Random Topics

Last time...

Logistic Regression

Today

- Review & finish logistic regression
- Weighted Least Squares
- Work on HW 5

Maximum Likelihood Estimation

- Logistic regression is used when our outcomes are categorical
- In logistic regression we are not so lucky to be able to use OLS
- Pick parameters of your model (b_0, b_1 etc.), and calculate the *likelihood* of the data, given those parameters. We do this iteratively until we find the best parameters -- the ones that *maximize* the *likelihood* of your data.

GLM in R

```
glm(formula,
    family = gaussian(link="
    data,
    weights,
    subset,
    na.action,
    start = NULL,
    etastart,
    mustart,
    offset,
    control = glm.control(..
    model = TRUE,
    method = ”glm.fit”,
    x = FALSE,
    y = TRUE,
    contrasts = NULL, ...)
```

The `family` argument specifies the distribution. In R, families have default links.

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

GLM in R

```
glm(y ~ X1+ X2 + X3 ,
    family = binomial,
    data = dataset)
```

Specify the model like you would with `lm`

GLM in R

```
glm(y ~ X1+ X2 + X3 ,
    family = binomial,
    data = dataset)
```

Specify the distribution you're working with. When binary outcomes, we'll use the binomial.

GLM in R

```
glm(y ~ X1+ X2 + X3 ,
    family = binomial,
    data = dataset)
```

Specify your dataset.

- b_1 is the predicted change in the logit for a 1-unit change in X, holding the other predictors constant
- For a 1-unit change in X, holding other predictors constant, the odds that Y = 1 changes by e^{b_1}
  - e.g,. b_1 = .4, $e^{.4}$ = 1.49
- For fitted values, need to use entire equation $\hat{Y} = e^{b_0 + b_1 X_1}$
- Turn to probabilities by: $\frac{\text{odds}}{(1+\text{odds})}$

Example

```
# 1 = not premature
mortality
```

```
## # A tibble: 300 × 4
##    Intelligence_Self Intelligence_Mate premature.d NOT.premature
##                <dbl>             <dbl> <fct>                <dbl>
##  1                22                19 normal                   1
##  2                22                18 normal                   1
##  3                21                21 normal                   1
##  4                22                17 normal                   1
##  5                19                18 normal                   1
##  6                19                20 premature                0
##  7                16                18 normal                   1
##  8                15                11 premature                0
##  9                16                21 normal                   1
## 10                19                22 normal                   1
## # ℹ 290 more rows
```

```
death.1 <- lm(NOT.premature ~ Intelligence_Self , data = mort
summary(death.1)
```

```
##
## Call:
## lm(formula = NOT.premature ~ Intelligence_Self, data = mortality)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.9030  0.1084  0.1538  0.1907  0.3355
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.641769   0.098636   6.506 3.25e-10 ***
## Intelligence_Self 0.011357   0.005807   1.956   0.0514 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3745 on 298 degrees of freedom
## Multiple R-squared:  0.01267,    Adjusted R-squared:  0.009362
## F-statistic: 3.826 on 1 and 298 DF,  p-value: 0.05141
```

```
death.2 <- glm(NOT.premature ~ Intelligence_Self , data = mor
summary(death.2)
```

```
##
## Call:
## glm(formula = NOT.premature ~ Intelligence_Self, data = mortality)
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       0.641769   0.098636   6.506 3.25e-10 ***
## Intelligence_Self 0.011357   0.005807   1.956   0.0514 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 0.1402466)
##
##     Null deviance: 42.330  on 299  degrees of freedom
## Residual deviance: 41.793  on 298  degrees of freedom
## AIC: 266.05
##
## Number of Fisher Scoring iterations: 2
```

```
anova(death.1)
```

```
## Analysis of Variance Table
##
## Response: NOT.premature
##                      Df Sum Sq Mean Sq F value  Pr(>F)
## Intelligence_Self     1  0.537 0.53653  3.8256 0.05141 .
## Residuals           298 41.793 0.14025
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(death.2)
```

```
## Analysis of Deviance Table
##
## Model: gaussian, link: identity
##
## Response: NOT.premature
##
## Terms added sequentially (first to last)
##
##
##                    Df Deviance Resid. Df Resid. Dev
## NULL                                  299     42.330
## Intelligence_Self  1  0.53653       298     41.793
```

```
death.3 <- glm(NOT.premature ~ Intelligence_Self,
               family = binomial, data = mortality)
summary(death.3)
```

```
##
## Call:
## glm(formula = NOT.premature ~ Intelligence_Self, family = binomial,
##     data = mortality)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.28695    0.67490   0.425   0.6707
## Intelligence_Self  0.08012    0.04143   1.934   0.0532 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 273.53  on 299  degrees of freedom
## Residual deviance: 269.75  on 298  degrees of freedom
## AIC: 273.75
##
## Number of Fisher Scoring iterations: 4
```

Interpretation

For a 1-unit change in X, holding other predictors constant, the odds that Y = 1 changes by e^{b_1}

```
exp(1)^.08012
```

```
## [1] 1.083417
```

For every 1-unit increase in Intelligence, the odds of living increase by 8%

Interpreting Odds Ratios

What Does the OR Mean?

So, what does an OR mean? Here it is in plain language.

- An OR of 1.2 means there is a 20% increase in the odds of an outcome with a given exposure.
- An OR of 2 means there is a 100% increase in the odds of an outcome with a given exposure. Or this could be stated that there is a doubling of the *odds* of the outcome. Note, this is not the same as saying a doubling of the *risk*.
- An OR of 0.2 means there is an 80% decrease in the odds of an outcome with a given exposure.

Summary

- Odds Ratio is a measure of the strength of association with an exposure and an outcome.
  - OR > 1 means greater odds of association with the exposure and outcome.
  - OR = 1 means there is no association between exposure and outcome.
  - OR < 1 means there is a lower odds of association between the exposure and outcome.
- If the 95% confidence interval for the OR includes 1, the results are not statistically significant.
- OR and RR are not the same.
- OR always overestimate RR, but...
- OR approximates RR when the outcome is rare but markedly overestimates it as outcome exceeds 10%.

Specific Values?

What if you want the probability of being a premature death for a given level of Intelligence? (Now that we've run our model and have parameters...)

For fitted values, need to use entire equation $\hat{Y} = e^{b_0 + b_1 X_1}$

```
# get odds with a given value of X (here 20)
exp(1)^(0.28695 + (.08012*20))
```

```
## [1] 6.615067
```

```
# now get probability
6.615067 / (1+6.615067)
```

Probit

We can have different link functions. When your response variable (DV) is truly binary -- the data generating process generates legit binary data -- logit is your pick.

What if your response variable is binary, but the underlying construct you are trying to measure is likely Gaussian? Ex: depressed vs. not depressed. But the underlying latent construct is continuous. More appropriate then is the **probit** link function.

Stack exchange thread if you're going down this route

Probit

```
death.4 <- glm(NOT.premature ~ Intelligence_Self,
        family = binomial(link = "probit"), data = mortality)
summary(death.4)
```

```
##
## Call:
## glm(formula = NOT.premature ~ Intelligence_Self, family = binomial(lin
##     data = mortality)
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)        0.21961    0.38376   0.572   0.5671
## Intelligence_Self  0.04513    0.02319   1.946   0.0516 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 273.53  on 299  degrees of freedom
## Residual deviance: 269.72  on 298  degrees of freedom
## AIC: 273.72
##
```

- You can include covariates
- You can have interactions...BUT

Interactions are super hard to interpret in logistic regression

- We have nonlinear mapping (S-function)
- Can express in terms of odds, probabilities, or logits
- Whether your observe an interaction *depends* on if you express the outcome in terms of odds, probabilities, or logits...You can get very different results!
- Ultimately, you might want to use other techniques

Weighted Least Squares

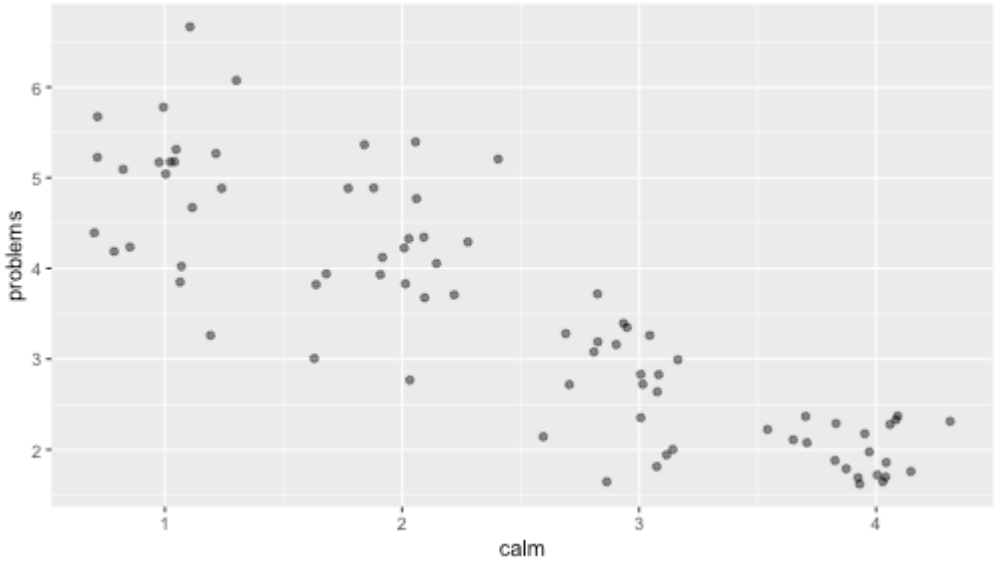
Estimation methods thus far

- OLS
- MLE

Why do we need another method?

Homoscedasticity

Homoscedasticity is the assumption that the variance of Y is constant across all levels of a predictor.



Do we meet this assumption?

Weighted least squares

Weighted least squares (WLS) is a commonly used remedial procedure for heteroscedasticity.

In an ordinary least squares (OLS) approach, each case in the dataset is given equal weight.

- WLS assigns each case a weight w_i, depending upon the precision of the observation of Y in that case.
- For observations for which the variance around the residuals around the regression line is low, the case is given a high weight.

OLS

Recall that an OLS estimation chooses values of b_0 and b_1 that minimizes the sum of squared residuals:

$$\min(\sum e_i^2) = \min \sum (Y_i - b_1 X_i - b_0)^2$$

WLS

Weights are taken into account, such that the values of b_0 and b_1 are chosen to minimize the sum of the **weighted** squared residuals:

$$\min(\sum w_i e_i^2) = \min \sum w_i (Y_i - b_1 X_i - b_0)^2$$

The value of the weights is the inverse of the conditional variance of the residuals corresponding to the specified value of X:

$$w_i = \frac{1}{\sigma^2_{Y-\hat{Y}|X}}$$

The value of $\sigma^2_{Y-\hat{Y}|X}$, the variance of the residuals in the population conditional on X, is not known and must be estimated.

A common procedure for estimating weights is to:

1. Estimate the usual OLS regression equation
2. Square the residuals
3. Regress the squared residuals onto X (x predicts squared residuals)

The weight is then estimated as the inverse of the predicted value for a case.

Our Data

Using our own data:

```
ols.model = lm(problems ~ calm, data = Data)
library(broom)
ols_aug = augment(ols.model)
head(ols_aug)
```

```
## # A tibble: 6 × 8
##   problems  calm .fitted .resid   .hat .sigma .cooksd .std.resid
##      <dbl> <dbl>   <dbl>  <dbl>  <dbl>  <dbl>   <dbl>      <dbl>
## 1        4 1.06     4.87 -0.872 0.0326  0.665 0.0296      -1.33
## 2        5 1.24     4.70  0.304 0.0278  0.671 0.00306      0.462
## 3        6 1.30     4.63  1.37  0.0263  0.653 0.0581       2.07
## 4        4 0.852    5.08 -1.08  0.0391  0.660 0.0558      -1.66
## 5        5 1.12     4.82  0.180 0.0311  0.672 0.00120      0.273
## 6        5 0.715    5.22 -0.223 0.0438  0.672 0.00266     -0.341
```

```
# square residuals
ols_aug$resid_sq = ols_aug$.resid^2
# regress squared resid on predictor
weight.mod = lm(resid_sq ~ calm, data = ols_aug)
```

```
coef(ols.model)
```

```
## (Intercept)        calm
##    5.941940   -1.005699
```

```
coef(weight.mod)
```

```
## (Intercept)        calm
##   1.0739505  -0.2591579
```

```
# extract predicted values
pred.resid = predict(weight.mod)
head(pred.resid)
```

```
##         1         2         3         4         5         6
## 0.7982546 0.7527584 0.7366885 0.8530291 0.7849316 0.8886457
```

```
# find inverse of predicted values
# use absolute value if some of your predicted values are neg
est.weights = 1/abs(pred.resid)
```

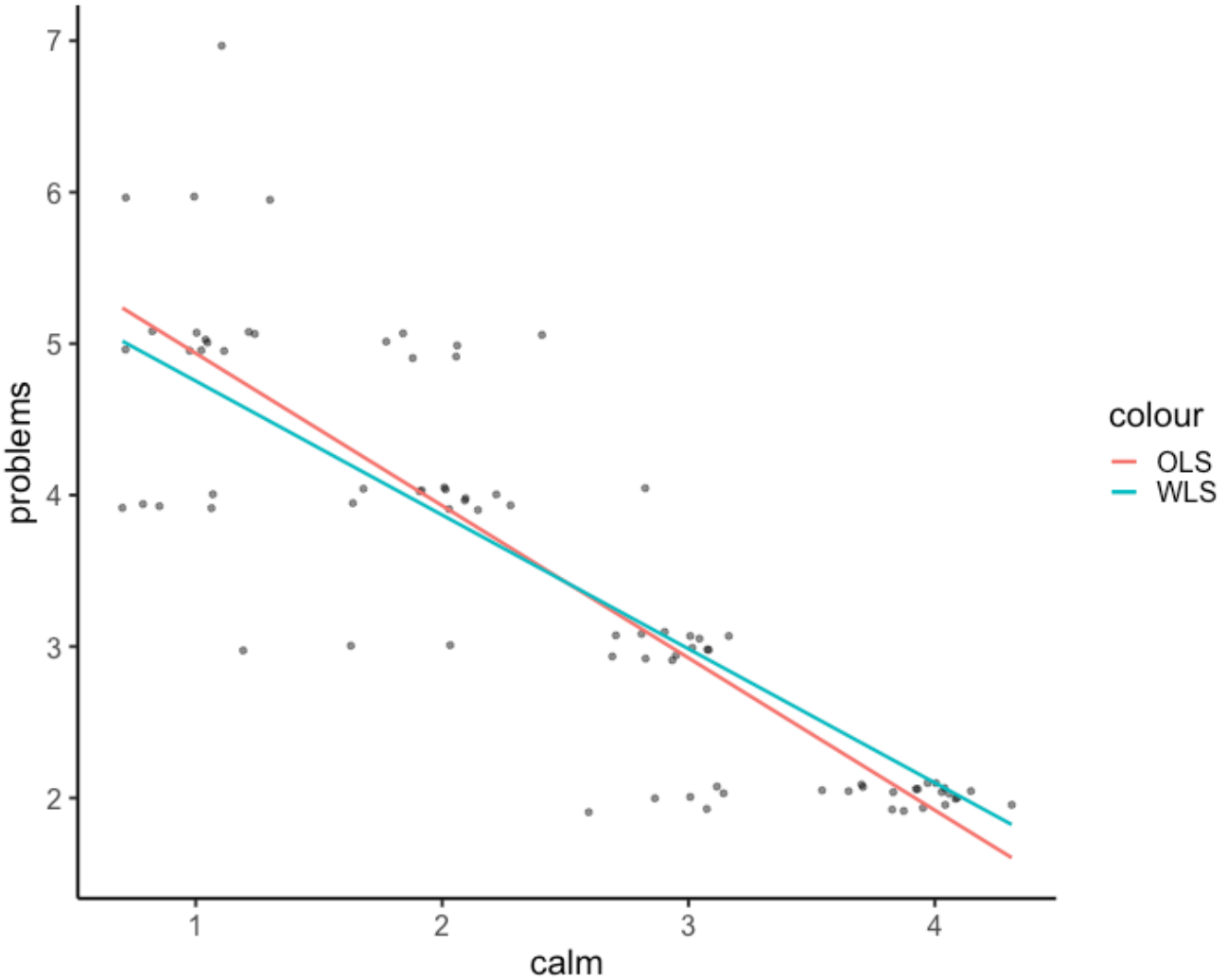
```
wls.model = lm(problems ~ calm, data = Data, weights = est.we
```

```
tidy(ols.model)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)     5.94    0.182       32.6 3.54e-47
## 2 calm           -1.01    0.0675     -14.9 1.58e-24
```

```
tidy(wls.model)
```

```
## # A tibble: 2 × 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    5.64     0.182       30.9 1.60e-45
## 2 calm          -0.884    0.0451     -19.6 7.58e-32
```


7
6
5
4
3
2
problems
1
2
3
4
calm
colour
OLS
WLS

When to use WLS

WLS is a **robust** method of estimation. **Robust statistics** are statistics with good performance for data drawn from a wide range of probability distributions, especially for distributions that are not normal." *(Wikipedia)*

Use when...

- Dealing with heteroscedasticity
- You know already that points should not be treated equally (some should be weighed more than others); do you know that some of your data was measured with less error?

Totally Random + HW Assignment

Checking Assumptions

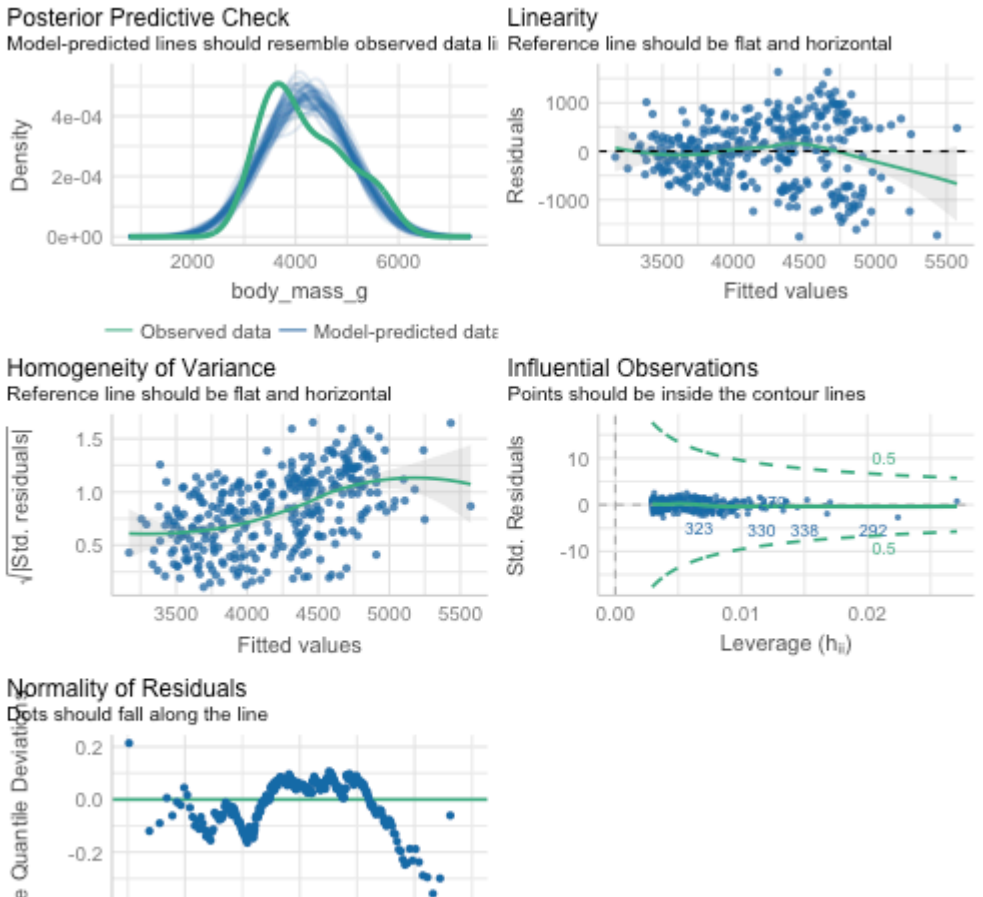
```
library(performance)
library(palmerpenguins)

model1 = lm(body_mass_g ~ bill_length_mm, data = penguins)
summary(model1)
```

```
##
## Call:
## lm(formula = body_mass_g ~ bill_length_mm, data = penguins)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1762.08  -446.98    32.59   462.31  1636.86
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     362.307    283.345   1.279    0.202
## bill_length_mm   87.415      6.402  13.654   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 645.4 on 340 degrees of freedom
```

Checking Assumptions

```
check_model(x = model1)
```



End of semester

- Tuesday = Resampling methods, namely bootstrapping
- Thursday = Machine learning; READ THE YARKONI & WESTFALL PAPER!
- Tuesday = Wrapping up, future directions, review
- Thursday = Exam 3. Oral exams will open after that.