SMI606: Week 9
Multiple Logistic Regression

Dr. Calum Webb
Sheffield Methods Institute, the University of Sheffield.
c.j.webb@sheffield.ac.uk

Sign In

Learning Objectives

What will I learn? | How does this week fit into my course?

By the end of this week you will:

- Be able to identify research questions suited to logistic regression.
- Be able to describe (at a basic level) how logistic regression works and what it can be used for.
- Be able to estimate logistic regression models in `R` using the `glm()` function.
- Be able to check the fit of logistic regression models in `R` using pseudo- R^2 statistics, model accuracy, and k-fold cross-validation.
- Be able to check whether your logistic regression model violates any assumptions of general linear regression.

Learning Objectives

What will I learn? | How does this week fit into my course?

- Logistic regression opens up a world of possible research questions that couldn't be adequately explored with a linear regression model.
- Your knowledge of model assumptions in general linear models will now be complete, enabling you to conduct quantitative research robustly.

Week 9: Logistic Regression — Part I

What is logistic regression and why learn it?

Multiple linear regression is great... But what if the thing we're interested in isn't on a continuous scale?

(Let alone one with nice normally distributed residuals...)

Wooclap activity

What things are you interested in researching that aren't on a continuous scale?

Join the wooclap activity by clicking this link, or scanning the QR code on the right, and try and contribute at least one variable that you would be interested in analysing but either can't be measured on, or you haven't seen measured on, a continuous scale.

Event ID: **AXUMRY**

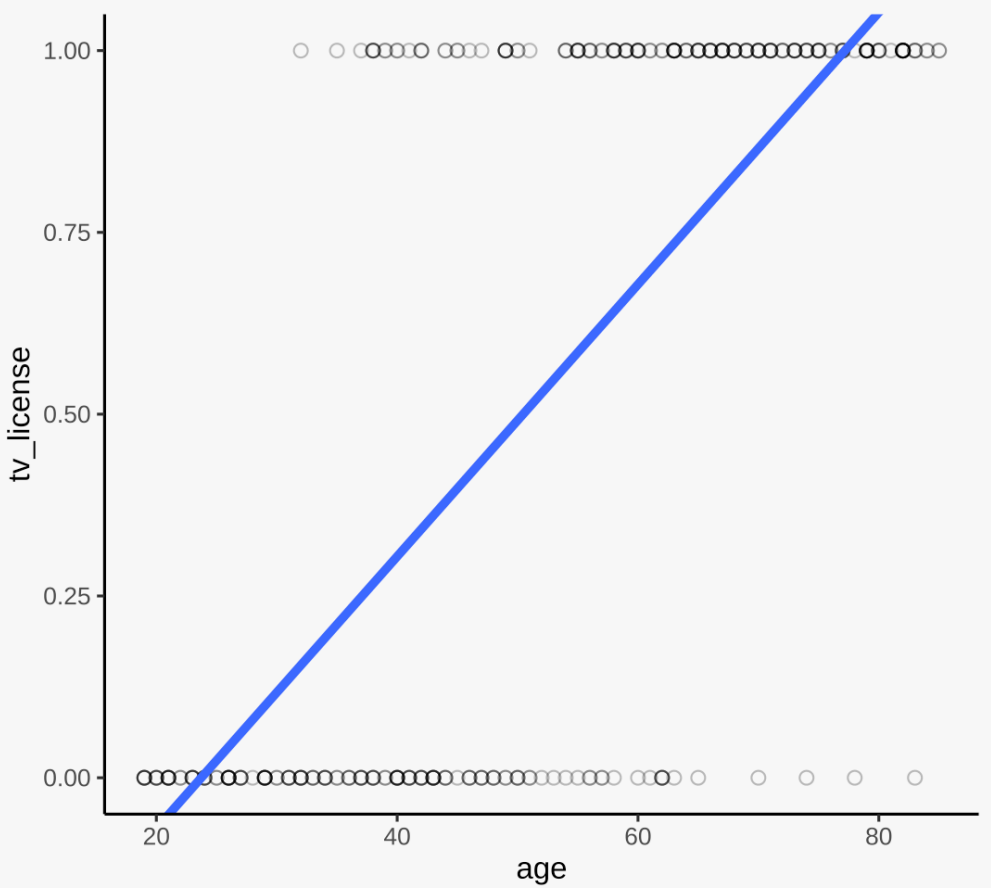
Predicting binary outcomes

We can use linear regression to predict binary outcomes. This is called an LPM — a **linear probability model** (not to be confused with a linear panel model!)

```
plm_mod <- lm(data = tv_licenses, formula = tv_license ~ age)

summary(plm_mod)
```

```
##
## Call:
## lm(formula = tv_license ~ age, data = tv_licenses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10876 -0.21049 -0.00056  0.21406  0.84565
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.444497   0.059934  -7.416 1.26e-12 ***
## age          0.018714   0.001082  17.295  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3538 on 298 degrees of freedom
## Multiple R-squared:  0.5009,    Adjusted R-squared:  0.4993
## F-statistic: 299.1 on 1 and 298 DF,  p-value: < 2.2e-16
```



Every **1 year increase in age** is associated with a **1.87 percentage point increase** in the probability of owning a TV license.

Predicting binary outcomes

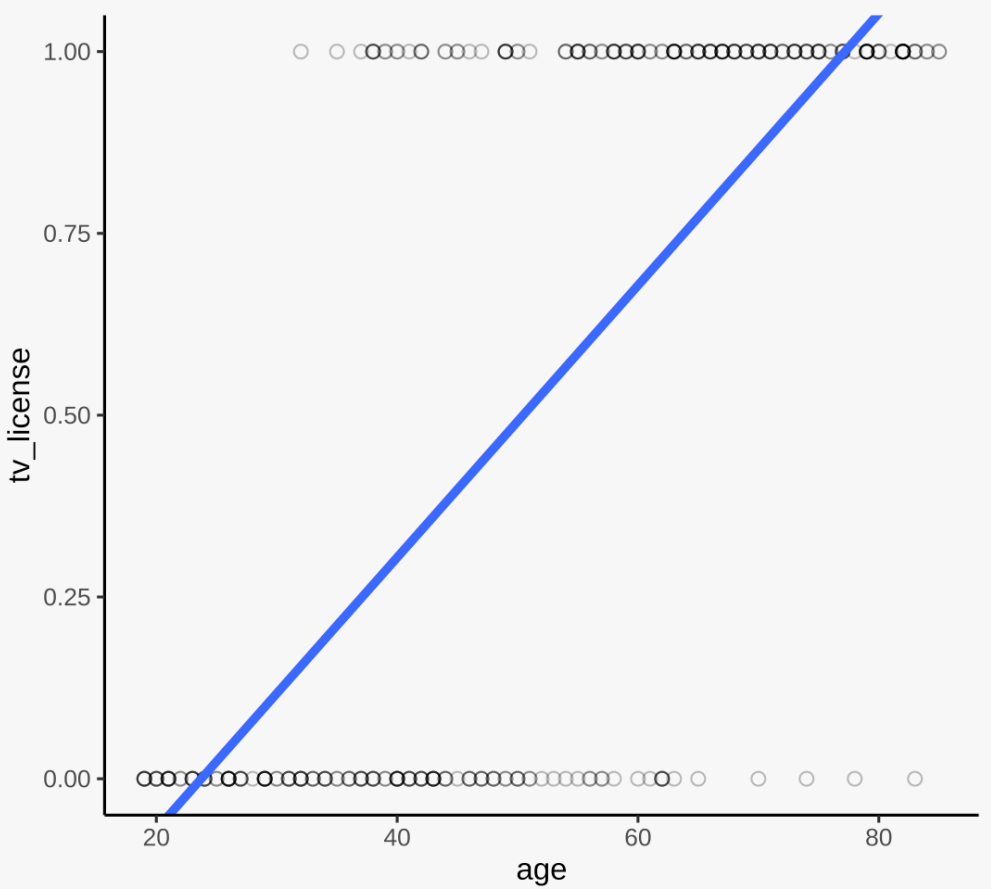
```
summary(plm_mod)
```

```
##
## Call:
## lm(formula = tv_license ~ age, data = tv_licenses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10876 -0.21049 -0.00056  0.21406  0.84565
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.444497   0.059934  -7.416 1.26e-12 ***
## age          0.018714   0.001082  17.295  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3538 on 298 degrees of freedom
## Multiple R-squared:  0.5009,    Adjusted R-squared:  0.4993
## F-statistic: 299.1 on 1 and 298 DF,  p-value: < 2.2e-16
```

But that means that if you are 85 your probability of owning a TV license would be...

$$\text{tv\_license} = -0.44 + 0.0187 * 85 = 1.1495$$

114.95%??



Predicting binary outcomes

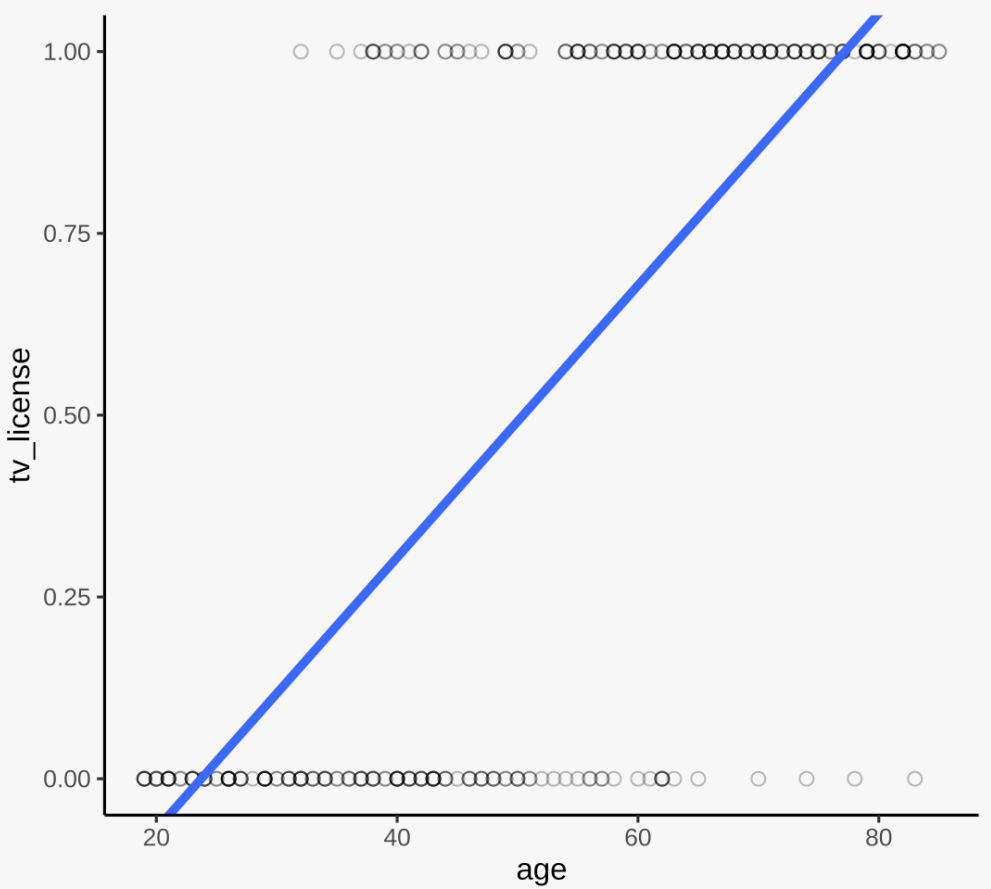
```
summary(plm_mod)
```

```
##
## Call:
## lm(formula = tv_license ~ age, data = tv_licenses)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.10876 -0.21049 -0.00056  0.21406  0.84565
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.444497   0.059934  -7.416 1.26e-12 ***
## age          0.018714   0.001082  17.295  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3538 on 298 degrees of freedom
## Multiple R-squared:  0.5009,    Adjusted R-squared:  0.4993
## F-statistic: 299.1 on 1 and 298 DF,  p-value: < 2.2e-16
```

And if you were 20 years old your predicted probability of owning a TV license would be...

$$\text{tv\_license} = -0.44 + 0.0187 * 20 = -0.066$$

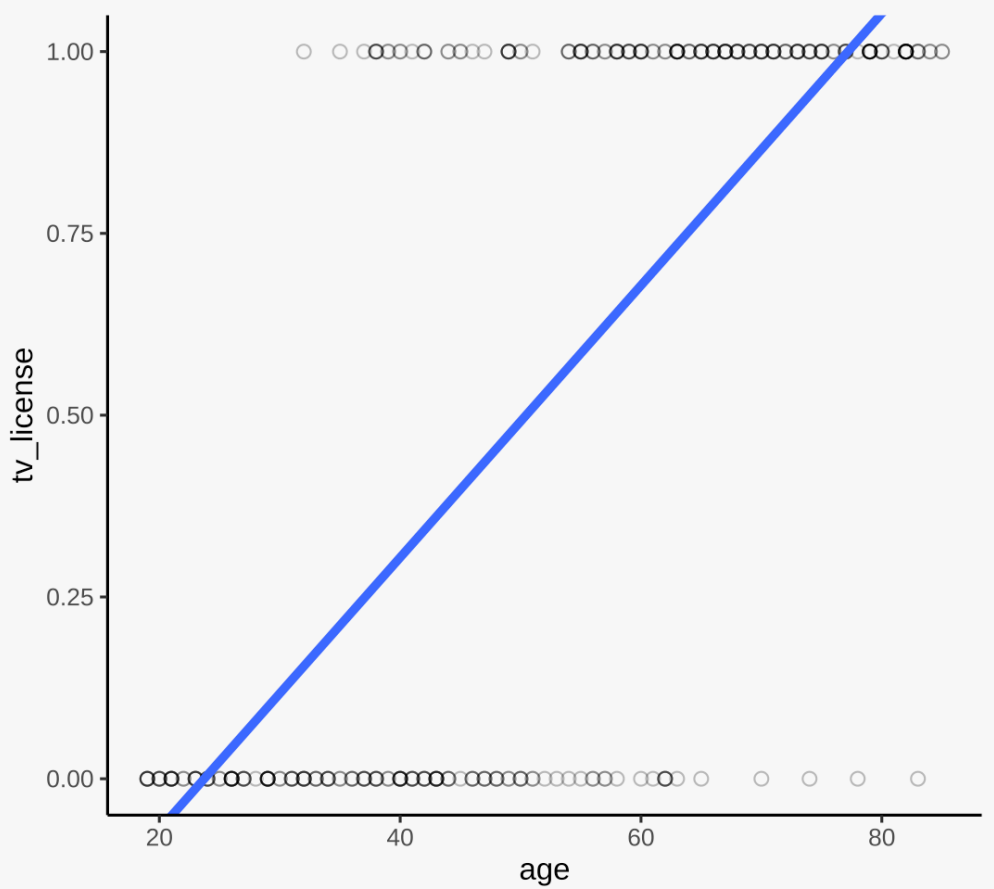
-6.6%??



Predicting binary outcomes

When we have binary outcomes to predict what we ideally want to do is constrain out model between either 1 (the thing happened, the person is in the group, etc.) and 0 (the thing didn't happen, the person wasn't in the group, etc.).

For this, we can use **logistic regression**.

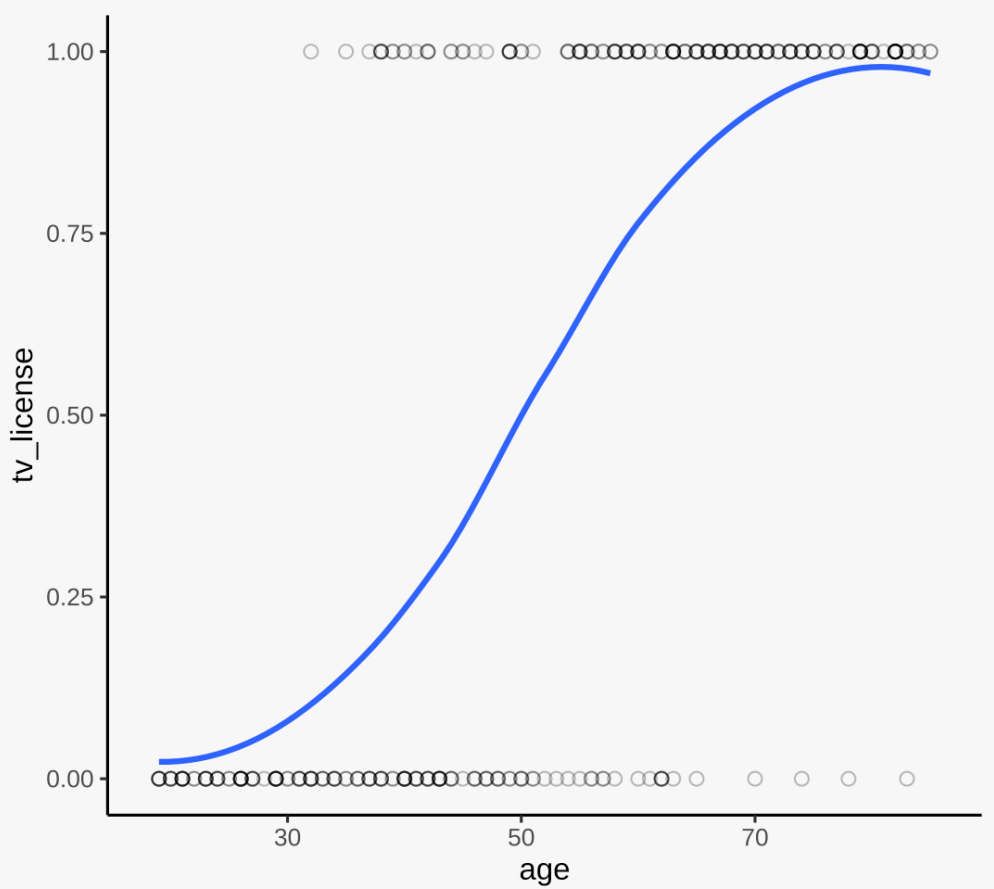


Predicting binary outcomes

When we have binary outcomes to predict what we ideally want to do is constrain out model between either 1 (the thing happened, the person is in the group, etc.) and 0 (the thing didn't happen, the person wasn't in the group, etc.).

For this, we can use **logistic regression**.

...Much better!



Week 9: Logistic Regression — Part II

How does logistic regression work?

Logistic regression

How does this magic work?

Unlike multiple linear regression (which usually uses an ordinary least squares estimator), logistic regression uses **Maximum Likelihood** estimation to calculate the best fitting probabilities for each data point.

Let's talk about what the ML estimator is doing...

Logistic regression

Logistic regression makes use of three things:

Odds

What are the odds of a thing happening?

If 150 people own a TV license for every 100 people who don't own a TV license, the odds of someone owning a TV license are 3:2 (usually just written as 1.5)

$$Odds|A = A/B$$

Log Odds

The odds can be transformed to their natural log to create 'log odds'

$$logodds = log(Odds)$$

This is very useful when combined with the logistic sigmoid function.

Logistic sigmoid function

The logistic sigmoid function is a function that returns a value that is always between 0 and 1. It can be defined as:

$$\frac{e^x}{(1+e^x)}$$

When x is our log odds of Y being equal to 1 the logistic sigmoid function returns a **predicted probability**.

Logistic regression

Example

150 people own TV licenses and 100 people do not. The **odds** of owning a TV license are equal to $150/100 = 1.5$.

Logistic regression

Example

150 people own TV licenses and 100 people do not. The **odds** of owning a TV license are equal to $150/100 = 1.5$.

The log odds of owning a TV license are equal to $log(1.5) = 0.4054651$

Logistic regression

Example

150 people own TV licenses and 100 people do not. The **odds** of owning a TV license are equal to $150/100 = 1.5$.

The log odds of owning a TV license are equal to $log(1.5) = 0.4054651$

The predicted probability of owning a TV license can be calculated using the logistic sigmoid function as:

$$\frac{e^{0.4054651}}{1 + e^{(0.4054651)}}$$

Or in `R` code: `exp(0.4054651) / (1 + exp(0.4054651)) = 0.6 = 60%`

Logistic regression

Example

150 people own TV licenses and 100 people do not. The **odds** of owning a TV license are equal to $150/100 = 1.5$.

The log odds of owning a TV license are equal to $log(1.5) = 0.4054651$

The predicted probability of owning a TV license can be calculated using the logistic sigmoid function as:

$$\frac{e^{0.4054651}}{1 + e^{(0.4054651)}}$$

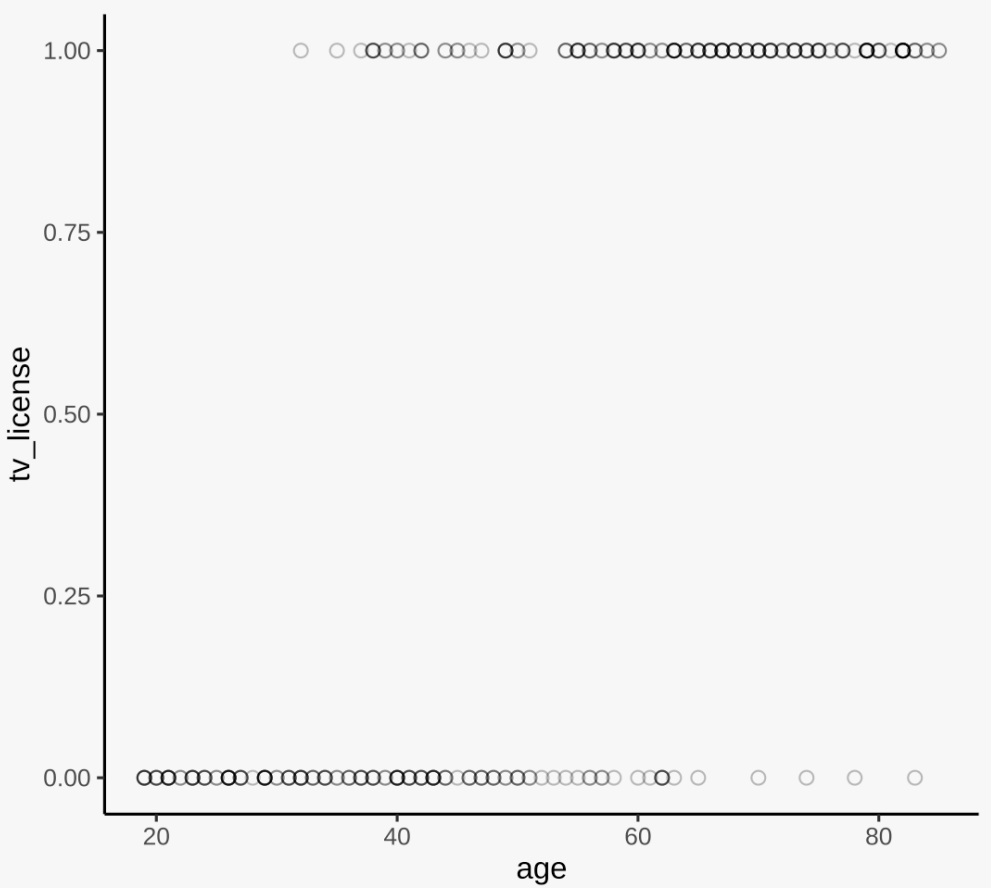
Or in `R` code: `exp(0.4054651) / (1 + exp(0.4054651)) = 0.6 = 60%`

This means that: if we know the log odds of something we can calculate its predicted probability. The maximum likelihood estimator takes advantage of the properties of log odds to model predicted probabilities that Y = 1 conditional on values of X (**Y=1|X**).

Logistic regression

Visual Explanation

Let's start off with our raw data. Along the top we have the people who do own a TV license (1), along the bottom we have people who do not own a TV license (0).

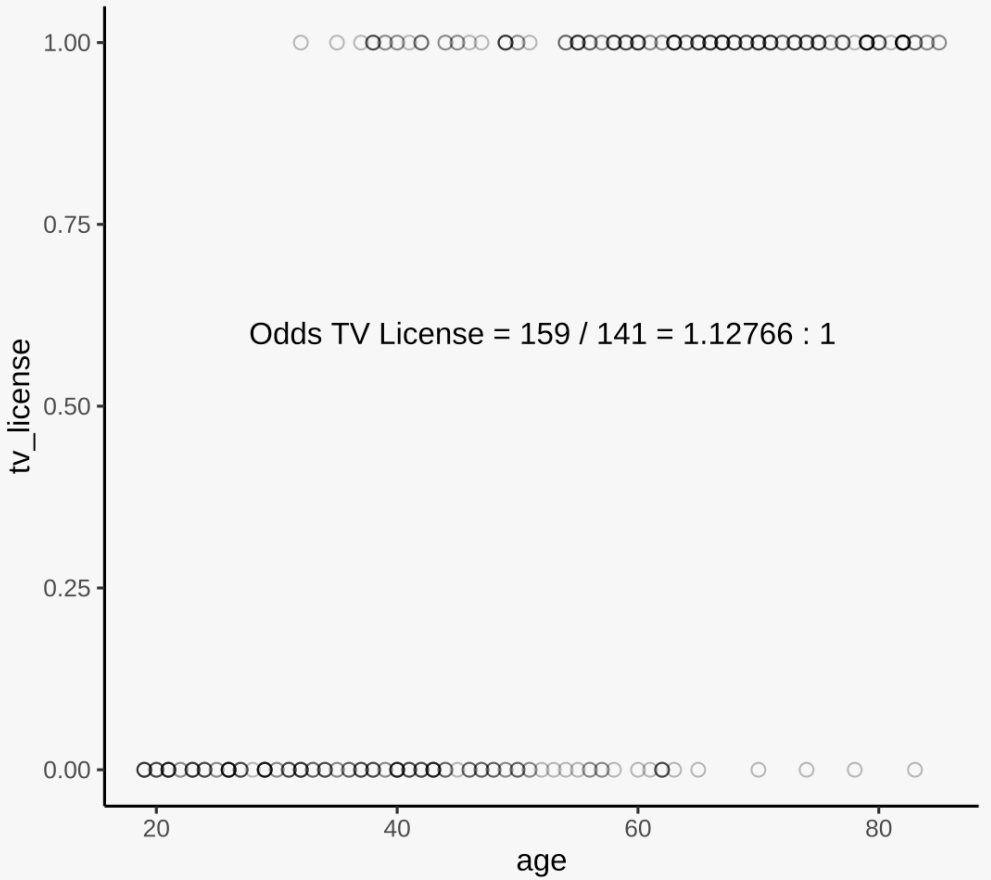


Logistic regression

Visual Explanation

Let's start off with our raw data. Along the top we have the people who do own a TV license (1), along the bottom we have people who do not own a TV license (0).

Maximum Likelihood is an iterative method, so let's start our first set of predictions at the general odds of owning a TV license (1.127 to 1) before we start to adjust the odds by age.



Logistic regression

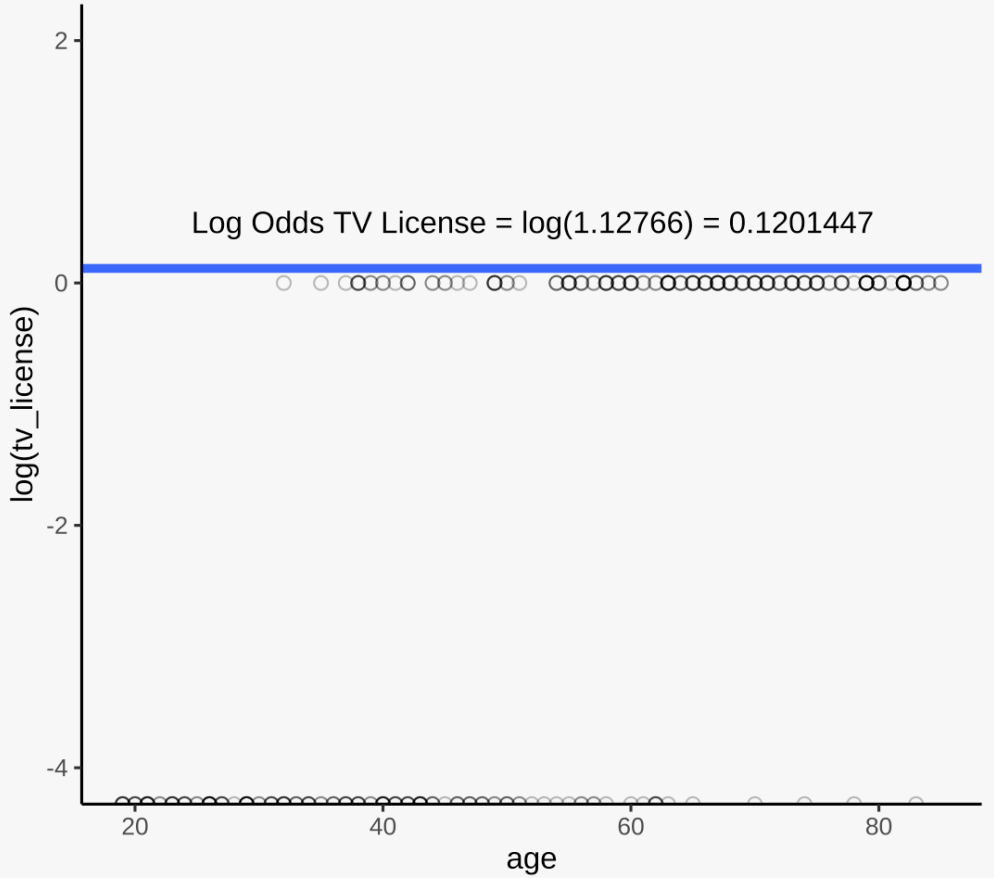
Visual Explanation

Maximum Likelihood is an iterative method, so let's start our first set of predictions at the general odds of owning a TV license (1.127 to 1) before we start to adjust the odds by age.

We can then convert our odds to log odds. The log odds equal 0.1201447.

We also log our real data

- log(1) = 0
- log(0) = -Infinity



Logistic regression

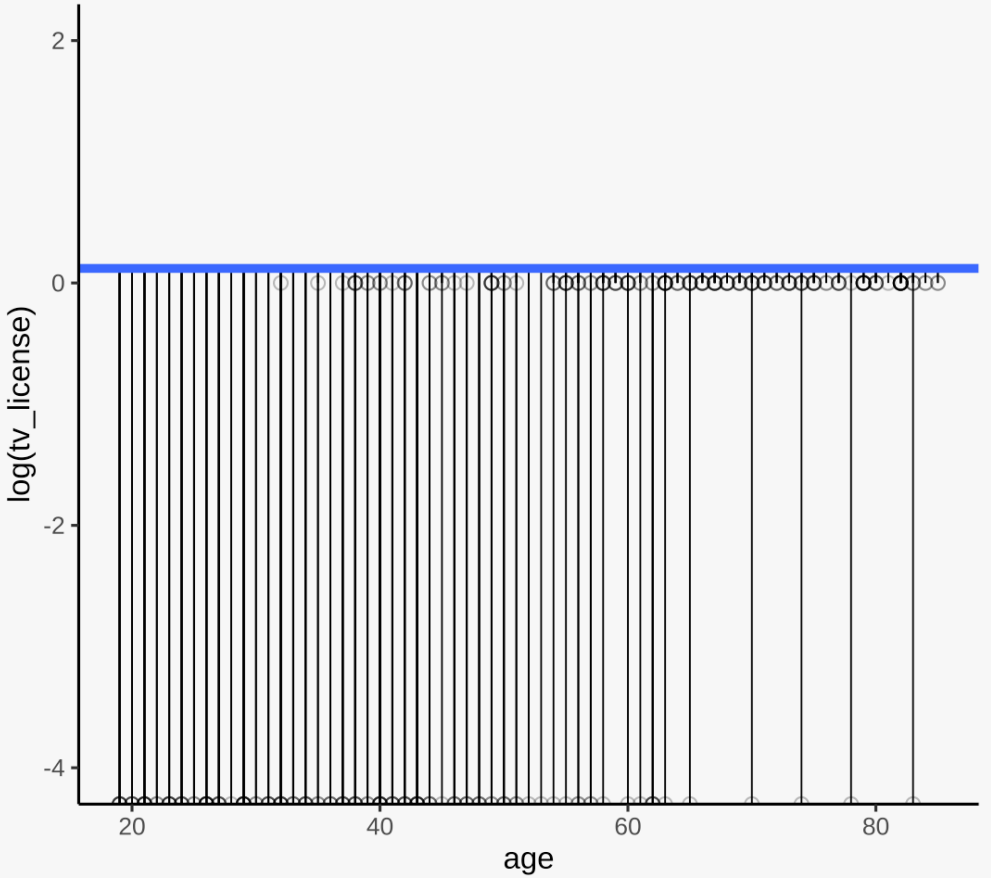
Visual Explanation

We can then convert our odds to log odds. The log odds equal 0.1201447.

We also log our real data

- log(1) = 0
- log(0) = -Infinity

We cannot really use negative infinity for anything, so we map our data points to the closest position on the line.

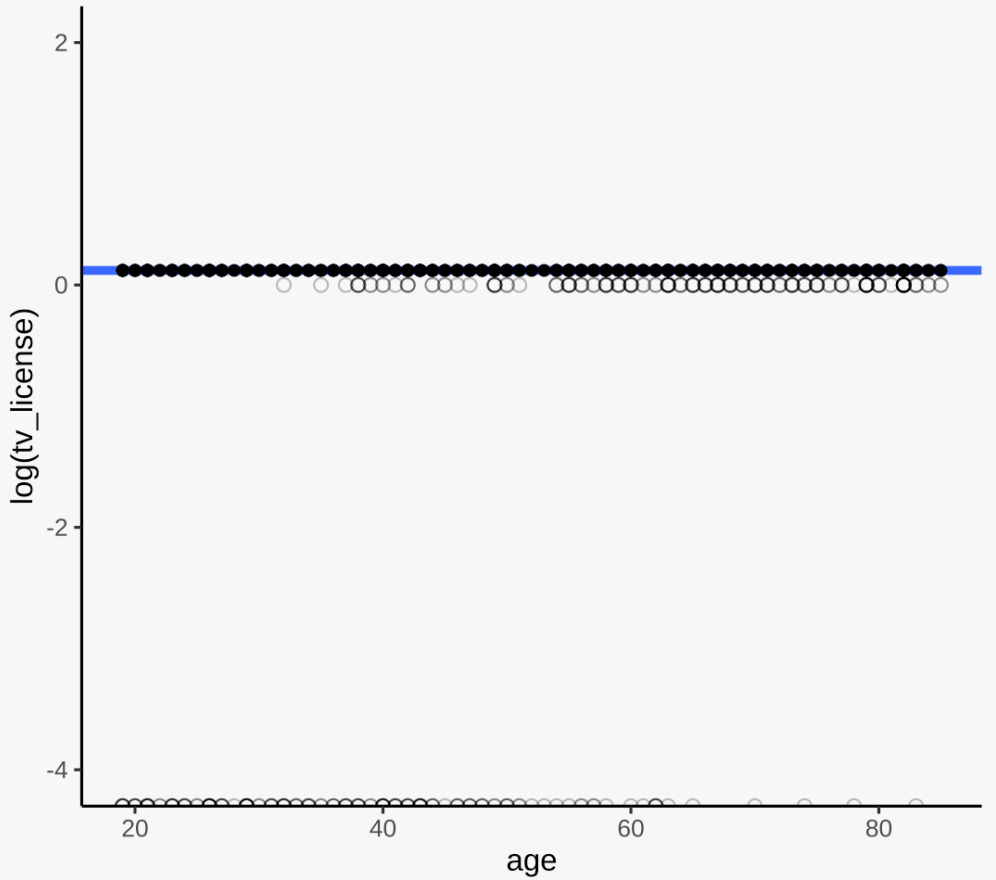


Logistic regression

Visual Explanation

We cannot really use negative infinity for anything, so we map our data points to the closest position on the line.

Now we've assigned each data point to its closest position on a line, we can give it a new value based on the position of the line. We can then transform these new log odds values back into a predicted probability using the logistic sigmoid function.



Logistic regression

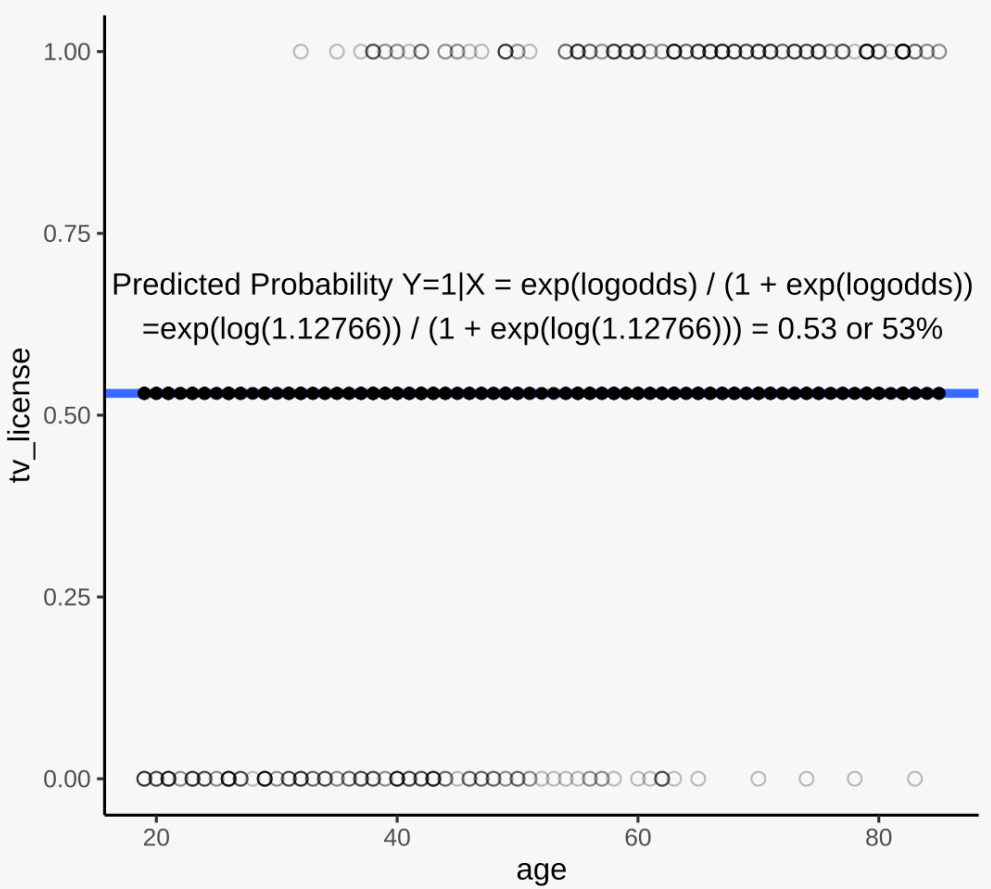
Visual Explanation

Now we've assigned each data point to its closest position on a line, we can give it a new value based on the position of the line. We can then transform these new log odds values back into a predicted probability using the logistic sigmoid function.

We can use the logistic sigmoid function below to calculate the predicted probability for each point on the line. Here, every point has the same value, so the calculation is relatively simple:

$$P(Y = 1|X) = \frac{e^x}{(1 + e^x)} = \frac{e^{0.1201447 + 0 * age}}{(1 + e^{0.1201447 + 0 * age})}$$

Doesn't something look familiar about part of our x?



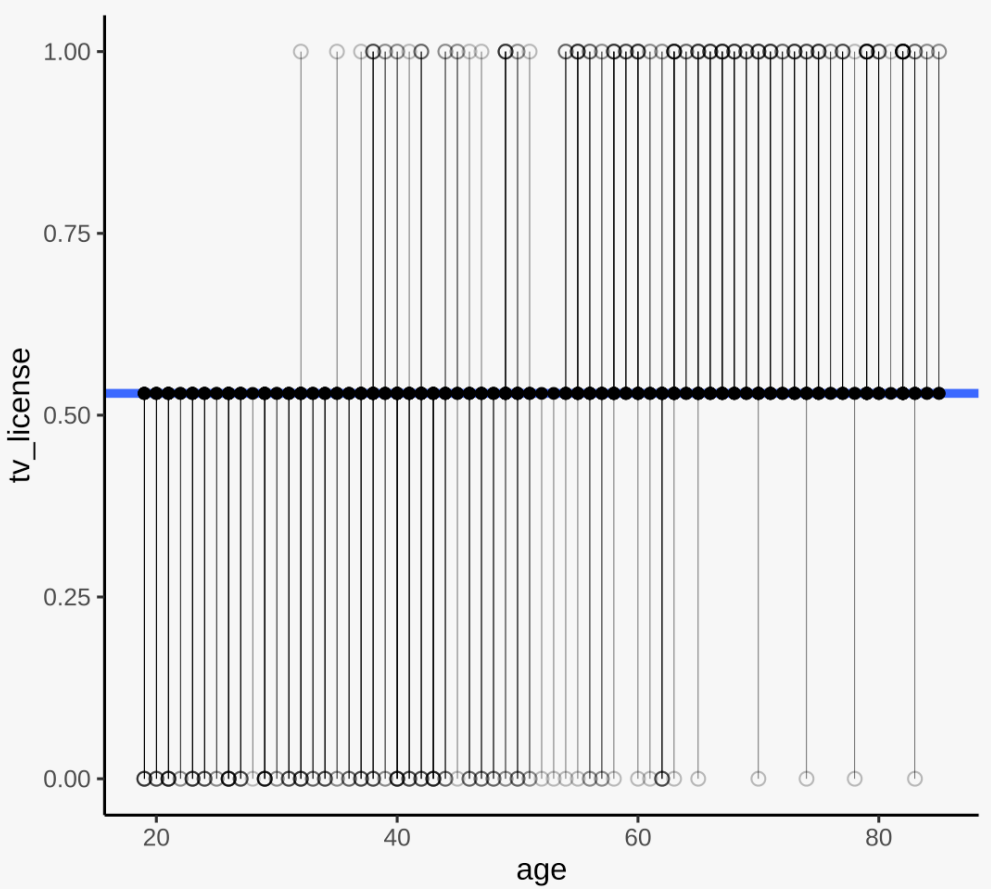
Logistic regression

Visual Explanation

$$P(Y = 1|X) = \frac{e^x}{(1 + e^x)} = \frac{e^{0.1201447+0*age}}{(1 + e^{0.1201447+0*age})}$$

Doesn't something look familiar about part of our x?

We can then get a picture of how good our predictions were from the real values, similar to how we use residuals in ordinary least squares — these ones are not looking very good, are they?

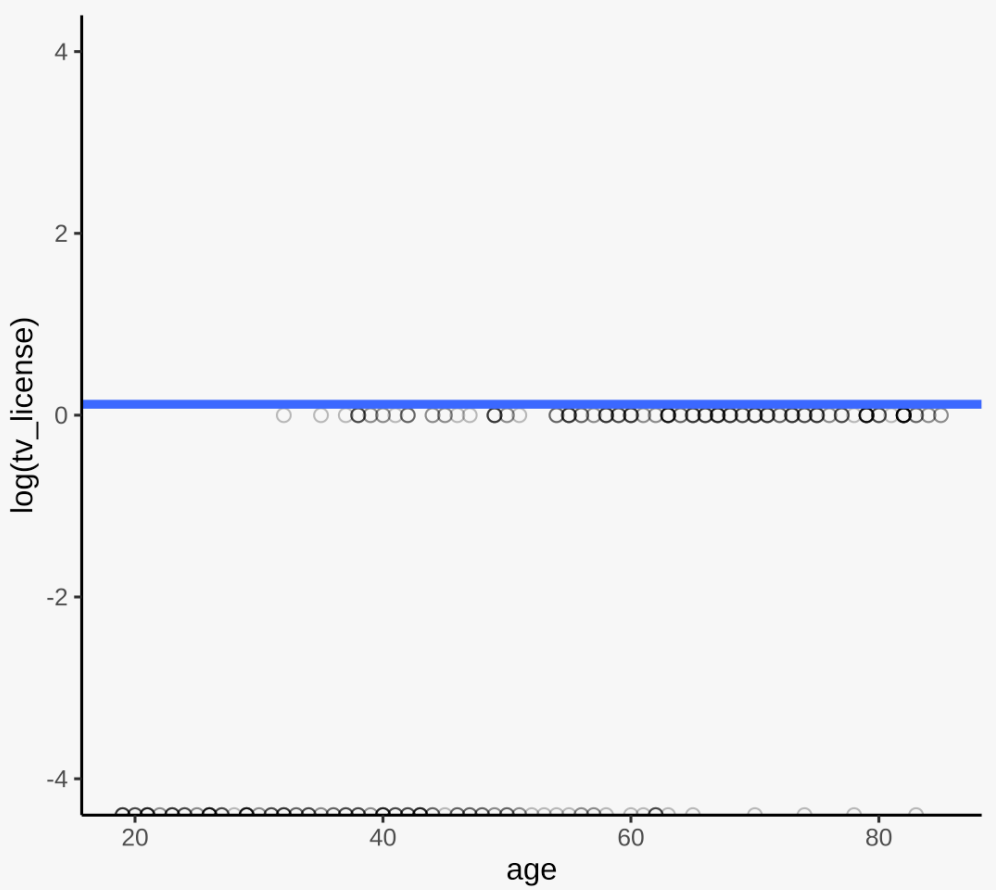


Logistic regression

Visual Explanation

We can then get a picture of how good our predictions were from the real values, similar to how we use residuals in ordinary least squares — these ones are not looking very good, are they?

Let's go back for another iteration — this time, **let's try giving the line a positive slope**.

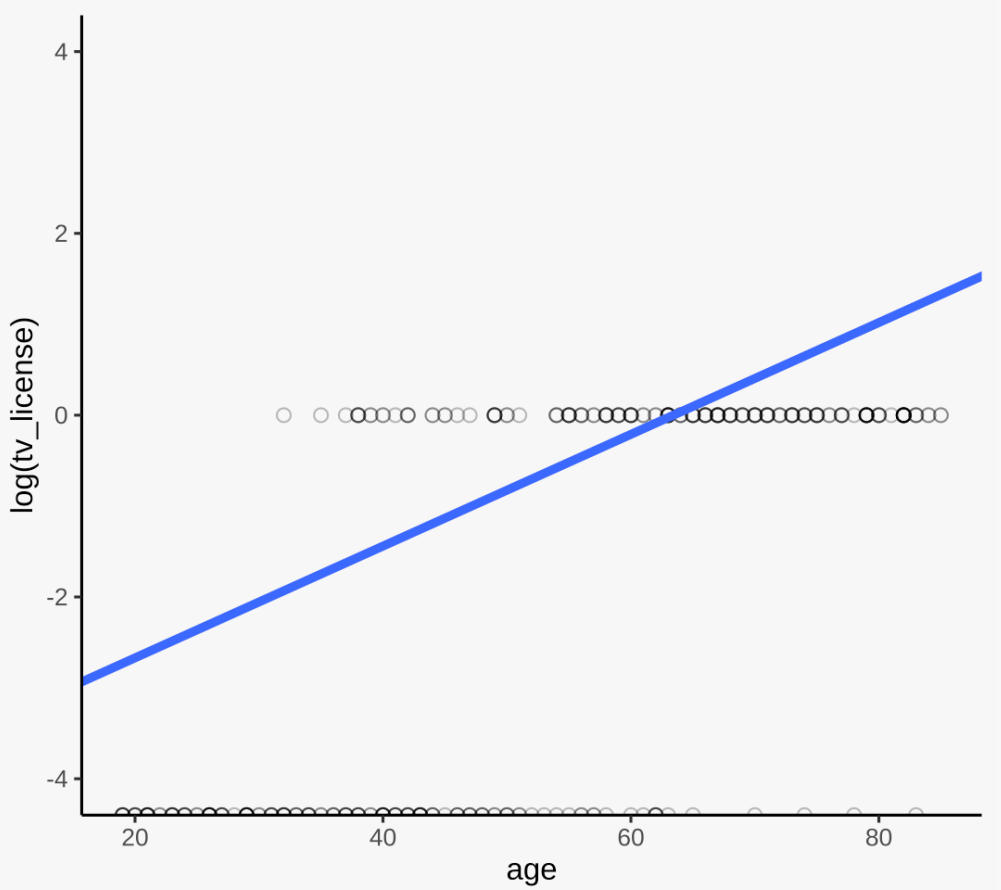


Logistic regression

Visual Explanation

Let's go back for another iteration — this time, **let's try giving the line a positive slope**.

Then we can repeat the process.



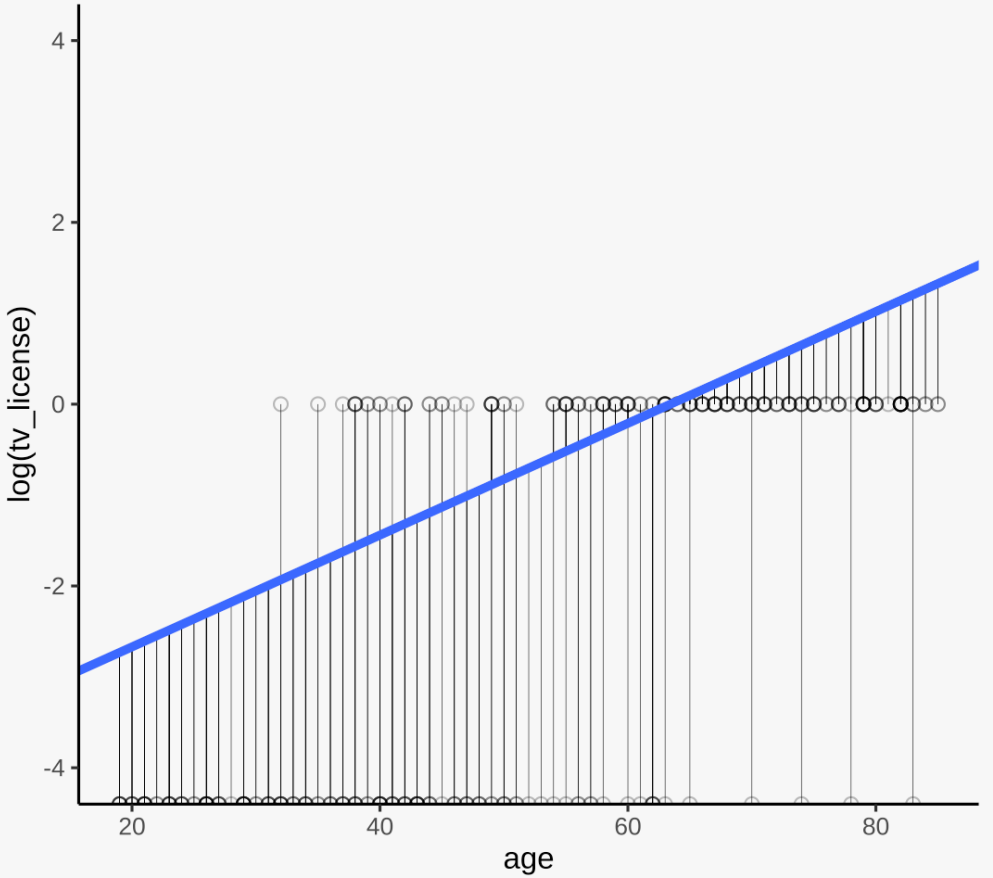
Logistic regression

Visual Explanation

Let's go back for another iteration — this time, **let's try giving the line a positive slope**.

Then we can repeat the process.

Let's map our negative infinity and 0 value data points back to their closest position on the log odds regression line.



Logistic regression

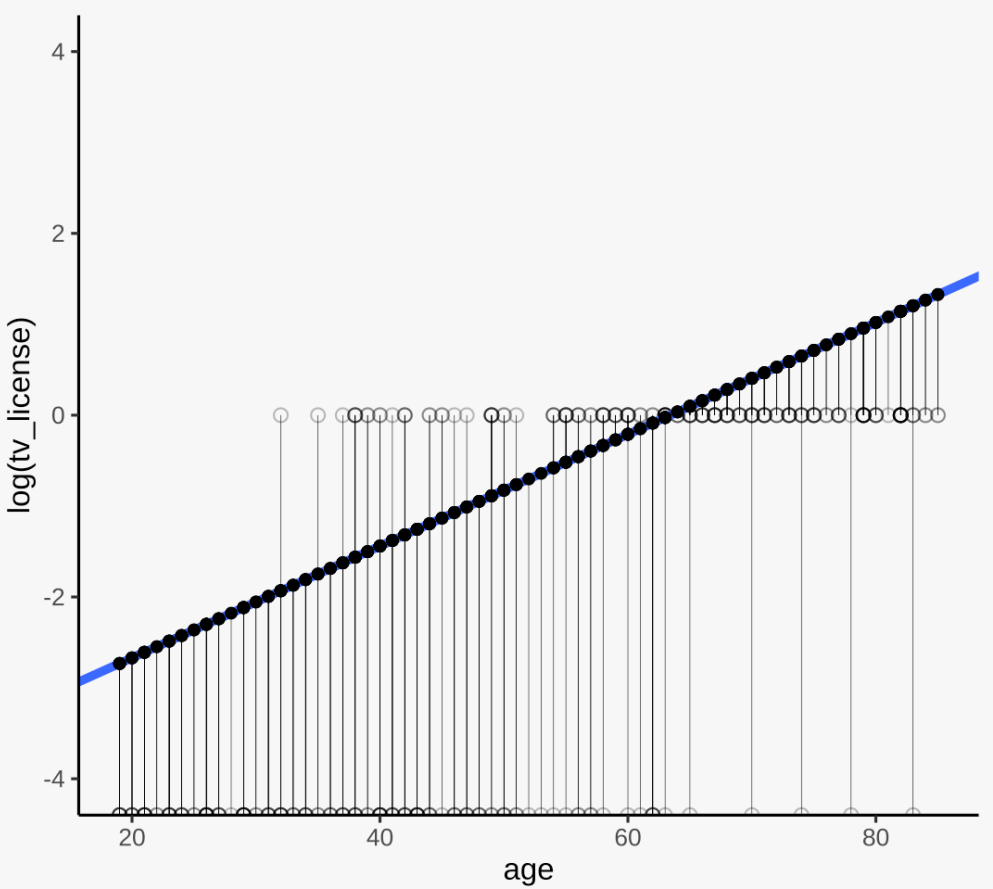
Visual Explanation

Let's go back for another iteration — this time, **let's try giving the line a positive slope**.

Then we can repeat the process.

Let's map our negative infinity and 0 value data points back to their closest position on the log odds regression line.

Then we just need to convert these new values back into predictions using the logistic sigmoid function.



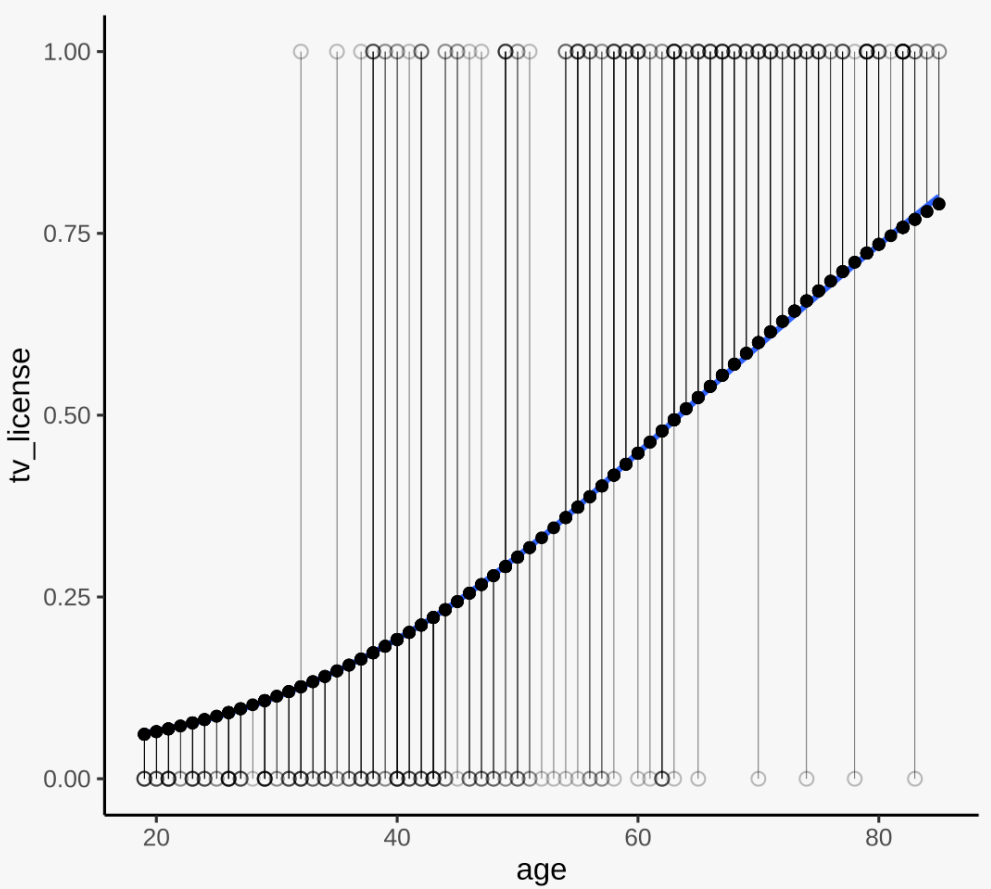
Logistic regression

Visual Explanation

Then we just need to convert these new values back into predictions using the logistic sigmoid function.

Our logistic sigmoid function is now:

$$P(Y = 1|X) = \frac{e^x}{(1 + e^x)} = \frac{e^{-3.9+0.0615*age}}{(1 + e^{-3.9+0.0615*age})}$$



Logistic regression

Visual Explanation

Our logistic sigmoid function is now:

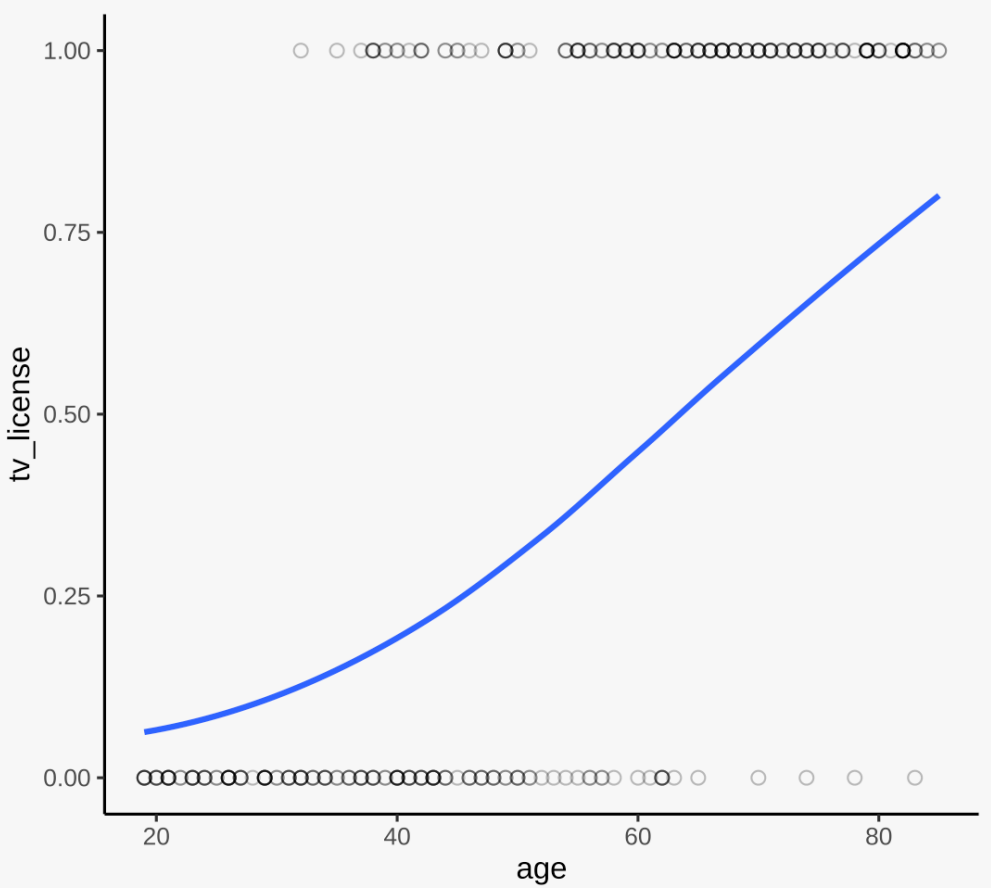
$$P(Y = 1|X) = \frac{e^x}{(1 + e^x)} = \frac{e^{-3.9+0.0615*age}}{(1 + e^{-3.9+0.0615*age})}$$

If we wanted to, we could use this to calculate the predicted probability of any given age. For example:

The predicted probability of owning a TV license when age = 80 is:

$$\frac{e^{-3.9+0.0615*80}}{(1 + e^{-3.9+0.0615*80})} = 0.7349$$

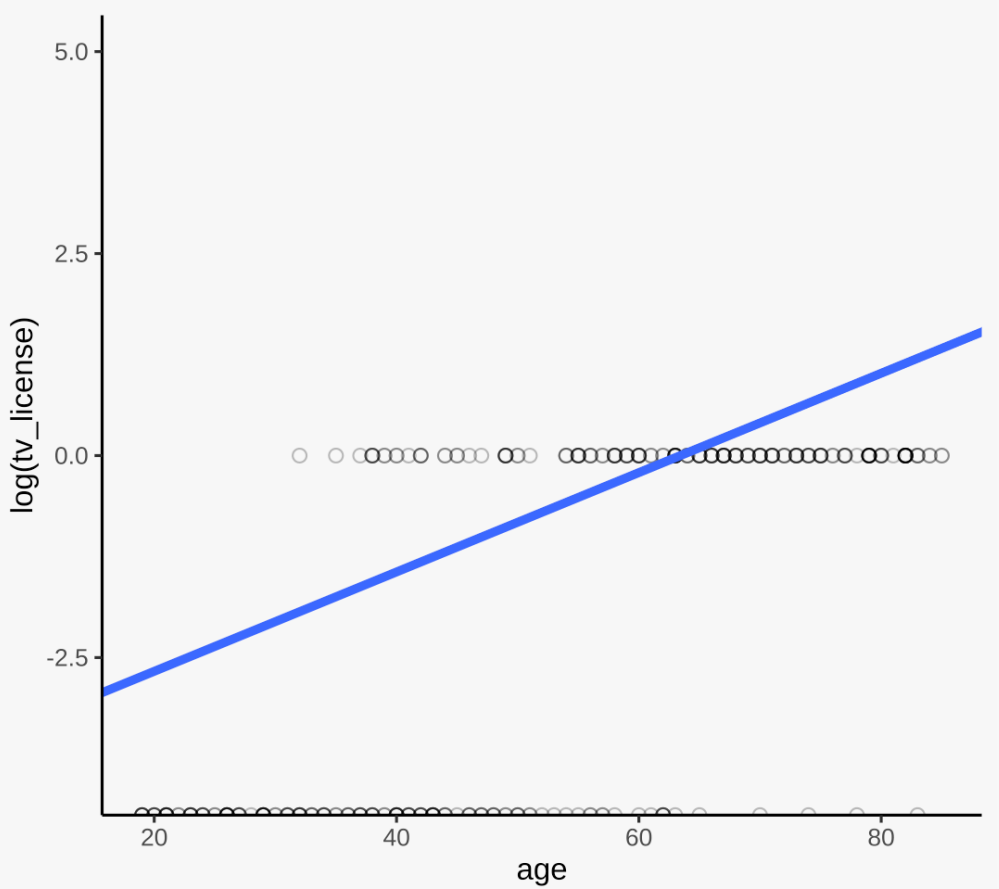
Or in `R` code: `exp(-3.9 + 0.0615*80)/(1 + exp(-3.9 + 0.0615*80)) = 0.7349`



Logistic regression

Visual Explanation

Can we do even better? What happens if we increase the regression line for the log odds even further.

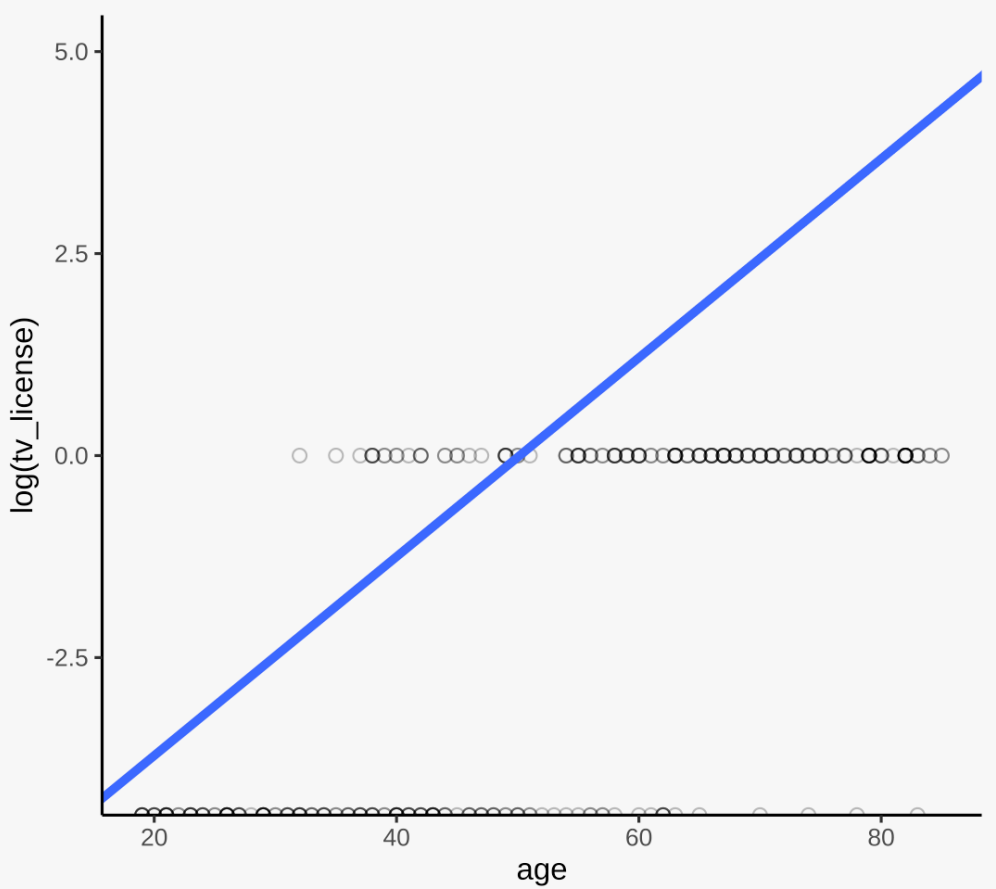


Logistic regression

Visual Explanation

Can we do even better? What happens if we increase the regression line for the log odds even further.

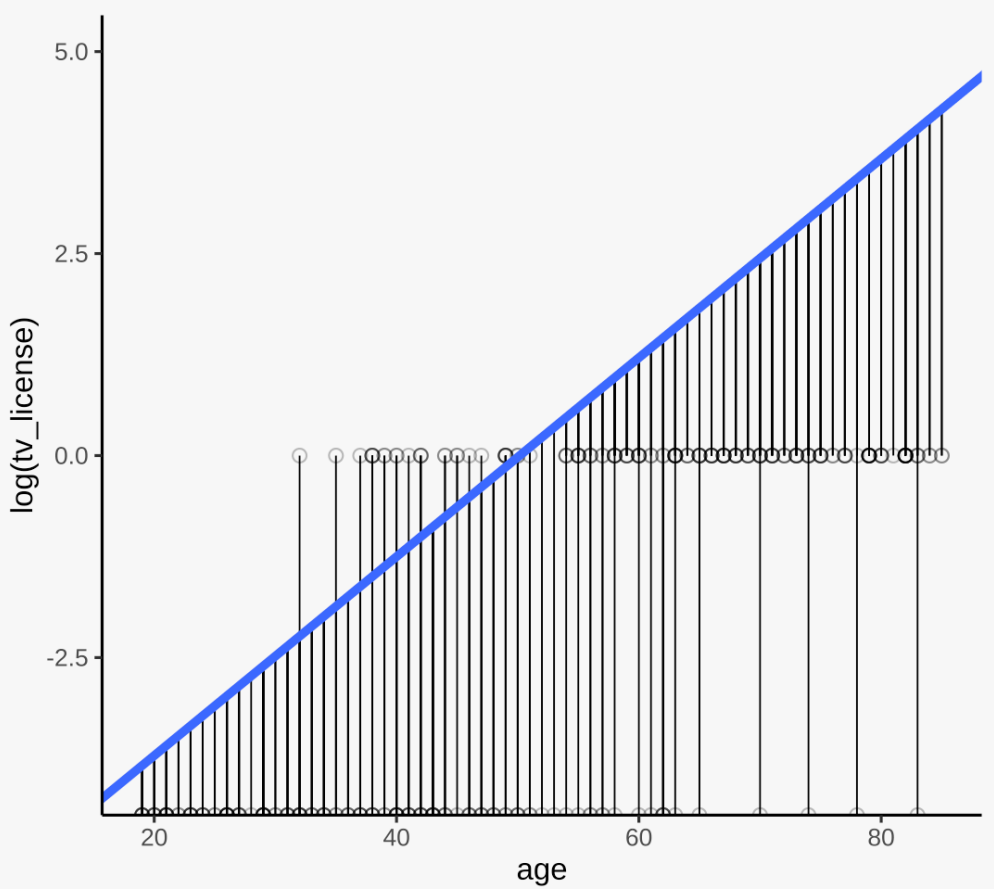
Note that, even though the distance between the points is getting larger in terms of log odds, this does not happen in the predictions due to them always being forced to be between 0 and 1.



Logistic regression

Visual Explanation

Let's repeat the process. First, let's project all of the points to their closest position on the line.

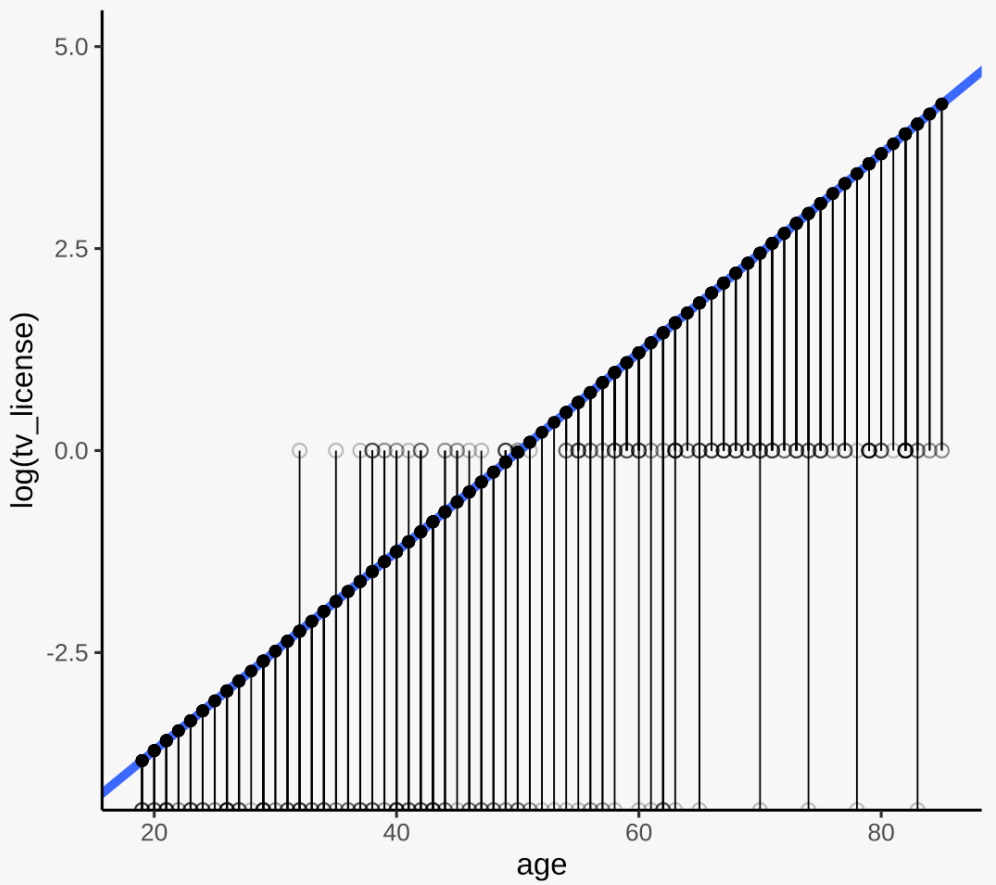


Logistic regression

Visual Explanation

Let's repeat the process. First, let's project all of the points to their closest position on the line.

Then we (or rather, the Maximum Likelihood Estimator) maps each point to its closest position on the line.



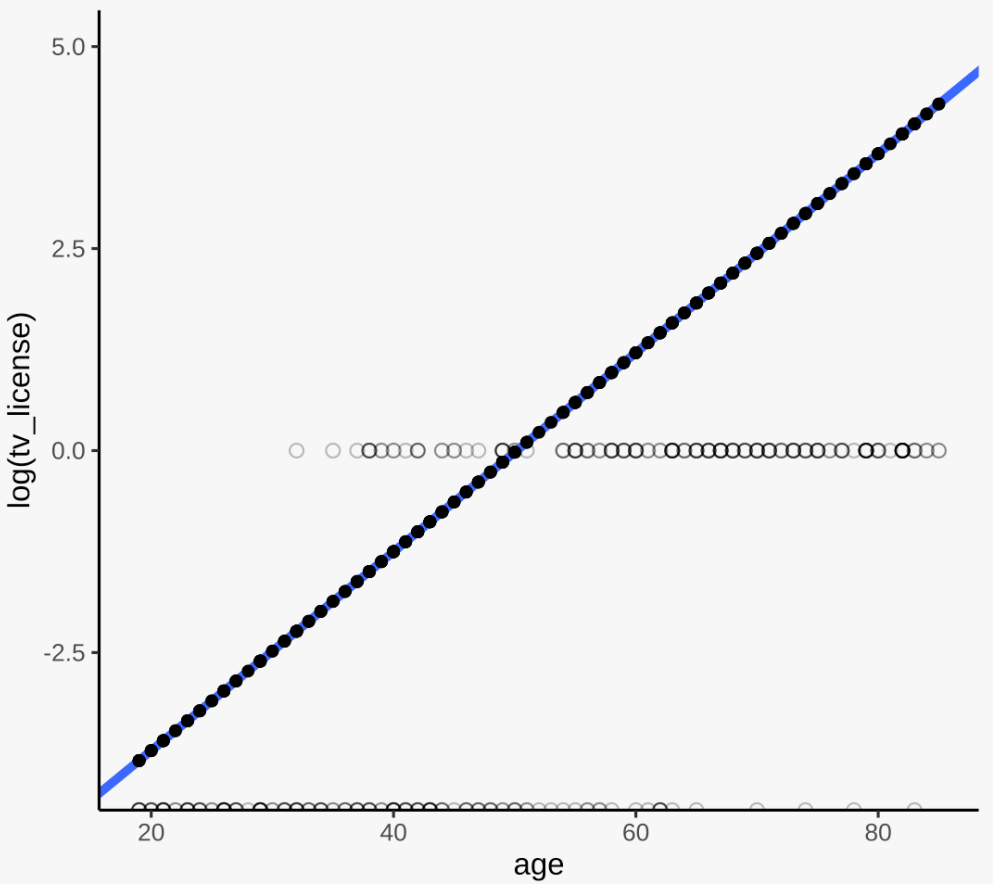
Logistic regression

Visual Explanation

Let's repeat the process. First, let's project all of the points to their closest position on the line.

Then we (or rather, the Maximum Likelihood Estimator) maps each point to its closest position on the line.

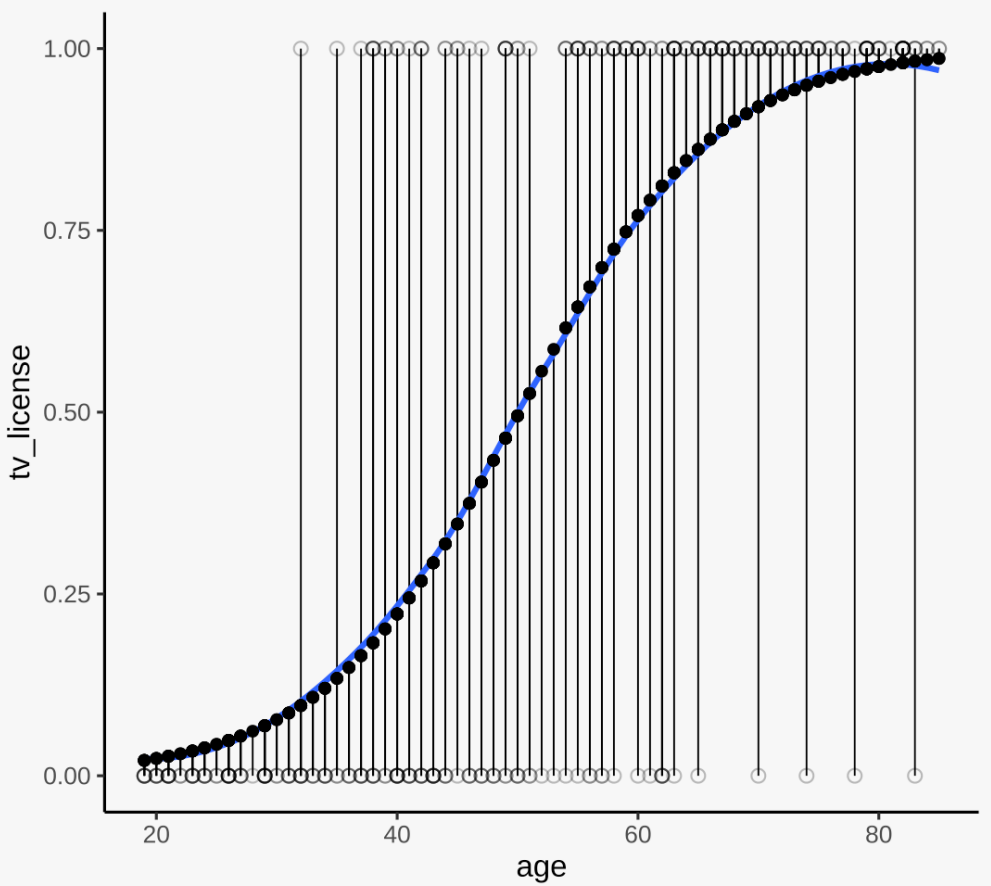
Then we can convert the resulting values for each point back into a predicted probability using the sigmoid function.



Logistic regression

Visual Explanation

Now we have another new set of predictions. Note that even though our logistic regression's sigmoid curve changed, our log odds regression remained a straight line (and could still be expressed in log odds as $\bar{Y} = B_0 + B_1X...$).



Logistic regression

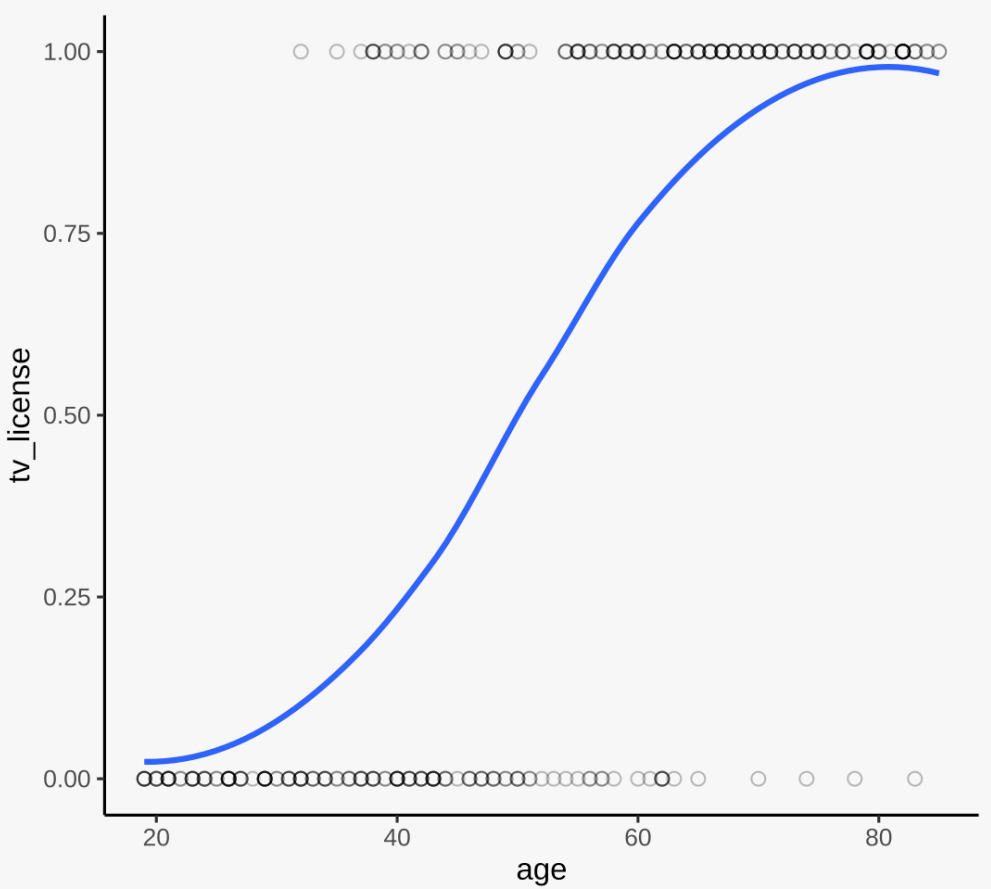
Visual Explanation

If these predictions are better than our last iteration, it would make sense to keep them!

We could try making the line steeper again and see if it improves them further. If not, we could make the line more shallow and keep going until we can't get any better predictions by changing the slope of the log odds regression line. When this happens it is known as **convergence**, and our logistic regression model gives us the outcome in the form of:

$$log(OddsY = 1|X) = B_0 + B_1X_1 + \ldots + B_nX_n$$

Which we can always convert back into predicted probability using the logistic sigmoid function! Pretty cool!

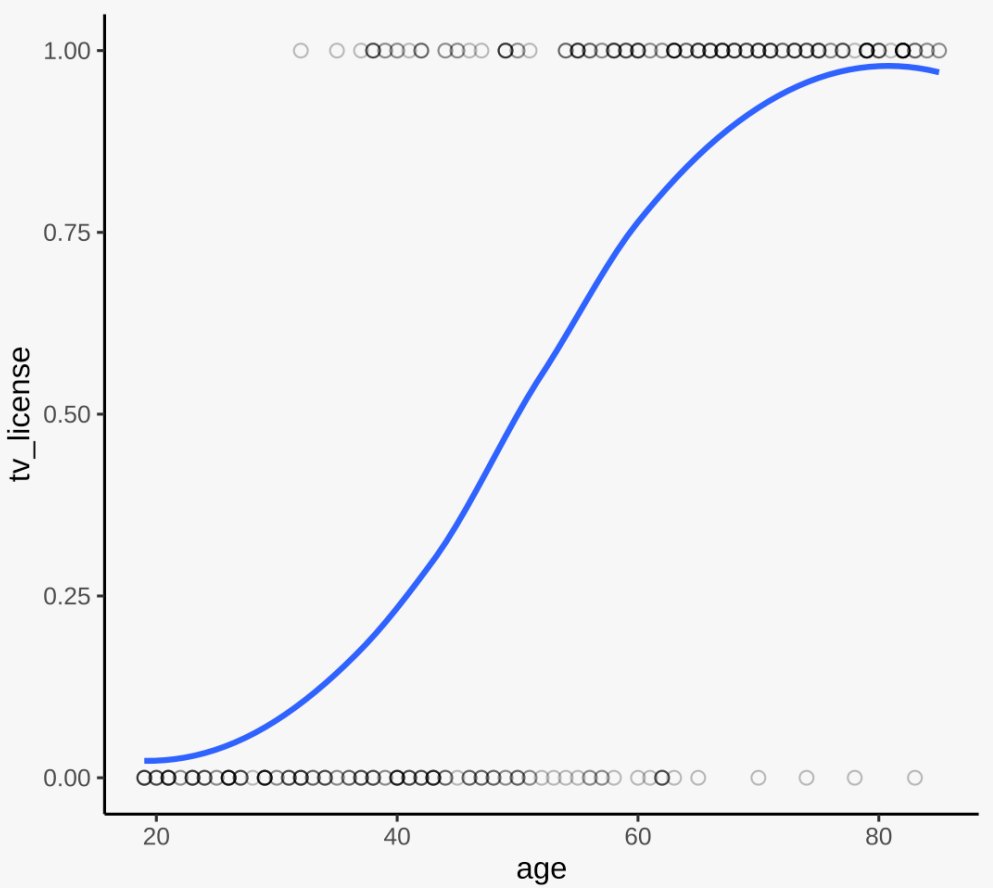


Logistic regression

Don't worry if this is too much maths/doesn't make much sense! The most important thing in this module is knowing how to run and interpret logistic regression models in R (which is very easy!)

You can use logistic regression even if you aren't confident using any of the formulas here or don't understand how maximum likelihood estimation works (I barely do)!

I wouldn't be doing my job as an educator if I didn't at least try and explain to you how it works!



Week 9: Logistic Regression — Part III

Estimating, interpreting, and communicating logistic regression in R.

Let's work through an example...

Who believes the government is hiding evidence of aliens?

Predicting belief in Alien conspiracy theories with logistic regression

```
conspiracy_data <- read_rds("conspiracy.rds")
```

```
conspiracy_data %>%
  janitor::tabyl(consp_alien)
```

```
##  consp_alien    n   percent
##            0 1025 0.6792578
##            1  484 0.3207422
```

How true do you think the following statement is: 'The government covers up knowledge of aliens'

(Recoded so that "Definitely true" and "Probably true" = 1, and "Probably false" and "Definitely false = 0)

Independent Variables

- Age (`rage`)
- Whether plays video games (`games`): Ref = Doesn't play video games
- Highest education qualification (`hedqual_rec`): Ref = No qualifications
- Openness to new experiences (`openness`)
  - 0 = Lowest openness, 3 = highest openness

Predicting belief in Alien conspiracy theories with logistic regression

Estimating a multiple logistic regression model

Code

```
# Model
glm(data = conspiracy_data,
    formula = consp_alien ~ rage + games +
                            hedqual_rec + openness,
    family = binomial(link = "logit"))
```

Predicting belief in Alien conspiracy theories with logistic regression

Estimating a multiple logistic regression model

- We need to use the `glm()` function (general linear model), rather than the `lm()` function we used for simple linear regression.

Code

```
# Model
glm(data = conspiracy_data,
    formula = consp_alien ~ rage + games +
                            hedqual_rec + openness,
    family = binomial(link = "logit"))
```

Predicting belief in Alien conspiracy theories with logistic regression

Estimating a multiple logistic regression model

- We need to use the `glm()` function (general linear model), rather than the `lm()` function we used for simple linear regression.
- We add in our formula in exactly the same way — our dependent variable should be a binary categorical variable which can be a factor, character, numeric, or logical class. The value being predicted will be **1** for a **numeric** class, **TRUE** for a **logical** class, the **highest value/non reference level** in a **factor** class. It will not work for a character class. If in doubt, just recode into a binary numeric!

Code

```
# Model
glm(data = conspiracy_data,
    formula = consp_alien ~ rage + games +
                            hedqual_rec + openness,
    family = binomial(link = "logit"))
```

Predicting belief in Alien conspiracy theories with logistic regression

Estimating a multiple logistic regression model

- We need to use the `glm()` function (general linear model), rather than the `lm()` function we used for simple linear regression.
- We add in our formula in exactly the same way — our dependent variable should be a binary categorical variable which can be a factor, character, numeric, or logical class. The value being predicted will be **1** for a **numeric** class, **TRUE** for a **logical** class, the **highest value/non reference level** in a **factor** class. It will not work for a character class. If in doubt, just recode into a binary numeric!
- Finally, and **importantly** we add `binomial(link = "logit")` to the `family =` argument. This tells `glm()` that we want a logistic regression, not a standard linear regression.

Code

```
glm(data = conspiracy_data,
    formula = consp_alien ~ rage + games +
                            hedqual_rec + openness,
    family = binomial(link = "logit"))
```

Predicting belief in Alien conspiracy theories with logistic regression

```
alien_model <- glm(data = conspiracy_data,
                   formula = consp_alien ~ rage + games +
                                           hedqual_rec + openness,
                   family = binomial(link = "logit"))

summary(alien_model)
```

```
##
## Call:
## glm(formula = consp_alien ~ rage + games + hedqual_rec + openness,
##     family = binomial(link = "logit"), data = conspiracy_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3461  -0.9378  -0.6988   1.2889   2.1832
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -0.003565   0.290870  -0.012  0.99022
## rage                          -0.007881   0.003547  -2.222  0.02631 *
## gamesYes                      -0.008412   0.127939  -0.066  0.94758
## hedqual_recLevel 1 quals      -0.041263   0.246457  -0.167  0.86704
## hedqual_recGCSEs/O levels     -0.445559   0.186221  -2.393  0.01673 *
## hedqual_recA levels           -0.649653   0.200278  -3.244  0.00118 **
## hedqual_recHigher Ed, no degree -0.900748   0.203448  -4.427 9.54e-06 ***
## hedqual_recFirst degree       -1.373859   0.220424  -6.233 4.58e-10 ***
## hedqual_recPostgraduate degree -1.844309   0.277238  -6.652 2.88e-11 ***
## openness                       0.215493   0.077718   2.773  0.00556 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1849.1  on 1468  degrees of freedom
## Residual deviance: 1764.4  on 1459  degrees of freedom
##   (40 observations deleted due to missingness)
## AIC: 1784.4
##
## Number of Fisher Scoring iterations: 4
```

Estimates

Estimates are now either the base log odds (Intercept) or the **change in log odds for a 1 unit change in X**.

p-values

p-values operate in the same way (albeit with a z-statistic rather than a t-statistic) and **test the hypothesis that the change in log odds associated with a 1-unit increase in X is equal to 0**.

But how do I communicate changes in log odds!?

Predicting belief in Alien conspiracy theories with logistic regression

```
alien_model <- glm(data = conspiracy_data,
                   formula = consp_alien ~ rage + games +
                                           hedqual_rec + openness,
                   family = binomial(link = "logit"))

summary(alien_model)
```

```
##
## Call:
## glm(formula = consp_alien ~ rage + games + hedqual_rec + openness,
##     family = binomial(link = "logit"), data = conspiracy_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3461  -0.9378  -0.6988   1.2889   2.1832
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -0.003565   0.290870  -0.012  0.99022
## rage                          -0.007881   0.003547  -2.222  0.02631 *
## gamesYes                      -0.008412   0.127939  -0.066  0.94758
## hedqual_recLevel 1 quals      -0.041263   0.246457  -0.167  0.86704
## hedqual_recGCSEs/O levels     -0.445559   0.186221  -2.393  0.01673 *
## hedqual_recA levels           -0.649653   0.200278  -3.244  0.00118 **
## hedqual_recHigher Ed, no degree -0.900748   0.203448  -4.427 9.54e-06 ***
## hedqual_recFirst degree       -1.373859   0.220424  -6.233 4.58e-10 ***
## hedqual_recPostgraduate degree  -1.844309   0.277238  -6.652 2.88e-11 ***
## openness                       0.215493   0.077718   2.773  0.00556 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1849.1  on 1468  degrees of freedom
## Residual deviance: 1764.4  on 1459  degrees of freedom
##   (40 observations deleted due to missingness)
## AIC: 1784.4
##
## Number of Fisher Scoring iterations: 4
```

Odds Ratios/ Exponentiated Coefficients

If we exponentiate (using `exp()`) the change in log odds associated with 1-unit increases in our independent variables, we get something called an **odds ratio**.

This **odds ratio** tells us the multiplicative change in the odds for a 1-unit increase — in other words **how many times as likely is it that Y = 1 when when the independent variable increases by 1**.

For example, **for a 1-unit increase in openness the odds of believing in alien conspiracy theories increases by** exp(0.215493) = **1.24 times**, or by around 24%.

Predicting belief in Alien conspiracy theories with logistic regression

```
alien_model <- glm(data = conspiracy_data,
                   formula = consp_alien ~ rage + games +
                                           hedqual_rec + openness,
                   family = binomial(link = "logit"))

summary(alien_model)
```

```
##
## Call:
## glm(formula = consp_alien ~ rage + games + hedqual_rec + openness,
##     family = binomial(link = "logit"), data = conspiracy_data)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.3461  -0.9378  -0.6988   1.2889   2.1832
##
## Coefficients:
##                                Estimate Std. Error z value Pr(>|z|)
## (Intercept)                   -0.003565   0.290870  -0.012  0.99022
## rage                          -0.007881   0.003547  -2.222  0.02631 *
## gamesYes                      -0.008412   0.127939  -0.066  0.94758
## hedqual_recLevel 1 quals      -0.041263   0.246457  -0.167  0.86704
## hedqual_recGCSEs/O levels     -0.445559   0.186221  -2.393  0.01673 *
## hedqual_recA levels           -0.649653   0.200278  -3.244  0.00118 **
## hedqual_recHigher Ed, no degree -0.900748   0.203448  -4.427 9.54e-06 ***
## hedqual_recFirst degree       -1.373859   0.220424  -6.233 4.58e-10 ***
## hedqual_recPostgraduate degree  -1.844309   0.277238  -6.652 2.88e-11 ***
## openness                       0.215493   0.077718   2.773  0.00556 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1849.1  on 1468  degrees of freedom
## Residual deviance: 1764.4  on 1459  degrees of freedom
##   (40 observations deleted due to missingness)
## AIC: 1784.4
##
## Number of Fisher Scoring iterations: 4
```

Odds Ratios/ Exponentiated Coefficients

Negative odds ratios can be a little bit more tricky to interpret, but are not too difficult with a bit of work.

For example, the multiplicative change in odds associated with having a postgraduate degree is exp(-1.844309) or **0.158**.

This means that **the odds of believing in alien conspiracy theories decrease by 84.2% among those with higher education degrees, compared to those with no qualifications** (1 - exp(-1.844309) or 1 - 0.158)

Predicting belief in Alien conspiracy theories with logistic regression

Quickly Adding Odds Ratios to your Model Output using `broom` and `mutate`

- Use `broom::tidy()` to turn your model output into a tidy tibble

```
# Convert model to tibble with broom::tidy
tidy_model <- broom::tidy(alien_model)
tidy_model
```

```
## # A tibble: 10 × 5
##    term                           estimate std.error statistic  p.value
##    <chr>                             <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)                    -0.00356   0.291     -0.0123 9.90e- 1
##  2 rage                           -0.00788   0.00355   -2.22   2.63e- 2
##  3 gamesYes                       -0.00841   0.128     -0.0657 9.48e- 1
##  4 hedqual_recLevel 1 quals       -0.0413    0.246     -0.167  8.67e- 1
##  5 hedqual_recGCSEs/O levels      -0.446     0.186     -2.39   1.67e- 2
##  6 hedqual_recA levels            -0.650     0.200     -3.24   1.18e- 3
##  7 hedqual_recHigher Ed, no degree -0.901    0.203     -4.43   9.54e- 6
##  8 hedqual_recFirst degree        -1.37      0.220     -6.23   4.58e-10
##  9 hedqual_recPostgraduate degree -1.84      0.277     -6.65   2.88e-11
## 10 openness                        0.215     0.0777     2.77   5.56e- 3
```

Predicting belief in Alien conspiracy theories with logistic regression

Quickly Adding Odds Ratios to your Model Output using `broom` and `mutate`

- Use `broom::tidy()` to turn your model output into a tidy tibble
- Use `mutate` to create an extra column of exponentiated coefficients (a.k.a. odds ratios)

```
# Create odds ratios column
tidy_model <- tidy_model %>%
  mutate(
    odds_ratios = exp(estimate)
  )

tidy_model
```

```
## # A tibble: 10 × 6
##    term                          estimate std.error statistic  p.value odds_ratios
##    <chr>                            <dbl>     <dbl>     <dbl>    <dbl>       <dbl>
##  1 (Intercept)                  -0.00356   0.291     -0.0123 9.90e- 1       0.996
##  2 rage                         -0.00788   0.00355   -2.22   2.63e- 2       0.992
##  3 gamesYes                     -0.00841   0.128     -0.0657 9.48e- 1       0.992
##  4 hedqual_recLevel 1 quals     -0.0413    0.246     -0.167  8.67e- 1       0.960
##  5 hedqual_recGCSEs/O levels    -0.446     0.186     -2.39   1.67e- 2       0.640
##  6 hedqual_recA levels          -0.650     0.200     -3.24   1.18e- 3       0.522
##  7 hedqual_recHigher Ed, no d… -0.901     0.203     -4.43   9.54e- 6       0.406
##  8 hedqual_recFirst degree      -1.37      0.220     -6.23   4.58e-10       0.253
##  9 hedqual_recPostgraduate de… -1.84      0.277     -6.65   2.88e-11       0.158
## 10 openness                     0.215     0.0777     2.77   5.56e- 3       1.24
```

Predicting belief in Alien conspiracy theories with logistic regression

Quickly Adding Odds Ratios to your Model Output using `broom` and `mutate`

- Use `broom::tidy()` to turn your model output into a tidy tibble
- Use `mutate` to create an extra column of exponentiated coefficients (a.k.a. odds ratios)

```
# Create odds ratios column
tidy_model <- tidy_model %>%
  mutate(
    odds_ratios = exp(estimate)
  )

tidy_model
```

```
## # A tibble: 10 × 6
##    term                          estimate std.error statistic  p.value odds_ratios
##    <chr>                            <dbl>     <dbl>     <dbl>    <dbl>       <dbl>
##  1 (Intercept)                  -0.00356   0.291     -0.0123 9.90e- 1       0.996
##  2 rage                         -0.00788   0.00355   -2.22   2.63e- 2       0.992
##  3 gamesYes                     -0.00841   0.128     -0.0657 9.48e- 1       0.992
##  4 hedqual_recLevel 1 quals     -0.0413    0.246     -0.167  8.67e- 1       0.960
##  5 hedqual_recGCSEs/O levels    -0.446     0.186     -2.39   1.67e- 2       0.640
##  6 hedqual_recA levels          -0.650     0.200     -3.24   1.18e- 3       0.522
##  7 hedqual_recHigher Ed, no d… -0.901     0.203     -4.43   9.54e- 6       0.406
##  8 hedqual_recFirst degree      -1.37      0.220     -6.23   4.58e-10       0.253
##  9 hedqual_recPostgraduate de… -1.84      0.277     -6.65   2.88e-11       0.158
## 10 openness                      0.215     0.0777     2.77   5.56e- 3       1.24
```

What was the associated change in the odds of believing the government hides information related to aliens for a 1 year increase in age?

Predicting belief in Alien conspiracy theories with logistic regression

What about if we want to look at the predictions themselves, not just the change in odds?

Predicting belief in Alien conspiracy theories with logistic regression

What about if we want to look at the predictions themselves, not just the change in odds?

We can use a very nice package called `ggeffects` to plot the predictied probabilities for different independent variables, while keeping the value of other independent variables constant at their mean.

First, load the `ggeffects` library.

```
library(ggeffects)
```

Predicting belief in Alien conspiracy theories with logistic regression

What about if we want to look at the predictions themselves, not just the change in odds?

We can use a very nice package called `ggeffects` to plot the predictied probabilities for different independent variables, while keeping the value of other independent variables constant at their mean.

First, load the `ggeffects` library.

Then, run the `ggeffect` function; make sure to specify the variable you want predictions for in the `terms` argument. Here, I want predictions for `openness`. This will give me some text output for a few possible values of openness.

```
library(ggeffects)

ggeffect(alien_model, terms = "openness")
```

```
## # Predicted probabilities of Conspiracy beliefs - government covers up knowledge of alier
##
## openness | Predicted |       95% CI
## -----------------------------------
##     0.00 |      0.24 | [0.20, 0.30]
##     0.33 |      0.26 | [0.22, 0.30]
##     0.67 |      0.27 | [0.24, 0.31]
##     1.00 |      0.29 | [0.26, 0.32]
##     1.67 |      0.32 | [0.29, 0.34]
##     2.00 |      0.33 | [0.30, 0.36]
##     2.33 |      0.35 | [0.31, 0.38]
##     3.00 |      0.38 | [0.33, 0.44]
```

Predicting belief in Alien conspiracy theories with logistic regression

What about if we want to look at the predictions themselves, not just the change in odds?

We can use a very nice package called `ggeffects` to plot the predictied probabilities for different independent variables, while keeping the value of other independent variables constant at their mean.

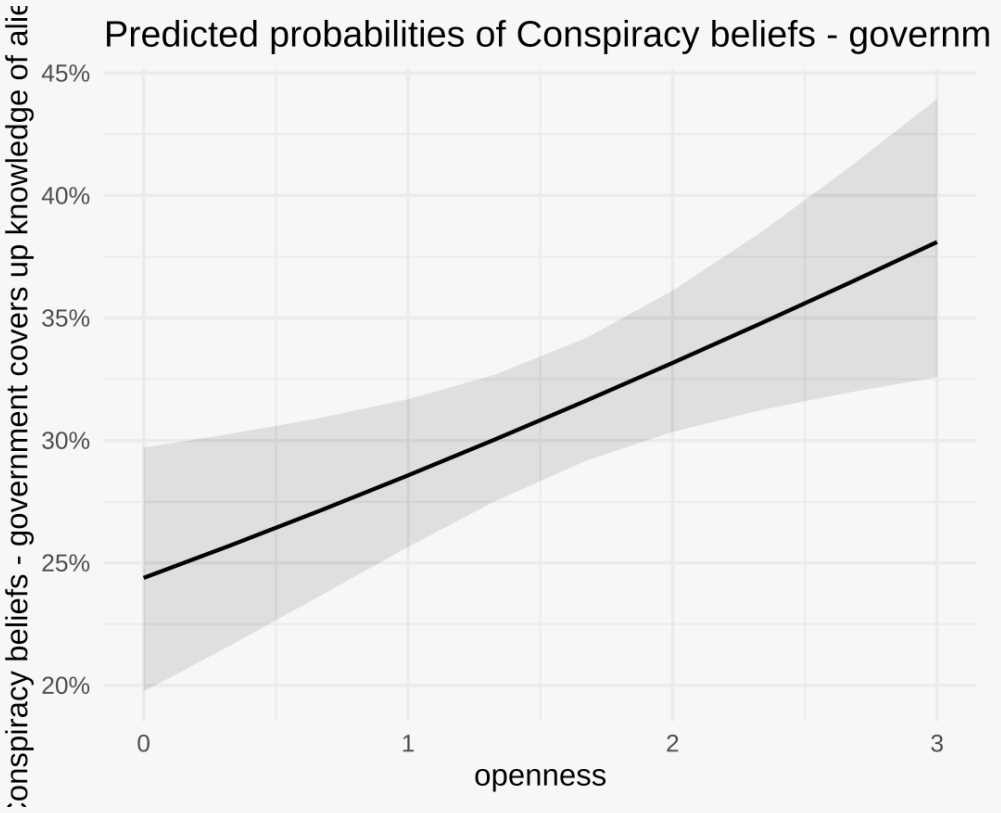
First, load the `ggeffects` library.

Then, run the `ggeffect` function; make sure to specify the variable you want predictions for in the `terms` argument. Here, I want predictions for `openness`. This will give me some text output for a few possible values of openness.

We can add a `plot()` function after this with a `%>%` to turn this into a ggplot! You can even customise it by using `ggplot` functions like you would normally.

```
library(ggeffects)

ggeffect(alien_model, terms = "openness") %>%
  plot()
```

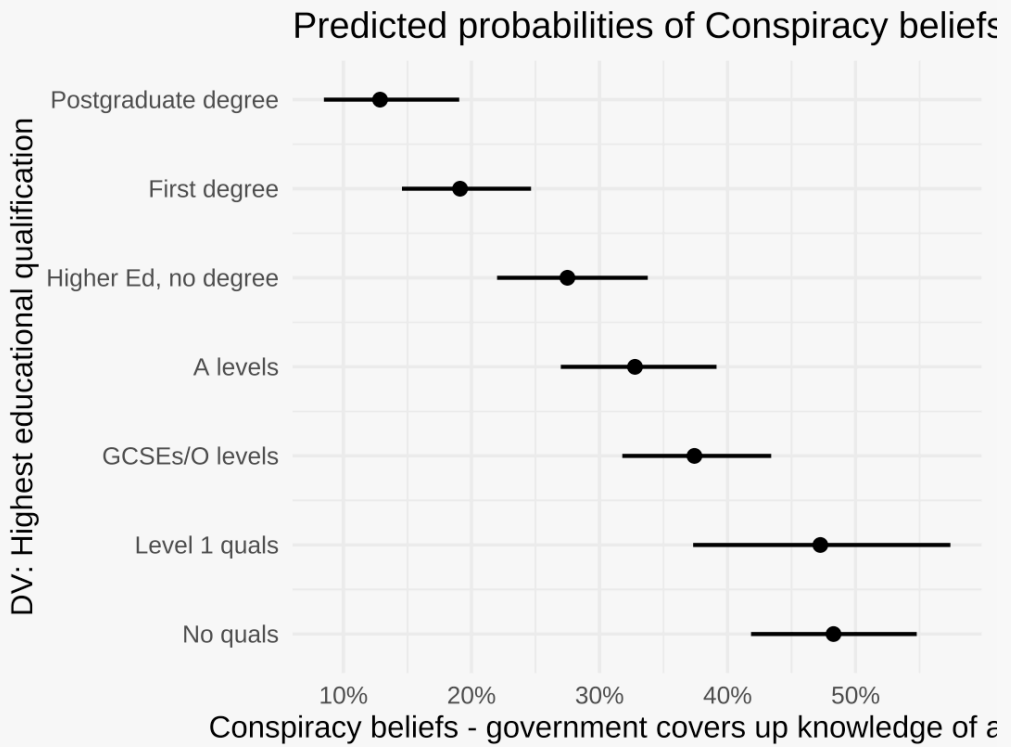


Predicting belief in Alien conspiracy theories with logistic regression

The plot you get will differ depending on whether your independent variable is categorical/ordinal or continuous.

```
library(ggeffects)

ggeffect(alien_model, terms = "hedqual_rec") %>%
  plot() +
  coord_flip()
```



Week 9: Logistic Regression — Part IV

Logistic regression model fit and accuracy.

But wait, what happened to our R-squared statistic? How good is our model overall?

Model fit/accuracy

With logistic regression we cannot calculate an R^2 value for model fit in the same way we would for a multiple linear regression model.

Model fit/accuracy

Pseudo R^2

One option is to use one of a range of Pseudo- R^2s. However, these do not have quite as straightforward a definition as OLS R^2.

- **McFadden R^2**: Values between 0.2 and 0.4 represent excellent fit to the data (Heshner & Stopher, 1979)
- **Cox & Snell R^2**: No exact interpretation. Closer to 1 = better fit.
- **Nagelkerke R^2**: No exact interpretation. Closer to 1 = better fit.
- **Efron R^2**: Based on the difference between the predicted probabilities and actual responses. Closer to 1 = better fit.

```
pseudo_r2s_alien <- DescTools::PseudoR2(alien_model, which = c("McFadden", "CoxSnell", "Nag

tibble(
  r2_names = names(pseudo_r2s_alien),
  r2_vals = pseudo_r2s_alien
)
```

```
## # A tibble: 4 × 2
##   r2_names   r2_vals
##   <chr>        <dbl>
## 1 McFadden    0.0458
## 2 CoxSnell    0.0560
## 3 Nagelkerke  0.0782
## 4 Efron       0.0554
```

Model fit/accuracy

Pseudo R^2

One option is to use one of a range of Pseudo- R^2s. However, these do not have quite as straightforward a definition as OLS R^2.

- **McFadden R^2**: Values between 0.2 and 0.4 represent excellent fit to the data (Heshner & Stopher, 1979)
- **Cox & Snell R^2**: No exact interpretation. Closer to 1 = better fit.
- **Nagelkerke R^2**: No exact interpretation. Closer to 1 = better fit.
- **Efron R^2**: Based on the difference between the predicted probabilities and actual responses. Closer to 1 = better fit.

However, Pseudo- R^2 values are of limited value for logistic regression — they can often be useful for **comparing the goodness of fit of different models**, but are not as useful as OLS R^2 for explaining how well a model fits the data overall (Hosmer, Lemeshow & Sturdivant, 2013).

However, we have two additional tools at our disposal for assessing our models: **accuracy** and **cross-validation**.

Model fit/accuracy

Accuracy

In the context of logistic regression, **accuracy** refers to whether our model's predictions are **able to correctly classify the underlying data**.

Model fit/accuracy

Accuracy

In the context of logistic regression, **accuracy** refers to whether our model's predictions are **able to correctly classify the underlying data**.

The first stage is to calculate our predictions.

- We start by using the `mutate` function to create a new column called `pred_consp_alien` using the `predict` function.
  - The first argument (`alien_model`) tells `R` the model to be used for prediction.
  - The second argument (`type = "response"`) tells `R` we want the predicted probability, not the logodds.
  - The third argument (`newdata = conspiracy_data`) provides `R` with the data that holds the independent variables required for the prediction of each case.

```
conspiracy_data <- conspiracy_data %>%
  mutate(
    pred_consp_alien = predict(alien_model,
                               type = "response",
                               newdata = conspiracy_data)
  )

conspiracy_data %>%
  select(consp_alien, pred_consp_alien)
```

```
## # A tibble: 1,509 × 2
##    consp_alien pred_consp_alien
##          <dbl>            <dbl>
##  1           1            0.338
##  2           1            0.376
##  3           1            0.516
##  4           1            0.440
##  5           1            0.347
##  6           1            0.431
##  7           1            0.403
##  8           0            0.521
##  9           0           NA
## 10           0            0.374
## # ℹ 1,499 more rows
```

Model fit/accuracy

Accuracy

The second stage is to convert our predicted probabilities into predicted outcomes.

- I use the `mutate` function again, this time to overwrite the `pred_consp_alien` variable with the outcome from a `case_when` function.
- The `case_when` function does the following:
  - Any `NA` predictions (because of missing independent variables) stay as missing (`NA_real_`).
  - When `pred_consp_alien` is greater than or equal to 0.5, it now becomes 1.
  - Else, `pred_cons_alien` is equal to 0 (whenever it's less than 0.5).

I then check the results with a temporary `select()`.

```
conspiracy_data <- conspiracy_data %>%
  mutate(
    pred_consp_alien = case_when(is.na(pred_consp_alien) ~ NA_real_,
                                 pred_consp_alien >= 0.5 ~ 1,
                                 TRUE ~ 0)
  )

conspiracy_data %>%
  select(consp_alien, pred_consp_alien)
```

```
## # A tibble: 1,509 × 2
##    consp_alien pred_consp_alien
##          <dbl>            <dbl>
##  1           1                0
##  2           1                0
##  3           1                1
##  4           1                0
##  5           1                0
##  6           1                0
##  7           1                0
##  8           0                1
##  9           0               NA
## 10           0                0
## # ℹ 1,499 more rows
```

Model fit/accuracy

Accuracy

Now I can calculate the accuracy of my model against the real data. A quick way to do this and get a collection of relevant statistics is using the `caret` package's `confusionMatrix` function.

- Note that I first change the predictions and original data into factors, and make 0 the reference level. This is because the `caret::confusionMatrix` function only accepts factor type variables.

```
library(caret)

confusionMatrix(
  data = relevel(factor( conspiracy_data$pred_consp_alien ),
                      ref = "0"),
  reference = relevel(factor( conspiracy_data$consp_alien ),
                          ref = "0"),
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 961 443
##          1  33  32
##
##                Accuracy : 0.676
##                  95% CI : (0.6514, 0.6999)
##     No Information Rate : 0.6767
##     P-Value [Acc > NIR] : 0.5346
##
##                   Kappa : 0.0441
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96680
##             Specificity : 0.06737
##          Pos Pred Value : 0.68447
##          Neg Pred Value : 0.49231
##              Prevalence : 0.67665
##          Detection Rate : 0.65419
##    Detection Prevalence : 0.95575
##       Balanced Accuracy : 0.51708
##
##        'Positive' Class : 0
##
```

Model fit/accuracy

Accuracy

There is a lot of information here, but for this class I just want to focus on four things:

- The **Confusion Matrix/Classification Table**
- The **Accuracy**
- The **No Information Rate**
- The **Acc > NIR hypothesis test result**

```
library(caret)

confusionMatrix(
  data = relevel(factor( conspiracy_data$pred_consp_alien ),
                         ref = "0"),
  reference = relevel(factor( conspiracy_data$consp_alien ),
                              ref = "0"),
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 961 443
##          1  33  32
##
##                Accuracy : 0.676
##                  95% CI : (0.6514, 0.6999)
##     No Information Rate : 0.6767
##     P-Value [Acc > NIR] : 0.5346
##
##                   Kappa : 0.0441
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96680
##             Specificity : 0.06737
##          Pos Pred Value : 0.68447
##          Neg Pred Value : 0.49231
##              Prevalence : 0.67665
##          Detection Rate : 0.65419
##    Detection Prevalence : 0.95575
##       Balanced Accuracy : 0.51708
##
##        'Positive' Class : 0
##
```

Model fit/accuracy

Accuracy

There is a lot of information here, but for this class I just want to focus on four things:

- The **Confusion Matrix/Classification Table**

The 2x2 table at the top of the output tells us how many values were correctly predicted (0 = 0 / 1 = 1) and how many were incorrectly predicted (false positives, predicted 1 when it should be 0; false negatives, predicted 0 when it should be 1). You can see that we had quite a large number of false negatives with this model: **443**!

- The **Accuracy**
- The **No Information Rate**
- The **Acc > NIR hypothesis test result**

```
library(caret)

confusionMatrix(
  data = relevel(factor( conspiracy_data$pred_consp_alien ),
                         ref = "0"),
  reference = relevel(factor( conspiracy_data$consp_alien ),
                              ref = "0"),
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 961 443
##          1  33  32
##
##                Accuracy : 0.676
##                  95% CI : (0.6514, 0.6999)
##     No Information Rate : 0.6767
##     P-Value [Acc > NIR] : 0.5346
##
##                   Kappa : 0.0441
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96680
##             Specificity : 0.06737
##          Pos Pred Value : 0.68447
##          Neg Pred Value : 0.49231
##              Prevalence : 0.67665
##          Detection Rate : 0.65419
##    Detection Prevalence : 0.95575
##       Balanced Accuracy : 0.51708
##
##        'Positive' Class : 0
##
```

Model fit/accuracy

Accuracy

There is a lot of information here, but for this class I just want to focus on four things:

- The **Confusion Matrix/Classification Table**
- The **Accuracy**

Accuracy is a straightforward calculation of the proportion of classifications that the model got correct. 961 + 32 predictions were correct (993), and 443 + 33 predictions were incorrect (476). This means that 993 out of 1469 cases were correctly predicted by our model (67.6%).

- The **No Information Rate**
- The **Acc > NIR hypothesis test result**

```
library(caret)

confusionMatrix(
  data = relevel(factor( conspiracy_data$pred_consp_alien ),
                         ref = "0"),
  reference = relevel(factor( conspiracy_data$consp_alien ),
                              ref = "0"),
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 961 443
##          1  33  32
##
##                Accuracy : 0.676
##                  95% CI : (0.6514, 0.6999)
##     No Information Rate : 0.6767
##     P-Value [Acc > NIR] : 0.5346
##
##                   Kappa : 0.0441
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96680
##             Specificity : 0.06737
##          Pos Pred Value : 0.68447
##          Neg Pred Value : 0.49231
##              Prevalence : 0.67665
##          Detection Rate : 0.65419
##    Detection Prevalence : 0.95575
##       Balanced Accuracy : 0.51708
##
##        'Positive' Class : 0
##
```

Model fit/accuracy

Accuracy

There is a lot of information here, but for this class I just want to focus on four things:

- The **Confusion Matrix/Classification Table**
- The **Accuracy**
- The **No Information Rate**

The No Information Rate is the proportion of observations we would expect to get correct by guessing without any additional information; equal to the proportion of the largest group (994 / 1469 = 67.67%).

- The **Acc > NIR hypothesis test result**

```
library(caret)

confusionMatrix(
  data = relevel(factor( conspiracy_data$pred_consp_alien ),
                         ref = "0"),
  reference = relevel(factor( conspiracy_data$consp_alien ),
                              ref = "0"),
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 961 443
##          1  33  32
##
##                Accuracy : 0.676
##                  95% CI : (0.6514, 0.6999)
##     No Information Rate : 0.6767
##     P-Value [Acc > NIR] : 0.5346
##
##                   Kappa : 0.0441
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96680
##             Specificity : 0.06737
##          Pos Pred Value : 0.68447
##          Neg Pred Value : 0.49231
##              Prevalence : 0.67665
##          Detection Rate : 0.65419
##    Detection Prevalence : 0.95575
##       Balanced Accuracy : 0.51708
##
##        'Positive' Class : 0
##
```

Model fit/accuracy

Accuracy

There is a lot of information here, but for this class I just want to focus on four things:

- The **Confusion Matrix/Classification Table**
- The **Accuracy**
- The **No Information Rate**
- The **Acc > NIR hypothesis test result**

Lastly, we have a hypothesis test for the hypothesis that our model's accuracy is not greater than the No Information Rate. A p-value less than 0.05 indicates our accuracy is significantly better than the No Information Rate.

```
library(caret)

confusionMatrix(
  data = relevel(factor( conspiracy_data$pred_consp_alien ),
                         ref = "0"),
  reference = relevel(factor( conspiracy_data$consp_alien ),
                              ref = "0"),
)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 961 443
##          1  33  32
##
##                Accuracy : 0.676
##                  95% CI : (0.6514, 0.6999)
##     No Information Rate : 0.6767
##     P-Value [Acc > NIR] : 0.5346
##
##                   Kappa : 0.0441
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.96680
##             Specificity : 0.06737
##          Pos Pred Value : 0.68447
##          Neg Pred Value : 0.49231
##              Prevalence : 0.67665
##          Detection Rate : 0.65419
##    Detection Prevalence : 0.95575
##       Balanced Accuracy : 0.51708
##
##        'Positive' Class : 0
##
```

Model fit/accuracy

Accuracy Weaknesses

While intuitive, we should be cautious about relying only on accuracy for measuring model fit.

Accuracy alone **doesn't tell us how good our predicted probabilities are**. Most of our predicted probabilities could be very marginal.

This means that small amounts of random bias or error in our specific sample could make our model perform much better in an accuracy test than it would in the population.

```
## # A tibble: 11 × 2
##    consp_alien pred_consp_alien
##          <dbl>            <dbl>
##  1           0            0.374
##  2           0            0.153
##  3           1            0.287
##  4           0            0.251
##  5           1            0.436
##  6           0            0.374
##  7           0            0.344
##  8           0            0.268
##  9           0            0.366
## 10           1            0.246
## 11           0            0.181
```

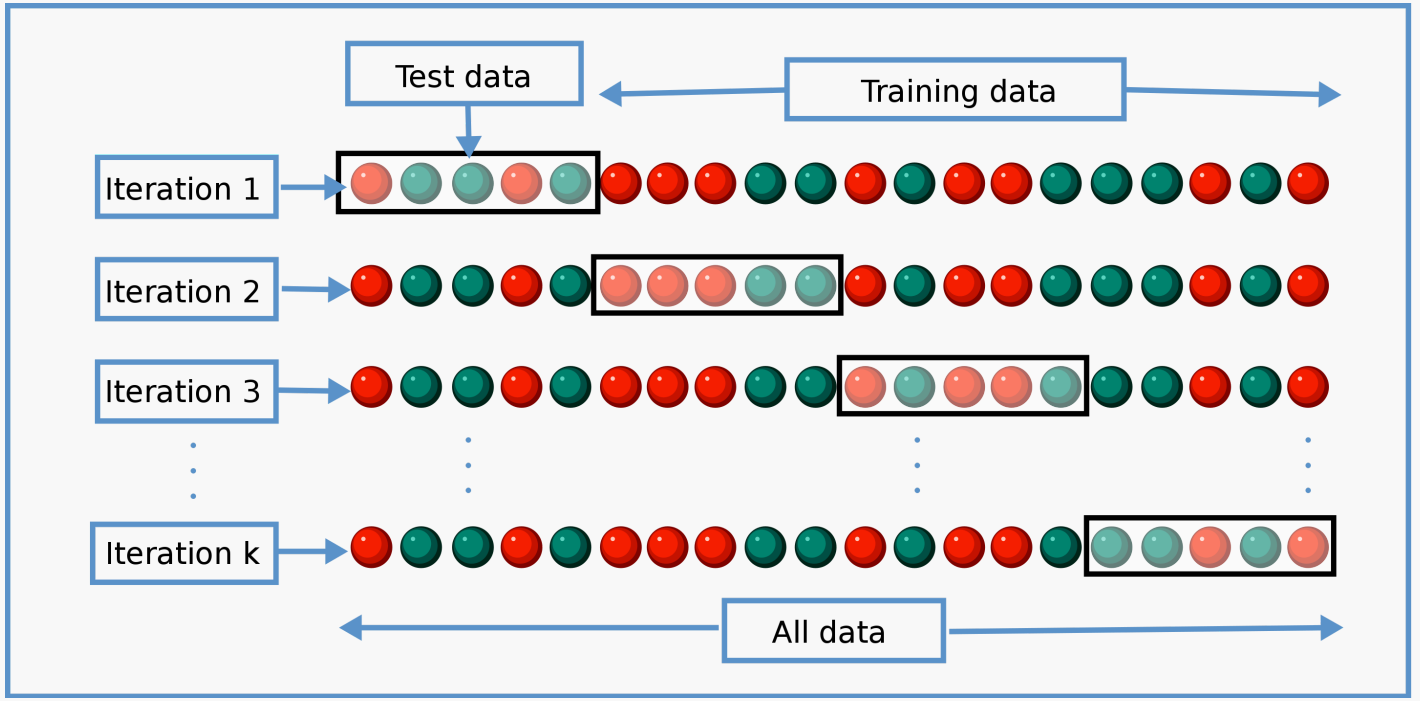
```
## # A tibble: 300 × 2
##    tv_license tv_license_pred
##         <dbl>           <dbl>
##  1          1          0.202
##  2          1          0.134
##  3          1          0.644
##  4          0          0.0302
##  5          1          0.464
##  6          0          0.495
##  7          1          0.936
##  8          0          0.293
##  9          1          0.644
## 10          0          0.0687
## # ℹ 290 more rows
```

Model fit/accuracy

K-fold Cross-Validation

One of the ways of getting a better representation of model accuracy (a measure less influenced by the idiosyncrasies in our sample) is the **K-fold Cross-validation**.

- Take a unique subset of data to 'hold out' of our model estimation.
- Estimate our model on the remaining data.
- Check the accuracy of this model on our hold out data.
- Repeat this k times and then look at the average accuracy and standard error. (Normally k = 5 or 10)



Model fit/accuracy

K-fold Cross-Validation

K-fold Cross-Validation is very easy to perform in `R` using the `performance` package.

- Set a randomiser seed if you need your results to be replicable.
- Use the `performance_accuracy` function to perform k-fold cross-validation.

Results

- The average accuracy for our model was 63.5% — about 4 percentage points worse than our raw accuracy.
- Average standard error estimate of the accuracy was around 5%

```r
# Set seed - Because the unique subsets are random
# for demonstration purposes I want to make sure
# the they are always the same
set.seed(306)

# Use the performance package to perform k-fold
# cross-validation
performance::performance_accuracy(alien_model, k = 5)
```

```
## # Accuracy of Model Predictions
##
## Accuracy (95% CI): 63.50% [56.66%, 67.97%]
## Method: Area under Curve
```

Week 9: Logistic Regression — Part V

Logistic regression assumptions.

Assumptions

- Linearity
- Homoscedasticity
- No outliers or leverage points
- Normality of residuals
- No multicollinearity

Assumptions

- **Linearity**
- Homoscedasticity
- **No outliers or leverage points**
- Normality of residuals
- **No multicollinearity**

Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity

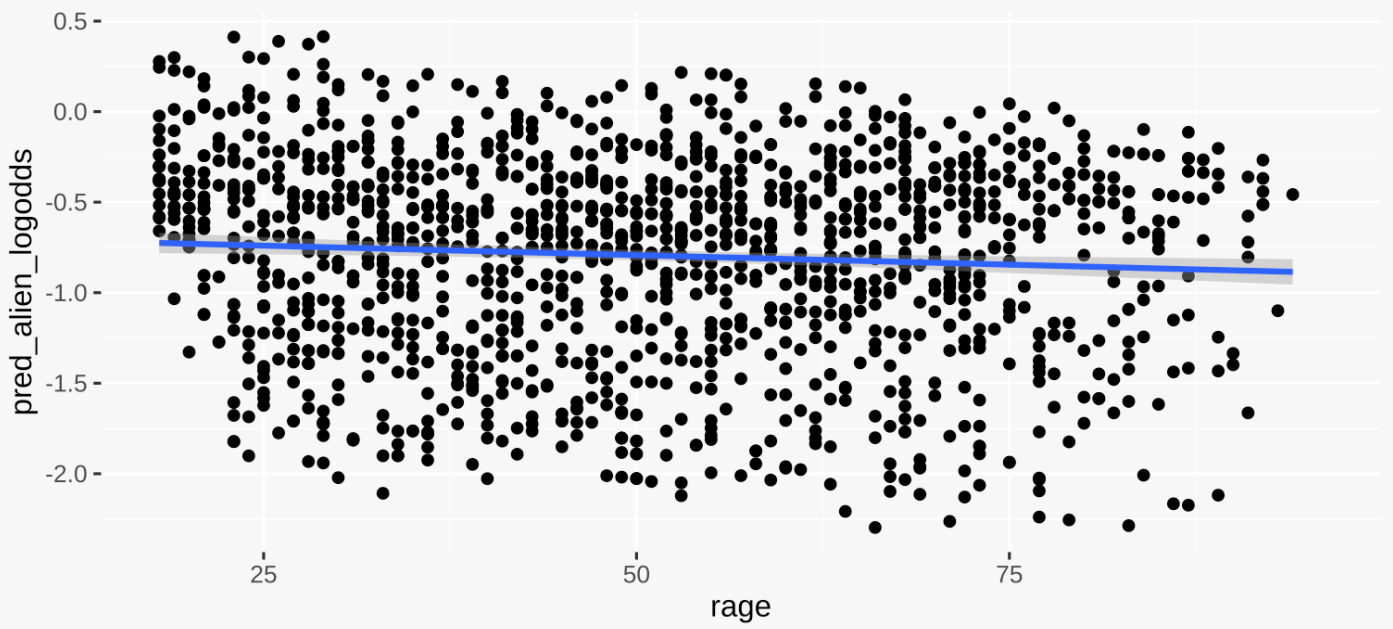
Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity
- No autocorrelation

Assumptions

- **Linearity**
  - For continuous predictors only
  - Is a straight line the best fit to the data or would a curved line fit better?
- No outliers or leverage points
- No multicollinearity
- No autocorrelation

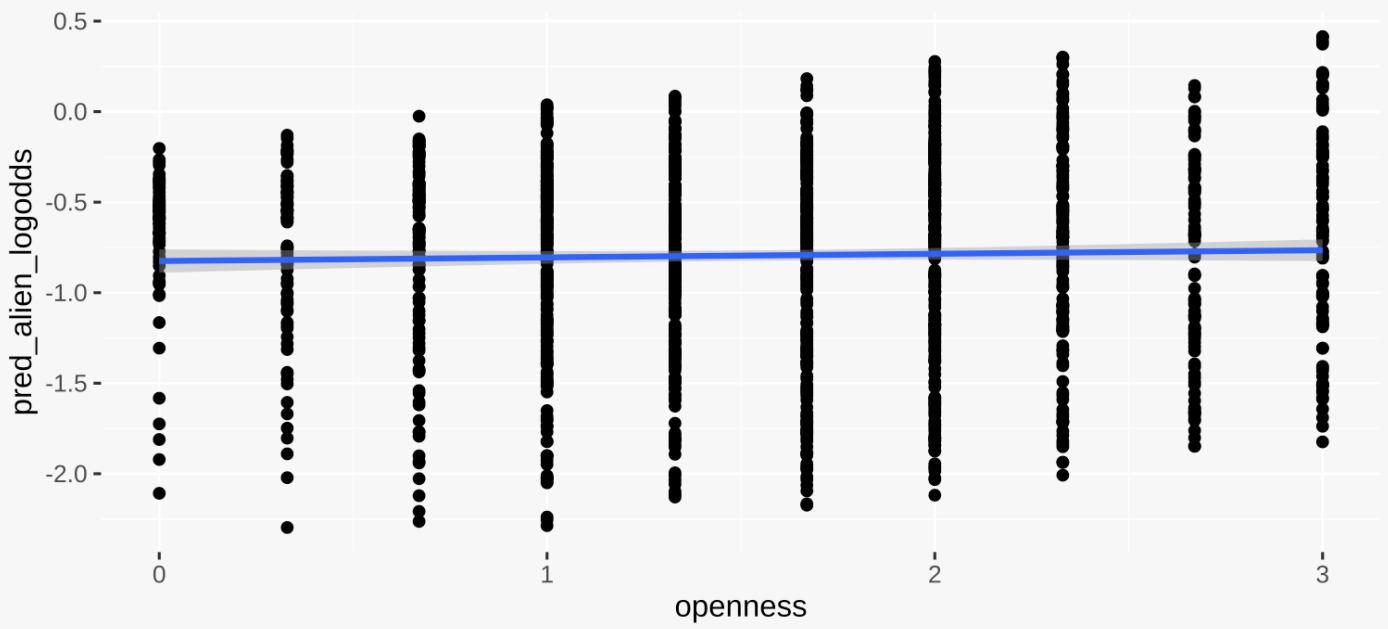
```
conspiracy_data %>%
  # Get predicted logodds
  mutate(
    pred_alien_logodds = predict(alien_model,
                                 type = "link",
                                 newdata = conspiracy_data)
  ) %>%
  # Plot logodds against continuous predictors
  ggplot() +
  geom_point(aes(x = rage, y = pred_alien_logodds)) +
  geom_smooth(aes(x = rage, y = pred_alien_logodds), method = "lm")
```



Assumptions

- **Linearity**
  - For continuous predictors only
  - Is a straight line the best fit to the data or would a curved line fit better?
  - Can use something called a Box-Tidwell test to check whether non-linearity is significant
- No outliers or leverage points
- No multicollinearity
- No autocorrelation

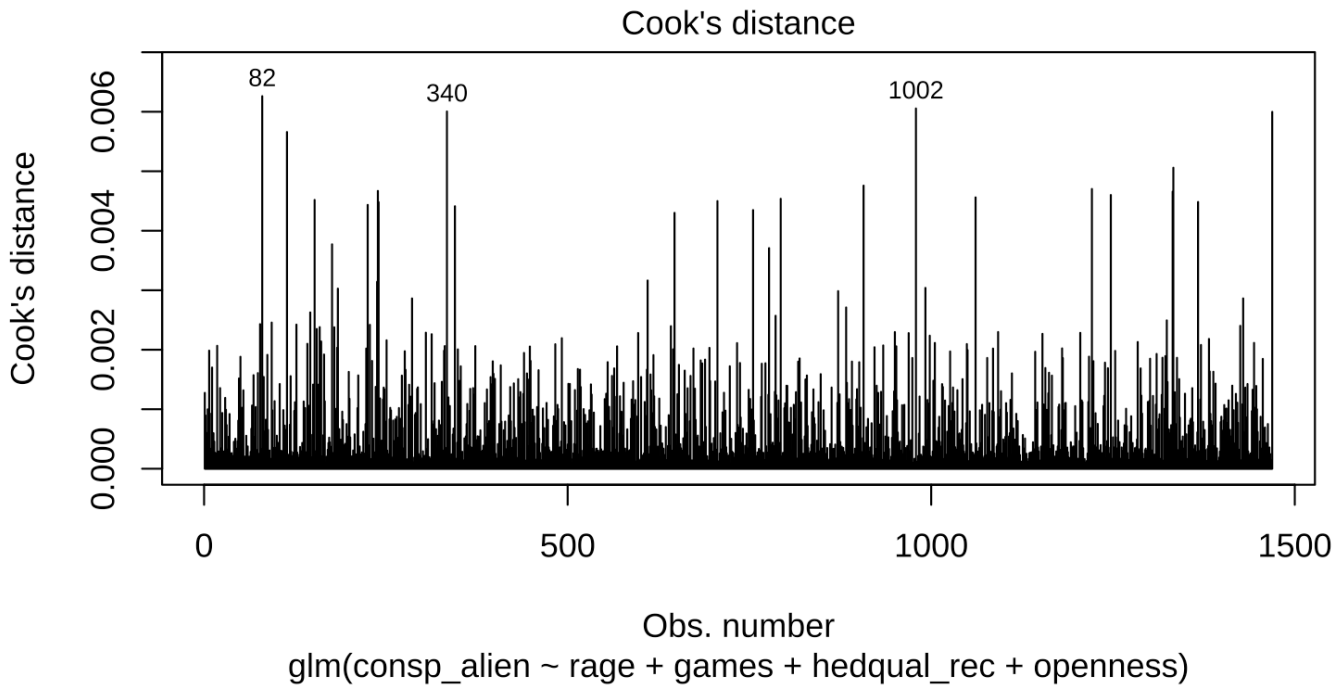
```
conspiracy_data %>%
  # Get predicted logodds
  mutate(
    pred_alien_logodds = predict(alien_model,
                                 type = "link",
                                 newdata = conspiracy_data)
  ) %>%
  # Plot logodds against continuous predictors
  ggplot() +
  geom_point(aes(x = openness, y = pred_alien_logodds)) +
  geom_smooth(aes(x = openness, y = pred_alien_logodds), method = "lm")
```



Assumptions

- Linearity
- **No outliers or leverage points**
    - Just like in multiple linear regression, we can use our Cook's Distance plots to try and identify any potential outliers.
- No multicollinearity
- No autocorrelation

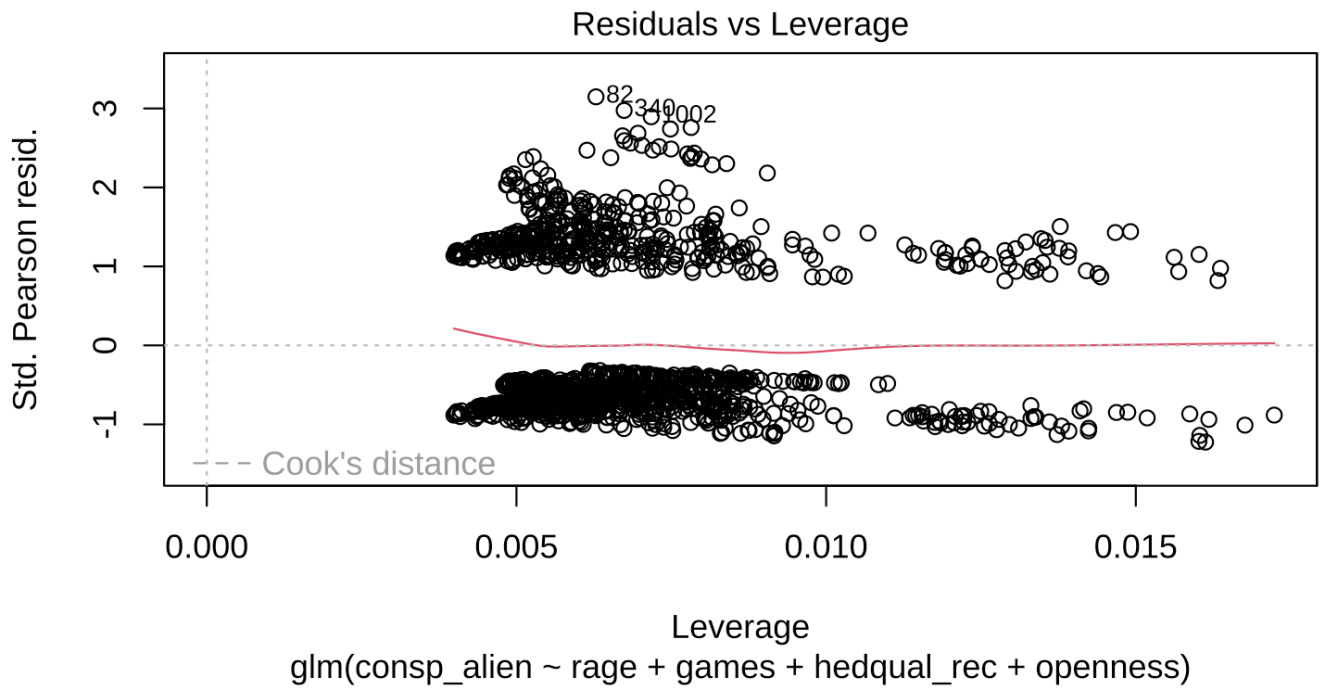
```
plot(alien_model, which = 4)
```



Assumptions

- Linearity
- **No outliers or leverage points**
    - Just like in multiple linear regression, we can use our Cook's Distance plots to try and identify any potential outliers.
- No multicollinearity
- No autocorrelation

```
plot(alien_model, which = 5)
```



Assumptions

- Linearity
- **No outliers or leverage points**
  - Just like in multiple linear regression, we can use our Cook's Distance plots to try and identify any potential outliers.
  - We can also use the convenience function `check_outliers` in the `performance` package
- No multicollinearity
- No autocorrelation

```
performance::check_outliers(alien_model)
```

```
## OK: No outliers detected.
## - Based on the following method and threshold: cook (0.8).
## - For variable: (Whole model)
```

Assumptions

- Linearity
- No outliers or leverage points
- **No multicollinearity**
  - Reminder: multicollinearity occurs when our independent variables are too closely related (linearly).
  - We can check using VIFs, the same was as with standard Multiple Linear Regression, using either the `car` or `performance` package. All of our independent variables' VIF should be less than 5.
- No autocorrelation

```
car::vif(alien_model)
```

```
##                GVIF Df GVIF^(1/(2*Df))
## rage       1.397780  1        1.182277
## games      1.242034  1        1.114466
## hedqual_rec 1.311885  6        1.022880
## openness   1.109380  1        1.053271
```

```
performance::multicollinearity(alien_model)
```

```
## # Check for Multicollinearity
##
## Low Correlation
##
##         Term  VIF   VIF 95% CI Increased SE Tolerance Tolerance 95% CI
##         rage 1.40 [1.32, 1.50]         1.18      0.72     [0.67, 0.76]
##        games 1.24 [1.18, 1.33]         1.11      0.81     [0.75, 0.85]
##  hedqual_rec 1.31 [1.24, 1.41]         1.15      0.76     [0.71, 0.81]
##     openness 1.11 [1.06, 1.19]         1.05      0.90     [0.84, 0.94]
```

Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity
- **No autocorrelation**

What is autocorrelation?

An assumption of general linear models that we have only briefly touched on (but is explored a lot in Advanced Quants methods' topics on multilevel modelling) is **autocorrelation**.

Just like our independent variables must be independent, our **observations should be independent**. Observations can become autocorrelated in a number of ways:

Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity
- **No autocorrelation**

What is autocorrelation?

An assumption of general linear models that we have only briefly touched on (but is explored a lot in Advanced Quants methods' topics on multilevel modelling) is **autocorrelation**.

Just like our independent variables must be independent, our **observations should be independent**. Observations can become autocorrelated in a number of ways:

- Multiple observations of the same person (over time, or in error)

Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity
- **No autocorrelation**

What is autocorrelation?

An assumption of general linear models that we have only briefly touched on (but is explored a lot in Advanced Quants methods' topics on multilevel modelling) is **autocorrelation**.

Just like our independent variables must be independent, our **observations should be independent**. Observations can become autocorrelated in a number of ways:

- Multiple observations of the same person (over time, or in error)
- Nested data structure (e.g. pupils in the same school will be more similar to each other than pupils in different schools)

Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity
- **No autocorrelation**

What is autocorrelation?

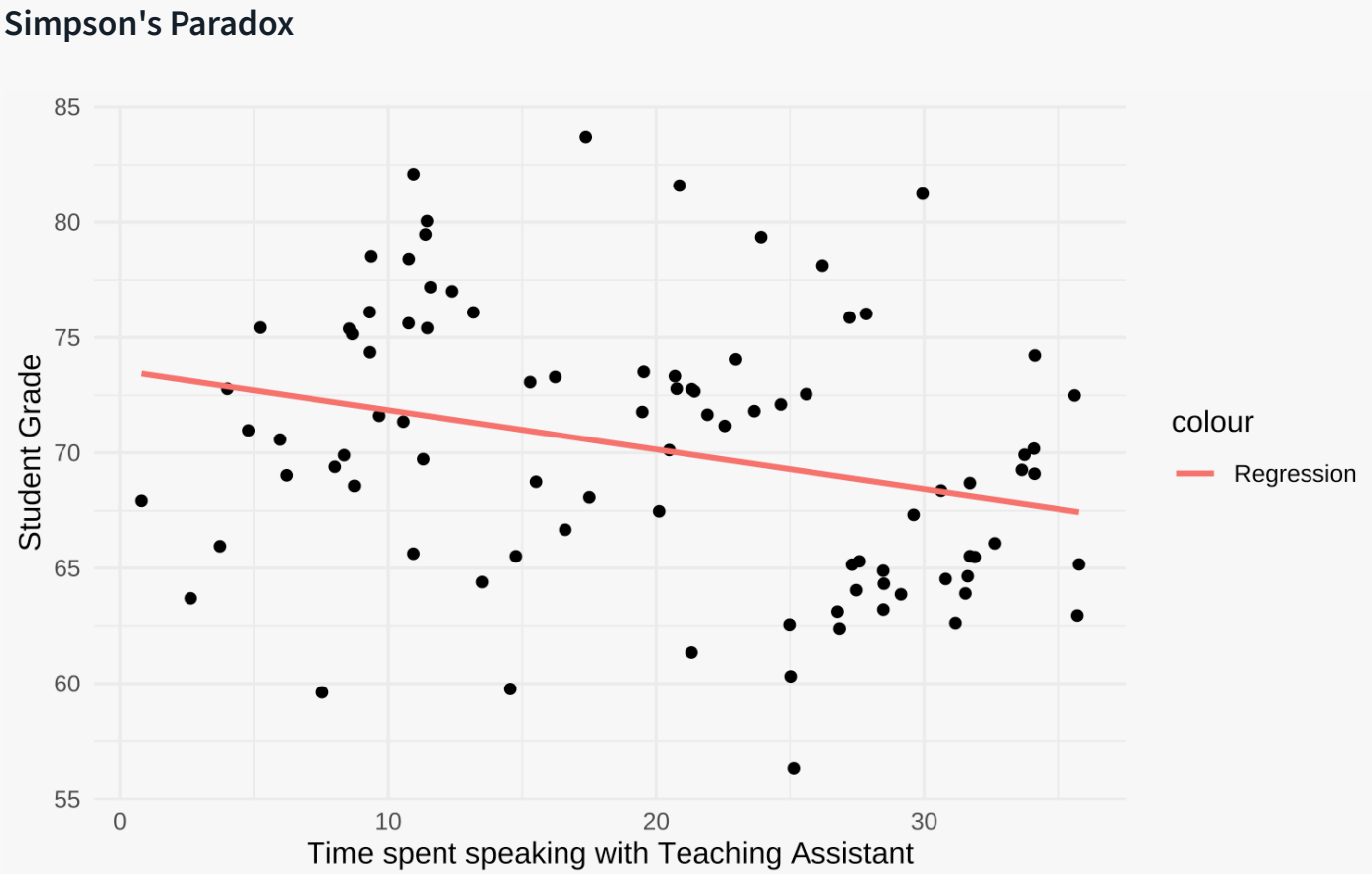
An assumption of general linear models that we have only briefly touched on (but is explored a lot in Advanced Quants methods' topics on multilevel modelling) is **autocorrelation**.

Just like our independent variables must be independent, our **observations should be independent**. Observations can become autocorrelated in a number of ways:

- Multiple observations of the same person (over time, or in error)
- Nested data structure (e.g. pupils in the same school will be more similar to each other than pupils in different schools)
- An important variable has not been included in the model, but is still exhibited in the data

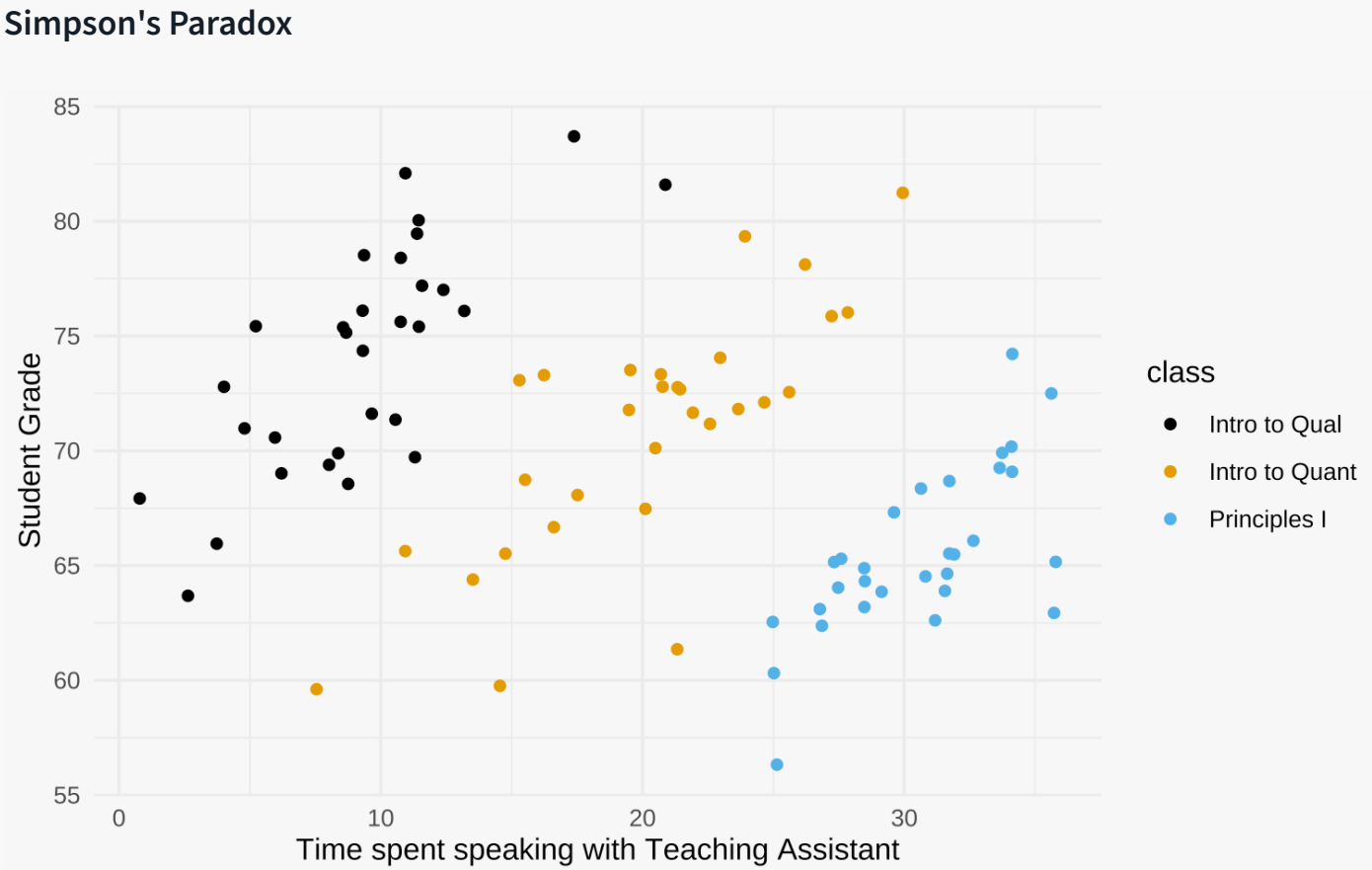
Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity
- **No autocorrelation**
  - Does spending more time with a Teaching Assistant improve a student's grades?



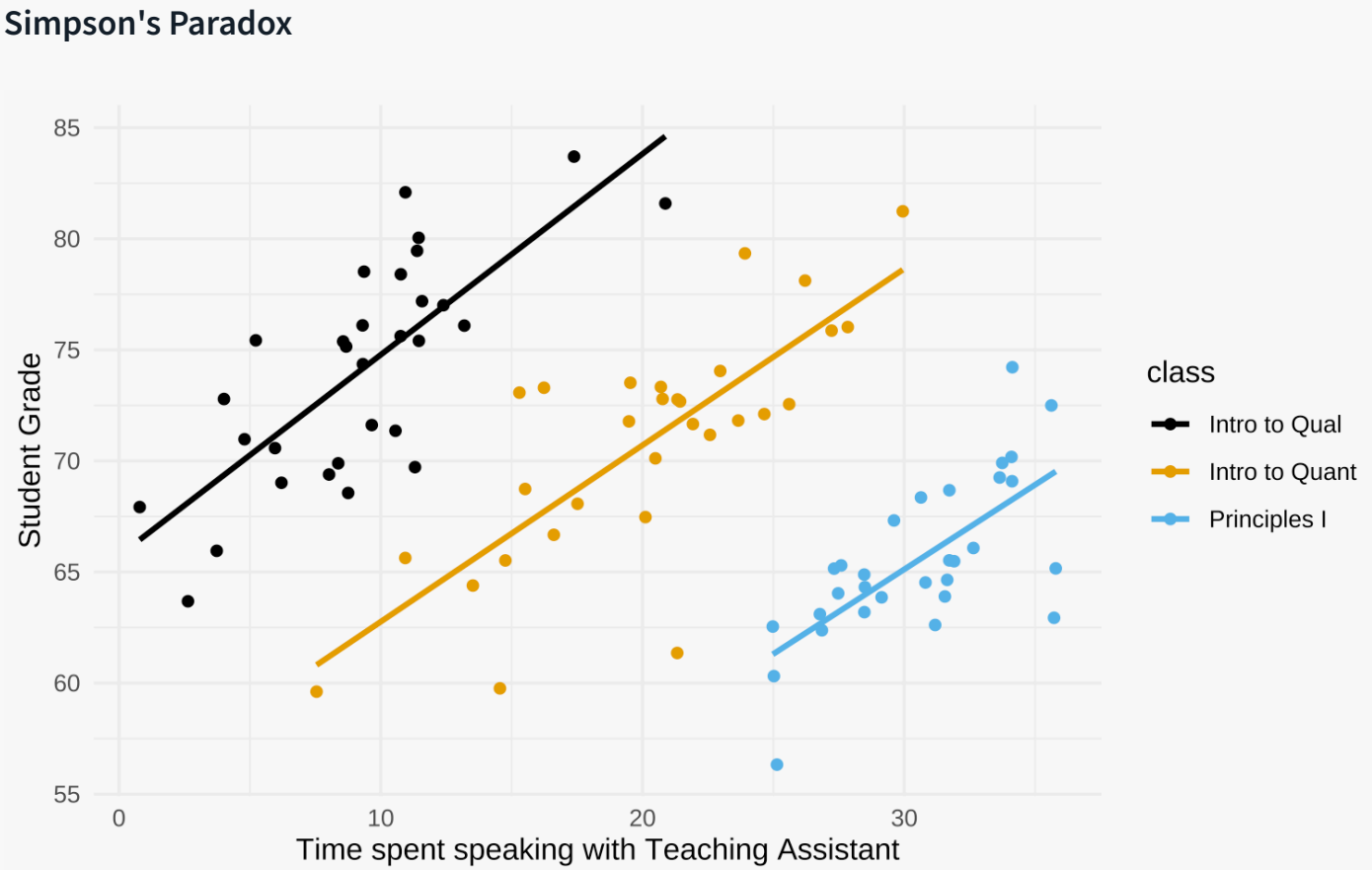
Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity
- **No autocorrelation**
  - Does spending more time with a Teaching Assistant improve a student's grades?



Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity
- **No autocorrelation**
  - Does spending more time with a Teaching Assistant improve a student's grades?



Assumptions: Autocorrelation

```
lm_without_class <- lm(sp_data,
                       formula = student_grade ~ time_with_ta)

summary(lm_without_class)
```

```
##
## Call:
## lm(formula = student_grade ~ time_with_ta, data = sp_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.9330  -3.7937  -0.0315   3.3053  13.1065
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  73.57857    1.36450  53.924  < 2e-16 ***
## time_with_ta -0.17174    0.06152  -2.792  0.00643 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.629 on 88 degrees of freedom
## Multiple R-squared:  0.08135,    Adjusted R-squared:  0.07091
## F-statistic: 7.793 on 1 and 88 DF,  p-value: 0.006433
```

```
lm_with_class <- lm(sp_data,
                    formula = student_grade ~ time_with_ta + class)

summary(lm_with_class)
```

```
##
## Call:
## lm(formula = student_grade ~ time_with_ta + class, data = sp_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.4383  -1.5488   0.3002   2.0286   6.6004
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              66.49609    0.92949   71.54   <2e-16 ***
## time_with_ta              0.82230    0.07913   10.39   <2e-16 ***
## classIntro to Quant     -12.23828    1.18316  -10.34   <2e-16 ***
## classPrinciples I       -26.07610    1.86911  -13.95   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.14 on 86 degrees of freedom
## Multiple R-squared:  0.7207,    Adjusted R-squared:  0.711
## F-statistic: 73.97 on 3 and 86 DF,  p-value: < 2.2e-16
```

The best defense against autocorrelation is knowing your data, its structure, and possible sources of omitted variable bias, as well as your subject area and theory!

Assumptions: Autocorrelation

However, we can test for autocorrelation using `car`'s Durbin Watson test...

```
car::durbinWatsonTest(lm_without_class)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1       0.2000415      1.559888   0.028
##  Alternative hypothesis: rho != 0
```

```
performance::check_autocorrelation(lm_without_class)
```

```
## Warning: Autocorrelated residuals detected (p = 0.044).
```

Or `performance`'s `check_autocorrelation` test.

```
car::durbinWatsonTest(lm_with_class)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     -0.03969624      2.026359   0.876
##  Alternative hypothesis: rho != 0
```

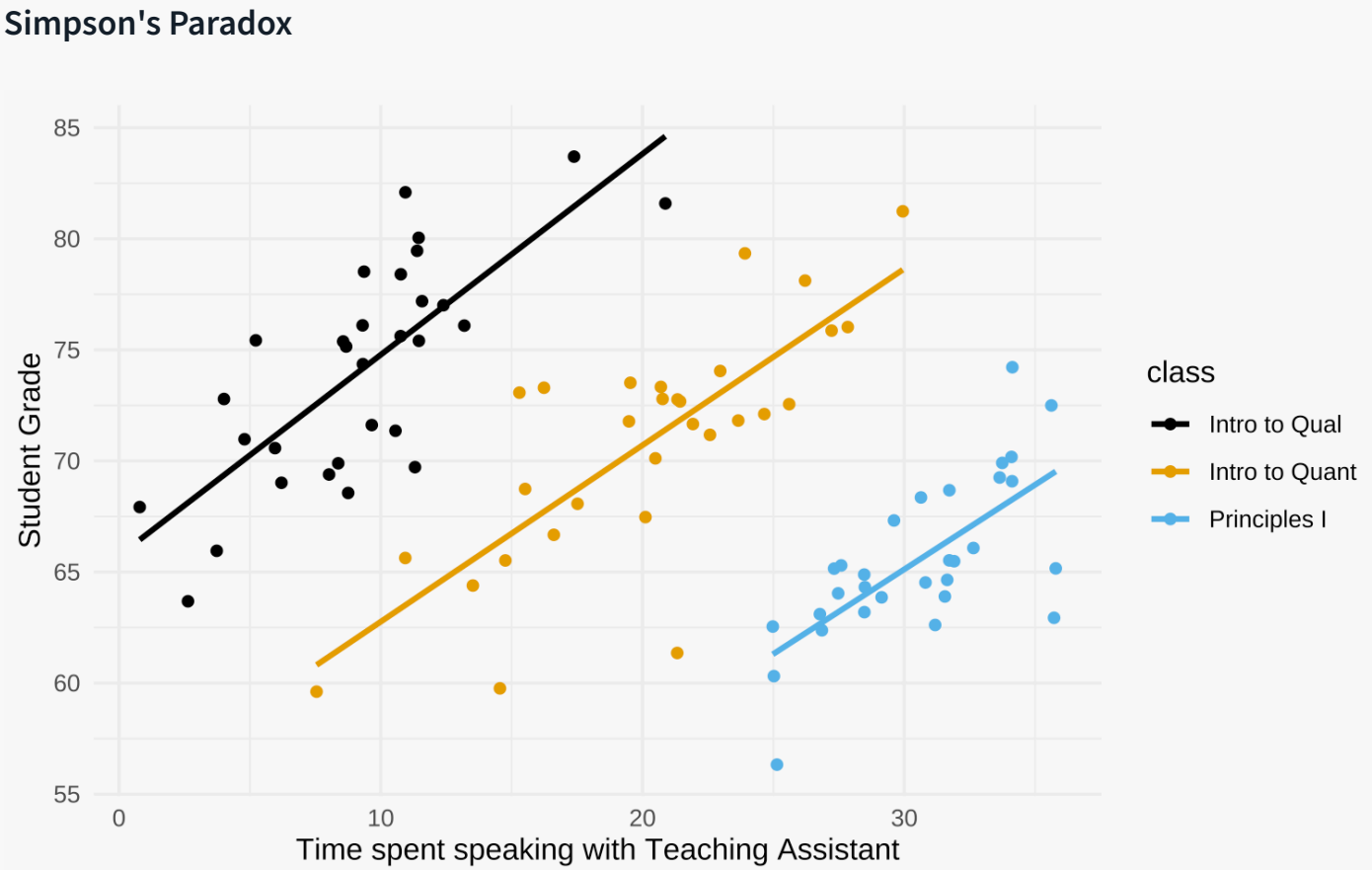
```
performance::check_autocorrelation(lm_with_class)
```

```
## OK: Residuals appear to be independent and not autocorrelated (p = 0.882).
```

p < 0.05 means we reject the idea that cases are independent (not autocorrelated).

Assumptions

- Linearity
- No outliers or leverage points
- No multicollinearity
- **No autocorrelation**
    - Observations are not independent.
    - Check using a Durbin-Watson test (but best to **know your data well**!)



Reminder - Cheat sheet on violated assumptions.

- **Linearity**

This means that your estimates will be over- and under-estimated at different parts of the line. For logistic regression, you can only resolve this by fitting a curvilinear model (non-linear relationship between X and Y) — bit more advanced but not too difficult!

- **Outliers and leverage points**

Outliers and leverage points can generally have an undue influence on the slope in a regression. Remove any error-based outliers (e.g. someone enters their age as 400 instead of 40). Compare results with 'true' outliers included and excluded to see how they differ, present both.

Reminder - Cheat sheet on violated assumptions.

- **Linearity**

This means that your estimates will be over- and under-estimated at different parts of the line. For logistic regression, you can only resolve this by fitting a curvilinear model (non-linear relationship between X and Y) — bit more advanced but not too difficult!

- **Outliers and leverage points**

Outliers and leverage points can generally have an undue influence on the slope in a regression. Remove any error-based outliers (e.g. someone enters their age as 400 instead of 40). Compare results with 'true' outliers included and excluded to see how they differ, present both.

- **No multicollinearity**

Try removing highly correlated variables and re-running the model to see how estimates change. Are the two+ variables measuring the same underlying process, are they poorly measured, is data highly aggregated? Report models with one variable removed to show bias caused.

Reminder - Cheat sheet on violated assumptions.

- **Linearity**

This means that your estimates will be over- and under-estimated at different parts of the line. For logistic regression, you can only resolve this by fitting a curvilinear model (non-linear relationship between X and Y) — bit more advanced but not too difficult!

- **Outliers and leverage points**

Outliers and leverage points can generally have an undue influence on the slope in a regression. Remove any error-based outliers (e.g. someone enters their age as 400 instead of 40). Compare results with 'true' outliers included and excluded to see how they differ, present both.

- **No multicollinearity**

Try removing highly correlated variables and re-running the model to see how estimates change. Are the two+ variables measuring the same underlying process, are they poorly measured, is data highly aggregated? Report models with one variable removed to show bias caused.

- **No autocorrelation**

Report Durbin-Watson test. Consider the source of autocorrelation. If from a measured variable (e.g. group), consider adding to the model. For very large numbers of groups (e.g. households), you may need a more complex **fixed effects** or **multilevel model**.

Summary & Practical.

Summary

- Logistic regression **allows us to explore any research question that uses a dependent variable that can be expressed as a binary outcome** with all the power of multiple linear regression.

Summary

- Logistic regression **allows us to explore any research question that uses a dependent variable that can be expressed as a binary outcome** with all the power of multiple linear regression.
- This works through the Maximum Likelihood Estimator using the odds, log odds, and logistic sigmoid function to **create a best fitting logistic sigmoid curve** to our underlying data.

Summary

- Logistic regression **allows us to explore any research question that uses a dependent variable that can be expressed as a binary outcome** with all the power of multiple linear regression.
- This works through the Maximum Likelihood Estimator using the odds, log odds, and logistic sigmoid function to **create a best fitting logistic sigmoid curve** to our underlying data.
- Logistic regression models can be estimated in `R` using the `glm()` function. Our results can be communicated as odds ratios by exponentiating our estimated change in log odds. Further, we can communicate our findings using the `ggeffects` package to produce predicted probabilities.

Summary

- Logistic regression **allows us to explore any research question that uses a dependent variable that can be expressed as a binary outcome** with all the power of multiple linear regression.
- This works through the Maximum Likelihood Estimator using the odds, log odds, and logistic sigmoid function to **create a best fitting logistic sigmoid curve** to our underlying data.
- Logistic regression models can be estimated in `R` using the `glm()` function. Our results can be communicated as odds ratios by exponentiating our estimated change in log odds. Further, we can communicate our findings using the `ggeffects` package to produce predicted probabilities.
- We can check our model fit using a range of tools, including pseudo-R^2s, model accuracy, and k-fold cross-validation.

Summary

- Logistic regression **allows us to explore any research question that uses a dependent variable that can be expressed as a binary outcome** with all the power of multiple linear regression.
- This works through the Maximum Likelihood Estimator using the odds, log odds, and logistic sigmoid function to **create a best fitting logistic sigmoid curve** to our underlying data.
- Logistic regression models can be estimated in `R` using the `glm()` function. Our results can be communicated as odds ratios by exponentiating our estimated change in log odds. Further, we can communicate our findings using the `ggeffects` package to produce predicted probabilities.
- We can check our model fit using a range of tools, including pseudo- R^2s, model accuracy, and k-fold cross-validation.
- We have a reduced number of assumptions to check for logistic regression, but we can check them using the `predict` function and `ggplot` (linearity with log odds), `plot()` function (outliers and leverage points), VIF statistics (multicollinearity), and Durbin-Watson test (autocorrelation). Autocorrelation also applies to OLS multiple linear models.

R Practical

What state-level variables are predictive of the presence of Neo-Nazi and anti-immigration hate groups in the United States?

- Download the `week-9-exercise.zip` file from Blackboard. Unzip the file and open the .Rproj file and associated .Rmd worksheet. Follow the steps to explore the above research question.