

Linear Models

Introduction to R - Day 3

Instructor: [Selina Baldauf](#)

Freie Universität Berlin - Theoretical Ecology

A little bit of theoretical background

Linear models background

The data:

- random sample of n data points ($Y_i, X_{i1}, X_{i2}, \dots, X_{ip}, i = 1 \dots n$)
 - $X_{i1}, X_{i2}, \dots, X_{ip}$ are p **independent predictor** variables
 - Y_i is the **dependent** observation and **response variable**

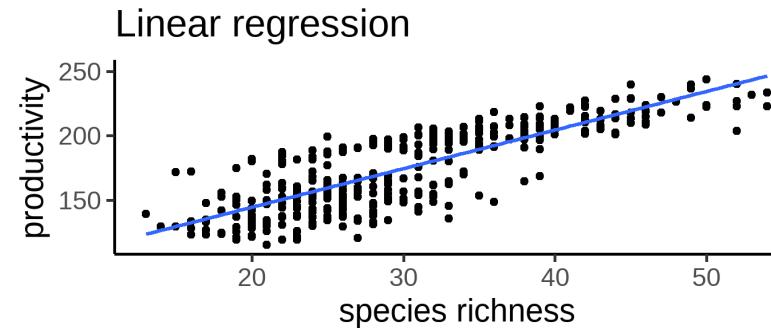
Aim of linear model:

- (How) do the predictor variables impact the response variable?
- Can we better predict values for the response variable, if we account for the predictor variables?

Linear models background

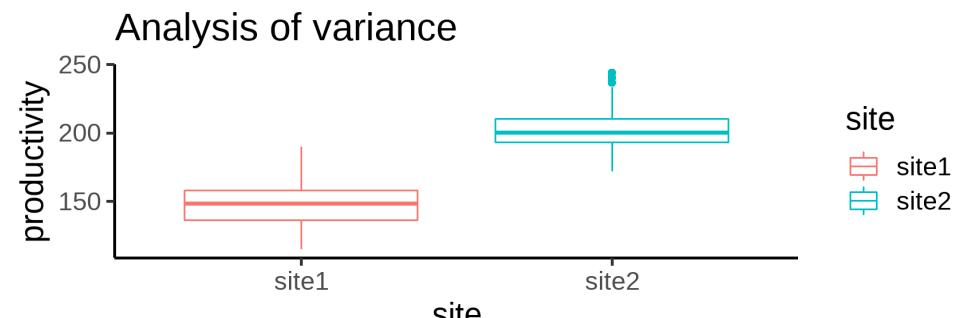
Linear regression

- numerical response
- numerical predictor(s)



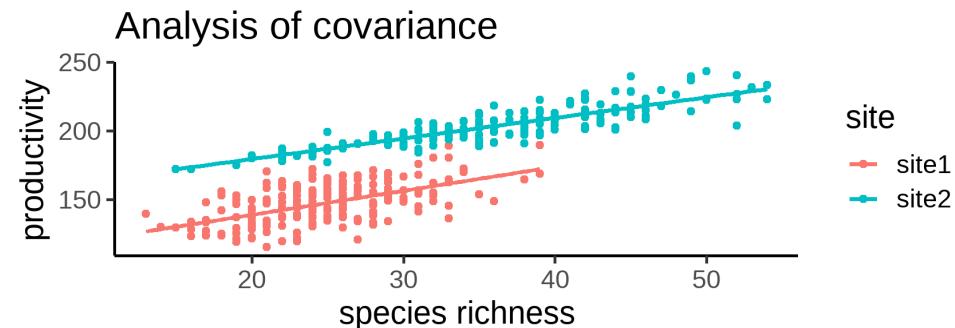
Analysis of variance

- numerical response
- categorical predictor(s)



Analysis of covariance

- numerical response
- numerical and categorical predictor(s)



Linear models background

Linear relation with two predictors X_1 and X_2 ($i = 1 \dots n$)

Without interaction

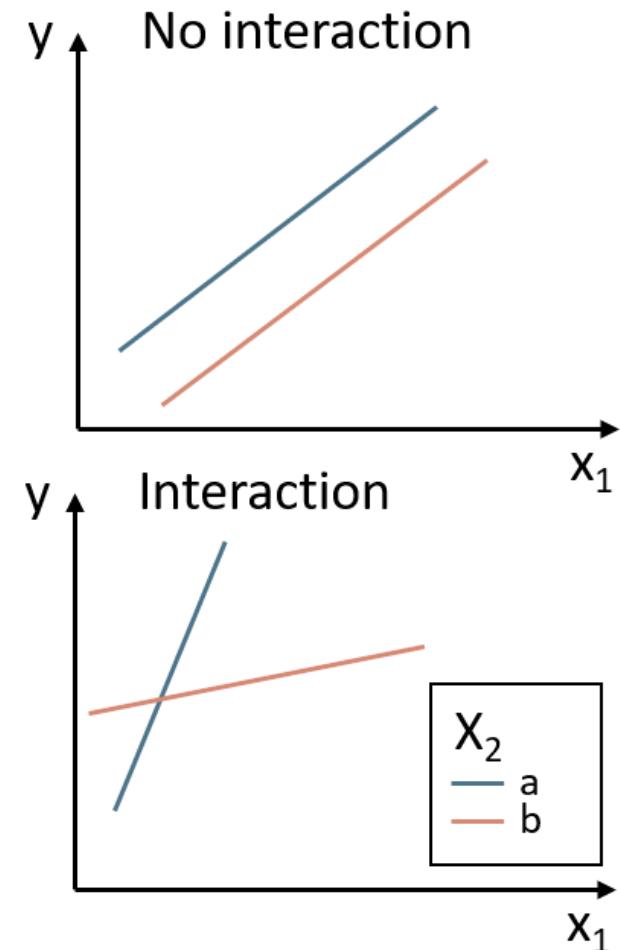
$$Y_i = \beta_0 + \beta_1 * X_{1,i} + \beta_2 * X_{2,i} + \epsilon$$

With interaction

$$Y_i = \beta_0 + \beta_1 * X_{1,i} + \beta_2 * X_{2,i} + \beta_3 * X_{1,i} * X_{2,i} + \epsilon$$

with

- Y_i value of response variable
- $\beta_0, \beta_1, \beta_2$ model coefficients
- $X_{1,i}$ value of predictor X_1
- $X_{2,i}$ value of predictor X_2
- ϵ error term



Goodness of fit linear model

Model residuals

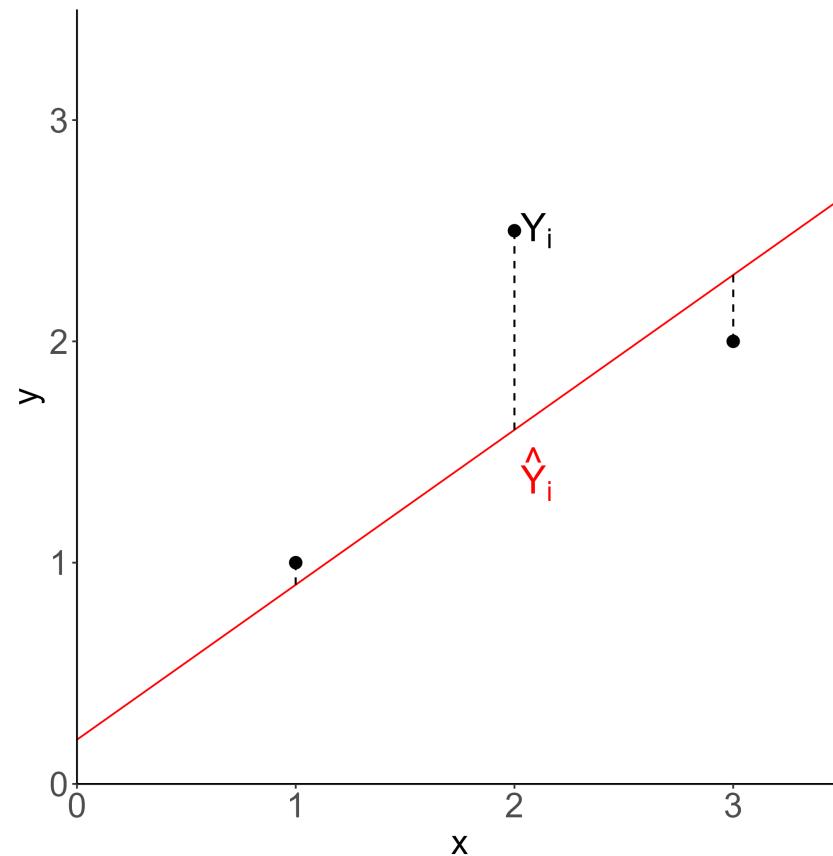
$$Y_i - \hat{Y}_i$$

Residual sum of squares (RSS)

$$RSS = \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

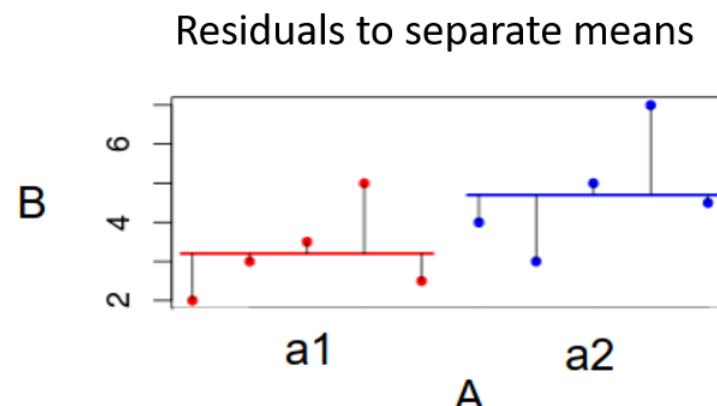
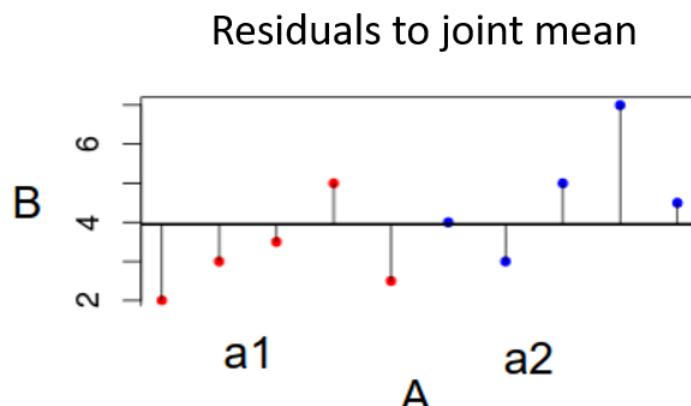
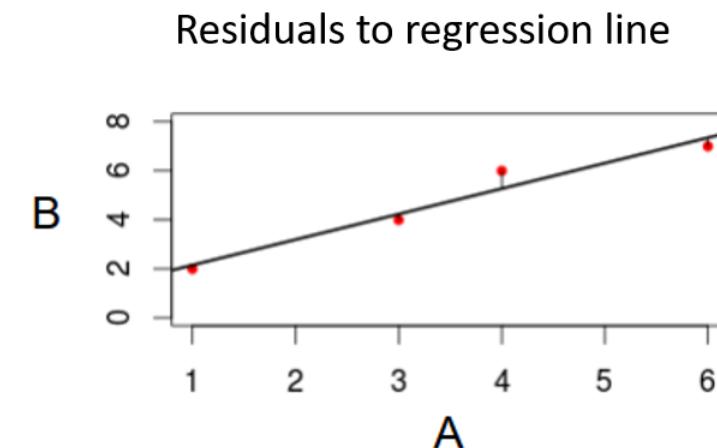
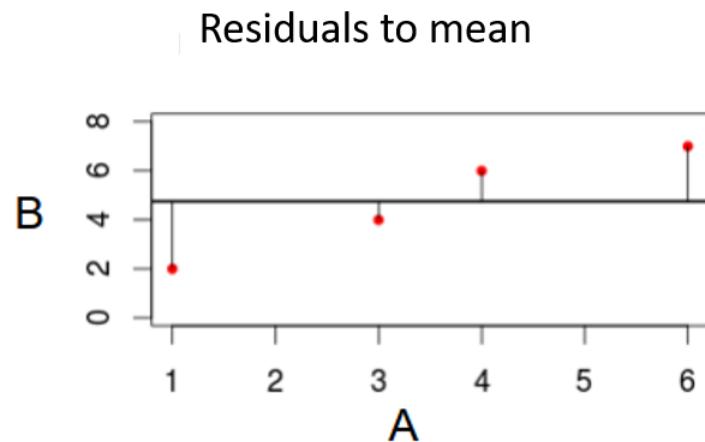
Aim of the model fitting:

Find the model coefficients that lead to the lowest sum of squares.



Significant effects

If we account for variable A: do the residuals become significantly smaller?



Assumptions for linear models

Finding the model estimates that lead to the lowest residual sum of squares works only if:

- 1) Residuals are normally distributed
- 2) Residual variance is constant
- 3) No strong outliers

This has to be checked for every linear model!

If assumptions are not fulfilled, a different model approach has to be chosen.

How to fit linear models in

The data

We use productivity data from grassland sites.

The data set is called `prod` and has 3 variables: site, productivity and richness (species richness):

```
prod
```

```
## # A tibble: 400 x 3
##   site   richness productivity
##   <fct>    <dbl>        <dbl>
## 1 site1      23        122.
## 2 site1      26        158.
## 3 site1      24        165
## 4 site1      25        165.
## # ... with 396 more rows
```

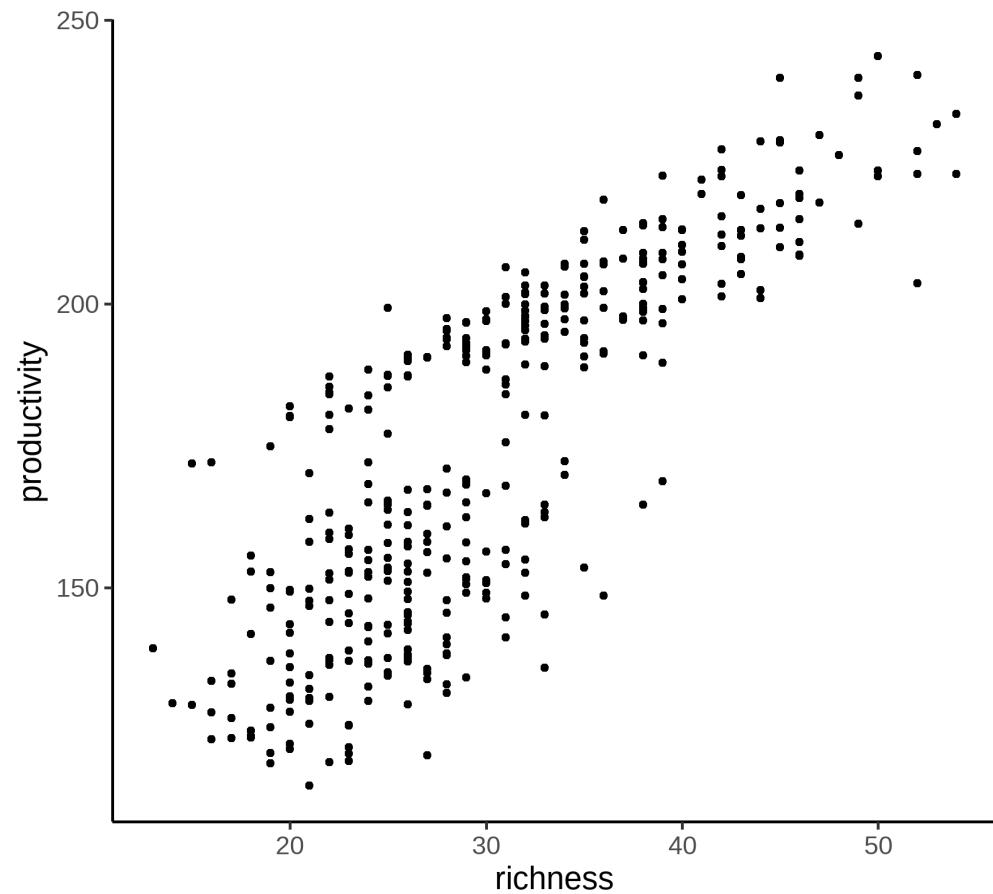
The variable site is a factor with two levels: site1 and site2

```
levels(prod$site) # get all levels of the factor
## [1] "site1" "site2"
```

Linear regression in

Linear regression in R

H_1 : Species richness increases productivity



Linear regression in R

Use the `lm` function to fit a linear model in R.

The general structure of the function call is like this:

```
lm ( formula = Y ~ X, data = dat )
```

with

- `Y` being the response variable
- `X` being the predictor(s)
- `dat` being the name of the data

Multiple predictors can be added with `+`, `:` or `*` depending on interaction:

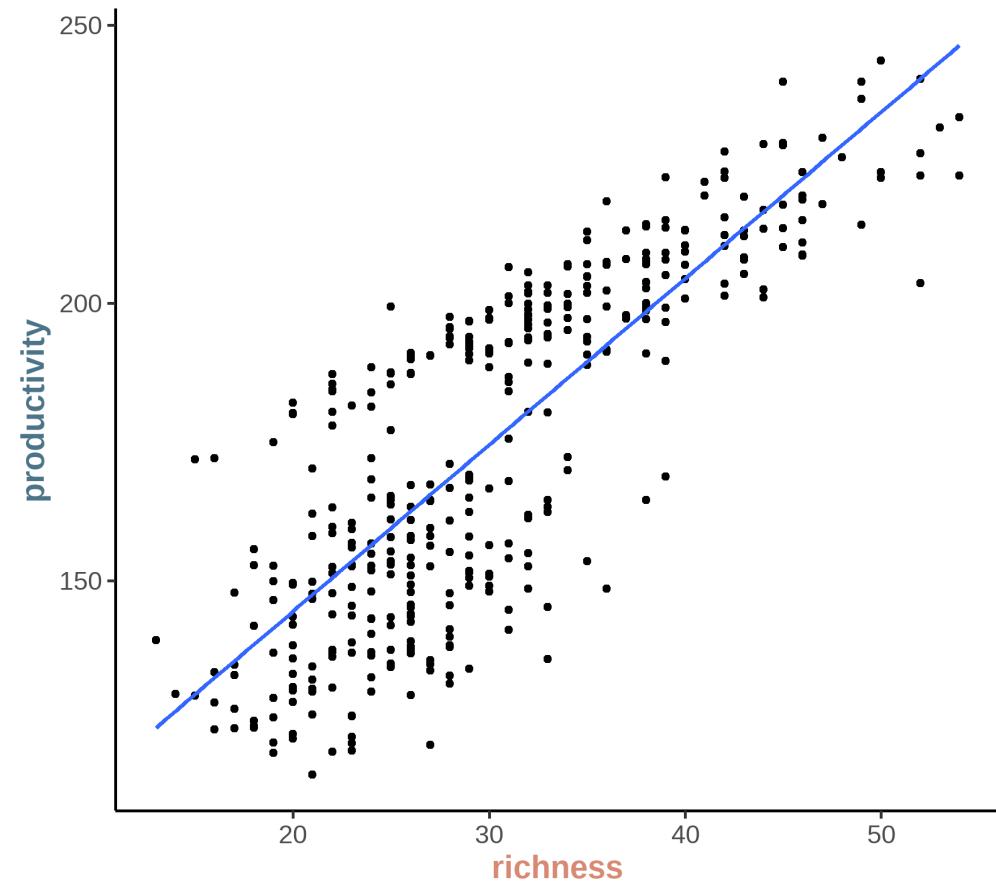
- `Y ~ x1 + x2` tests effects without interaction
- `Y ~ x1 + x2 + x1:x2` tests single effects and interaction between `x1` and `x2`
- `Y ~ x1 * x2` is short for testing all single effects and all interactions (here same as line above)

Linear regression in R

Let's fit a linear model to test our hypothesis

```
lm ( formula = Y ~ X, data = dat )
```

```
prod_lm <- lm(productivity ~ richness,  
                 data = prod)
```



Linear regression in R

Is the effect of richness on productivity significant?

Or in other words

Does the model with richness as predictor significantly reduce the residual sum of squares?

Hypothesis testing using F-Tests

Compare the complex model with a simple model that does not contain the predictor

H_0 : The error variance in the simple model is not significantly higher than in the more complex model

- H_0 accepted: simplification was justified → use simple model without predictor
- H_0 rejected: simplification reduced explanatory power → use complex model with predictor

Linear regression in R

Hypothesis testing using F-Tests in R

H_0 : The error variance in the simple model is not significantly higher than in the more complex model

`drop1` deletes single terms from the model and performs an F-test:

```
drop1(prod_lm, test = "F")
## Single term deletions
##
## Model:
## productivity ~ richness
##           Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>          132779 2326.0
## richness  1     254119 386899 2751.8   761.71 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Result: Reject $H_0 \rightarrow$ Richness increases productivity ($F_{1,398} = 761.71, p < 0.001$)

Extracting the coefficients

Look at model coefficients using the model `summary`

```
summary(prod_lm)
##
## Call:
## lm(formula = productivity ~ richness, data = prod)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -47.593 -13.683   0.029  13.970  42.295 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 84.6982    3.3987  24.92   <2e-16 ***
## richness    2.9938    0.1085  27.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.27 on 398 degrees of freedom
## Multiple R-squared:  0.6568,    Adjusted R-squared:  0.6559 
## F-statistic: 761.7 on 1 and 398 DF,  p-value: < 2.2e-16
```

Summary table

```
summary(prod_lm)
##
## Call:
## lm(formula = productivity ~ richness, data = prod)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -47.593 -13.683   0.029  13.970  42.295
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 84.6982    3.3987  24.92 <2e-16 ***
## richness    2.9938    0.1085  27.60 <2e-16 ***
## ---
## Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.27 on 398 degrees of freedom
## Multiple R-squared:  0.6568,   Adjusted R-squared:  0.6559
## F-statistic: 761.7 on 1 and 398 DF,  p-value: < 2.2e-16
```

Calculated model

Distribution of residuals (errors)

Summary table

```
summary(prod_lm)
##
## Call:
## lm(formula = productivity ~ richness, data = prod)
## Model coefficients:
## 1. Intercept = 84.7
## 2. Slope = 3
## -47.593 -13.683   Median 0.029  Standard error of coefficients 18.970   Max 42.295
## 
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 84.6982   3.3987  24.92 <2e-16 ***
## richness    2.9938   0.1085  27.60 <2e-16 ***
## ---
## Signif. codes:  0 '****' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 18.27 on 398 degrees of freedom
## Multiple R-squared:  0.6568, Adjusted R-squared:  0.6559 
## F-statistic: 761.7 on 1 and 398 DF, p-value: < 2.2e-16
```

Summary table

```
summary(prod_lm)
##
## Call:
## lm(formula = productivity ~ richness, data = prod)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -47.593 -13.683   0.029  13.970  42.295 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 84.6982    3.3987  24.92   <2e-16 ***
## richness    2.9938    0.1085  27.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.27 on 398 degrees of freedom
## Multiple R-squared:  0.6568, Adjusted R-squared:  0.6559 
## F-statistic: 761.7 on 1 and 398 DF, p-value: < 2.2e-16
```

t-statistics: ratio of coefficients to their standard error

Summary table

```
summary(prod_lm)
##
## Call:
## lm(formula = productivity ~ richness, data = prod)
##
## Residuals:
##     Min      1Q  Median      3Q     Max 
## -47.593 -13.683   0.029  13.970  42.295 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 84.6982    3.3987  24.92   <2e-16 ***
## richness    2.9938    0.1085  27.60   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.01 '**' 0.05 '*' 0.1 '.' 1
## 
## Coefficient of determination R2
## 
## Residual standard error: 18.27 on 398 degrees of freedom
## Multiple R-squared:  0.6568, Adjusted R-squared:  0.6559 
## F-statistic: 761.7 on 1 and 398 DF, p-value: < 2.2e-16
```

Test model assumptions

Test model assumptions to make sure that:

- 1) Residuals are normally distributed
- 2) Variance is constant (homogeneous)
- 3) There are no strong outliers or very influential observations

We can check that by looking at diagnostic plots

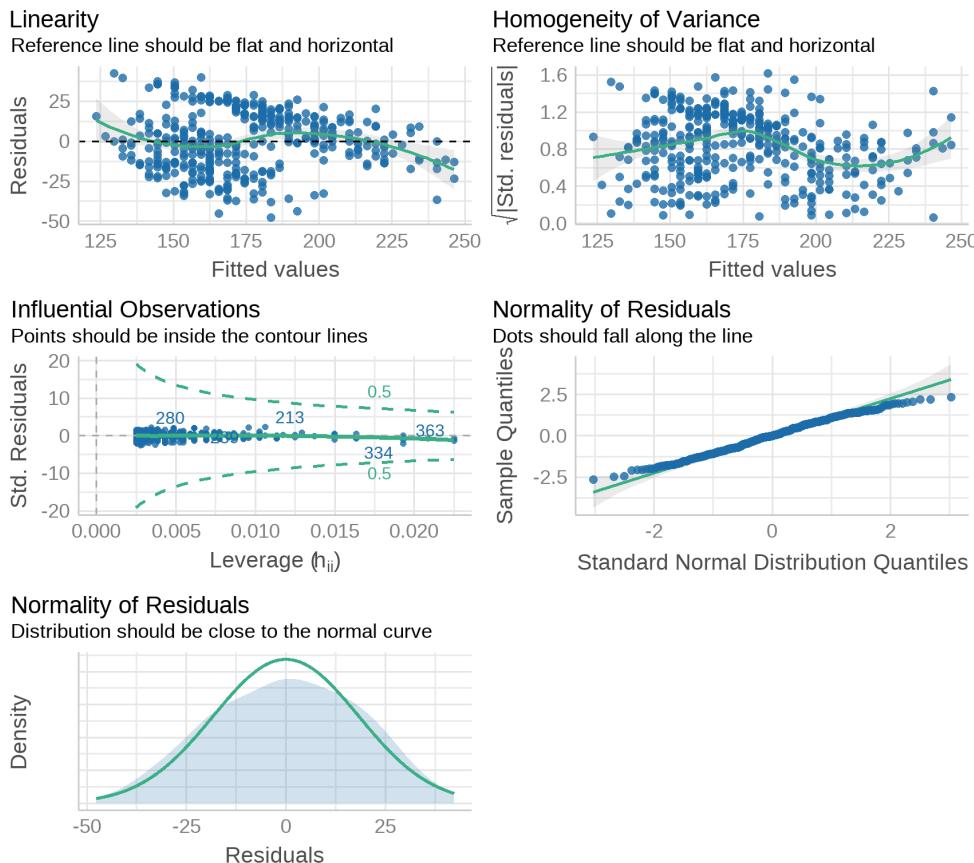
```
# install.packages("performance")
performance::check_model(prod_lm)
```

Another option without an additional package using base R would be:

```
par(mfrow = c(2,2)) # change graphics settings to fit 4 plots
plot(prod_lm) # The actual diagnostic plots
par(mfrow=c(1,1)) # change graphics settings back to default
```

Test model assumptions

```
performance::check_model(prod_lm)
```



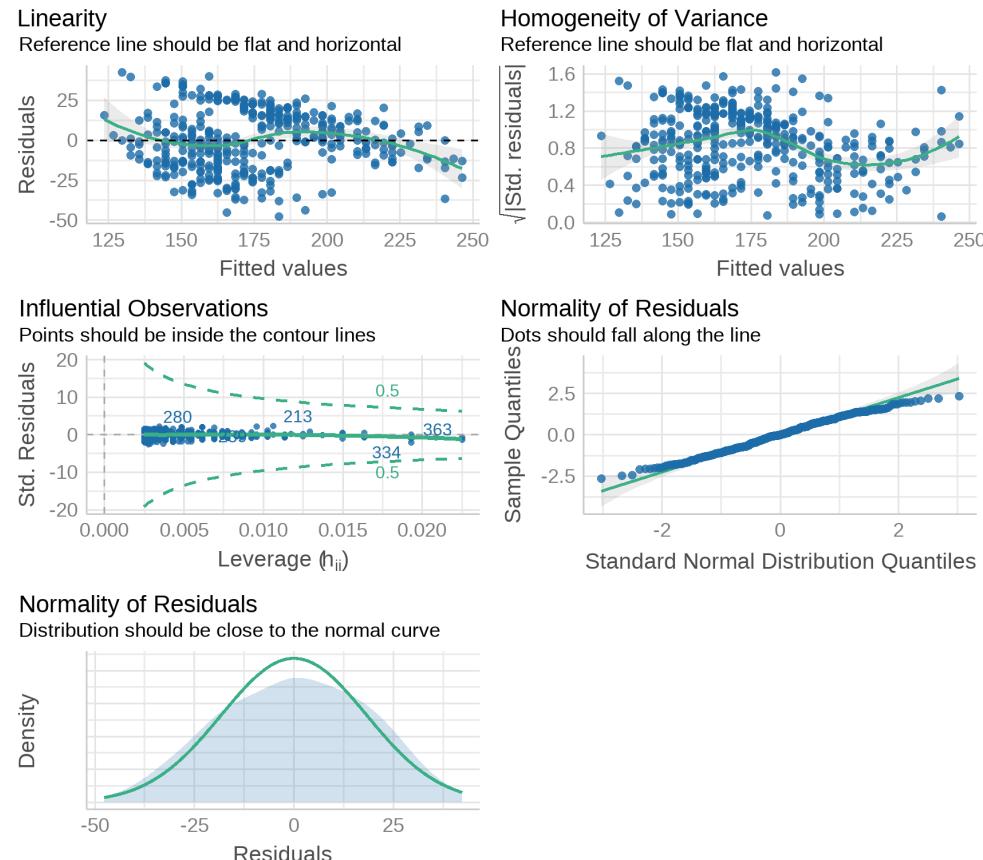
Test model assumptions

Check the assumptions

- Residuals are normally distributed ✓
- Variance is constant (homogeneous) ✓
- There are no strong outliers or very influential observations ✓

Checking diagnostic plots needs some experience

- Real life data (almost) never perfectly fit the assumptions
- Linear models are to some extent robust against violations of the assumptions



Plot the model

Option 1

Extract coefficients (slope + intercept) of the model and add the regression line

```
# These are the coefficients of the lm  
prod_lm$coefficients  
## (Intercept) richness  
## 84.698163 2.993774
```

```
intercept <- prod_lm$coefficients[1]  
slope <- prod_lm$coefficients[2]
```

Plot the model

Option 1

Extract coefficients of the model and add the regression line

```
# These are the coefficients of the lm  
prod_lm$coefficients  
## (Intercept)    richness  
##     84.698163    2.993774
```

```
intercept <- prod_lm$coefficients[1]  
slope <- prod_lm$coefficients[2]
```

Add a line defined by slope and intercept using

`geom_abline()`:

```
ggplot(prod, aes(x=richness, y=productivity)) +  
  geom_point() +  
  geom_abline(slope = slope,  
             intercept = intercept)
```

Plot the model

Option 2

Add the regression line directly as a ggplot layer

Use `geom_smooth()` to add the model directly:

```
ggplot(prod, aes(x = richness, y =  
productivity)) +  
  geom_point() +  
  geom_smooth(method = "lm")
```

Plot the model

Option 2

Add the regression line directly as a ggplot layer

Use `geom_smooth()` to add the model directly:

```
ggplot(prod, aes(x = richness, y =  
productivity)) +  
  geom_point() +  
  geom_smooth(method = "lm")
```

Or without the confidence interval:

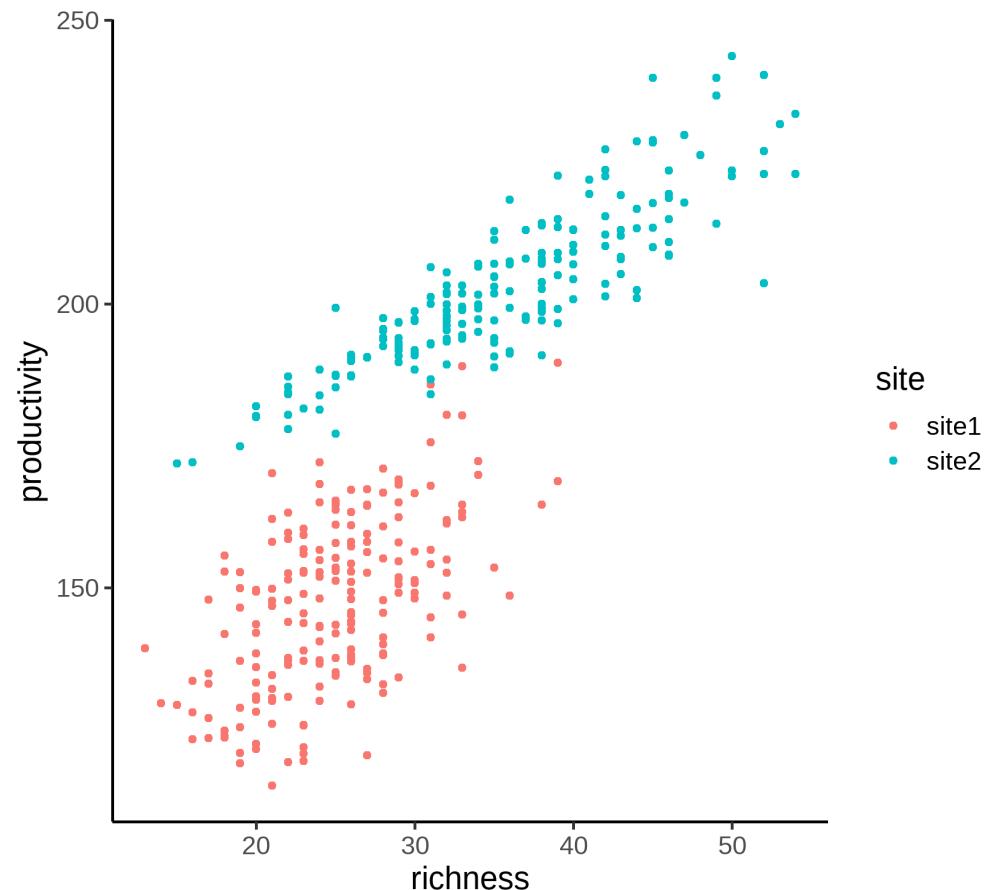
```
ggplot(prod, aes(x = richness, y =  
productivity)) +  
  geom_point() +  
  geom_smooth(method = "lm", se = FALSE)
```

Analysis of covariance

Analysis of covariance

One categorical and one numerical predictor variable

Two possible models:



Analysis of covariance

Fit the model with `lm`

Without interaction

Species richness has an effect on productivity and there is a difference between sites.

```
prod_lm2a <- lm(productivity ~ richness + site, data = prod)
```

With interaction

Species richness has an effect on productivity and the effect differs between sites.

```
prod_lm2b <- lm(productivity ~ richness + site + richness:site, data = prod)
# or the same in short version:
# prod_lm2b <- lm(productivity ~ richness * site, data = prod)
```

Analysis of covariance

Are the effects significant?

Without interaction

```
drop1(prod_lm2a, test = "F") # no interaction
## Single term deletions
##
## Model:
## productivity ~ richness + site
##           Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>            40064 1848.7
## richness   1     42908  82973 2137.9  425.18 < 2.2e-16 ***
## site       1     92715 132779 2326.0  918.72 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Both, the removal of site and richness significantly increase the model's RSS → both site and richness are significant predictors

Analysis of covariance

Are the effects significant?

With interaction

```
drop1(prod_lm2b, test = "F") # interaction
## Single term deletions
##
## Model:
## productivity ~ richness + site + richness:site
##           Df Sum of Sq   RSS   AIC F value Pr(>F)
## <none>            39850 1848.6
## richness:site     1    214.53 40064 1848.7  2.1319 0.1451
```

The removal of the interaction does not significantly increase the model's RSS → the interaction is not significant and the simpler model without interaction is just as good in predicting productivity.

Extracting model coefficients

```
summary(prod_lm2a)
##
## Call:
## lm(formula = productivity ~ richness + site, data = prod)
##
## Residuals:
##       Min     1Q Median     3Q    Max
## -30.1743 -6.2895 -0.0352  5.6955 28.9276
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 108.36550   2.02582  53.49 <2e-16 ***
## richness     1.56699   0.07599  20.62 <2e-16 ***
## sitesite2    38.78575   1.27962  30.31 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 10.05 on 397 degrees of freedom
## Multiple R-squared:  0.8964,    Adjusted R-squared:  0.8959 
## F-statistic: 1718 on 2 and 397 DF,  p-value: < 2.2e-16
```

Extracting model coefficients

Call:

```
lm(formula = productivity ~ richness + site, data = prod)
```

Residuals:

| Min | 1Q | Median |
|----------|---------|---------|
| -30.1743 | -6.2895 | -0.0352 |

Intercept site1 = 108.4

Slope = 1.6

Intercept site2 = 108.4 + 38.8

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) | |
|-------------|-----------|------------|---------|----------|-----|
| (Intercept) | 108.36550 | 2.02582 | 53.49 | <2e-16 | *** |
| richness | 1.56699 | 0.07599 | 20.62 | <2e-16 | *** |
| sitesite2 | 38.78575 | 1.27962 | 30.31 | <2e-16 | *** |
| --- | | | | | |

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 10.05 on 397 degrees of freedom

Multiple R-squared: 0.8964, Adjusted R-squared: 0.8959

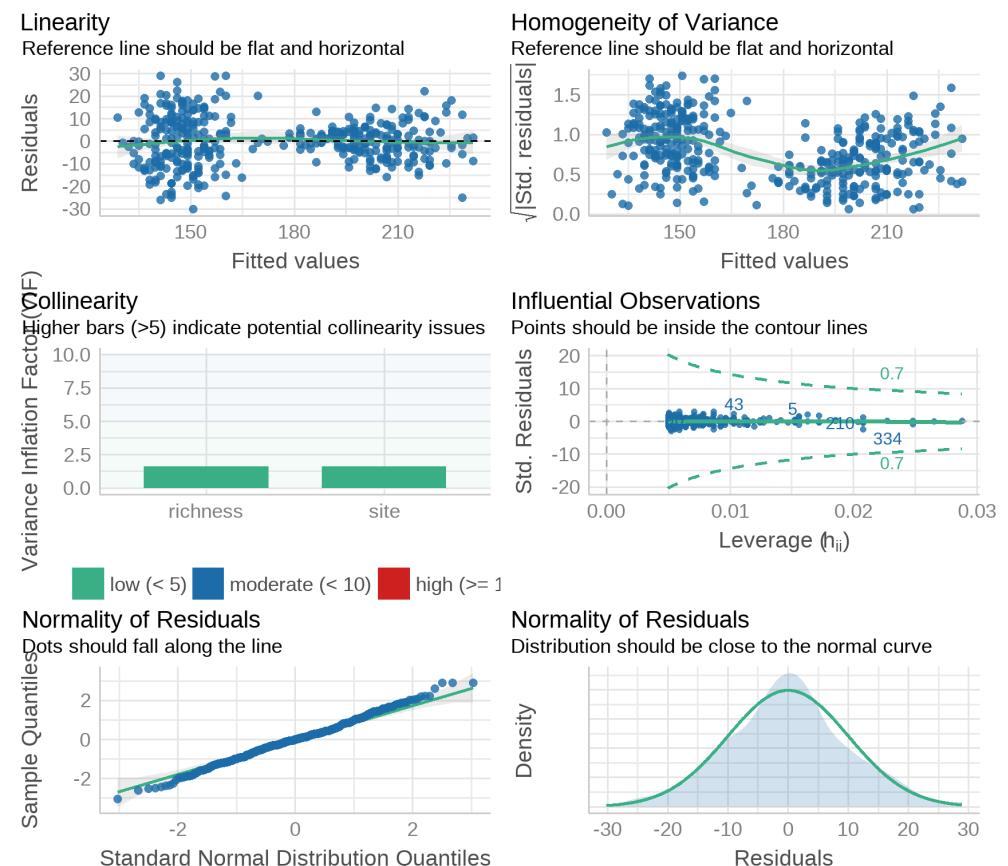
F-statistic: 1718 on 2 and 397 DF, p-value: < 2.2e-16

Test model assumptions

Check the assumptions

- Residuals are normally distributed ✓
- Variance is constant (homogeneous) ✓
- There are no strong outliers or very influential observations ✓

```
performance::check_model(prod_lm2a)
```



Plot the model

If we want to be precise, we could not use `geom_smooth()` for this model because it plots the full model with interaction between the predictors

```
ggplot(prod, aes(x = richness,  
                  y = productivity,  
                  color = site)) +  
  geom_point() +  
  geom_smooth(method = "lm")
```

This plot is not 100% appropriate if the model you present is without interaction

Plot the model

Option 1: Extract coefficients and add regression line

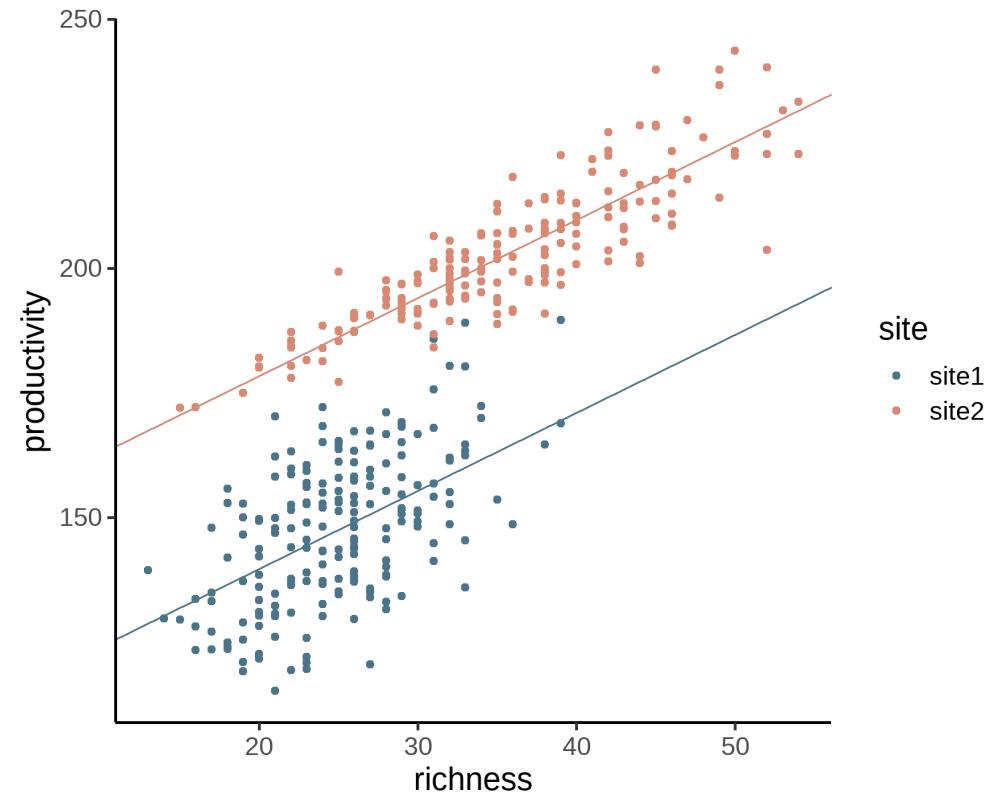
```
# these are the model coefficients  
prod_lm2a$coefficients  
## (Intercept) richness sitesite2  
## 108.365497 1.566994 38.785753
```

```
slope <- prod_lm2a$coefficients[2]  
intercept1 <- prod_lm2a$coefficients[1]  
intercept2 <- prod_lm2a$coefficients[1] + prod_lm2a$coefficients[3]
```

Plot the model

Option 1: Extract coefficients and add regression line

```
ggplot(prod, aes(x = richness,
                  y = productivity,
                  color = site)) +
  geom_point() +
  scale_color_manual(
    values = c("#4C7488", "#D78974")) +
  geom_abline(
    slope = slope,
    intercept = intercept1,
    color = "#4C7488"
  ) +
  geom_abline(
    slope = slope,
    intercept = intercept2,
    color = "#D78974"
  )
```



Plot the model

Option 2: Use the `predict()` function

```
# step 1: create some data to predict from
pred_data <- tidyverse::expand_grid(
  richness = min(prod$richness) : max(prod$richness),
  site = c("site1", "site2")
)
```

```
## # A tibble: 84 x 2
##   richness site
##       <int> <chr>
## 1        13 site1
## 2        13 site2
## 3        14 site1
## # ... with 81 more rows
```

- `tidyverse::expand_grid` returns a tibble with all combinations of site and richness given as input
 - The `tidyverse` package is part of the tidyverse
- The column names of the tibble that you create in this step have to correspond to the predictor names in the linear model

Plot the model

Option 2: Use the `predict()` function

```
# step 1: create some data to predict from
pred_data <- tidyverse::expand_grid(
  richness = min(prod$richness) : max(prod$richness),
  site = c("site1", "site2")
)
```

```
# step2: predict productivity values from pred_data
predictions <- predict(prod_lm2a, newdata = pred_data)
```

- `predict` uses the `prod_lm2a` model and the data produced with `expand_grid` to predict productivity values for each combination of site and richness.

Plot the model

Option 2: Use the `predict()` function

```
# step 1: create some data to predict from
pred_data <- tidyverse::expand_grid(
  richness = min(prod$richness) : max(prod$richness),
  site = c("site1", "site2")
)
```

```
# step 2: predict productivity values from pred_data
predictions <- predict(prod_lm2a, newdata = pred_data)
```

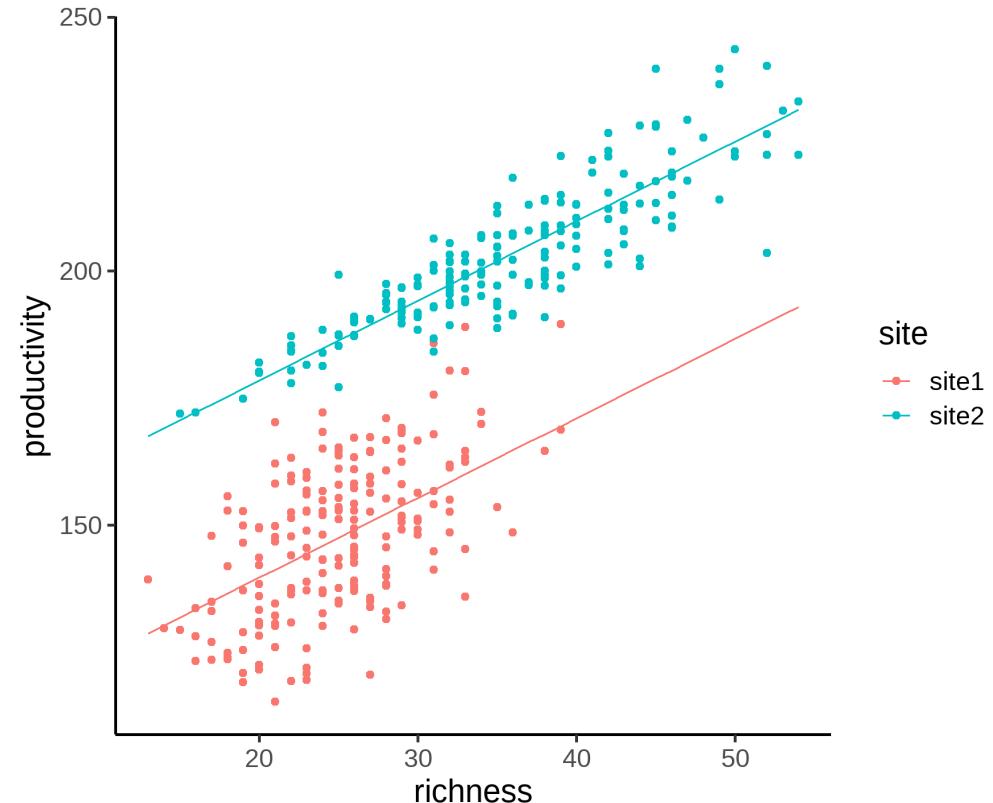
```
# step 3: add predictions to the tibble
pred_data$productivity <- predictions
```

Plot the model

Option 2: Use the `predict()` function

Add a new `geom_*` layer to the plot from the prediction data:

```
# step 4: Add predictions with geom_line
ggplot(prod, aes(x = richness,
                  y = productivity,
                  color = site)) +
  geom_point() +
  geom_line(data = pred_data)
```



- The aesthetic mapping is inherited from the top level `ggplot` call to the `geom_line`

Plot the model

Option 2: Use the `predict()` function

Summary of general workflow

- **Step 1:** Use `expand_grid` to create a tibble with values for all predictor variables (the columns have to have the exact same name as the original data)
- **Step 2:** Use `predict` to predict response variable with the model using input predictor values from tibble in step 1
- **Step 3:** Add predictions as column to the tibble from step 1
- **Step 4:** Add predictions to the plot using a new `geom_line()` layer based on the tibble from step 3

Analysis of variance (Anova)

Anova

Only categorical predictors.

Example: Data set `chickwts` about the weight of chickens fed with different diets.

```
chickwts
```

```
## # A tibble: 71 x 2
##   weight feed
##   <dbl> <fct>
## 1    179 horsebean
## 2    160 horsebean
## 3    136 horsebean
## 4    227 horsebean
## # ... with 67 more rows
```

Anova

H_1 : The diet has an effect on the weight of chicken.

Fit a linear model with `lm` and test the significance of the predictor:

```
lm_chicken <- lm(weight ~ feed, data = chickwts)
drop1(lm_chicken, test = "F")
## Single term deletions
##
## Model:
## weight ~ feed
##      Df Sum of Sq    RSS    AIC F value    Pr(>F)
## <none>          195556 574.39
## feed     5     231129 426685 619.78  15.365 5.936e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Removing the predictor `feed` from the model significantly increases the RSS → The model with `feed` explains the data better than the model without `feed`, so the effect of `feed` on the chicken weight is significant.

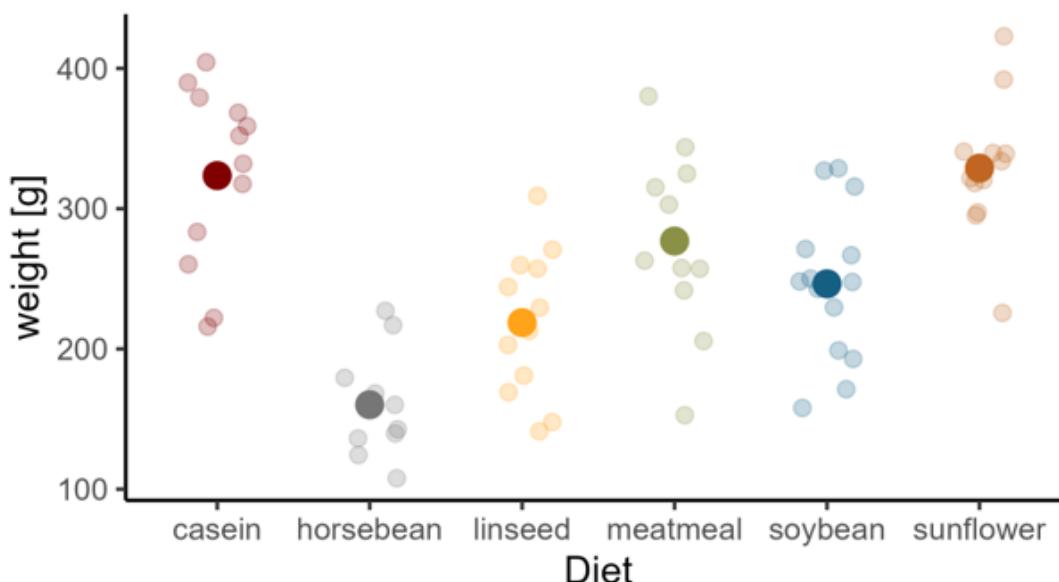
Extracting model coefficients

```
summary(lm_chicken)
##
## Call:
## lm(formula = weight ~ feed, data = chickwts)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -123.909 -34.413     1.571   38.170  103.091
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 323.583   15.834  20.436 < 2e-16 ***
## feedhorsebean -163.383   23.485 -6.957 2.07e-09 ***
## feedlinseed   -104.833   22.393 -4.682 1.49e-05 ***
## feedmeatmeal   -46.674   22.896 -2.039 0.045567 *
## feedsoybean    -77.155   21.578 -3.576 0.000665 ***
## feedsunflower      5.333   22.393   0.238 0.812495
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 54.85 on 65 degrees of freedom
## Multiple R-squared:  0.5417,    Adjusted R-squared:  0.5064
## F-statistic: 15.36 on 5 and 65 DF,  p-value: 5.936e-10
```

Extracting model coefficients

Coefficients:

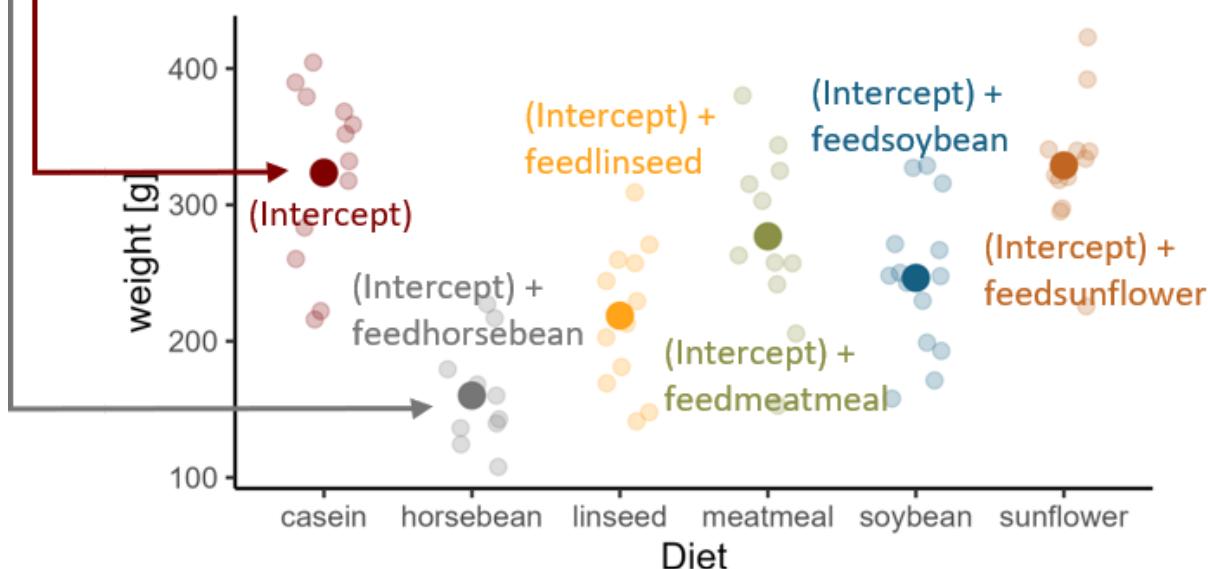
| | Estimate | Std. Error | t value | Pr(> t) | |
|---------------|----------|------------|---------|----------|-----|
| (Intercept) | 323.583 | 15.834 | 20.436 | < 2e-16 | *** |
| feedhorsebean | -163.383 | 23.485 | -6.957 | 2.07e-09 | *** |
| feedlinseed | -104.833 | 22.393 | -4.682 | 1.49e-05 | *** |
| feedmeatmeal | -46.674 | 22.896 | -2.039 | 0.045567 | * |
| feedsoybean | -77.155 | 21.578 | -3.576 | 0.000665 | *** |
| feedsunflower | 5.333 | 22.393 | 0.238 | 0.812495 | |



Extracting model coefficients

Coefficients:

| | Estimate | Std. Error | t value | Pr(> t) | |
|---------------|----------|------------|---------|----------|-----|
| (Intercept) | 323.583 | 15.834 | 20.436 | < 2e-16 | *** |
| feedhorsebean | -163.383 | 23.485 | -6.957 | 2.07e-09 | *** |
| feedlinseed | -104.833 | 22.393 | -4.682 | 1.49e-05 | *** |
| feedmeatmeal | -46.674 | 22.896 | -2.039 | 0.045567 | * |
| feedsoybean | -77.155 | 21.578 | -3.576 | 0.000665 | *** |
| feedsunflower | 5.333 | 22.393 | 0.238 | 0.812495 | |
| --- | | | | | |

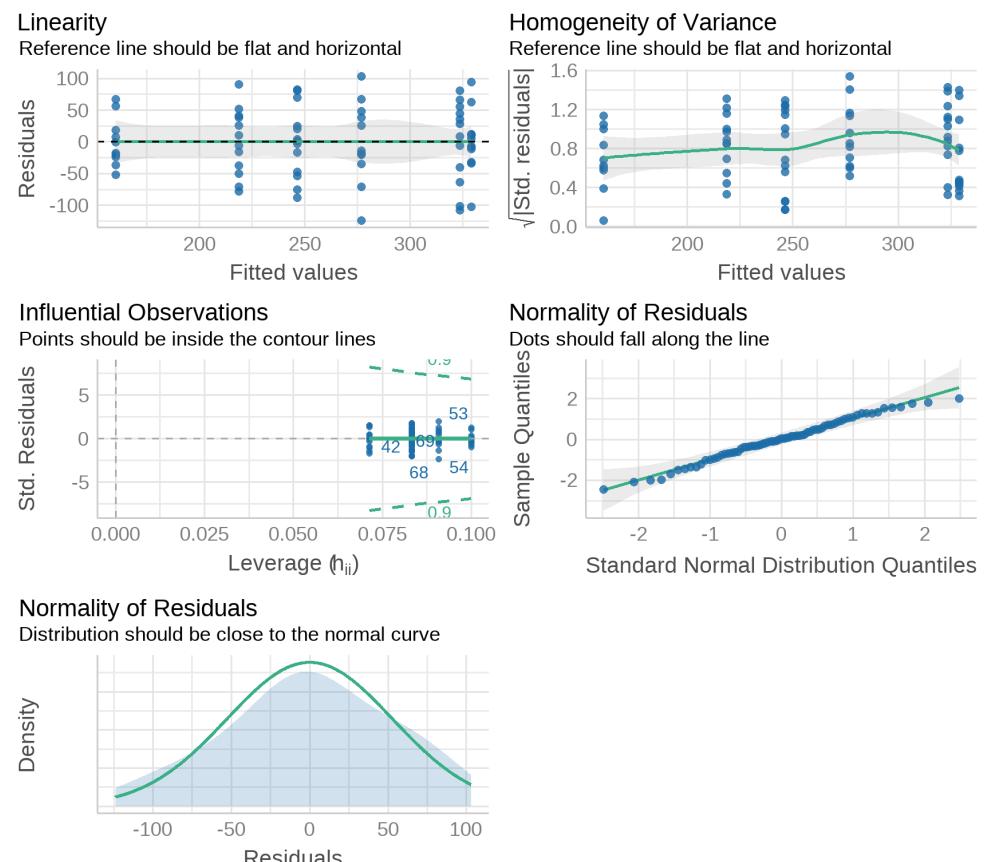


Test model assumptions

Check the assumptions

- Residuals are normally distributed ✓
- Variance is constant (homogeneous) ✓
- There are no strong outliers or very influential observations ✓

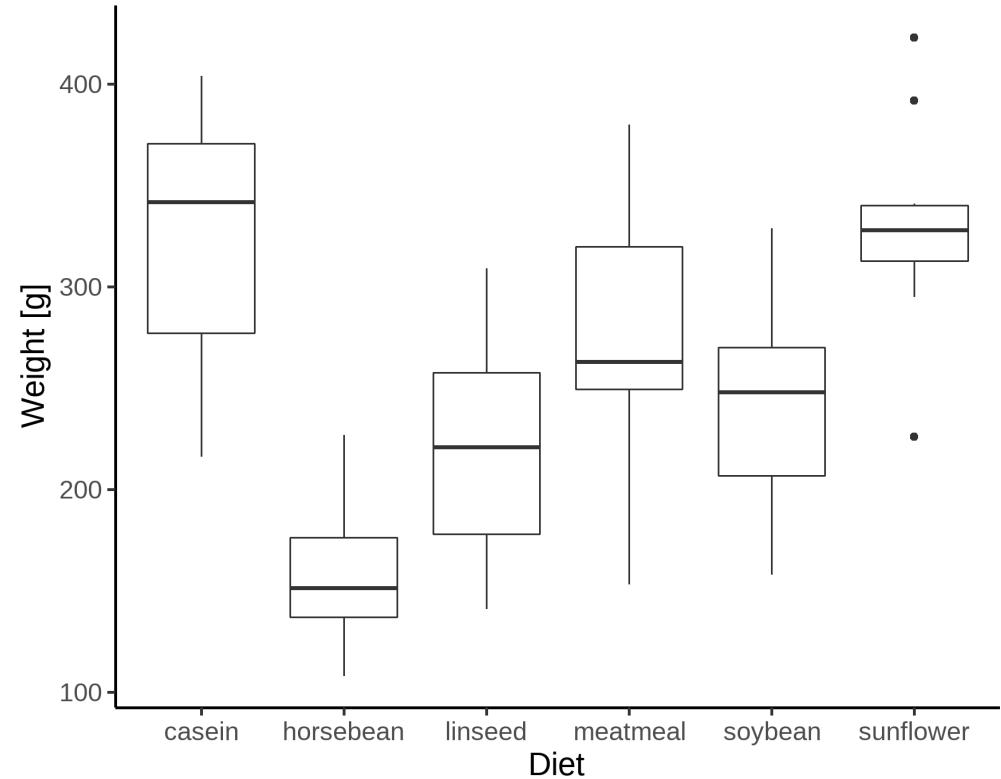
```
performance::check_model(lm_chicken)
```



Plot results

Plot anova results e.g. in a boxplot:

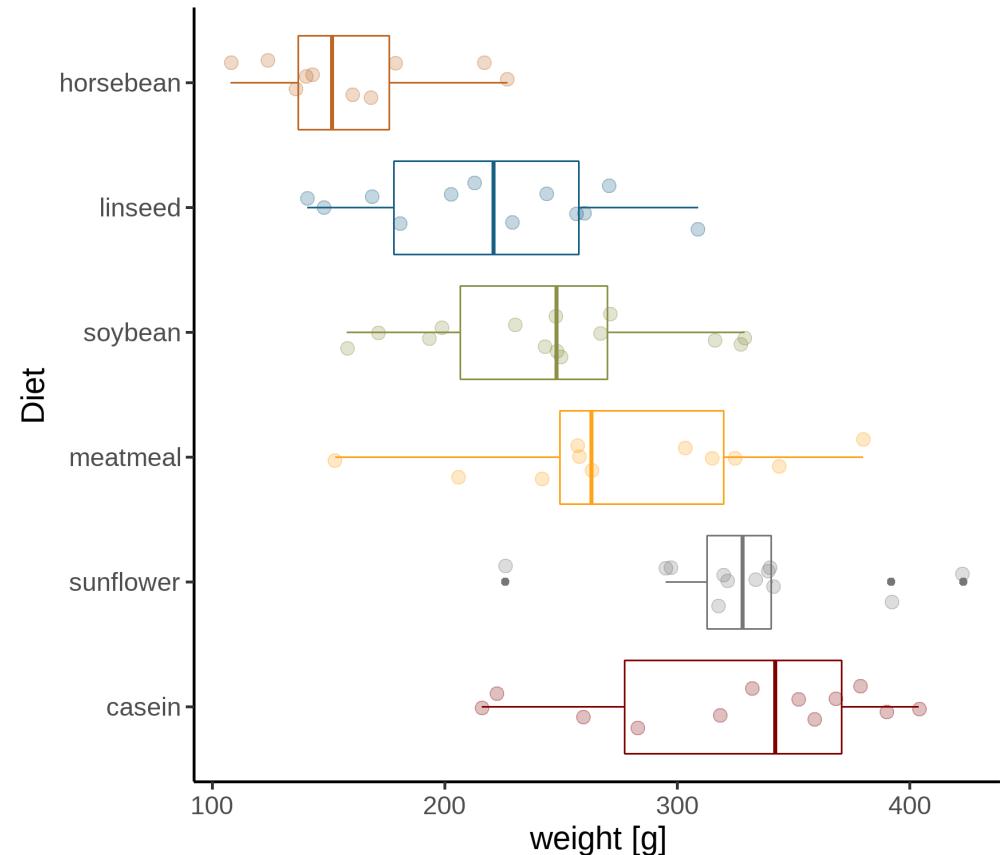
```
ggplot(chickwts, aes(feed, weight)) +  
  geom_boxplot() +  
  labs(x = "Diet", y= "Weight [g]")
```



Plot results

If you want a slightly nicer box plot, you could do something like this:

```
chickwts %>%
  mutate(feed = as.factor(feed)) %>%
  mutate(feed = fct_reorder(feed, -weight)) %>%
  ggplot(aes(
    x = feed,
    y = weight,
    color = feed
  )) +
  geom_boxplot() +
  geom_point(
    size = 3, alpha = 0.25,
    position = position_jitter(width = 0.2,
    seed = 0)
  ) +
  coord_flip() +
  ggsci::scale_color_uchicago() +
  labs(y = "weight [g]", x = "Diet") +
  theme(legend.position = "none")
```



Linear models step by step

Step 1: Explore the data with plots

Step 2: Write down a **question/hypothesis** and think of the **model** to test it

Step 3: Fit the linear model using the `lm` function:

```
lm(formula = Y ~ x1 + x2 + x1:x2, data = dat)
```

- Use `+` for additive effects and `:` for interactions between predictors

Step 4: Check model **assumptions** by looking at the diagnostic plots

```
performance::check_model(mod)
```

- Normally distributed residuals
- Constant variance
- No strong outliers / influential data points

Linear models step by step

Step 5: Check significant variables by conducting F-tests with `drop1`

```
drop1(mod, test = "F")
```

- If the removal of a variable significantly increases the RSS of the model, the predictor is significant

Step 6: Check effect size e.g. in the `summary` table

```
summary(mod)
```

- Extract coefficients from the model with `mod$coefficients`

Step 7: Plot model

- Regression: regression line and scatterplot
- Analysis of covariance: regression lines and scatter plot
- Analysis of variance: boxplots, barplot, mean + sem or similar

Options for model plotting

Option 1: `geom_smooth(method = "lm")`

- Plots a regression line with all interactions between variables
 - only use it if this is what you want to plot

Option 2: extract coefficients and use `geom_abline()`

- Extract slopes and intercepts from the model
 - `mod$coefficients`
- Add a `geom_abline` layer to your plot
 - `geom_abline(slope = your_slope, intercept = your_intercept)`

Options for model plotting

Option 3: Use `predict` function

- This is the most flexible plotting option
- Step 1: Create a new tibble with data to predict from
 - Column names have to be same as predictors of linear model
 - Use e.g. `tidyverse::expand_grid()` to create variable combinations
- Step 2: Predict the response for all value combinations from step 1

```
predict(mod, newdata = pred_data)
```

- Step 3: Add predicted response to tibble from step 1
- Step 4: Add a `geom_line` layer with the new data to your plot

```
ggplot(orig_dat, aes(x = some_x, y = some_y, color = some_col)) +  
  geom_point() +  
  geom_line(data = pred_data)
```

Now you

Task 2: Linear models with the penugin data set

Find the task description [here](#)

Linear models with transformed response variable

When to transform?

Transformation of the response variable Y can help with potential violations of the model assumptions:

- residuals non-normally distributed
 - skewed distribution
 - outliers
- trends in the residuals
- non-constant variance

Example

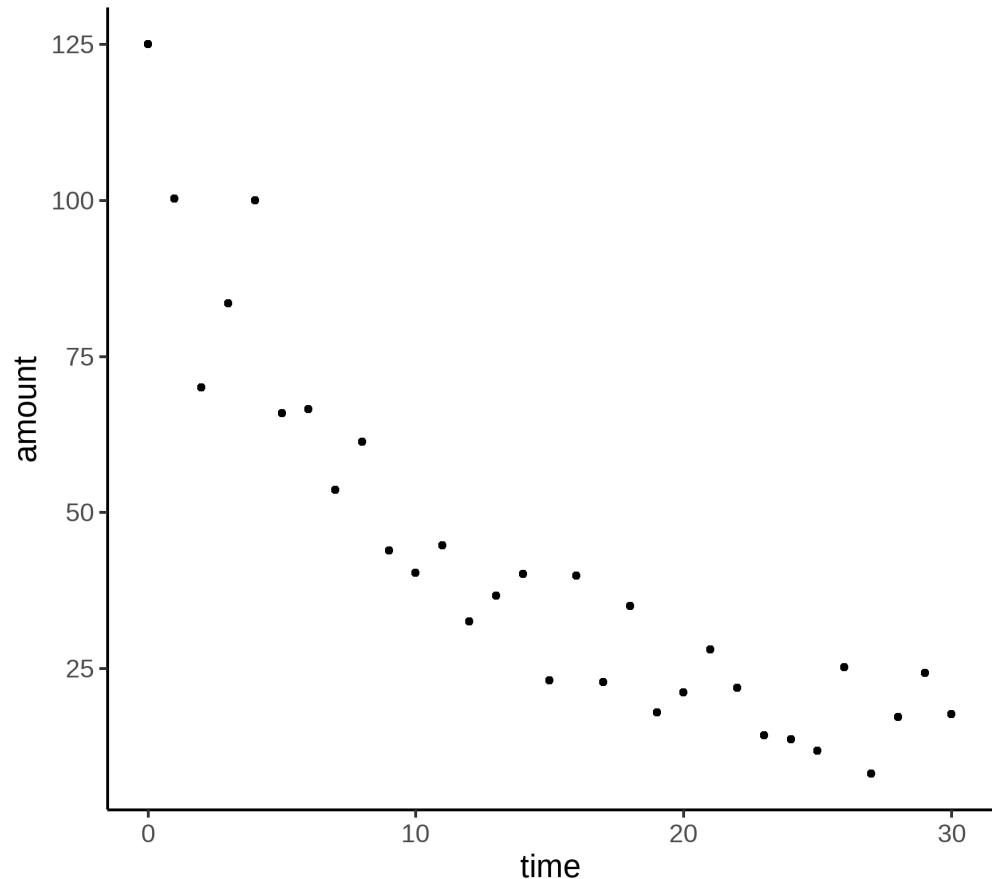
- Non-linear function $y = a * e^{b*x}$
- Transformed to a linear model: $\log(y) = \log(a) + b * x$
- Common natural processes: exponential growth and decay

Example

Data set `decay` on the decay of soil organic matter over time.

Two variables:

- amount of organic matter
- time



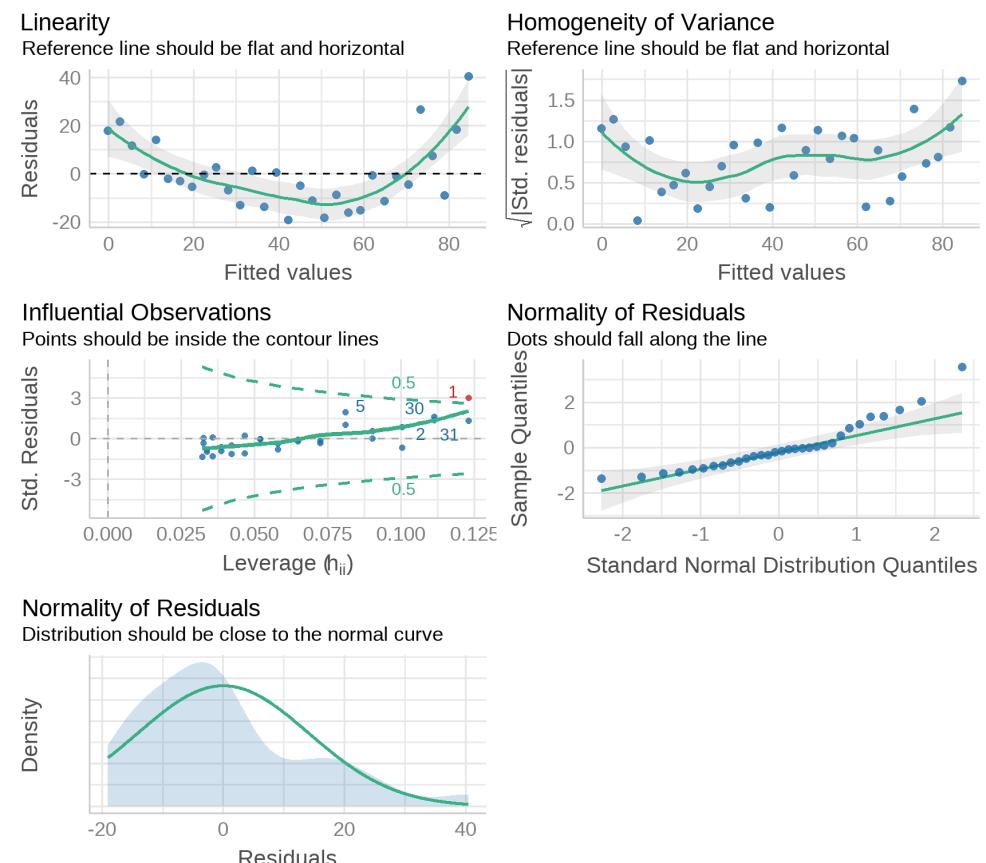
Example

First, let's fit a linear model to the untransformed data and look at the diagnostic plots:

```
mod1 <- lm(amount ~ time, data = decay)
performance::check_model(mod1)
```

Diagnostic plots look bad:

- pattern in the residuals
- skewed distribution



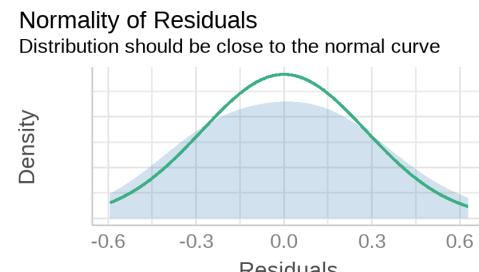
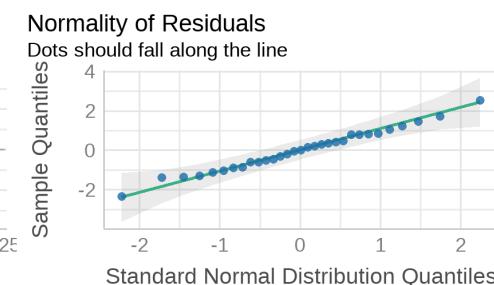
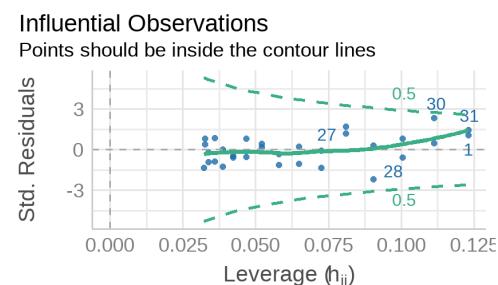
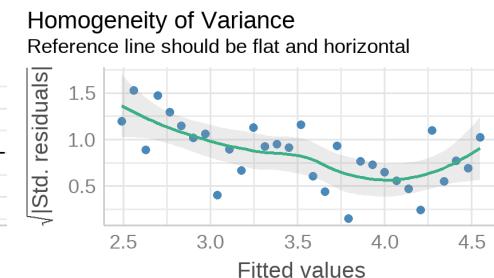
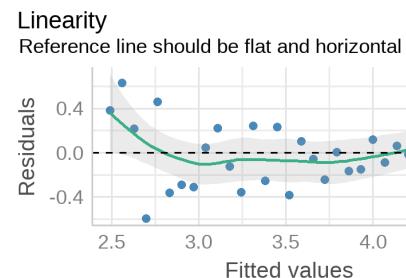
Exercise

Let's refit the model to transformed data (log-transformed response variable):

```
mod2 <- lm(log(amount) ~ time, data = decay)  
performance::check_model(mod2)
```

Diagnostic plots look much better (though not perfect)

- no patterns anymore



Test for significant effects

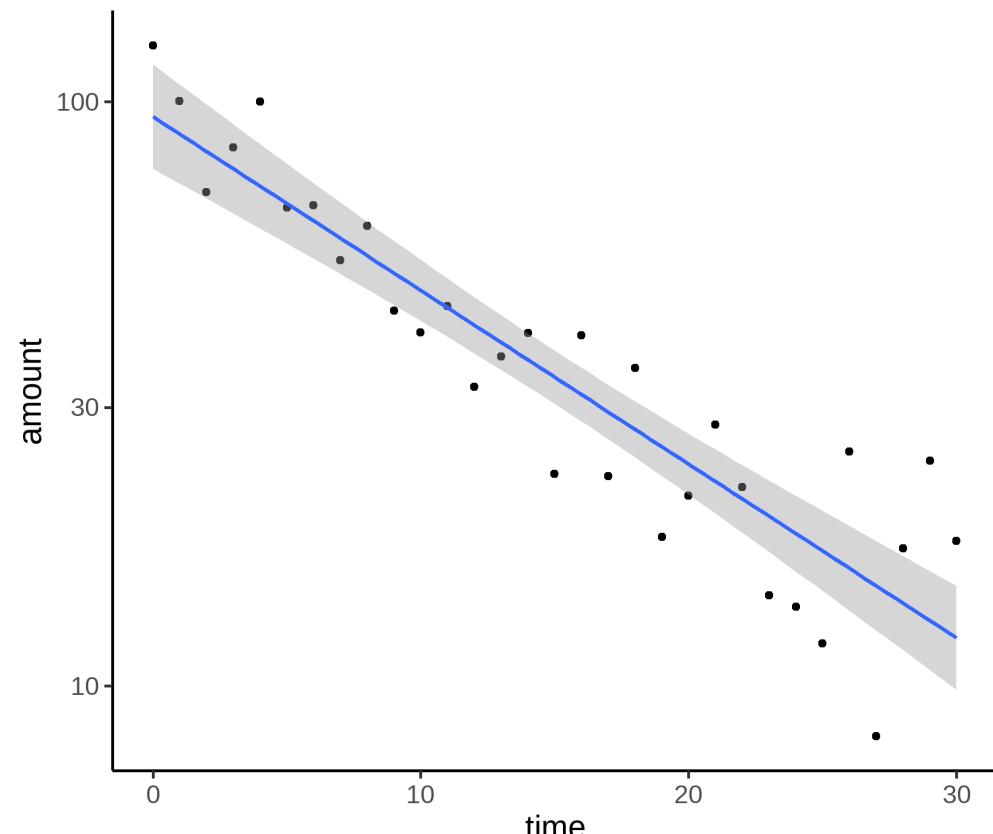
Is the effect of time on the amount of organic matter significant?

```
drop1(mod2, test = "F")
## Single term deletions
##
## Model:
## log(amount) ~ time
##          Df Sum of Sq    RSS      AIC F value    Pr(>F)
## <none>            2.372 -75.678
## time     1   11.646 14.018 -22.602  142.39 1.038e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Plot the model

To plot the results of the model **on the transformed scale**, we can just use `geom_smooth` in combination with `scale_y_log10`

```
ggplot(decay, aes(x = time, y = amount)) +  
  geom_point() +  
  scale_y_log10() +  
  geom_smooth(method = "lm")
```

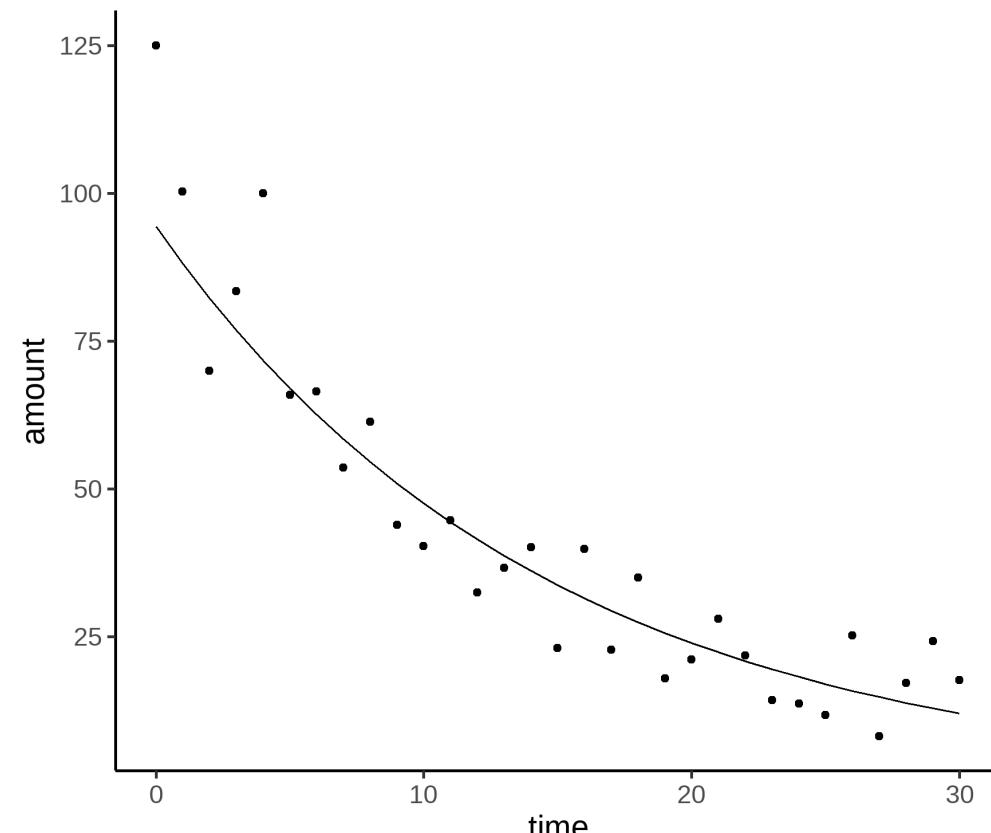


Plot the model

However, we mostly want to plot the model **on the scale of the original data**. For this, we need to use `predict`

```
# step 1: create some data to predict from
pred_data <- tibble(time = 0:30)
# step 2: make the prediction
amount_pred <- predict(mod, newdata =
pred_data)
# step 3: backtransform to original scale and
# add to prediction data
pred_data$amount <- exp(amount_pred)
# step 4: add model to plot
ggplot(decay, aes(x= time, y=amount)) +
  geom_point() +
  geom_line(data = pred_data)
```

- The trick here is to **backtransform** the predictions using `exp` as the inverse function of `log`



Transformations step by step

1. Plot raw data
2. Fit a model to untransformed data
3. Check diagnostic plots
4. Apply a transformation
5. Re-fit model to transformed data
6. Check diagnostic plots again
7. Plot raw data and add model predictions **based on the transformed data**
 - Backtransform predictions appropriately:
 - $\log(y) <=> \exp(y)$
 - $\sqrt{y} <=> y^2$

Now you

Task 3: Linear models with transformations (75 min)

Find the task description [here](#)