Methods 3: Multilevel Statistical Modeling and Machine Learning

Week 03: *Link functions and fitting generalised linear multilevel models*
September 17, 2024

The course plan

Week 1: *Introduction*

Instructor sessions: *Setting up* R *and* Python *and recollection of the general linear model*

Week 2: *Multilevel linear regression*

Instructor sessions: *Modelling subject level effects – and how do they differ from group level effects?*

Week 3: *Link functions and fitting generalised linear multilevel models*

Instructor sessions: *What to do when the response variable is not continuous?*

Week 4: *Evaluating Generalised linear mixed models*

Instructor sessions: *How do we assess how models compare to one another?*

Week 5: *Explanation and Prediction*

Instructor sessions: *Code review*

Week 6: *Mid-way evaluation and Machine Learning Intro*

Instructor sessions: *Getting Python Running*

Week 7: *Linear regression revisited (machine learning)*

Instructor sessions: *How to constrain our models to make them more predictive*

Week 8: *Logistic regression revisited (machine learning)*

Instructor sessions: *Categorizing responses based on informed guesses*

Week 9: *Dimensionality Reduction, Principled Component Analysis (PCA)*

Instructor sessions: *What to do with very rich data?*

Week 10: *Outlook, unsupervised classification and neural networks*

Instructor sessions: *Data with no labels and networks*

Week 11: *Organising and preprocessing messy data*

Instructor sessions: *Code review*

Week 12: Final evaluation and wrap-up of course

Instructor sessions: *Ask anything!*

Recap – week 02

- Design matrices
  - reflect the choices we make in modelling
  - sometimes, our choices cannot be fit
  - the variance-covariance matrix will sometimes reveal why
- Partial pooling
  - allows for subject weighting or average weighting on a case-by-case basis, depending on the reliability of information available

Learning goals and outline

Link functions and fitting generalised linear multilevel models

1) Understanding that we can extend the scope of our general linear model by using appropriate link functions and data distributions
2) Understanding that we can extend this scope to multilevel modelling as well
3) Getting an understanding of maximum likelihood estimation

Extending the scope of the general linear model by using appropriate link functions and data distributions

With a focus on logistic regression (*Booleans*) and Poisson regression (*counts*)

Generalised linear model

At least four ingredients needed

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

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

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

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

Gelman A, Hill J (2006) Data Analysis Using Regression and Multilevel/Hierarchical Models. Cambridge University Press: Chapter 6

Logistic regression

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

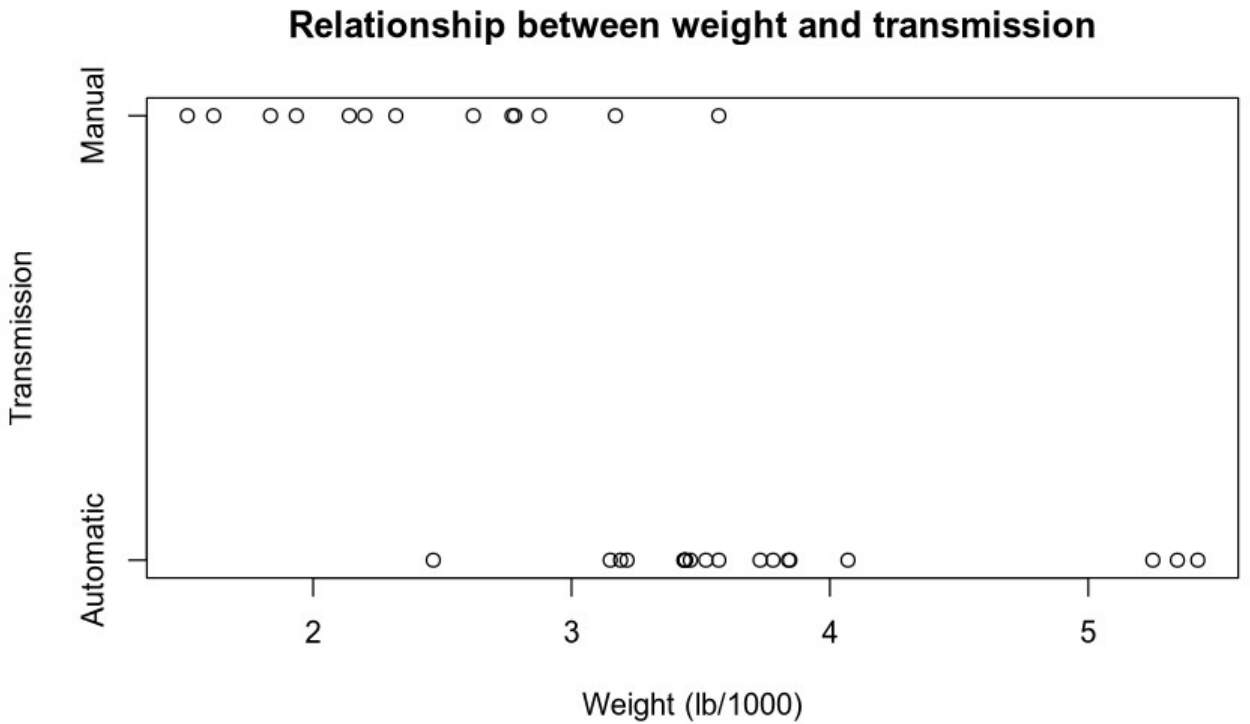
```
data('mtcars')
print(y <- mtcars$am)
```

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

```
print(n <- length(y))
```

```
## [1] 32
```

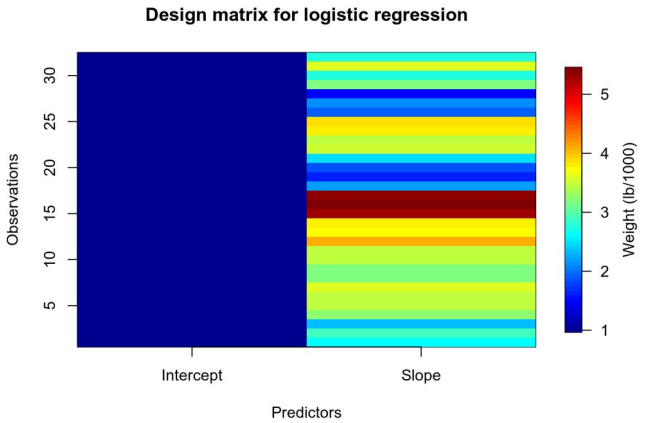
1) A data vector : $y = (y_1, \ldots, y_n)$



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

```
X <- matrix(nrow=n, ncol=2) ## choosing two predictors
X[, 1] <- 1 ## modelling an intercept
X[, 2] <- mtcars$wt ## modelling a linear predictor based on weight
library(fields)
image.plot(x=1:dim(X)[2], y=1:dim(X)[1], z=t(X),
           xlab='Predictors', ylab='Observations', xaxt='n',
           main='Design matrix for logistic regression',
           legend.lab='Weight (lb/1000)')
axis(1, 1:dim(X)[2], c('Intercept', 'Slope'))
```

$X =$

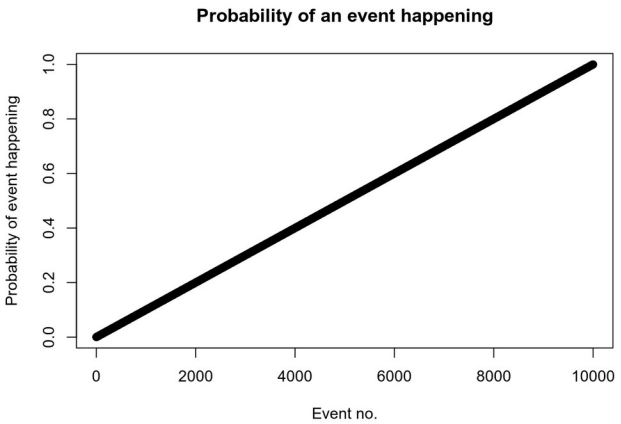


$\beta =$ to be estimated

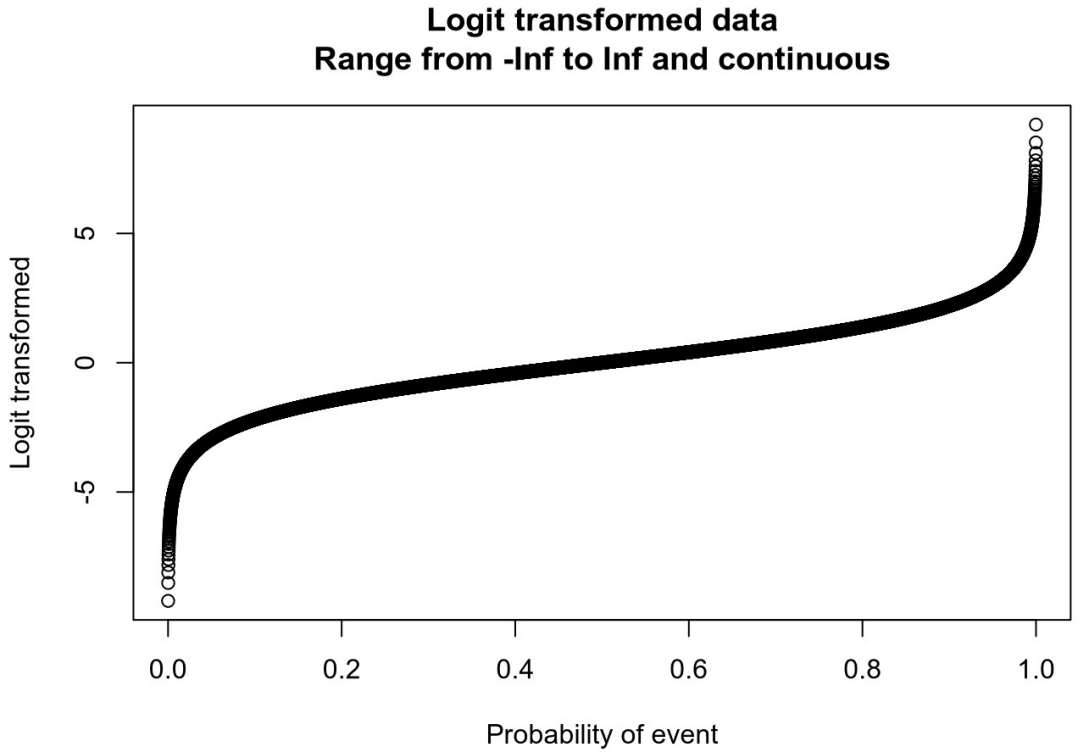
3) A link function g: yielding a vector of transformed data $\hat{y} = g^{-1}(X\beta)$ that are used to model the data

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

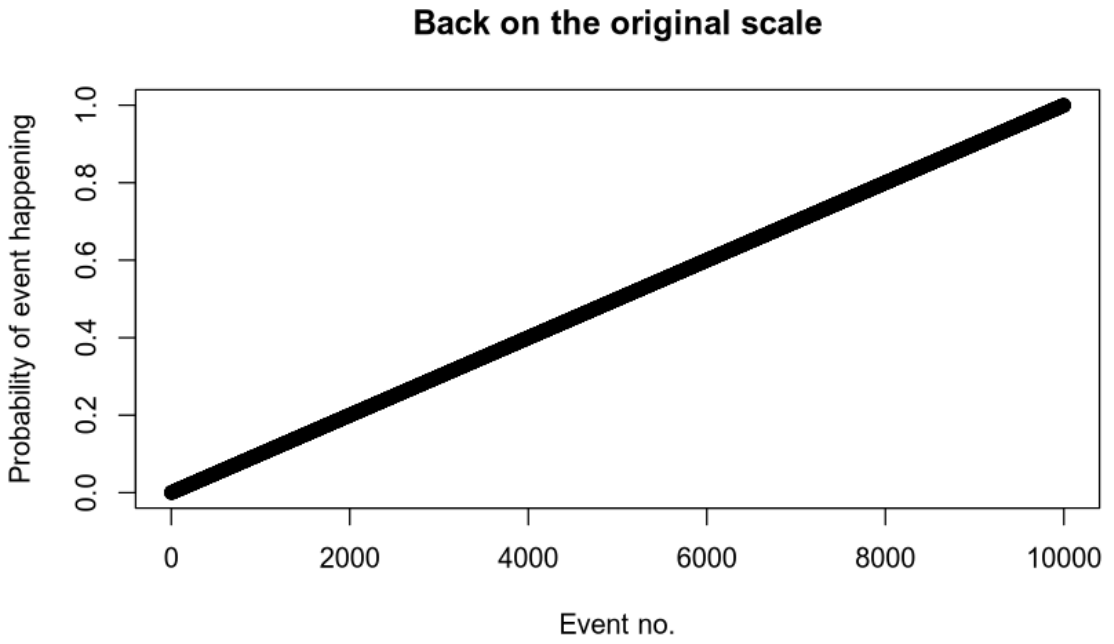
```r
y <- seq(0.0001, 0.9999, 0.0001)
x <- 1:length(y)
plot(x, y, main='Probability of an event happening', xlab='Event no.',
     ylab='Probability of event happening')
```



```
plot(y, g(y), xlab='Probability of event', ylab='Logit transformed',
     main='Logit transformed data\nRange from -Inf to Inf and continuous')
```



```
plot(x, inv.g(g(y)), main='Back on the original scale',
     xlab='Event no.', ylab='Probability of event happening')
```



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

$$\text{PMF}_{\text{Bernouilli}} = p^k(1-p)^{1-k}$$

$$p \in [0,1]; k \in \{0,1\}$$

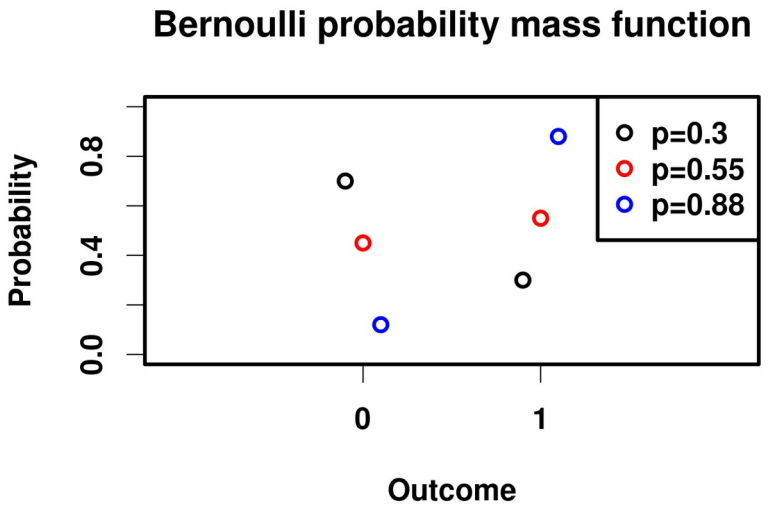
```r
dbernoul <- function(p, k) p^k * (1 - p)^(1 - k)

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)
```



Bringing it all together

```
Call:  glm(formula = am ~ wt, family = "binomial", data = mtcars)
Coefficients:
(Intercept)           wt
     12.040       -4.024
Degrees of Freedom: 31 Total (i.e. Null);  30 Residual
Null Deviance:      43.23
Residual Deviance: 19.18        AIC: 23.18
```

$\hat{\beta}$

```r
log.reg        <- glm(am ~ wt, data=mtcars, family='binomial')
y              <- log.reg$y
X              <- model.matrix(log.reg)
beta.hat       <- coef(log.reg)
lin.pred       <- X %*% beta.hat
y.hat          <- inv.g(lin.pred) # identical to fitted(log.reg)
likelihoods    <- dbinom(y, size=1, prob=y.hat) # bernoulli
likelihood     <- prod(likelihoods)
log.likelihood <- log(likelihood) ## check for yourself with logLik(log.reg)
res.deviance   <- -2 * log.likelihood # null deviance is logLik(glm(am ~ 1))
n.predictors   <- length(beta.hat)
AIC            <- 2 * n.predictors + res.deviance
print(log.reg)
```

live coding

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

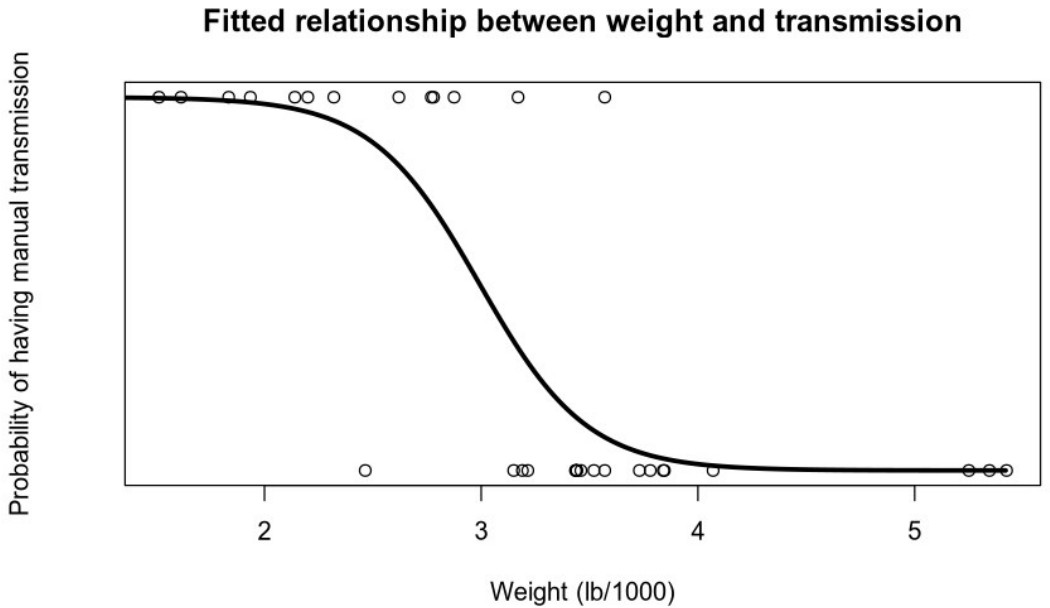
$p(y|\hat{y})$: the likelihood of observing y when our fitted value is $\hat{y}$

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

$p(y|\hat{y}) = \hat{y}^{y}(1-\hat{y})^{1-y}$: Bernoulli distribution

$y \in \{0, 1\}; \hat{y} \in (0, 1)$

```r
x.pred <- seq(0, max(mtcars$wt), 0.01)
y.pred <- inv.g(coef(log.reg)[1] + coef(log.reg)[2] * x.pred)
plot(am ~ wt, data=mtcars, ylab='Probability of having manual transmission'
     , xlab='Weight (lb/1000)',
     yaxt='n', main='Fitted relationship between weight and transmission',
     ylim=c(-0.0, 1.0))
lines(x.pred, y.pred, lwd=3)
```



Why we need a data distribution

WE NEED A LIKELIHOOD TO MAXIMISE

$$L(\hat{\beta}|y) = \prod_{i=1}^{n} \hat{y}_i^{y_i} (1 - \hat{y}_i)^{(1-y_i)}$$

$$l(\hat{\beta}|y) = \sum_{i=1}^{n} y_i \log(\hat{y}_i) + \log(1 - y_i)(1 - \hat{y}_i)$$

Why is the log-likelihood preferred?

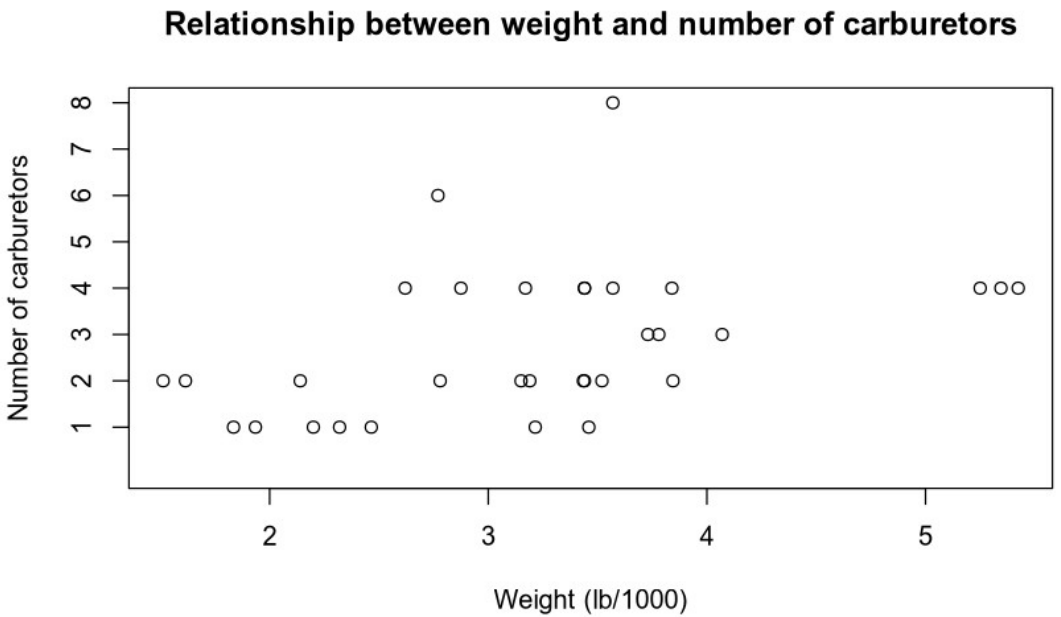
Poisson regression

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

```
data('mtcars')
print(y <- mtcars$carb)
```

```
##  [1] 4 4 1 1 2 1 4 2 2 4 4 3 3 3 4 4 4 1 2 1 1 2 2 4 2 1 2 2 4 6 8 2
```

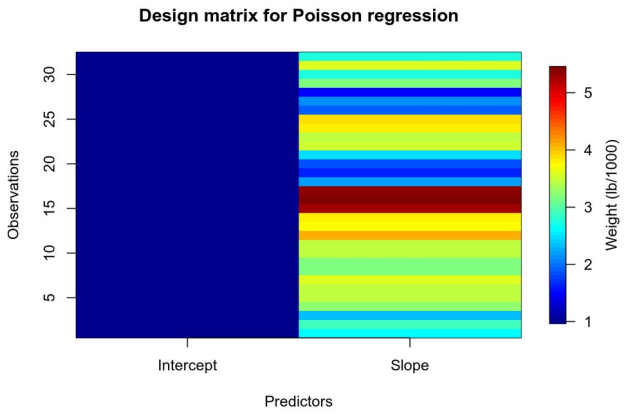
```
print(n <- length(y))
```



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

```
X <- matrix(nrow=n, ncol=2) ## choosing two predictors
X[, 1] <- 1 ## modelling an intercept
X[, 2] <- mtcars$wt ## modelling a linear predictor based on weight
library(fields)
image.plot(x=1:dim(X)[2], y=1:dim(X)[1], z=t(X),
           xlab='Predictors', ylab='Observations', xaxt='n',
           main='Design matrix for Poisson regression',
           legend.lab='Weight (lb/1000)')
axis(1, 1:dim(X)[2], c('Intercept', 'Slope'))
```

$X =$

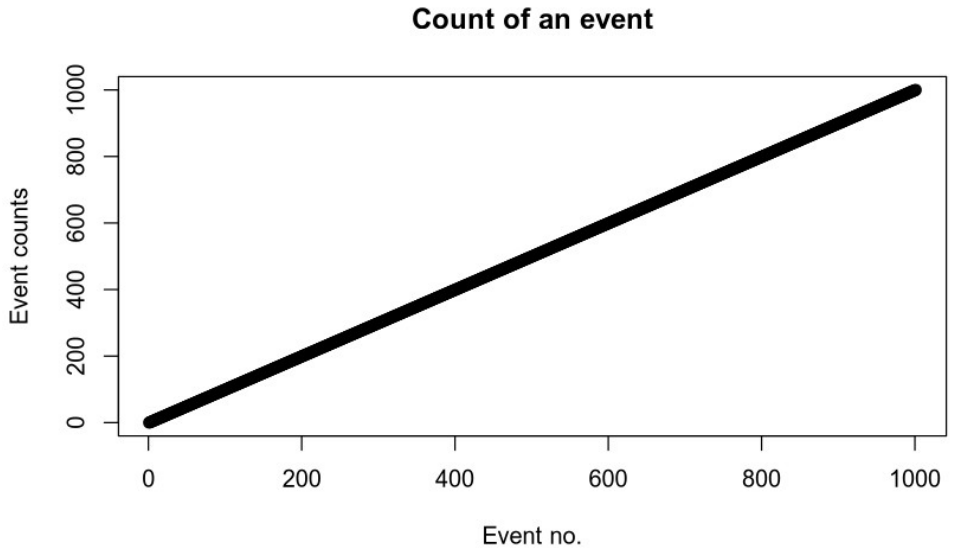


$\beta =$ to be estimated

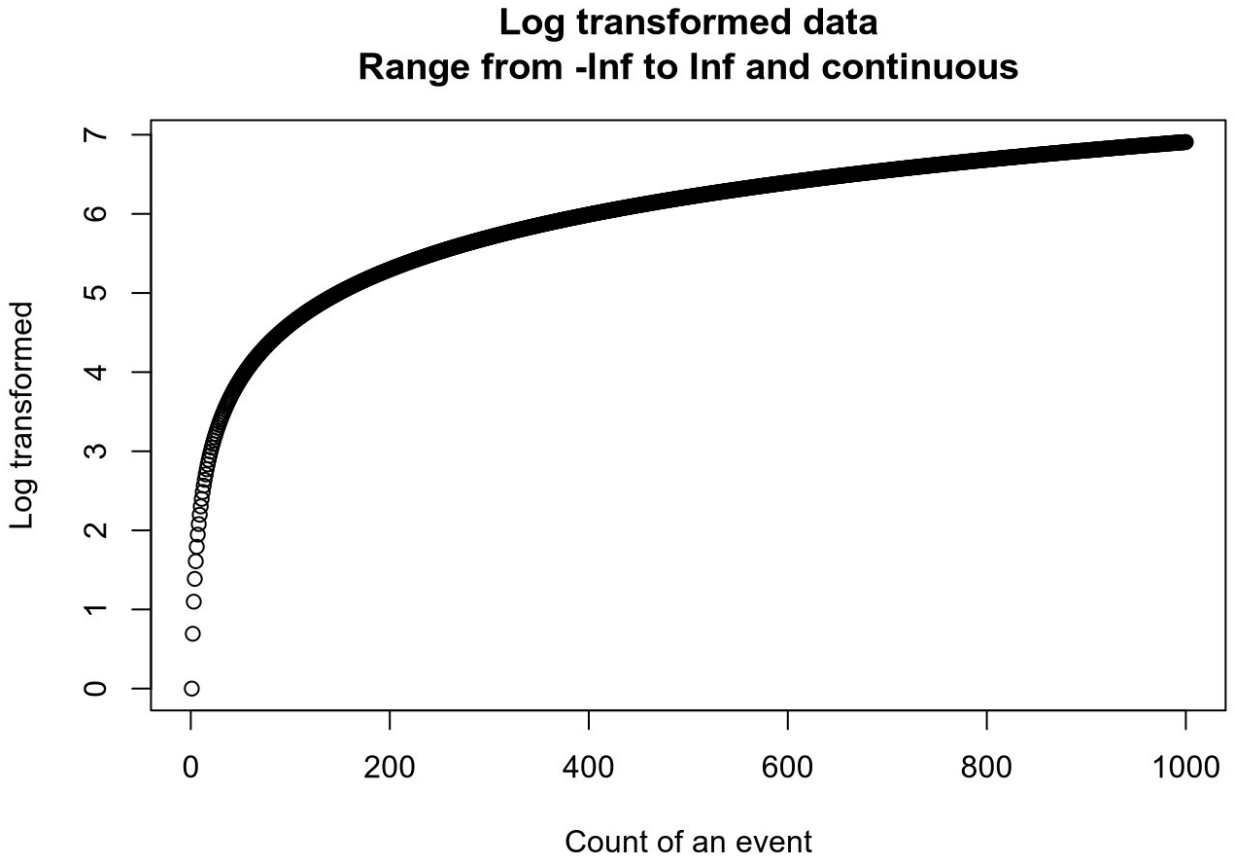
3) A link function g: yielding a vector of transformed data $\hat{y} = g^{-1}(X\beta)$ that are used to model the data

```
g     <- function(x) log(x)
inv.g <- function(x) exp(x)
```

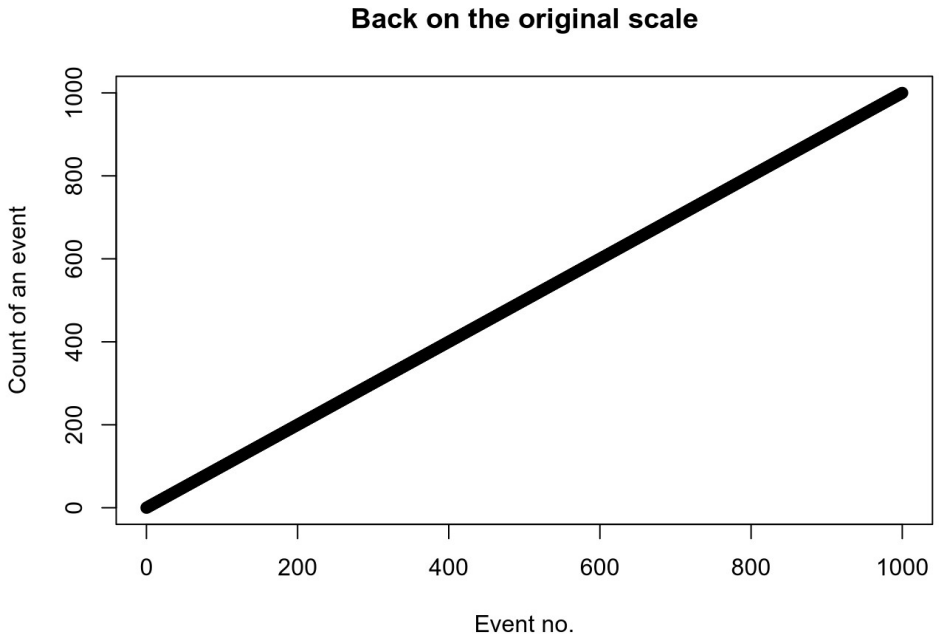
```
y <- seq(0, 1000, 1)
x <- 1:length(y)
plot(x, y, main='Count of an event', xlab='Event no.',
     ylab='Event counts')
```



```
plot(y, g(y), xlab='Count of an event', ylab='Log transformed',
     main='Log transformed data\nRange from -Inf to Inf and continuous')
```



```
plot(y, inv.g(g(y)), main='Back on the original scale',
     xlab='Event no.', ylab='Count of an event')
```



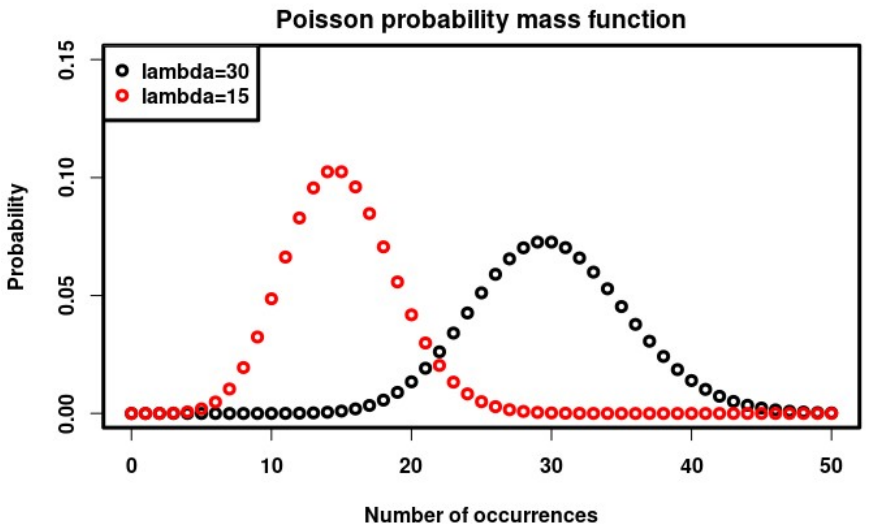
4) A data distribution: $p(y \mid \hat{y})$

$\boldsymbol{Poisson(\lambda)}$

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

$k \in N_0$

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



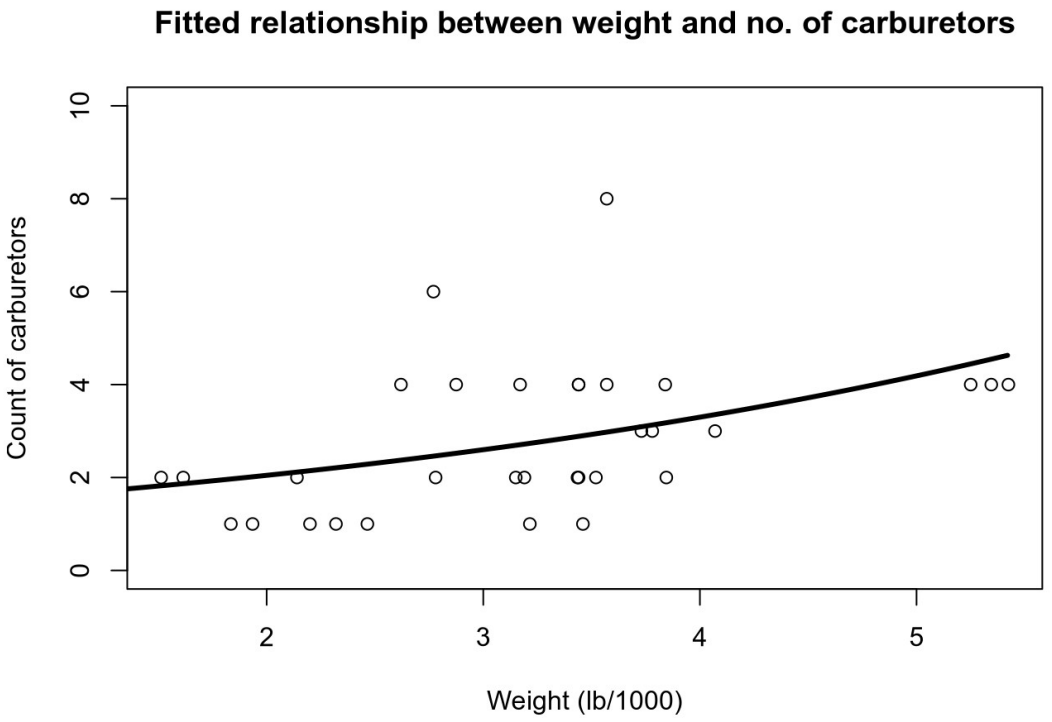
Bringing it all together

```
Call:  glm(formula = carb ~ wt, family = "poisson", data = mtcars)
Coefficients:
(Intercept)           wt
     0.2391       0.2386
Degrees of Freedom: 31 Total (i.e. Null);  30 Residual
Null Deviance:      27.04
Residual Deviance: 21.96        AIC: 115.3
```

```r
pois.reg       <- glm(carb ~ wt, data=mtcars, family='poisson')
y              <- pois.reg$y
X              <- model.matrix(pois.reg)
beta.hat       <- coef(pois.reg)
lin.pred       <- X %*% beta.hat
y.hat          <- inv.g(lin.pred) # identical to fitted(pois.reg)
likelihoods    <- dpois(y, y.hat) # poisson
likelihood     <- prod(likelihoods)
log.likelihood <- log(likelihood) ## check for yourself with logLik(pois.reg)
res.deviance   <- 2 * sum(y * log(y/y.hat) - (y - y.hat))
n.predictors   <- length(beta.hat)
AIC            <- 2 * n.predictors - 2*log.likelihood
print(pois.reg)
```

live coding

```r
x.pred <- seq(0, max(mtcars$wt), 0.01)
y.pred <- inv.g(coef(pois.reg)[1] + coef(pois.reg)[2] * x.pred)
plot(carb ~ wt, data=mtcars, ylab='Count of carburetors'
     , xlab='Weight (lb/1000)',
      main='Fitted relationship between weight and no. of carburetors',
     ylim=c(0.0, 10.0))
lines(x.pred, y.pred, lwd=3)
```



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

$p(y|\hat{y})$: the likelihood of observing y when our fitted value is $\hat{y}$

$$\hat{y} = \lambda$$

$$p(y|\hat{y}) = \frac{\hat{y}^{y} e^{-\hat{y}}}{y!}$$: Poisson distribution

$$y \in \mathbb{N}_0; \hat{y} \in [0, \infty)$$

Why we need a data distribution

WE NEED A LIKELIHOOD TO MAXIMISE

$$L(\hat{\beta}|y) = \prod_{i=1}^{n} \frac{\hat{y}_i^{y_i} e^{-\hat{y}_i}}{y_i!}$$

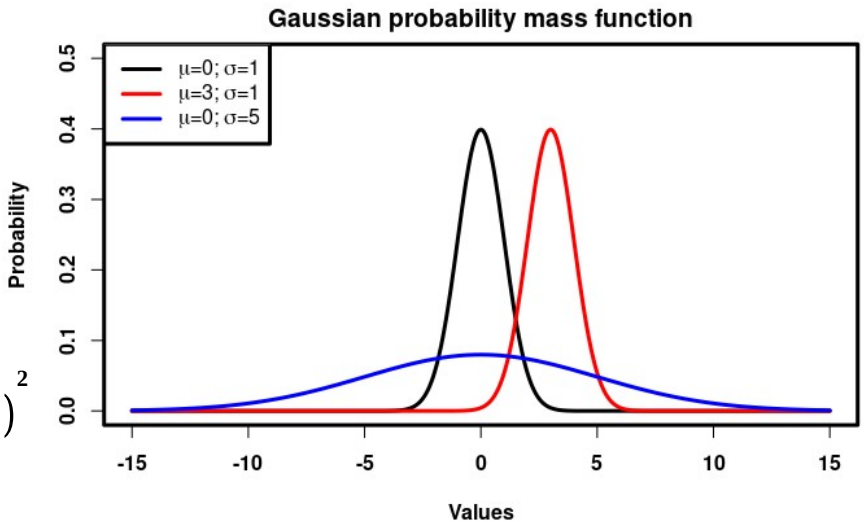
$$l(\hat{\beta}|y) = \sum_{i=1}^{n} y_i \log(\hat{y}_i) - \hat{y}_i - \log(y_i!)$$

... the identity link

```r
g     <- function(x) identity(x)
inv.g <- function(x) identity(x) ## identity⁻¹ = identity
```

$$N(\mu, \sigma^2)$$
$$\mu \in R$$
$$\sigma^2 \in R_{>0}$$
$$x \in R$$

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



Bringing it all together

```
Call:  glm(formula = mpg ~ wt, family = "gaussian", data = mtcars)
Coefficients:
(Intercept)           wt
     37.285       -5.344
Degrees of Freedom: 31 Total (i.e. Null);  30 Residual
Null Deviance:      1126
Residual Deviance: 278.3        AIC: 166
```

```r
lin.reg          <- glm(mpg ~ wt, data=mtcars, family='gaussian')
y                <- lin.reg$y
n                <- length(y)
X                <- model.matrix(lin.reg)
beta.hat         <- coef(lin.reg)
lin.pred         <- X %*% beta.hat
y.hat            <- inv.g(lin.pred) # identical to fitted(lin.reg)
sigma.hat.squared <- 1/n * sum((y - y.hat)^2)
log.likelihood   <- -n/2*log(2*pi) - n/2*log(sigma.hat.squared) - n/2
res.deviance     <- sum((y - y.hat)^2)
n.predictors     <- length(beta.hat) + 1 # sigma hat squared
AIC              <- 2 * n.predictors - 2*log.likelihood
print(lin.reg)
```

$$L(\beta, \sigma^2|y) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - X\beta)^2}{2\sigma^2}}$$

$$l(\beta, \sigma^2|y) = \frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - X\beta)^2$$

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

$$l(\hat{\beta}, \hat{\sigma}^2|y) = -\frac{n}{2}\log(2\pi) - \frac{n}{2}\log(\hat{\sigma}^2) - \frac{n}{2}$$

Summary

- We can extend the General Linear Model to a Generalised Linear Model by ...
  - choosing appropriate link functions
    - we can define our own
  - and using appropriate data distributions
    - such that we have a likelihood function to maximise
- This can be used to extend the General Linear Multilevel Model to the Generalised Linear Multilevel Model
  - And we can get the benefits of partial pooling among other things

Maximising likelihood for multilevel models

Beyond the scope of the present course

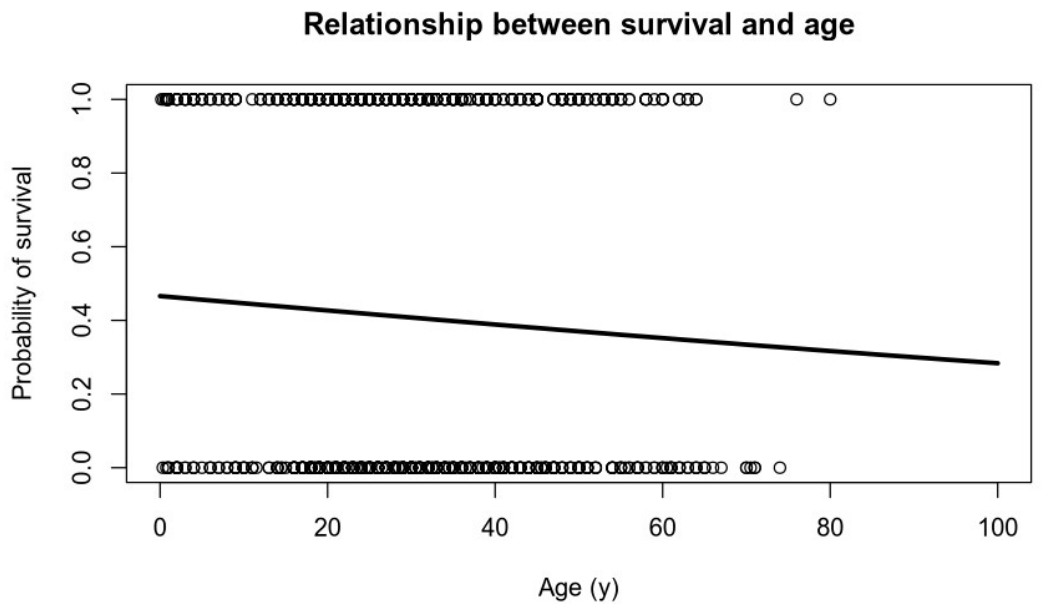
Bates D, Mächler M, Bolker B, Walker S (2015) Fitting Linear Mixed-Effects Models Using lme4. Journal of Statistical Software 67:1–48. https://doi.org/10.18637/jss.v067.i01

Example of how to fit

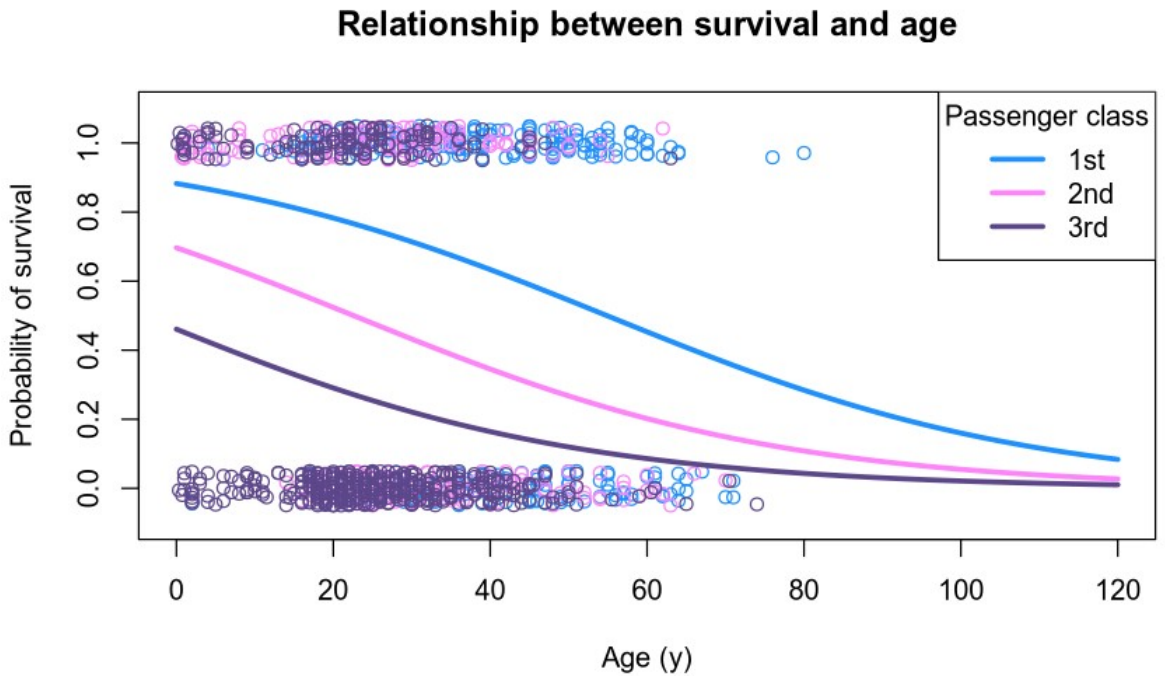
```r
library(carData)
library(lme4)
inv.g <- function(x) exp(x) / (1 + exp(x)) ## logit⁻¹
data('TitanicSurvival')
titanic <- TitanicSurvival

titanic$survived <- ifelse(titanic$survived == 'yes', 1, 0)

model.glm <- glm(survived ~ age, data=titanic, family='binomial')
```



```
model.glmm <- glmer(survived ~ age + (1 | passengerClass), data=titanic,
                    family='binomial')
```



Poisson example

```
data('Wool')
str(Wool)
```

```
## 'data.frame':    27 obs. of  4 variables:
##  $ len   : int  250 250 250 250 250 250 250 250 250 300 ...
##  $ amp   : int  8 8 8 9 9 9 10 10 10 8 ...
##  $ load  : int  40 45 50 40 45 50 40 45 50 40 ...
##  $ cycles: int  674 370 292 338 266 210 170 118 90 1414 ...
```

```
pois.glmm <- glmer(cycles ~ 1 + (1 | len) + (1 | amp) + (1 | load), data=Wool,
                   family='poisson')
```

Learning goals and outline

Link functions and fitting generalised linear multilevel models

1) Understanding that we can extend the scope of our general linear model by using appropriate link functions and data distributions
2) Understanding that we can extend this scope to multilevel modelling as well
3) Getting an understanding of maximum likelihood estimation

The course plan

Week 1: *Introduction*

Instructor sessions: *Setting up* R *and* Python *and recollection of the general linear model*

Week 2: *Multilevel linear regression*

Instructor sessions: *Modelling subject level effects – and how do they differ from group level effects?*

Week 3: *Link functions and fitting generalised linear multilevel models*

Instructor sessions: *What to do when the response variable is not continuous?*

Week 4: *Evaluating Generalised linear mixed models*

Instructor sessions: *How do we assess how models compare to one another?*

Week 5: *Explanation and Prediction*

Instructor sessions: *Code review*

Week 6: *Mid-way evaluation and Machine Learning Intro*

Instructor sessions: *Getting Python Running*

Week 7: *Linear regression revisited (machine learning)*

Instructor sessions: *How to constrain our models to make them more predictive*

Week 8: *Logistic regression revisited (machine learning)*

Instructor sessions: *Categorizing responses based on informed guesses*

Week 9: *Dimensionality Reduction, Principled Component Analysis (PCA)*

Instructor sessions: *What to do with very rich data?*

Week 10: *Outlook, unsupervised classification and neural networks*

Instructor sessions: *Data with no labels and networks*

Week 11: *Organising and preprocessing messy data*

Instructor sessions: *Code review*

Week 12: Final evaluation and wrap-up of course

Instructor sessions: *Ask anything!*

Next time

- Overdispersion and other link functions
- Comparison of models
  - R^2
  - Log-likelihood testing
  - Akaike Information Criteria
  - Regularisation

Reading questions

- Chapter 14; focus on sections 14.1 and 14.2
  - Read for examples of advanced multilevel models and what they can be good for
- Chapter 15; focus on section 15.1
  - What is overdispersion?
- Suggestion: re-read pp. 31-42 of: Winter B (2013) Linear models and linear mixed effects models in R with linguistic applications. arXiv:13085499 [cs]