Linear Models

Programming in R for Data Science

Anders Stockmarr, Kasper Kristensen, Anders Nielsen

Linear models

Statistical models of a linear relationship between variables:

$$Y_i = \alpha + \beta X_i + \varepsilon_i, \qquad i = 1, \ldots, n.$$

- Dependent variable: Y.
- Independent variable: X.
- Stochastic term/error term: ε.

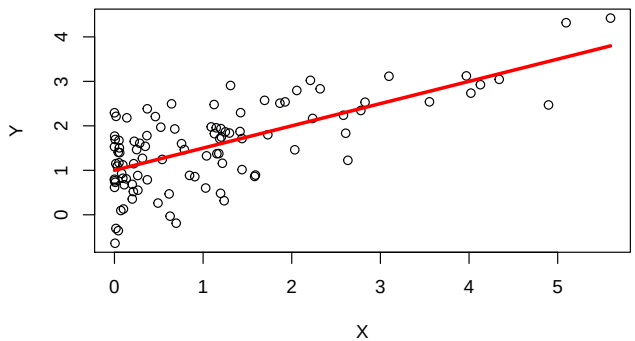
The ε_i's should be a) **stochastically independent**, and b) **identically normally distributed**, with mean 0, and variance σ^2 for some positive number $\sigma^2 > 0$.

- Model parameters: α, β and σ^2.

Linear models: Example

$$Y = 1 + 0.5X + \varepsilon$$

```
> plot(X,Y,xlab='X',ylab='Y')
> lines(sort(X),1+0.5*sort(X),lwd=3,col="red")
```



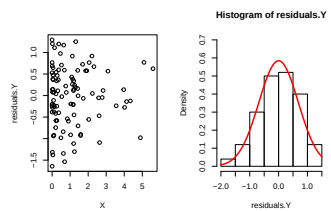
Linear models: Example

Model residuals: The random/stochastic term.

```
> residuals.Y<-Y-(1+0.5*X)
```

should be: a) **stochastically independent**, and b) **identically normally distributed**, with mean 0, and variance σ^2 for some positive number $\sigma^2 > 0$.

```
f<-function(x){
    dnorm(x,sd=sd(residuals.Y))}
par(mfrow=c(1,2))
plot(X,residuals.Y)
hist(residuals.Y,
     probability=TRUE,
     ylim=c(0,0.5))
curve(f,col="red", lwd=3,
      add=TRUE)
```



Fitting linear models: The lm() function

$$Y = \alpha + \beta X + \varepsilon$$

In R, linear models can be fitted to data with the `lm() function`:

```
> analysis<-lm(Y~X)
> analysis

Call:
lm(formula = Y ~ X)

Coefficients:
(Intercept)            X
     0.9702       0.5155
```

$\hat{\alpha}$ is the intercept 0.97, while $\hat{\beta}$ is the estimated coefficient to X, 0.52.

Model formulas

The argument to `lm()` is a *formula* object.

- A linear model is specified by a formula object, which t.ex. may look like this:

```
> my.formula<-formula(y~x+z+w)
> fit<-lm(my.formula)
```

Corresponding linear Model:

$$y_i = \alpha + \beta_x x_i + \beta_z z_i + \beta_w w_i + \varepsilon_i, \quad i = 1, \ldots n$$

- Intercept: R by default assumes that your model contains the intercept (α). You can get rid of the common intercept by adding either a `0` or a `-1` to the model formula:

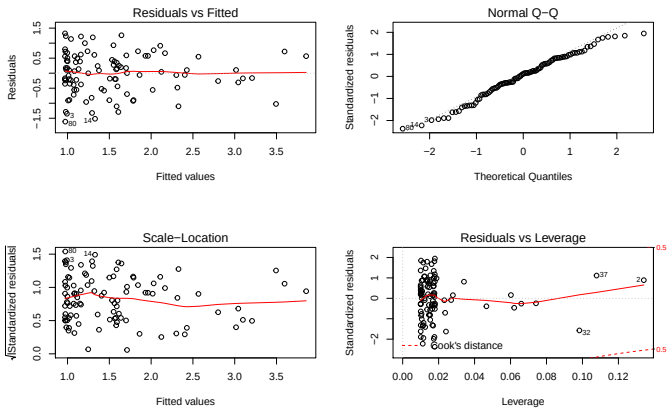
```
> fit<-lm(y~0+x+z+w)
```

or

```
> fit<-lm(y~-1+x+z+w)
```

The lm object: Model diagnostics

```
> analysis<-lm(Y~X); par(mfrow=c(2,2)); plot(analysis)
```



The lm object: Contents

- An lm object is a list, and contains a lot of information. See the contents with the *names()* function:

```
> analysis<-lm(Y~X)
> names(analysis)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [7] "qr"            "df.residual"   "xlevels"       "call"
```

- Access the contents with the $ operator; eg.

```
> analysis$coef
(Intercept)           X
  0.9701906   0.5154684
```

- Some of the 12 components of the list are lists themselves. Find more information by applying `str()`.

The lm object: Summaries

The summary() fuction may be applied to lm objects as well:

```
> analysis<-lm(Y~X)
> summary(analysis)

Call:
lm(formula = Y ~ X)

Residuals:
     Min       1Q   Median       3Q      Max
-1.61297 -0.40132  0.07808  0.55124  1.32380

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.97019    0.09182  10.566  < 2e-16 ***
X            0.51547    0.05410   9.527 1.29e-15 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.6861 on 98 degrees of freedom
Multiple R-squared:  0.4808,        Adjusted R-squared:  0.4755
F-statistic: 90.77 on 1 and 98 DF,  p-value: 1.286e-15
```

The lm object: Summaries

The summary is a R list object itself, with sub-elements that can be accessed:

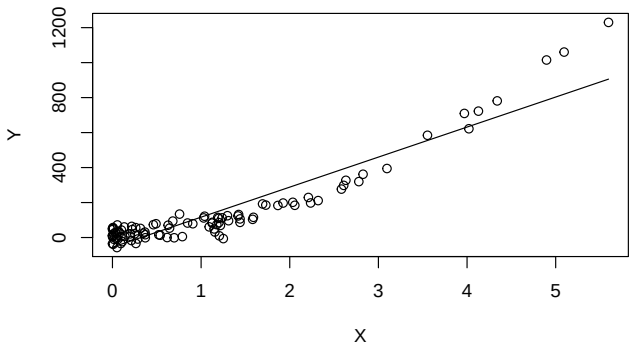
```
> analysis<-lm(Y~X)
> names(summary(analysis))
 [1] "call"          "terms"         "residuals"     "coefficients"  "al
 [7] "df"            "r.squared"     "adj.r.squared" "fstatistic"    "co
```

We can find the estimate $\hat{\sigma}^2$ for σ^2 as

```
> summary(analysis)$sigma^2
[1] 0.4707802
```

Modeling nonlinear relations with lm()

```
> plot(X,Y)
> lines(sort(X),predict(lm(Y~X))[order(X)])
```

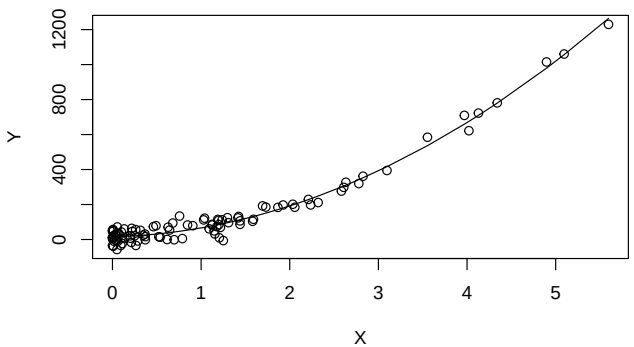


Relationship with Y and X is not linear. How to proceed with `lm()`?

Modeling nonlinear relations with lm()

- The I-operator in formulas:

```
> analysis<-lm(Y~X+I(X^2))
> plot(X,Y)
> lines(sort(X),predict(analysis)[order(X)],type="l" )
```



'Linear' in `lm()` is relative to the 'right' independent variables.

Extraction functions

- Some important extraction functions for obtaining information:

`coef()`	Estimated model parameters
`confint()`	Confidence intervals for estimated model parameters
`residuals()`	Raw residuals
`rstandard()`	Standardized residuals
`model.matrix()`	The design matrix
`predict()`	Predictions from model
`vcov()`	Covariance matrix for estimated model parameters
`anova()`	Anova test table for model reduction
`drop1()`	Test for dropping one term from model
`summary()`	A summary printout, and access to summary statistics

- Statistical tests: `drop1()` is usually the function to use.

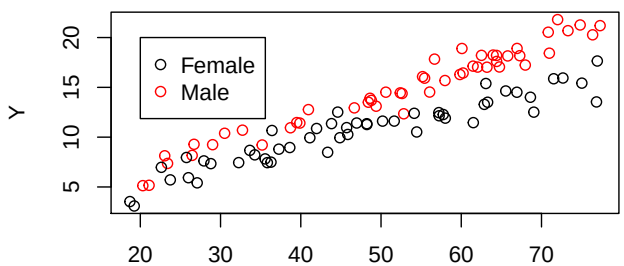
Factors and interactions

A dataset on Sex, Age, and a response Y:

```
> summary(my.data)
```

```
     Sex          Age              Y          
 Female:50   Min.   :18.70   Min.   : 3.091  
 Male  :50   1st Qu.:36.38   1st Qu.: 9.274  
             Median :51.12   Median :12.430  
             Mean   :49.99   Mean   :12.711  
             3rd Qu.:63.22   3rd Qu.:15.972  
             Max.   :77.31   Max.   :21.800  
```

```
plot(my.data$Age, my.data$Y,xlab='',ylab='Y',col=my.data$Sex)
legend(20,20,c("Female","Male"),col=1:2,pch=1)
```



Factors and interactions

Model: Interaction between Sex and Age. Testing the interaction term with `drop1()`:

```
> analysis<-lm(Y~Age+Sex+Age:Sex,data=my.data)
> drop1(analysis,test="F")
Single term deletions

Model:
Y ~ Age + Sex + Age:Sex
        Df Sum of Sq    RSS    AIC F value    Pr(>F)
<none>                102.77 10.737
Age:Sex  1    33.131 135.91 36.679  30.947 2.391e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The interaction is for real and cannot be removed.

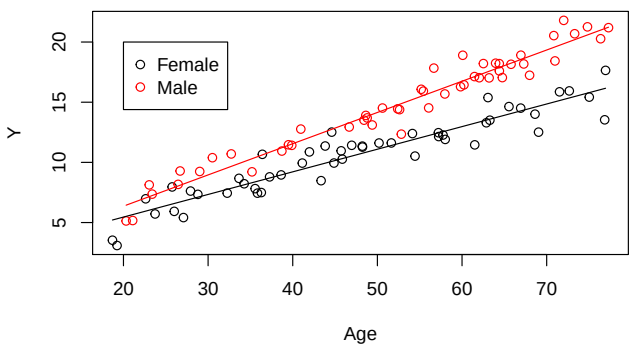
Factors and interactions

```
> summary(analysis)

Call:
lm(formula = Y ~ Age + Sex + Age:Sex, data = my.data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.60300 -0.53551  0.00317  0.59830  2.43544

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.696993   0.452879   3.747 0.000305 ***
Age          0.187921   0.009045  20.777  < 2e-16 ***
SexMale     -0.525623   0.677086  -0.776 0.439479
Age:SexMale  0.071599   0.012871   5.563 2.39e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.035 on 96 degrees of freedom
Multiple R-squared:  0.945,        Adjusted R-squared:  0.9433
F-statistic: 550.2 on 3 and 96 DF,  p-value: < 2.2e-16
```

- R selects the first level of the Sex variable; similarly for the interaction term.

Factors and interactions

```
> my.data2<-data.frame(my.data[order(my.data$Age),],
+                      predicted=predict(analysis)[order(my.data$Age)])
> with(my.data2,{
+   plot(Age,Y,xlab='Age',ylab='Y',col=Sex)
+   lines(Age[Sex=="Female"],predicted[Sex=="Female"],col=1,type="l")
+   lines(Age[Sex=="Male"],predicted[Sex=="Male"],col=2,type="l")
+   legend(20,20,c("Female","Male"),col=1:2,pch=1)
+   })
```



More on formula objects

- Model formulae are *symbolic*. We have seen the use of '+' and ':', and adding a 0 or -1.
- The product '*' *crosses* variables: Expands to main effects and interactions:

  `y ~ x*z`

  corresponds to

  `$y~x+z+x:z$`
- *Powers* expands effects to the specified order:

  `y~(x+z+w)^2`

  corresponds to

  `y~x+z+w+x:z+x:w+z:w`
- The subtraction function '-' removes variables if possible:

  `y~(x+z+w)^2-x:z-a:b`

  corresponds to

  `y~x+z+w+x:w+z:w`

More on formula objects

The `I()` function overrides the symbolic interpretation, and invokes the usual arithmetic instead.

Observe that

`y~(x*z)^2 = y~(x+z)^2`

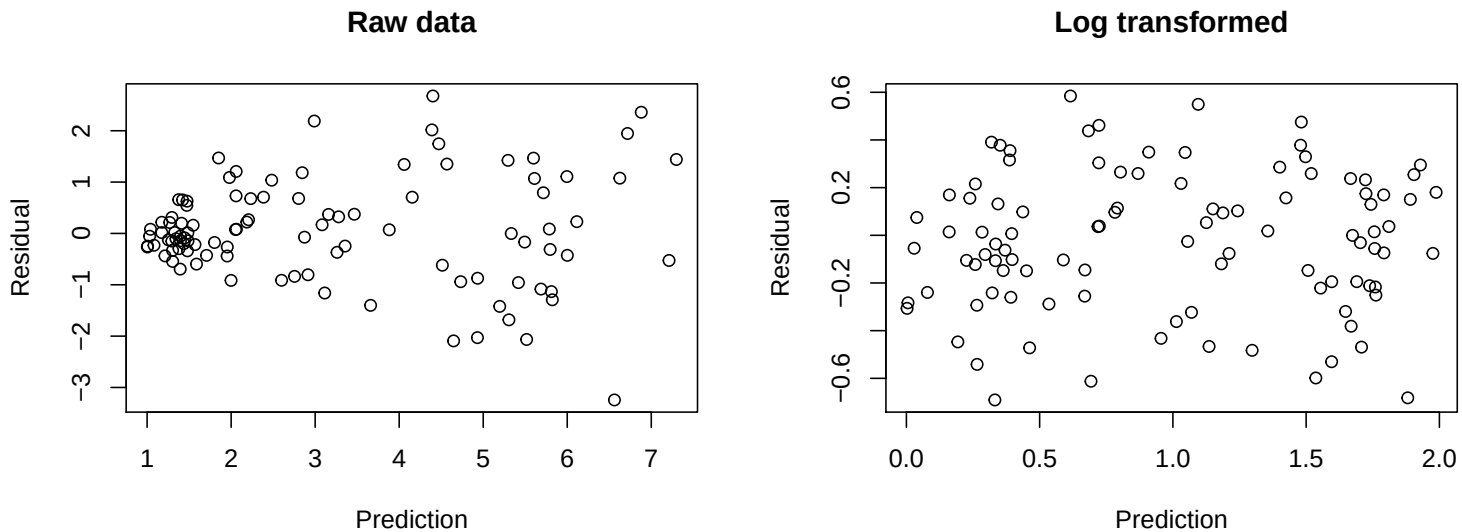
But

`y~I((x*z)^2) and y~I((x+z)^2)`

are two different model formulas; regressing y on $x^2 z^2$ and $x^2 + z^2 + 2xz$, respectively.

Formulas when transforming data into normality

- Sometimes it is possible to *transform data*, such that it matches a linear model.
- For instance if the variance is increasing with the mean



- A log transformation is often appropriate in this case.
- This may be done directly in a formula object. T. ex:

```
log(y)~log(x)+log(z)
```

Generalized linear models - the glm() function

- Some types of observations can never be transformed into normality
- Example: binary data; ones and zeroes.
- For a wide class of distributions, the so called *exponential families*, we can use *generalized linear models*:
- Formulate linear models for a transformation of the mean value.
- No transformation of observations, thereby preserving their distributional properties.
- Allows **easy** modeling in R with the `glm()` function, nearly identical to `lm()`.
- Standard example: Logistic regression.

GLM vs GLM

General linear models	**Generalized linear models**
Normal distribution	Exponential dispersion family
Mean value linear	Function of mean value linear
Independent observations	Independent observations
Same variance	Variance function of mean
`lm()` easy to apply	`glm()` almost as easy to apply

Types of response variables

i. Count data ($y_1 = 57, \ldots, y_n = 59$ accidents) - Poisson distribution.

ii. Binary response variables ($y_1 = 0, y_2 = 1, \ldots, y_n = 0$), or frequencies of counts ($y_1 = 15/297, \ldots, y_n = 144/285$) - Binomial distribution.

iii. Count data, waiting times - Negative Binomial distribution.

iv. Multiple ordered categories "Unsatisfied", "Neutral", "Satisfied" - Multinomial distribution.

v. Count data, multiple categories - Multinomial distribution..

vi. Continuous responses, constant variance ($y_1 = 2.567, \ldots, y_n = 2.422$) - Normal distribution.

vii. Continuous positive responses with constant coefficient of variation - Gamma distribution.

Logistic regression example

In a study of developmental toxicity of a chemical compound, a specified amount of an ether was dosed daily to pregnant mice, and after 10 days all fetuses were examined. The size of each litter and the number of stillborns were recorded:

Index	Number of stillborn, z_i	Number of fetuses, n_i	Fraction still-born, y_i	Concentration [mg/kg/day], x_i
1	15	297	0.0505	0.0
2	17	242	0.0702	62.5
3	22	312	0.0705	125.0
4	38	299	0.1271	250.0
5	144	285	0.5053	500.0

Table: Results of a dose-response experiment on pregnant mice. Number of stillborn fetuses found for various dose levels of a toxic agent.

Reported in Price et al. (1987).

Logistic regression example

Let Z_i denote the number of stillborns at dose concentration x_i.

We shall assume $Z_i \sim B(n_i, p_i)$, that is a binomial distribution corresponding to n_i independent trials (fetuses), and the probability, p_i, of stillbirth being the same for all n_i fetuses.

We want to model $Y_i = Z_i/n_i$. In particular, we will look for a model for $E[Y_i] = p_i$.

Logistic regression example

- A natural quantity to consider is the odds, $p/(1-p)$; varies on $(0;\infty)$, more natural than $(0;1)$ where p varies.
- since effects on the odds are often multiplicative, we take the log to convert the effects to additive form.
- we arrive at the logit function:

$$logit(p) = \log\left(\frac{p}{1-p}\right).$$

for this model, the logit function is our *link function*. We will formulate a linear model for the mean values transformed with the link function:

$$\eta_i = logit(p_i), \quad i = 1, \ldots, 5.$$

The linear model is

$$\eta_i = \alpha + \beta x_i, \quad i = 1, \ldots, 5.$$

- The inverse transformation, which gives the probabilities, p_i, for stillbirth is the so-called *logistic function*:

$$p_i = \frac{\exp(\alpha + \beta x_i)}{1 + \exp(\alpha + \beta x_i)}, \quad i = 1, \ldots, 5.$$

Logistic regression example

```
> mice<-data.frame(
+   stillb=c(15, 17, 22, 38, 144),
+   total=c(297, 242, 312, 299, 285),
+   conc=c(0, 62.5, 125, 250, 500))
> mice$resp <- cbind(mice$stillb,mice$total-mice$stillb)
```

Note that the response variable is composed by the vector of the number of stillborns, z_i, and the number of live fetuses, $n_i - z_i$.

We fit the model with the `glm()` function:

```
> mice.glm <- glm(resp ~ conc,
+                 family = binomial(link = logit),
+                 data = mice)
```

Logistic regression example

```
> summary(mice.glm)

Call:
glm(formula = resp ~ conc, family = binomial(link = logit), data = mice)

Deviance Residuals:
      1        2        3        4        5
 1.1317   1.0174  -0.5968  -1.6464   0.6284

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) -3.2479337  0.1576602   -20.6   <2e-16 ***
conc         0.0063891  0.0004348    14.7   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 259.1073  on 4  degrees of freedom
Residual deviance:   5.7775  on 3  degrees of freedom
AIC: 35.204

Number of Fisher Scoring iterations: 4
```

Logistic regression example

The linear predictor, $\hat{y}_i = \hat{\alpha} + \hat{\beta} x_i$:

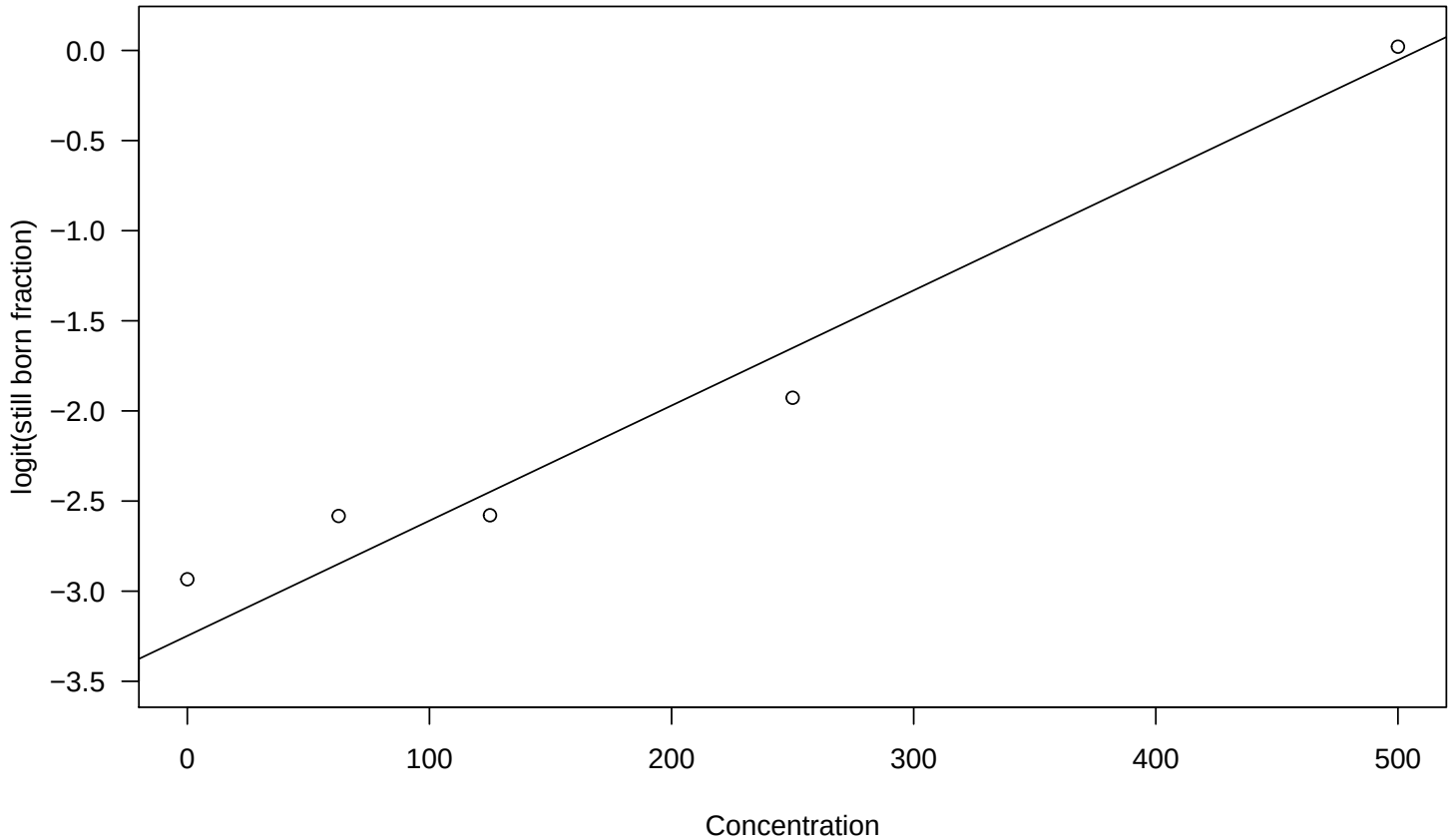


Figure: Logit transformed observations and corresponding linear predictions for dose response assay.

Logistic regression example

Predicted stillborn fractions, $\hat{p}_i = exp(\hat{y}_i)/(1 + exp(\hat{y}_i))$:

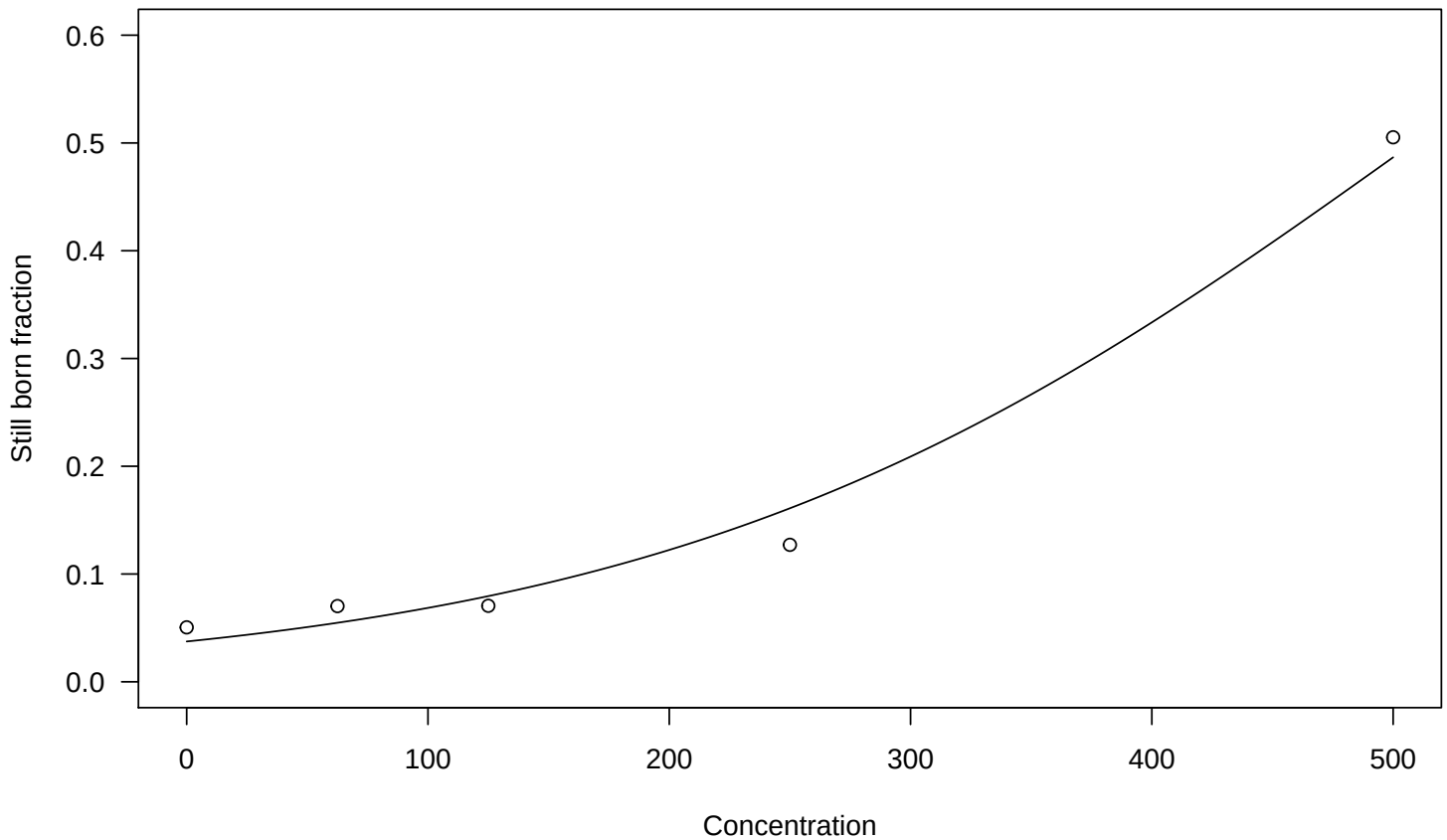


Figure: Observed stillborn fractions and corresponding fitted values under logistic regression for dose response assay.

Specification of a generalized linear model in glm()

```
> mice.glm <- glm(formula = resp ~ conc,
+                 family = binomial(link = logit),
+                 data = mice
+                )
```

- formula: As in general linear models (lm())
- family:
  - binomial(link = logit | probit | cauchit | log | cloglog)
  - gaussian(link = identity | log | inverse)
  - Gamma(link = inverse | identity | log)
  - inverse.gaussian(link = 1/mu^2 | inverse | identity | log)
  - poisson(link = log | identity | sqrt)
  - quasi(link = ... , variance = ...))
  - quasibinomial(link = logit | probit | cauchit | log | cloglog)
  - quasipoisson(link = log | identity | sqrt)

Application: Classification models

Diabetes data: diabetes information for 12795 Caucasians.

```
> dim(diabetes.data)

[1] 12795    18

> names(diabetes.data)

 [1] "gender"                    "age"                       "weight"
 [4] "discharge_disposition_id"  "time_in_hospital"          "payer_code"
 [7] "num_procedures"            "number_outpatient"         "number_emergency"
[10] "number_inpatient"          "diag_1"                    "number_diagnoses"
[13] "max_glu_serum"             "glipizide"                 "insulin"
[16] "diabetesMed"               "admission_type_description" "readmi_class"
```

The `readmi_class` variable contains readmission status to hospitals, in terms of YES and NO. We need to predict it.

Application: Classification models

Building a logistic regression model; first the formula:

```
> my.model<-"gender"
> for(i in 2:17){my.model<-paste(my.model,
+                                "+",
+                                names(diabetes.data)[i])}
> my.formula<-as.formula(paste("readmi_class~",my.model))
```

then the analysis itself:

```
> analysis<-glm(my.formula,
+               family=binomial(link=logit),
+               data=diabetes.data)
```

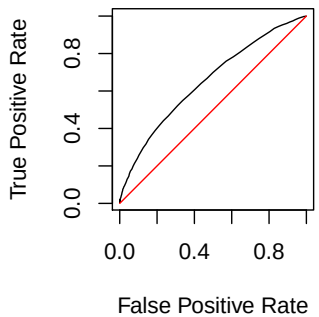
Application: Classification models

- New dataset: `new.diabetes.data`. containing 10.000 new observations.
- Can we predict the readmission class in new.diabetes.datafrom the analysis object?
- Predict status to be "YES" if the estimator for p is above a certain level.
- We use the `predict()` function in a new way, and calculate the rate of false positives and true positives:

```
> logit<-function(p){log(p/(1-p))}
> fp<-numeric(99)
> tp=numeric(99)
> for(i in 1:99){
+   predict.diabetes<-1*(predict(analysis,
+                        newdata=new.diabetes.data)>logit(i/100))
+   temp<-tapply(predict.diabetes,
+                new.diabetes.data$readmi_class,
+                mean)
+   fp[i]<-temp[1]
+   tp[i]<-temp[2]
+   }
```

Application: Classification models

Performance testing: ROC curve.Plot of `fp` and `tp`:



Area under the curve: 0.66, well above 0.5.