Introduction to R: **Linear Regression**

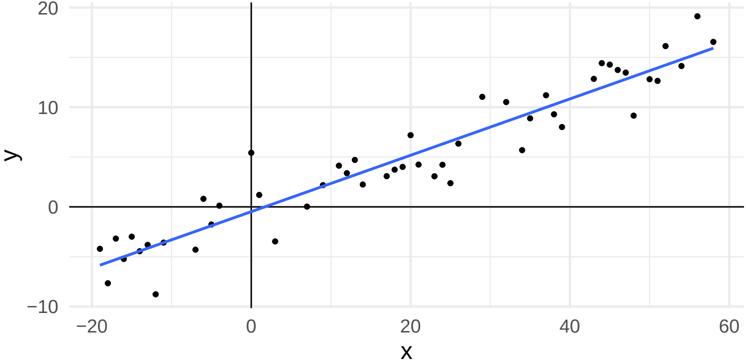
Day 3, Part B

In this lecture

1. Linear regression in a nutshell
2. Why do regression?
3. How to do regression
4. Useful things you can do with regressions
5. Problems and common pitfalls
6. Other forms of regression models

Linear regression in a nutshell

At the simplest level, linear regression is just a way to draw a straight line through the middle of some data.



Linear regression in a nutshell

What is the equation for a line?

Linear regression in a nutshell

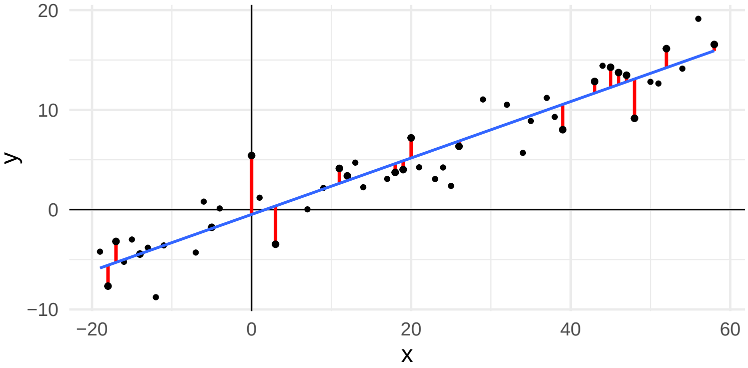
What is the equation for a line?

$$y = \beta_0 + \beta_1 \cdot x_1 + \epsilon$$

Linear regression is a method to solve for the “best” β_0 and β_1 so that the line goes through the middle of the data.

Linear regression in a nutshell

The “best fit” line is the one that minimizes the residuals, i.e., the distance from the line to each point.



Technically, we minimize the sum of the squared residuals.

Why do regression?

Statistics is all about measuring expected values

- Identify the "signal" in noisy data
- Form an expectation where there is no data

Regression techniques come with a lot of useful characteristics

- Calculates p-values and confidence intervals
- Easy to operate in multiple dimensions (i.e., "control" for other variables)

How to do regression?

How do I find the coefficients that minimize the residuals?

- Guess and check
- Matrix algebra ($(X'X)^{-1}X'Y$)
- Make your computer do it

The `lm()` function in R solves a linear model given data

The `lm` function

The `lm` function has two important arguments:

- `formula` - a `formula`-class object describing the x and y variables
- `data` - the name of a `data.frame` where R can find the variables

It creates a new type of object of the class `lm`, which contains the coefficients, confidence intervals, fit statistics, etc.

It comes with many associated functions:

- `summary()` - summarize the output and fit
- `coef()` - return the coefficients as a vector
- `confint()` - return the confidence intervals of the coefficients as a matrix
- `predict()` - return fitted values as a vector (including among `newdata`)

Example data

```
> data[1:20,]
     x         y
1    3 -3.466733
2   43 12.844468
3   12  3.383433
4   48  9.154658
5   52 16.134537
6  -17 -3.183839
7   19  4.012321
8   46 13.737814
9   20  7.195334
10  58 16.553953
11  47 13.471601
12  11  4.134794
13  26  6.345221
14  18  3.735093
15 -14 -4.451178
16  39  8.013233
17  -5 -1.769793
18 -18 -7.663491
19   0  5.422390
20  45 14.269905
```

Example regression output

```
> fit <- lm(y ~ x, data)
> summary(fit)

Call:
lm(formula = y ~ x, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-4.9009 -1.2432  0.0592  1.6096  5.9010

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.47859    0.40516  -1.181    0.243
x            0.28279    0.01344  21.042   <2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.234 on 48 degrees of freedom
Multiple R-squared:  0.9022,    Adjusted R-squared:  0.9002
F-statistic: 442.7 on 1 and 48 DF,  p-value: < 2.2e-16
```

Example data

```
> main_dir <- "C:/Users/ngraetz/Documents/repos/r_training_penn/"
> data <- read.csv(paste0(main_dir, "data/mmr_data_time_series.csv"))
> data <- data[, c("location_name", "super_region_name", "region_name",
+                  "year_id", "mmr", "maternal_education", "ldi")]
> head(data)
   location_name super_region_name    region_name year_id      mmr
1 United Kingdom       High-income Western Europe    1990 31.36912
2 United Kingdom       High-income Western Europe    1995 33.76302
3 United Kingdom       High-income Western Europe    2000 27.00985
4 United Kingdom       High-income Western Europe    2005 30.50040
5 United Kingdom       High-income Western Europe    2010 26.52158
6 United Kingdom       High-income Western Europe    2015 33.02467
  maternal_education      ldi
1           10.98886 22304.07
2           11.74322 24543.84
3           12.42620 28378.34
4           13.06440 32807.42
5           13.67682 34932.26
6           14.24481 35569.39
```

Multivariate regression

```
> mod <- lm(mmr ~ ldi + maternal_education, data = data)
```

```
> summary(mod)

Call:
lm(formula = mmr ~ ldi + maternal_education, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-395.32 -166.52  -20.38  114.87  702.29

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)        821.281812  73.515093  11.172 5.55e-16 ***
ldi                  0.011897   0.004959   2.399   0.0197 *
maternal_education -94.429170  16.384075  -5.763 3.50e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 239.9 on 57 degrees of freedom
Multiple R-squared:  0.5407,    Adjusted R-squared:  0.5246
F-statistic: 33.55 on 2 and 57 DF,  p-value: 2.342e-10
```

Interpretation of regression output

```
> summary(mod)$coefficients
                      Estimate  Std. Error   t value     Pr(>|t|)
(Intercept)        821.28181228 73.515092605 11.171608 5.548333e-16
ldi                  0.01189679  0.004958562  2.399242 1.972142e-02
maternal_education -94.42917000 16.384074579 -5.763473 3.498165e-07
```

There are multiple valid ways to "read" the coefficients:

- **As a best fit line**: "for every 1-unit increase in `maternal_education`, the expected value of `mmr` declined by 94.4"
- **As an average**: "the mean `mmr` when `ldi` and `maternal_education` are zero was estimated at 821.3, the mean `mmr` when maternal education is 1 was (821.3 + -94.4)"
- **As correlation**: "`maternal_education` had a statistically significant negative correlation with `mmr` controlling for `ldi`; `ldi` had a small, positive correlation with `mmr`, controlling for `maternal_education`"

Note: Nothing about this is causal, or even a statement about the meaning of the data. **Regression coefficients are just statements about the relationships observed in the data.**

Dummy variables

A "dummy variable" is the same as a binary variable (something with just 0's and 1's).

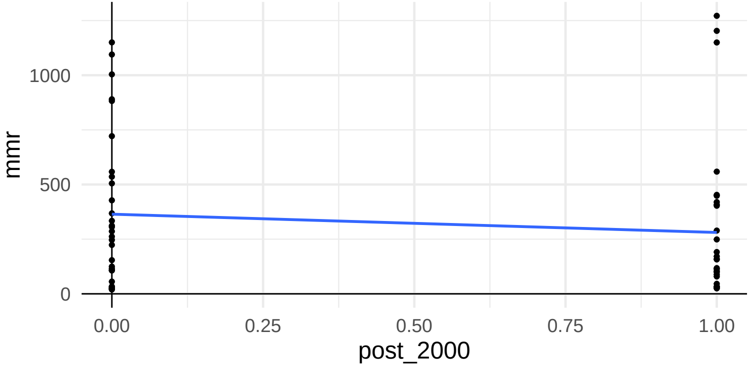
```
> data$post_2000 <- ifelse(data$year_id>2000, 1, 0)
> head(data[,c(1:5,8)])
   location_name super_region_name    region_name year_id      mmr post_2000
1 United Kingdom       High-income Western Europe    1990 31.36912         0
2 United Kingdom       High-income Western Europe    1995 33.76302         0
3 United Kingdom       High-income Western Europe    2000 27.00985         0
4 United Kingdom       High-income Western Europe    2005 30.50040         1
5 United Kingdom       High-income Western Europe    2010 26.52158         1
6 United Kingdom       High-income Western Europe    2015 33.02467         1
```

This can be very useful in linear regression:

```
> mod <- lm(mmr ~ post_2000, data=data)
> summary(mod)$coefficients
             Estimate Std. Error    t value     Pr(>|t|)
(Intercept) 364.38105   63.58588  5.7305339 3.774187e-07
post_2000   -83.64283   89.92401 -0.9301501 3.561488e-01
```

Dummy variables

What did the regression do with the dummy variable?



Dummy variables

R will take any factor variable and turn it into a series of dummy variables for you.

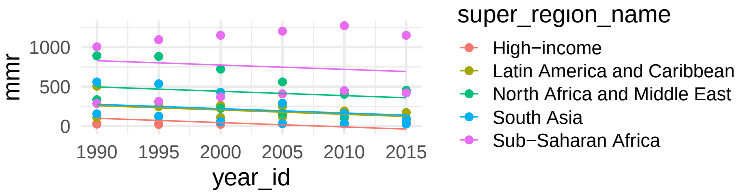
```
> unique(data$super_region_name)
[1] High-income                       Latin America and Caribbean
[3] North Africa and Middle East South Asia
[5] Sub-Saharan Africa
5 Levels: High-income ... Sub-Saharan Africa
> mod <- lm(mmr ~ factor(super_region_name), data=data)
> summary(mod)$coefficients[,c(1,4)]
                                                      Estimate     Pr(>|t|)
(Intercept)                                           29.85047 6.752600e-01
factor(super_region_name)Latin America and Caribbean  159.87394 1.164220e-01
factor(super_region_name)North Africa and Middle East 397.13799 2.157322e-04
factor(super_region_name)South Asia                   176.71777 8.343038e-02
factor(super_region_name)Sub-Saharan Africa           729.81611 1.304003e-09
```

Question: Where did the “high income” super region go?

Dummy variables

Another way to think about a dummy variable is as an adjustment to the intercept:

```
> mod <- lm(mmr ~ factor(super_region_name) + year_id, data=data)
> summary(mod)$coefficients[,c(1,4)]
                                                      Estimate     Pr(>|t|)
(Intercept)                                       11023.381041 1.395437e-01
factor(super_region_name)Latin America and Caribbean   159.873938 1.125894e-01
factor(super_region_name)North Africa and Middle East   397.137992 1.899649e-04
factor(super_region_name)South Asia                  176.717767 8.022992e-02
factor(super_region_name)Sub-Saharan Africa          729.816105 1.060479e-09
year_id                                              -5.489903 1.405793e-01
```



In other words, this allows for a different intercept for each super region.

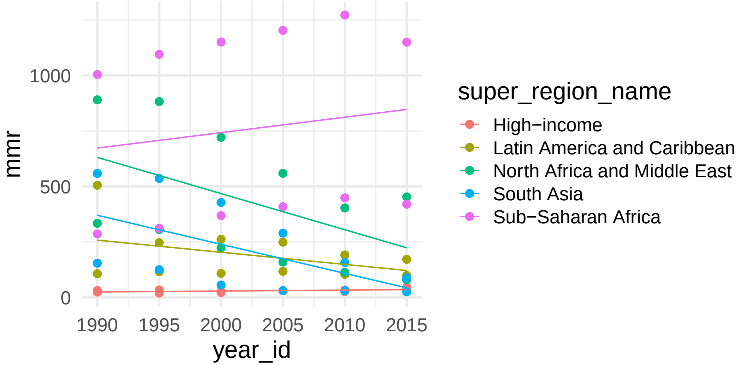
Interactions

An interaction term allows a different slope at different levels of a variable. It is literally just the arithmetic product of other variables.

In R, you specify it in the formula with the * symbol (R assumes you want the product and the "main effects")

```
> mod <- lm(mmr ~ factor(super_region_name) * year_id, data=data)
> summary(mod)$coefficients[,c(1,4)]
                                                                  Estimate  Pr(>|t|)
(Intercept)                                                  -8.180138e+02 0.9599404
factor(super_region_name)Latin America and Caribbean          1.194394e+04 0.6045337
factor(super_region_name)North Africa and Middle East         3.386656e+04 0.1457280
factor(super_region_name)South Asia                           2.720399e+04 0.2408022
factor(super_region_name)Sub-Saharan Africa                  -1.234397e+04 0.5925198
year_id                                                       4.234029e-01 0.9584795
factor(super_region_name)Latin America and Caribbean:year_id -5.884677e+00 0.6093631
factor(super_region_name)North Africa and Middle East:year_id -1.671382e+01 0.1504075
factor(super_region_name)South Asia:year_id                  -1.349677e+01 0.2438253
factor(super_region_name)Sub-Saharan Africa:year_id           6.528731e+00 0.5708937
```

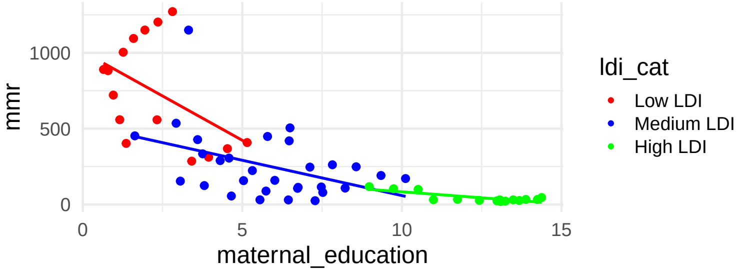
Interactions



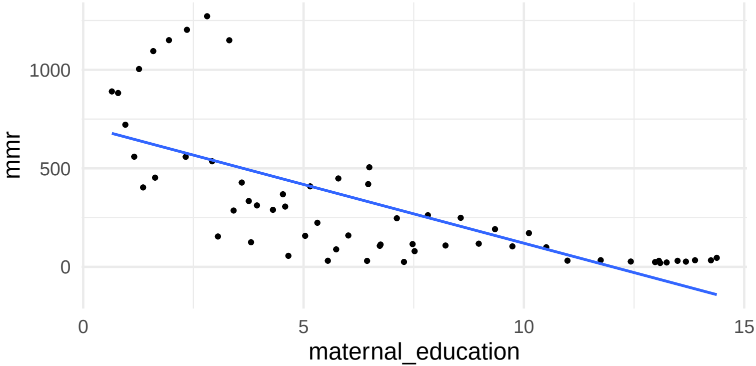
Interactions

Interactions can also be applied to two continuous variables

```
> mod <- lm(mmr ~ maternal_education * ldi, data=data)
> summary(mod)$coefficients
                            Estimate   Std. Error   t value     Pr(>|t|)
(Intercept)              861.060283950 74.069576229 11.625020 1.477376e-16
maternal_education       -84.496920809 16.650175454 -5.074837 4.578632e-06
ldi                       -0.029088371  0.020471254 -1.420937 1.608796e-01
maternal_education:ldi     0.002756018  0.001337822  2.060079 4.404946e-02
```



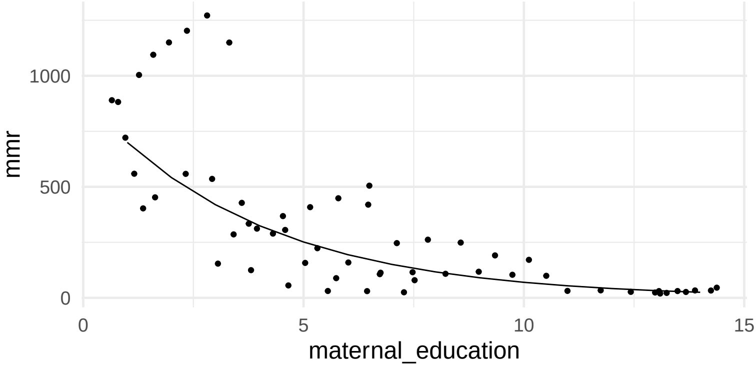
Often the relationship between `x` and `y` is poorly-approximated by a straight line:



Transformations

Transformations to the data can help approximate a linear relationship, and can be added directly into the regression formula

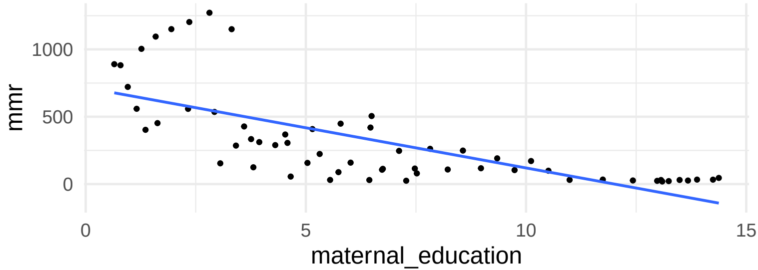
```
> mod <- lm(log(mmr) ~ maternal_education, data=data)
```



Predictions

"Fitted values" can be obtained using simple algebra:

```
> mod <- lm(mmr ~ maternal_education, data=data)
> betas <- coef(mod)
> betas
      (Intercept) maternal_education
        716.13807          -59.61979
> betas[1] + betas[2]*5
(Intercept)
   418.0391
```



Predictions

The `predict()` function can be used to obtain fitted values more conveniently:

```
> data$predictions <- predict(mod)
> head(data[, c(1, 2, 3, 5, 6, 8)], 3)
  location_name super_region_name    region_name      mmr
1 United Kingdom       High-income Western Europe 31.36912
2 United Kingdom       High-income Western Europe 33.76302
3 United Kingdom       High-income Western Europe 27.00985
  maternal_education post_2000
1           10.98886         0
2           11.74322         0
3           12.42620         0
```

It also allows you to pass new data to it:

```
> mod <- lm(mmr ~ year_id, data = data)
> more_data <- data.frame(year_id = seq(1990, 2020))
> more_data$predictions <- predict(mod, newdata = more_data)
> head(more_data, 3)
  year_id predictions
1    1990    391.1834
2    1991    385.6935
3    1992    380.2036
```

Predictions

You can also use `predict()` to extract confidence or prediction intervals:

```
> preds <- predict(mod, interval = "confidence")
> head(preds, 3)
       fit      lwr      upr
1 391.1834 231.9338 550.4330
2 363.7339 244.1761 483.2917
3 336.2844 242.6848 429.8840
>
> data <- cbind(data, preds)
> head(data[, c(1, 5, 12, 13, 14)])
   location_name      mmr      fit       lwr      upr
1 United Kingdom 31.36912 391.1834 231.93380 550.4330
2 United Kingdom 33.76302 363.7339 244.17611 483.2917
3 United Kingdom 27.00985 336.2844 242.68482 429.8840
4 United Kingdom 30.50040 308.8349 215.23530 402.4344
5 United Kingdom 26.52158 281.3854 161.82756 400.9432
6 United Kingdom 33.02467 253.9358  94.68623 413.1855
```

Problems and common pitfalls

Linear regression relies on a number of assumptions in order to say it's the "best-fit" line:

- Linearity - the relationship between X and Y is well-approximated by a straight line
- Homoscedasticity - the variance of Y is unrelated to the level of X
- Zero autocorrelation - the magnitude of the residual is unrelated to the level of X (or the magnitude of other residuals)
- Normally-distributed residuals with mean of 0 - the residuals tend toward zero, symmetrically falling on either side of the line

Problems and common pitfalls

There are potential fixes to violations of each of these problems:

- Linearity - transform the data, use a different form of regression (GLM)
- Homoscedasticity - compute "Homoscedasticity-consistent standard errors"
- Zero autocorrelation - control for previous values of X (with additional variables), use a different form of regression (AR models, random effects, etc.)
- Normally-distributed residuals with mean of 0 - use a different form of regression (GLM)

Other forms of regression models

There are different "flavors" of regression for every type of data:

- Continuous, normal data - ordinary linear regression
- Count data - Poisson regression (simpler), Negative binomial regression (better for "real-world" data)
- Binary data - Logistic regression
- Categorical data - Multinomial logistic regression

And many ways to account for complex residuals:

- Random effects - similar to fixed effects, but draws on more assumptions about normality, incorporates hierarchies better
- AR and ARIMA models - builds autocorrelation of residuals into the fitting process
- Zero-inflated models - combines logistic regression and another form of regression to approximate data with more complex data generating processes
- Proportional hazards models - estimates "hazards" instead of odds, accounts for censored data

Other forms of regression models

There is a function (and package) for fitting any model you can imagine:

Function	Package	Model types
`glm()`	`stats`	Generalized linear models (e.g., logistic regression, Poisson regression, etc.)
`glm.nb()`	`MASS`	Negative binomial GLMs
`lmer()`	`lme4`	Linear mixed effects models
`glmer()`	`lme4`	Generalized linear mixed effects models
`arima()`	`stats`	ARIMA time series models
`coxph()`	`surv`	Cox proportional hazards model
`inla()`	`INLA`	Linear models, GLMs, mixed effects models, and generalized additive models (Bayesian)