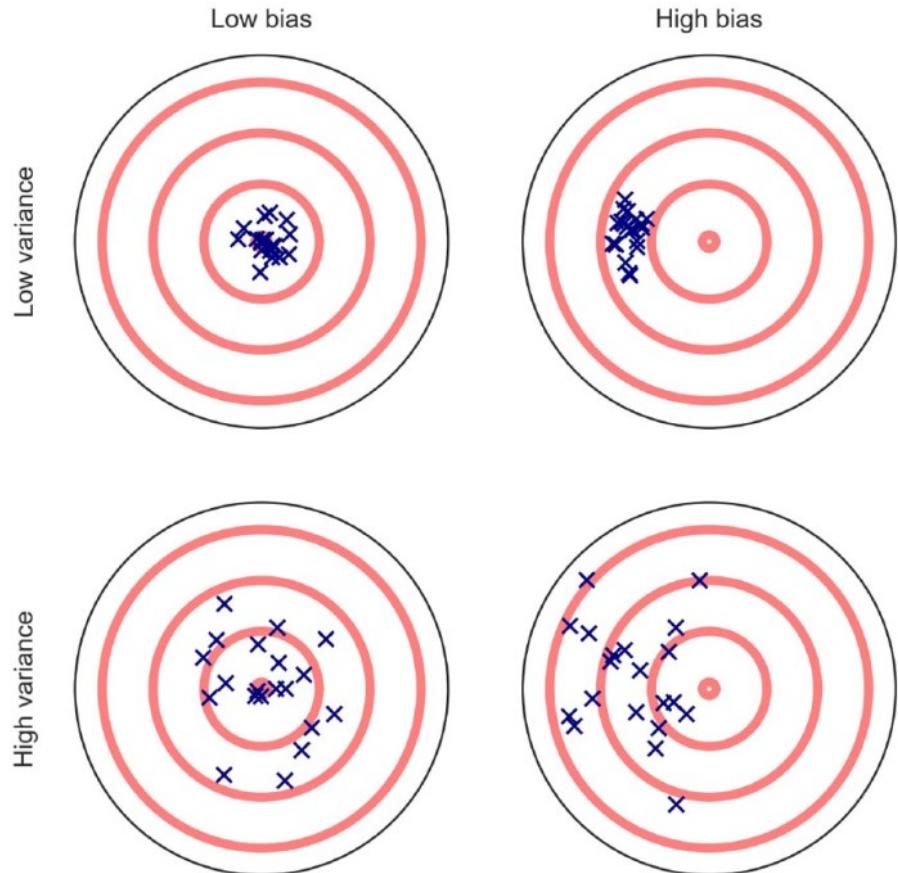


Fig. 2. An estimator's predictions can deviate from the desired outcome (or true scores) in two ways. First, the predictions may display a systematic tendency (or *bias*) to deviate from the central tendency of the true scores (compare right panels with left panels). Second, the predictions may show a high degree of *variance*, or imprecision (compare bottom panels with top panels).

(Yarkoni and Westfall, 2017)

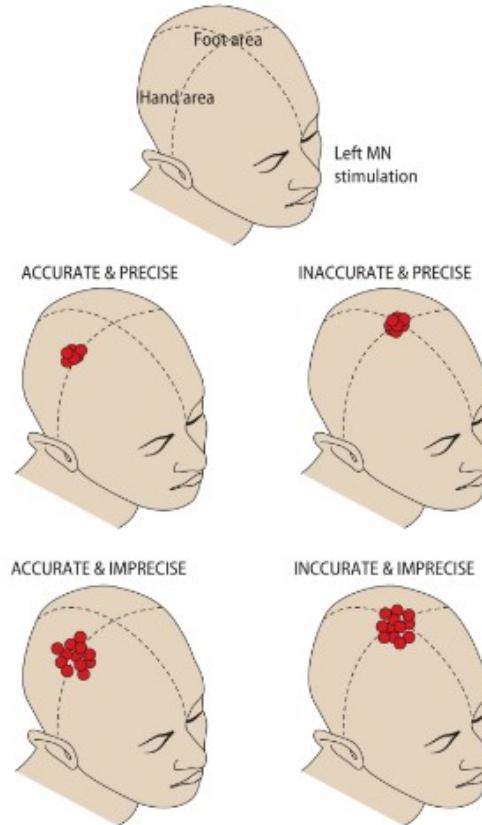


FIGURE 3.7. Accuracy versus precision. A schematic illustration of the differences between accuracy and precision of source localization. After left median-nerve stimulation, activations is expected in the right-hemisphere hand region of the primary somatosensory cortex. The foot area is shown at the top of the head. See text for further explanation.

(Hari and Puce, 2017)

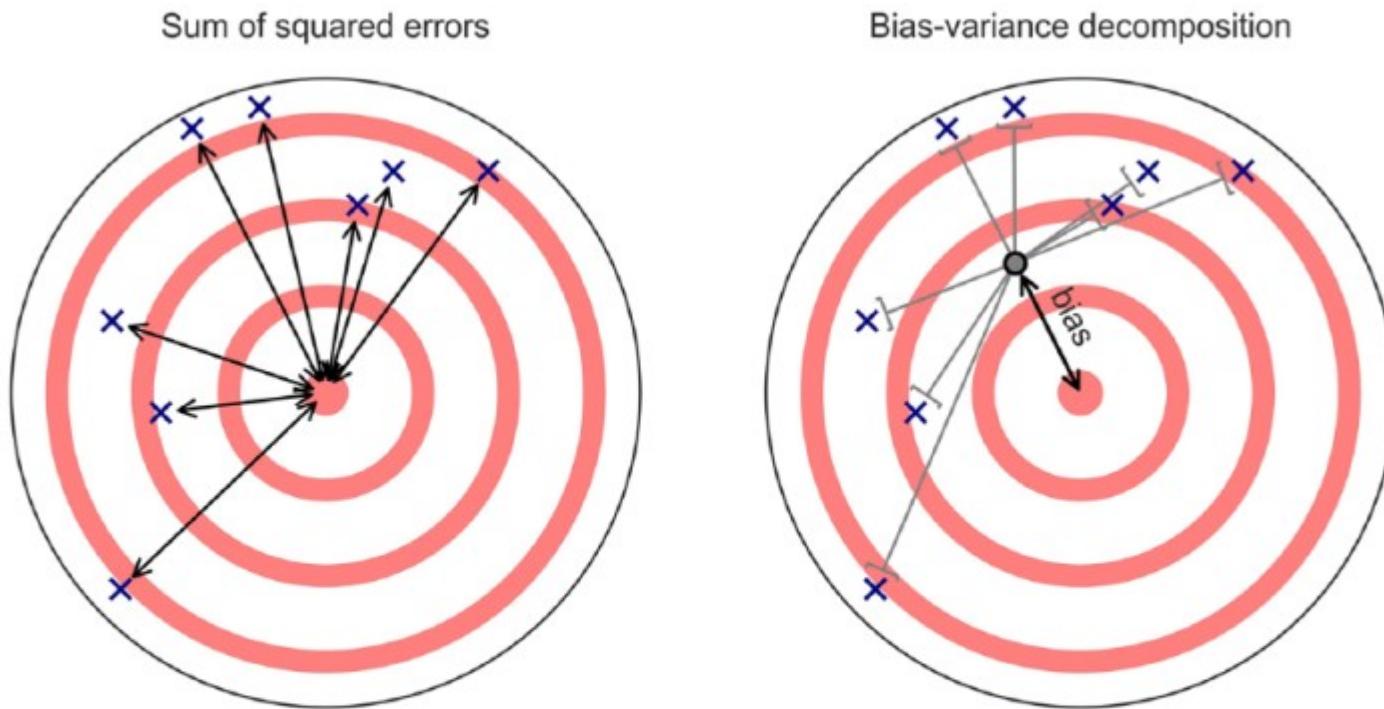


Fig. 3. Schematic illustration of the bias-variance decomposition. (Left) Under the classical error model, prediction error is defined as the sum of squared differences between true scores and observed scores (black lines). (Right) The bias-variance decomposition partitions the total sum of squared errors into two separate components: a bias term that captures a model's systematic tendency to deviate from the true scores in a predictable way (black line) and a variance term that represents the deviations of the individual observations from the model's expected prediction (gray lines).

Multilevel modelling as a *bias introducer*

“For example, some readers may be surprised to learn that multilevel modeling approaches to analyzing clustered data—which have recently seen a dramatic increase in adoption in psychology—improve on ordinary least squares (OLS) approaches to estimating individual cluster effects by deliberately biasing (through “shrinking” or “pooling”) the cluster estimates toward the estimated population average”

(Yarkoni and Westfall, 2017)

Introducing *bias*

“In a widely used form of penalized regression called lasso regression (Tibshirani, 1996, 2011), this leastsquares criterion is retained, but the overall cost function that the estimation seeks to minimize now includes an additional penalty term that is proportional to the sum of the absolute values of the coefficients.”

(Yarkoni and Westfall, 2017)

The benefit of large samples

(Yarkoni and Westfall, 2017)

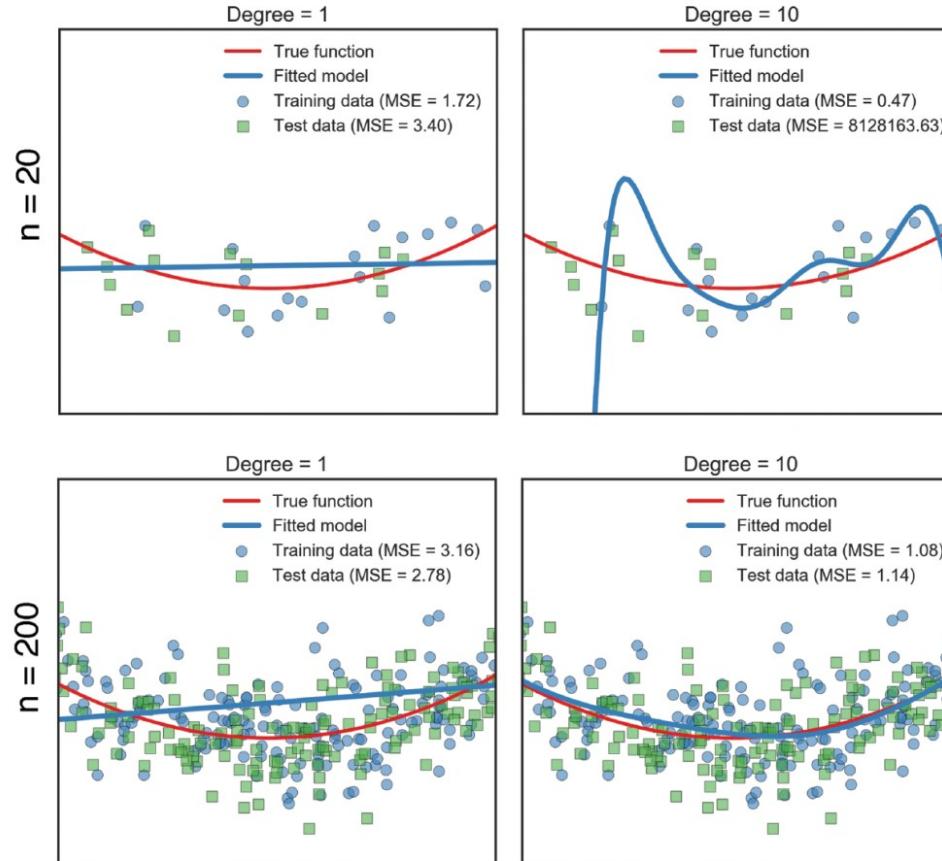


Fig. 4. Large samples guard against overfitting. See text for explanation.

References

- Gelman, A., Hill, J., 2006. Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press.
- Hari, R., Puce, A., 2017. MEG-EEG Primer. Oxford University Press, New York, NY, US.
- Yarkoni, T., Westfall, J., 2017. Choosing Prediction Over Explanation in Psychology: Lessons From Machine Learning. *Perspect Psychol Sci* 12, 1100–1122.
<https://doi.org/10.1177/1745691617693393>