Logistic regression

Gerko Vink

Fundamental Techniques in Data Science with R

Packages and functions used

```
library(magrittr) # pipes
library(dplyr)    # data manipulation
library(mice)     # data
library(ggplot2)  # plotting
library(DAAG)     # data sets and functions
```

- `glm()` Generalized linear models
- `predict()` Obtain predictions based on a model
- `confint()` Obtain confidence intervals for model parameters
- `coef()` Obtain a model's coefficients
- `DAAG::CVbinary()` Cross-validation for regression with a binary response

So far

At this point we have covered the following models:

- Simple linear regression (SLR)

$$y = \alpha + \beta x + \epsilon$$

The relationship between a numerical outcome and a numerical or categorical predictor

- Multiple linear regression (MLR)

$$y = \alpha + \beta_1 x_1 + \beta_2 x_2 + \ldots \beta_p x_p + \epsilon$$

*The relationship between a numerical outcome and **multiple** numerical or categorical predictors*

What remains

We have not yet covered how to handle outcomes that are not categorical or how to deal with predictors that are nonlinear or have a strict dependency structure.

What should you know

At this point you should know how to

- fit SLR and MLR models
- select MLR models
- interpret model parameters
- perform hypothesis test on slope and intercept parameters
- perform hypothesis test for the whole regression model
- calculate confidence intervals for regression parameters
- obtain prediction intervals for fitted values
- study the influence of single cases
- study the validity of linear regression assumptions:
  - linearity, constant residual variance
- study the residuals, leverage and Cook's distance

Rewriting what we know

Instead of modeling

$$y = \alpha + \beta x + \epsilon$$

we can also consider

$$\mathbb{E}[y] = \alpha + \beta x$$

They're the same. Different notation, different framework.

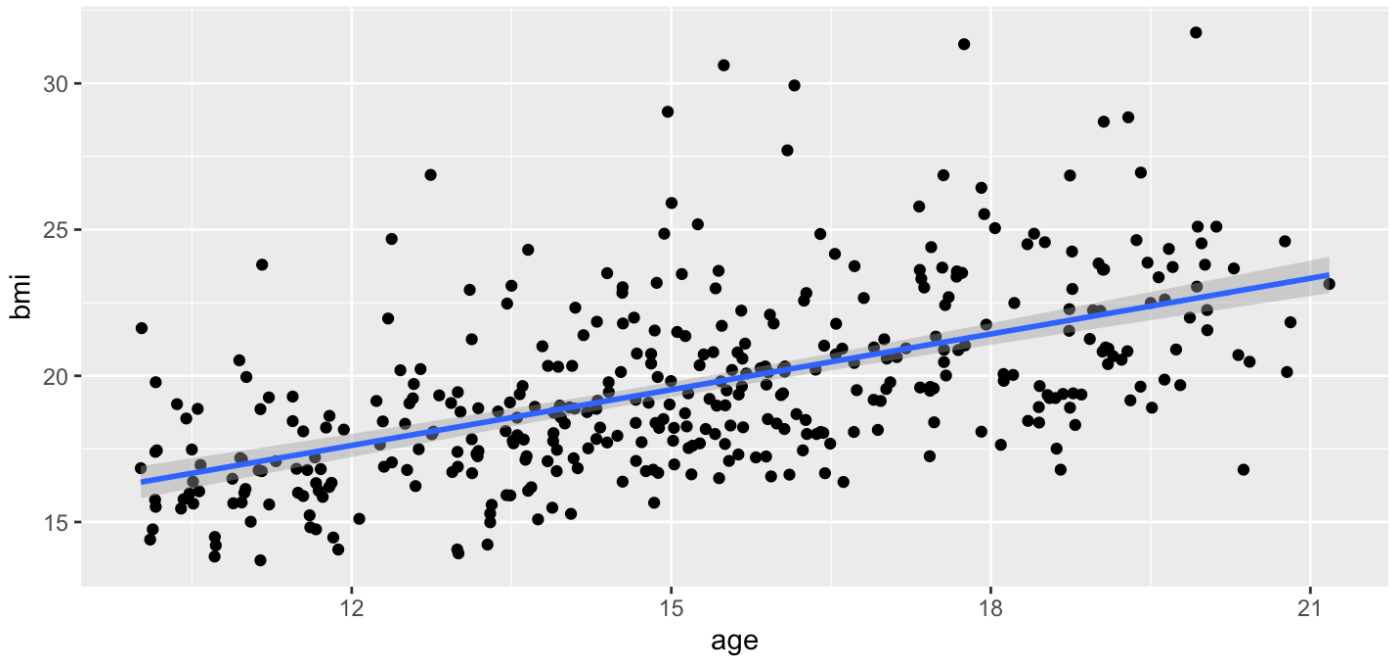
The upside is that we can now use a function for the expectation $\mathbb{E}$ to allow for transformations. This would enable us to change $\mathbb{E}[y]$ such that $f(\mathbb{E}[y])$ has a linear relation with x.

This is what we will be doing today

Illustration of the problem

Example: boys' `bmi`

```
boys %>%
  filter(age > 10) %>%
  ggplot(aes(y = bmi, x = age)) + geom_point() + geom_smooth(method = "lm")
```



Adding a column to the data

We have information on `bmi`. We know that a `bmi > 25` would indicate being overweight. We could add this information to the data

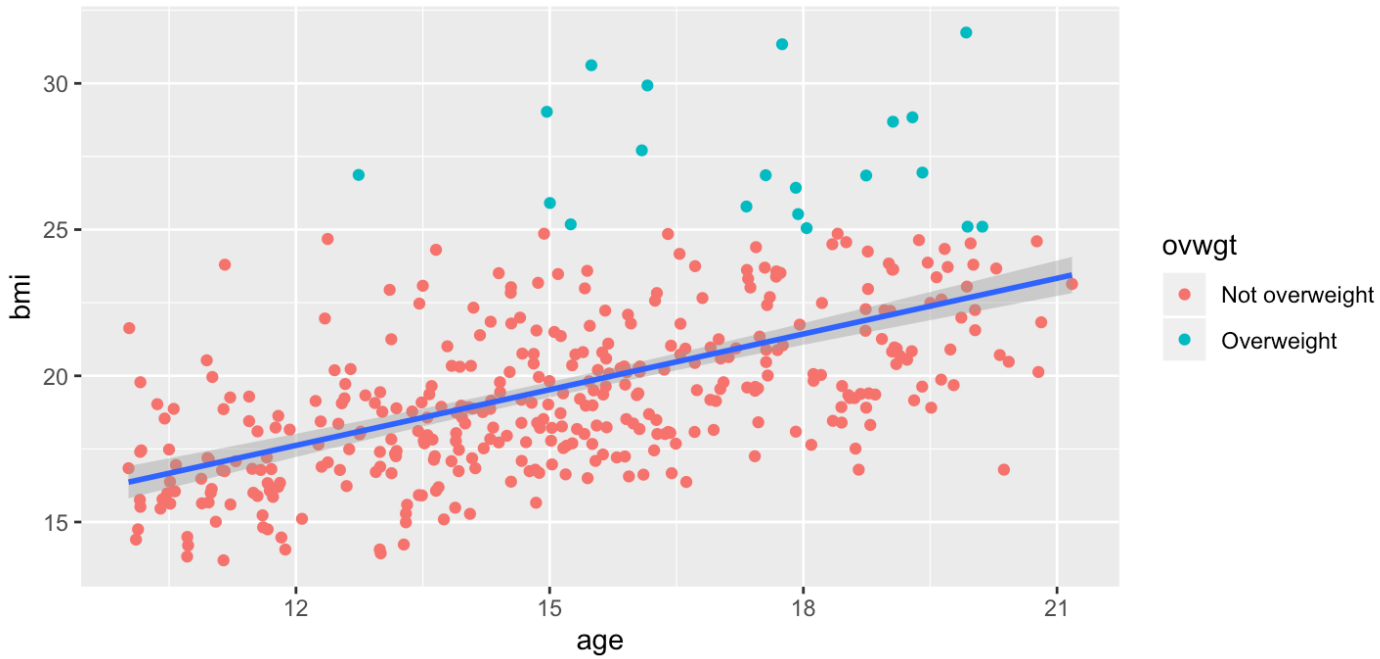
```
boys.ovwgt <- boys %>%
  filter(age > 10, !is.na(bmi)) %>%
  mutate(ovwgt = cut(bmi,
                     breaks = c(0, 25, Inf),
                     labels = c("Not overweight", "Overweight")))

boys.ovwgt %>% filter(bmi > 24) %>% head
```

```
##      age   hgt  wgt   bmi   hc  gen  phb tv   reg          ovwgt
## 1 12.375 157.2 61.0 24.68 54.0   G1   P1  3 south Not overweight
## 2 12.741 172.0 79.5 26.87 55.0   G2   P3  8 south     Overweight
## 3 13.656 175.4 74.8 24.31 59.2   G4   P5 20  city Not overweight
## 4 14.934 156.0 60.5 24.86 56.9 <NA> <NA> NA  east Not overweight
## 5 14.967 174.1 88.0 29.03 54.4   G3   P4 10  east     Overweight
## 6 15.003 188.0 91.6 25.91 59.8   G4   P4 12 south     Overweight
```

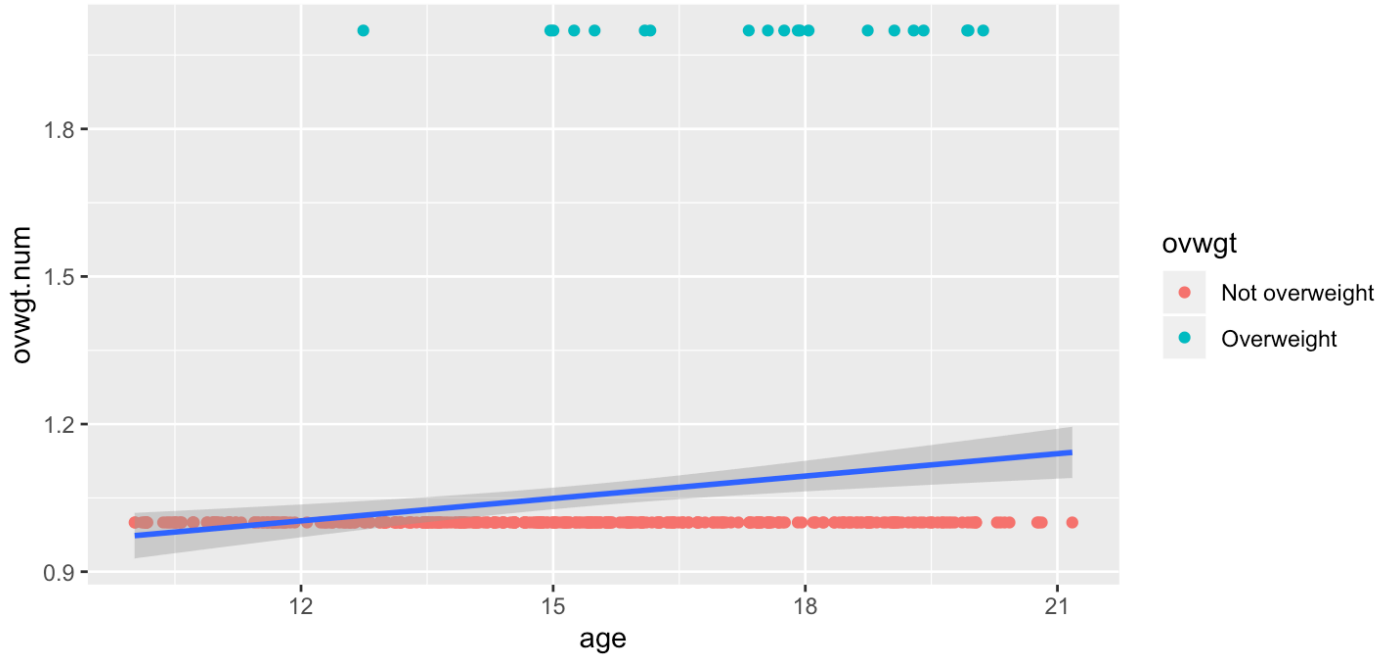
Example: boys' `bmi`

```
boys.ovwgt %>%
  ggplot(aes(y = bmi, x = age)) + geom_point(aes(color = ovwgt)) +
  geom_smooth(method = "lm")
```



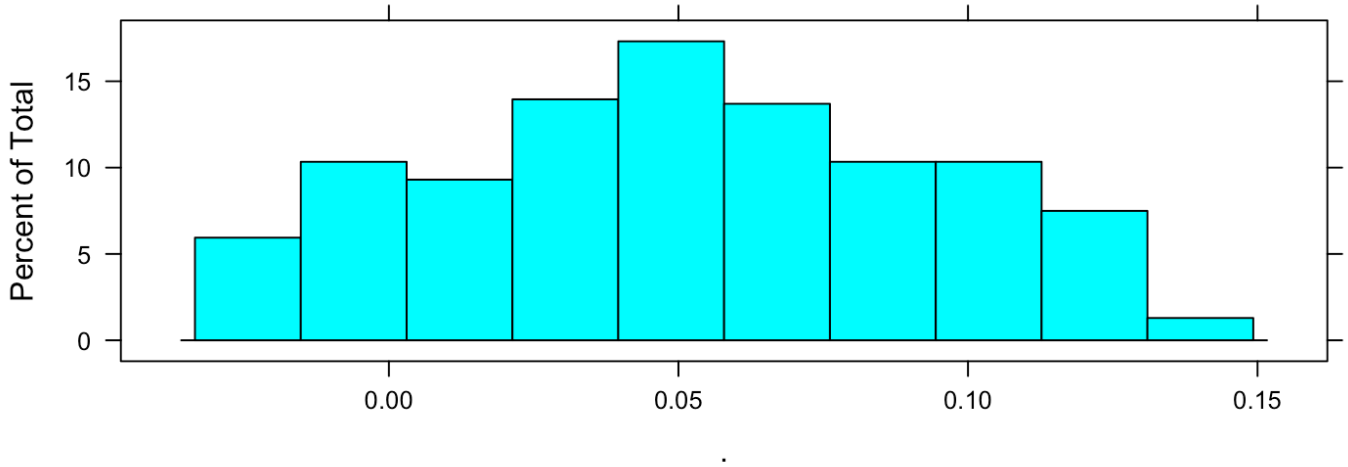
Example: boys' **overweight**

```
boys.ovwgt %>%
  mutate(ovwgt.num = as.numeric(ovwgt)) %>%
  ggplot(aes(y = ovwgt.num, x = age)) + geom_point(aes(color = ovwgt)) +
  geom_smooth(method = "lm")
```



Modeling `ovwgt ~ age` #1

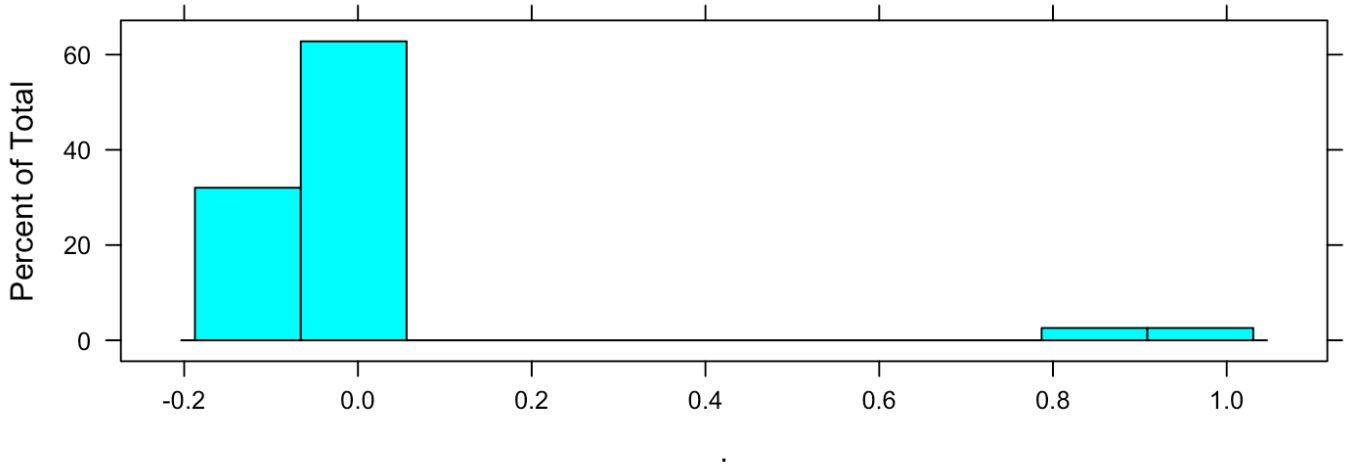
```
fit <- boys.ovwgt %>%
  mutate(ovwgt.num = as.numeric(ovwgt) - 1) %$%
  lm(ovwgt.num ~ age)
fit$fitted.values %>% histogram
```



The predicted values for the outcome are all very low. the large number of `Not overweight` boys heavily influences the estimation. No boy is predicted to be `Overweight`.

Modeling `ovwgt ~ age` #2

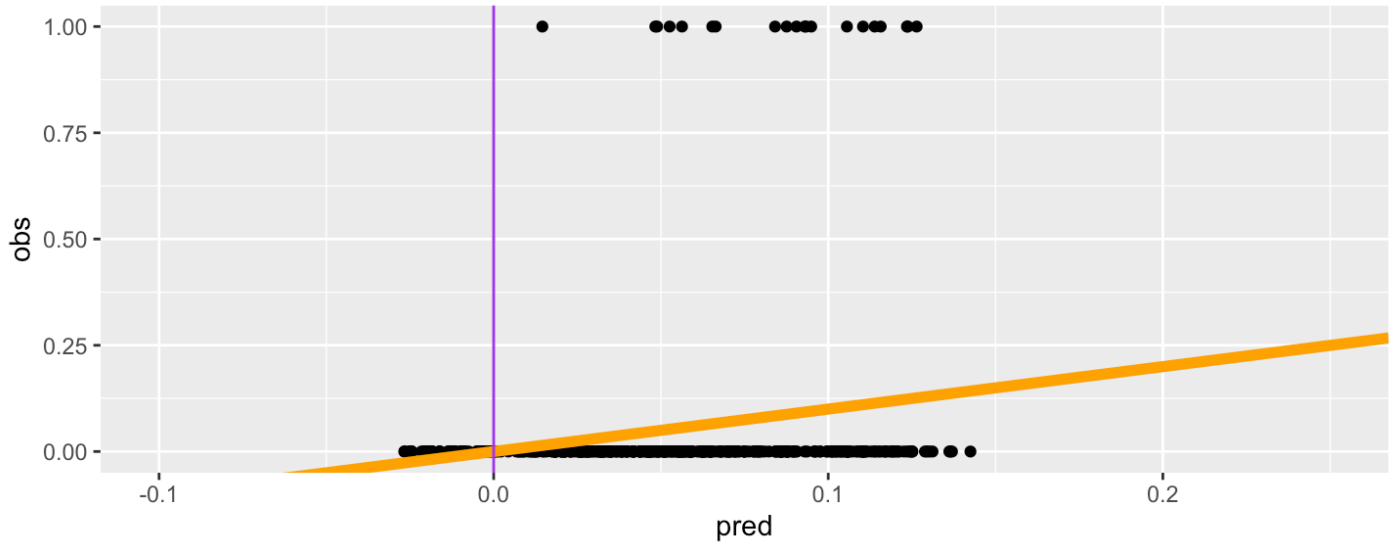
```
fit$residuals %>% histogram
```



Naturally, these residuals are not normally distributed.

Modeling **ovwgt ~ age** #3

```
plotdata <- data.frame(obs = fit$model$ovwgt.num,
                       pred = fit %>% predict)
plotdata %>% ggplot(aes(x = pred, y = obs)) +
  geom_point() + geom_abline(slope = 1, intercept = 0, color = "orange", lwd = 2) +
  xlim(-.1, .25) + geom_vline(xintercept = 0, color = "purple")
```



The relation between the observed outcome (black) and the predicted outcome (orange) is by no means equivalent. The assumption of linearity is heavily violated.

Example: `simulated` data

To further illustrate why the linear model is not an appropriate model for discrete data I propose the following simple simulated data set:

```
set.seed(123)
simulated <- data.frame(discrete = c(rep(0, 50), rep(1, 50)),
                        continuous = c(rnorm(50, 10, 3), rnorm(50, 15, 3)))

simulated %>% summary
```

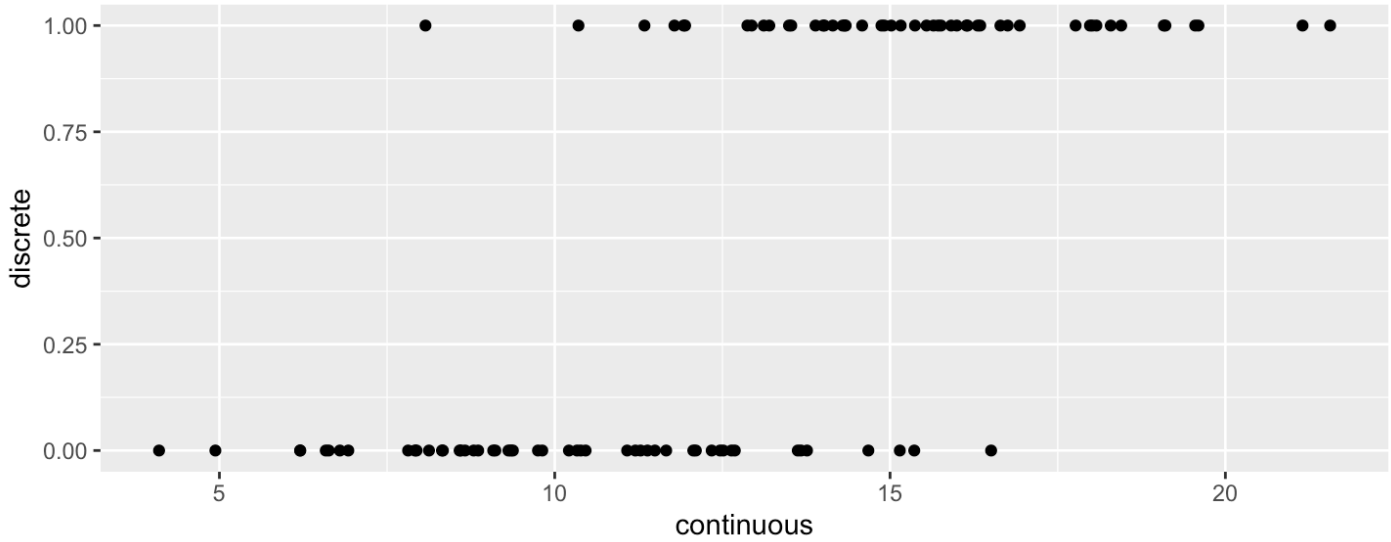
```
##     discrete     continuous
##  Min.   :0.0   Min.   : 4.100
##  1st Qu.:0.0   1st Qu.: 9.656
##  Median :0.5   Median :12.904
##  Mean   :0.5   Mean   :12.771
##  3rd Qu.:1.0   3rd Qu.:15.570
##  Max.   :1.0   Max.   :21.562
```

This data allows us to illustrate modeling the relation between the `discrete` outcome and the `continuous` predictor with logistic regression.

Remember that fixing the random seed allows for a replicable random number generator sequence.

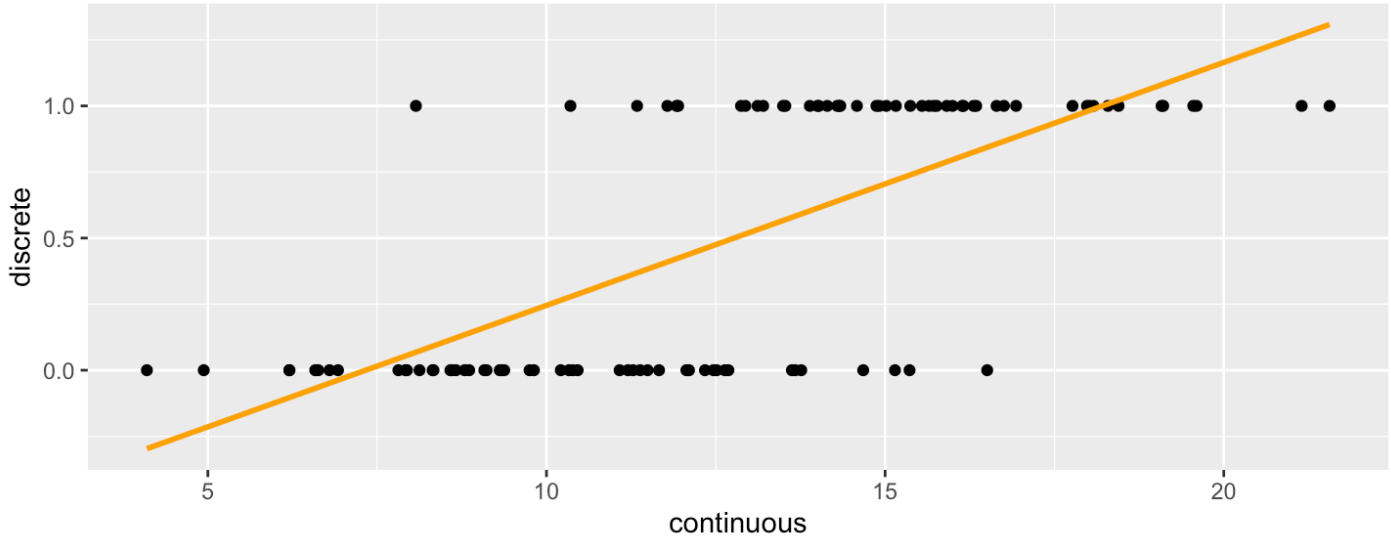
Visualizing `simulated` data

```
simulated %>% ggplot(aes(x = continuous, y = discrete)) +
  geom_point()
```



Modeling `simulated` with `lm`

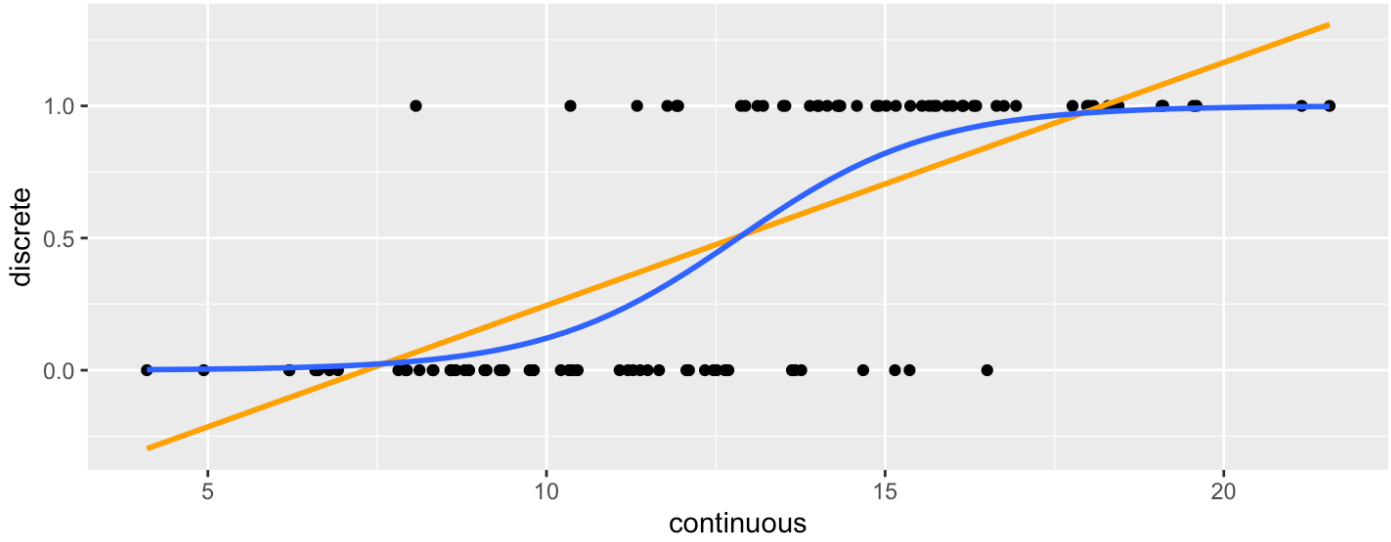
```
simulated %>% ggplot(aes(x = continuous, y = discrete)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE, color = "orange")
```



The orange line represents the `lm` linear regression line. It is not a good representation for our data, as it assumes the data are continuous and projects values outside of the range of the observed data.

Modeling `simulated` with `glm`

```
simulated %>% ggplot(aes(x = continuous, y = discrete)) +
  geom_point() + geom_smooth(method = "lm", se = FALSE, color = "orange") +
  geom_smooth(method = "glm", method.args = list(family = "binomial"), se = FALSE)
```



The blue `glm` logistic regression line represents this data infinitely better than the orange `lm` line. It assumes the data to be `0` or `1` and does not project values outside of the range of the observed data.

How does this work?

Generalized linear modeling

There is a very general way of addressing this type of problem in regression. The models that use this *general way* are called generalized linear models (GLMs).

Every generalized linear model has the following three characteristics:

1. A probability distribution that describes the outcome
2. A linear predictor model in the form
3. A link function that relates the linear predictor to the the parameter of the outcome's probability distribution.

The linear predictor model in (2) is

$$\eta = \mathbf{X}\beta$$

where η denotes a linear predictor and the link function in (3) is

$$\mathbf{X}\beta = \mathbf{g}(\mu)$$

The technique to model a binary outcome based on a set of continuous or discrete predictors is called *logistic regression*. **Logistic regression is an example of a generalized linear model.**

The link function

The link function for logistic regression is the `logit link`

$$\mathbf{X}\beta = \ln(\frac{\mu}{1-\mu})$$

where

$$\mu = \frac{\exp(\mathbf{X}\beta)}{1+\exp(\mathbf{X}\beta)} = \frac{1}{1+\exp(-\mathbf{X}\beta)}$$

Before we continue with discussing the link function, we are first going to dive into the concept of odds.

Properly understanding odds is necessary to perform and interpret logistic regression, as the `logit` link is connected to the odds

Modeling the odds

Odds are a way of quantifying the probability of an event E

The odds for an event E are

$$\text{odds}(E) = \frac{P(E)}{P(E^c)} = \frac{P(E)}{1 - P(E)}$$

The odds of getting heads in a coin toss is

$$\text{odds}(\text{heads}) = \frac{P(\text{heads})}{P(\text{tails})} = \frac{P(\text{heads})}{1 - P(\text{heads})}$$

For a fair coin, this would result in

$$\text{odds}(\text{heads}) = \frac{.5}{1 - .5} = 1$$

Another odds example

The game Lingo has 44 balls: 36 blue, 6 red and 2 green balls

- The odds of a player choosing a blue ball are

$$\text{odds(blue)} = \frac{36}{8} = \frac{36/44}{8/44} = \frac{.8182}{.1818} = 4.5$$

- The odds of a player choosing a red ball are

$$\text{odds(blue)} = \frac{6}{38} = \frac{6/44}{36/44} = \frac{.1364}{.8636} \approx .16$$

- The odds of a player choosing a blue ball are

$$\text{odds(blue)} = \frac{2}{42} = \frac{2/44}{42/44} = \frac{.0455}{.9545} \approx .05$$

Odds of 1 indicate an equal likelihood of the event occuring or not occuring. Odds $<$ 1 indicate a lower likelihood of the event occuring vs. not occuring. Odds $>$ 1 indicate a higher likelihood of the event occuring.

GLM's continued

Remember that

$$y = \alpha + \beta x + \epsilon,$$

and that

$$\mathbb{E}[y] = \alpha + \beta x.$$

As a result

$$y = \mathbb{E}[y] + \epsilon.$$

and residuals do not need to be normal (heck, y probably isn't, so why should ϵ be?)

Logistic regression

Logistic regression is a GLM used to model a **binary categorical variable** using **numerical** and **categorical** predictors.

In logistic regression we assume that the true data generating model for the outcome variable follows a binomial distribution.

- it is therefore intuitive to think of logistic regression as modeling the probability of succes p for any given set of predictors.

How

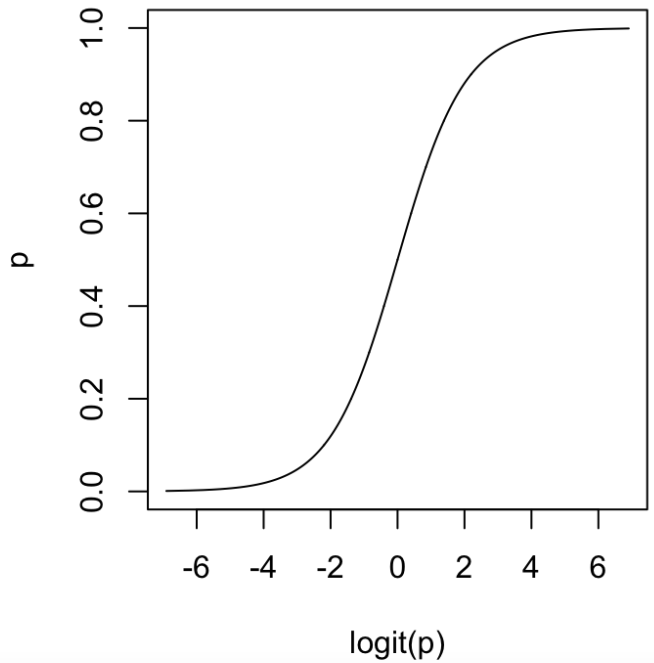
We specify a reasonable link that connects η to p. Most common in logistic regression is the *logit* link

$$logit(p) = \log(\frac{p}{1-p}), \text{ for } 0 \leq p \geq 1$$

We might recognize $\frac{p}{1-p}$ as the odds.

Logit

$$\text{logit}(p) = \log(\frac{p}{1-p}) = \log(p) - \log(1-p)$$



Logit continued

Logit models work on the $\log(\text{odds})$ scale

$$\log(\text{odds}) = \log(\frac{p}{1-p}) = \log(p) - \log(1-p) = \text{logit}(p)$$

The logit of the probability is the log of the odds.

Logistic regression allows us to model the $\log(\text{odds})$ as a function of other, linear predictors.

Interpreting the logit

The logit function takes a value between 0 and 1 and maps it to a value between $-\infty$ and ∞.

Inverse logit (logistic) function

$$g^{-1}(x) = \frac{\exp(x)}{1 + \exp(x)} = \frac{1}{1 + \exp(-x)}$$

The inverse logit function takes a value between $-\infty$ and ∞ and projects it to a value between 0 and 1.

Again, this formulation is quite useful as we can interpret the `logit` as the log odds of a success.

log(odds) explained

Remember our data example?

```
fit <- simulated %$%
  glm(discrete ~ continuous, family = binomial())
```

We can use this model to obtain predictions in the scale of the linear predictor (i.e. the logodds):

```
linpred <- predict(fit, type = "link")
```

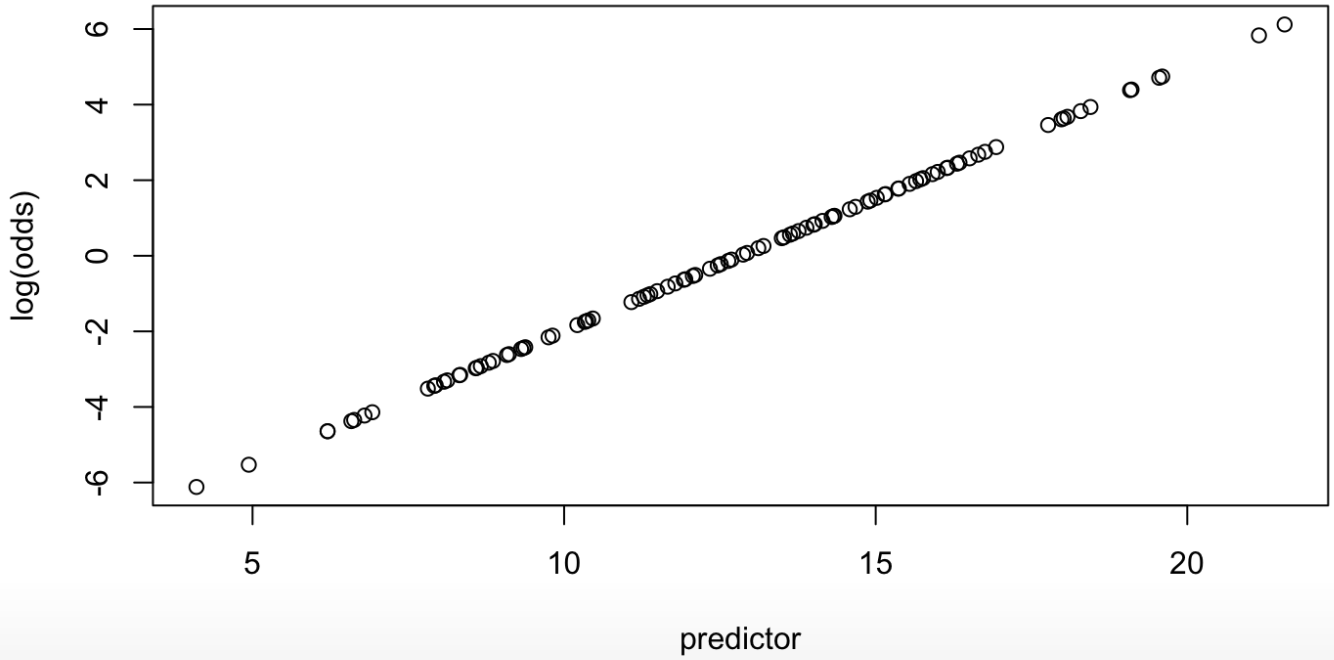
or in the scale of the response (i.e. the probability)

```
response <- predict(fit, type = "response")
```

log(odds) **explained**

Now if we visualize the relation between our predictor and the logodds

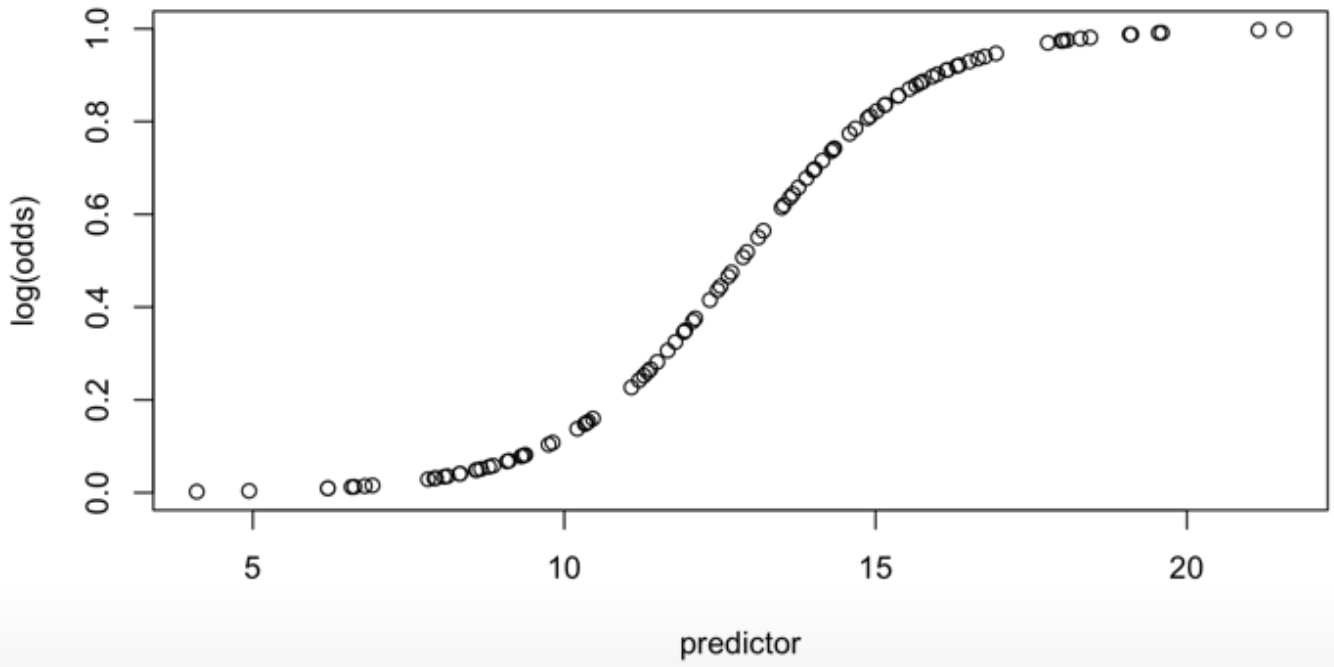
```
plot(simulated$continuous, linpred, xlab = "predictor", ylab = "log(odds)")
```



log(odds) **explained**

And the relation between our predictor and the probability

```
plot(simulated$continuous, response, xlab = "predictor", ylab = "log(odds)")
```



logit^{-1} explained

The inverse of the logit brings us back to the probability

```
invlogit <- exp(linpred) / (1 + exp(linpred))
invlogit %>% sort() %>% head()
```

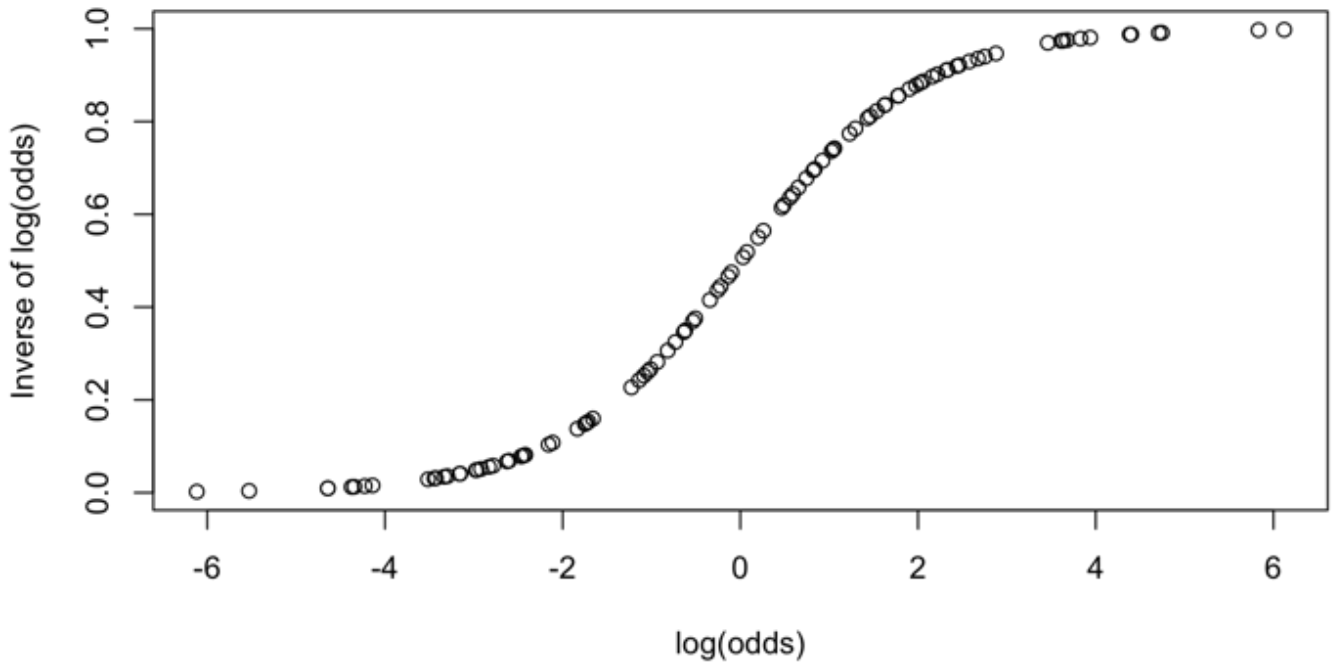
```
##          18          26          43           8          29          46
## 0.002200057 0.003956052 0.009538921 0.009545580 0.012428777 0.012822635
```

```
invlogit %>% sort() %>% tail()
```

```
##        95        54        56        98        70        97
## 0.9876591 0.9878611 0.9910757 0.9913709 0.9970761 0.9978074
```

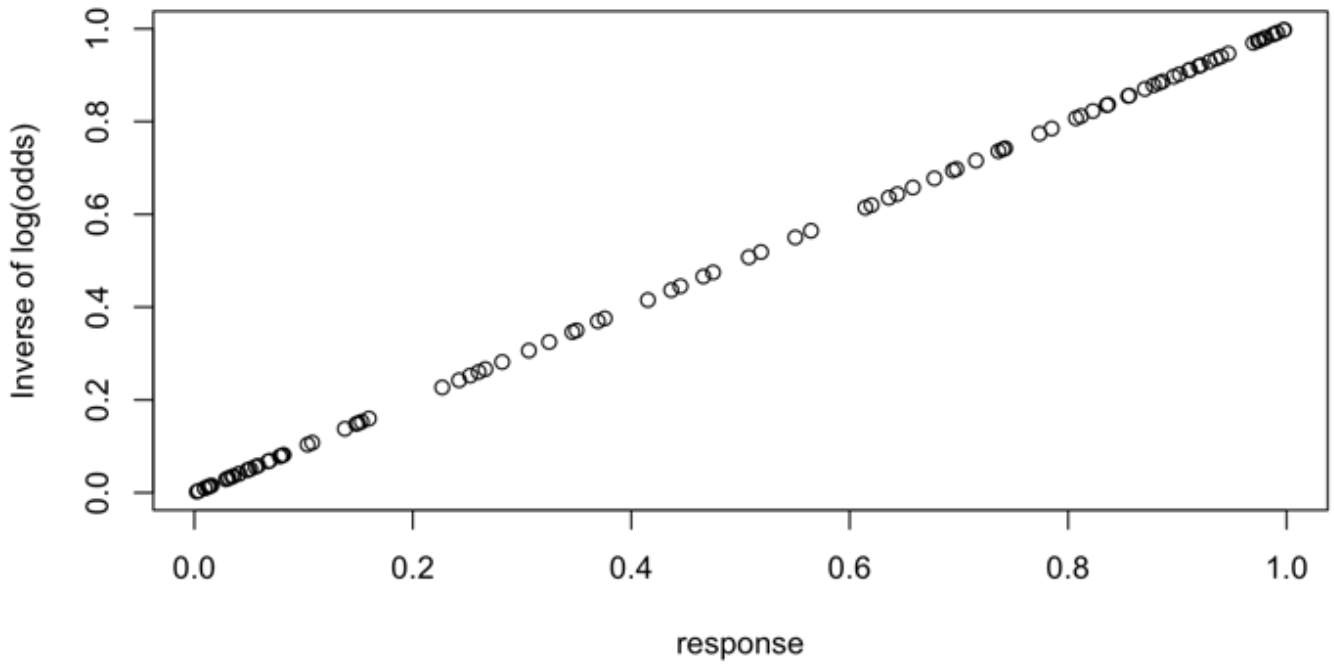
logit^{-1} explained

```
plot(linpred, invlogit, xlab = "log(odds)", ylab = "Inverse of log(odds)")
```



logit^{-1} explained

```
plot(response, invlogit, xlab = "response", ylab = "Inverse of log(odds)")
```



Logistic regression

Logistic regression

With linear regression we had the `Sum of Squares (SS)`. Its logistic counterpart is the `Deviance (D)`.

- Deviance is the fit of the observed values to the expected values.

With logistic regression we aim to maximize the `likelihood`, which is equivalent to minimizing the deviance.

The likelihood is the (joint) probability of the observed values, given the current model parameters.

In normally distributed data: $\mathrm{SS} = \mathrm{D}$.

The logistic regression model

Remember the three characteristics for every generalized linear model:

1. A probability distribution that describes the outcome
2. A linear predictor model in the form
3. A link function that relates the linear predictor to the the parameter of the outcome's probability distribution.

For the logistic model this gives us:

1. $y_i \sim \text{Binom}(p_i)$
2. $\eta = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n$
3. $\text{logit}(p) = \eta$

Simple substitution brings us at

$$p_i = \frac{\exp(\eta)}{1 + \exp(\eta)} = \frac{\exp(\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_n x_{n,i})}{1 + \exp(\beta_0 + \beta_1 x_{1,i} + \cdots + \beta_n x_{n,i})}$$

Fitting a logistic regression

The `anesthetic` data

```
anesthetic %>% head(n = 10)
```

```
##    move conc    logconc nomove
## 1     0  1.0  0.0000000      1
## 2     1  1.2  0.1823216      0
## 3     0  1.4  0.3364722      1
## 4     1  1.4  0.3364722      0
## 5     1  1.2  0.1823216      0
## 6     0  2.5  0.9162907      1
## 7     0  1.6  0.4700036      1
## 8     1  0.8 -0.2231436      0
## 9     0  1.6  0.4700036      1
## 10    1  1.4  0.3364722      0
```

Thirty patients were given an anesthetic agent maintained at a predetermined level (`conc`) for 15 minutes before making an incision. It was then noted whether the patient moved, i.e. jerked or twisted.

Fitting a logistic regression model

Fitting a `glm` in `R` is not much different from fitting a `lm`. We do, however, need to specify what type of `glm` to use by specifying both the `family` and the type of `link` function we need.

For logistic regression we need the **binomial** family as the binomial distribution is the probability distribution that describes our outcome. We also use the `logit` link, which is the default for the binomial `glm` family.

```
fit <- anesthetic %$%
  glm(nomove ~ conc, family = binomial(link="logit"))
fit
```

```
##
## Call:  glm(formula = nomove ~ conc, family = binomial(link = "logit"))
##
## Coefficients:
## (Intercept)         conc
##      -6.469        5.567
##
## Degrees of Freedom: 29 Total (i.e. Null);  28 Residual
## Null Deviance:       41.46
## Residual Deviance: 27.75     AIC: 31.75
```

The model parameters

```
fit %>% summary
```

```
##
## Call:
## glm(formula = nomove ~ conc, family = binomial(link = "logit"))
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.76666  -0.74407   0.03413   0.68666   2.06900
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)   -6.469      2.418  -2.675  0.00748 **
## conc           5.567      2.044   2.724  0.00645 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 41.455  on 29  degrees of freedom
## Residual deviance: 27.754  on 28  degrees of freedom
## AIC: 31.754
##
## Number of Fisher Scoring iterations: 5
```

The regression parameters

```
fit %>% summary %>% .$coefficients
```

```
##              Estimate Std. Error   z value    Pr(>|z|)
## (Intercept) -6.468675   2.418470 -2.674697 0.007479691
## conc         5.566762   2.043591  2.724010 0.006449448
```

With every unit increase in concentration `conc`, the log odds of **not moving** increases with 5.5667617. This increase can be considered different from zero as the p-value is 0.0064494.

In other words; an increase in `conc` will lower the probability of moving. We can verify this by modeling `move` instead of `nomove`:

```
anesthetic %$%
  glm(move ~ conc, family = binomial(link="logit")) %>%
  summary %>% .$coefficients
```

```
##              Estimate Std. Error   z value    Pr(>|z|)
## (Intercept)  6.468675   2.418470  2.674697 0.007479691
## conc        -5.566762   2.043591 -2.724010 0.006449448
```

A different approach

```
anestot <- aggregate(anesthetic[, c("move","nomove")],
                     by = list(conc = anesthetic$conc), FUN = sum)
anestot
```

```
##   conc move nomove
## 1  0.8    6      1
## 2  1.0    4      1
## 3  1.2    2      4
## 4  1.4    2      4
## 5  1.6    0      4
## 6  2.5    0      2
```

A different approach

We can summarize the same information in a frequency table

```
anestot$total <- apply(anestot[, c("move","nomove")], 1 , sum)
anestot
```

```
##   conc move nomove total
## 1  0.8    6      1     7
## 2  1.0    4      1     5
## 3  1.2    2      4     6
## 4  1.4    2      4     6
## 5  1.6    0      4     4
## 6  2.5    0      2     2
```

We can add the proportion to this table

```
anestot$prop <- anestot$nomove / anestot$total
anestot %>% head(n = 3)
```

```
##   conc move nomove total      prop
## 1  0.8    6      1     7 0.1428571
## 2  1.0    4      1     5 0.2000000
## 3  1.2    2      4     6 0.6666667
```

A different approach

We can then calculate the same model on the frequency table, by specifying the number of times each scenario (row) is observed (`total`) as the `weights` argument in `glm`:

```
anestot %$% glm(prop ~ conc, family = binomial(link="logit"), weights = total) %>%
  summary() %>% .$coefficients
```

```
##             Estimate Std. Error   z value    Pr(>|z|)
## (Intercept) -6.468675   2.418537 -2.674624 0.007481321
## conc         5.566762   2.043649  2.723932 0.006450977
```

We now model the proportion. Naturally, the proportion multiplied with the total equals the observation. So in this case, the `glm` function performs these calculations and its expansion to cases internally.

Next week

In the next lecture we'll dive more into the ins and outs of interpreting the logistic regression output.