

Examples of regression and mixed models

Marc Allasonnière-Tang (UMR 7206)

Contents

| | |
|---------------------|---|
| Notes | 1 |
| Basic settings | 1 |
| Data visualization | 2 |
| Linear regression | 3 |
| Linear mixed models | 6 |

Notes

This is the supplementary materials for running regression and mixed models. The code uses the palmer penguin dataset as an example (<https://allisonhorst.github.io/palmerpenguins/>).

Basic settings

First, we read the basic packages and set the seed for reproducibility. The most relevant is the tidyverse work environment. See <https://www.tidyverse.org/> for more details.

```
# load the package with penguin data
library(palmerpenguins)
# save the data as a named data frame
data <- penguins
# load basic packages
library(readxl)
library(tidyverse)
library(knitr)
# packages for correlation paired plot
library(GGally)
# packages for model performance
library(caret)
library(sjPlot)
library(MuMIn)
# set seed for reproducibility
set.seed(10)
```

```
# set visual options for numbers
options(scipen=999)
```

Data visualization

Have a look at the raw data, which has continuous and categorical variables.

```
glimpse(data)
```

```
## Rows: 344
## Columns: 8
## $ species      <fct> Adelie, Adelie, Adelie, Adelie, Adelie, Adelie, Adel~
## $ island       <fct> Torgersen, Torgersen, Torgersen, Torgersen, Torgerse~
## $ bill_length_mm <dbl> 39.1, 39.5, 40.3, NA, 36.7, 39.3, 38.9, 39.2, 34.1, ~
## $ bill_depth_mm <dbl> 18.7, 17.4, 18.0, NA, 19.3, 20.6, 17.8, 19.6, 18.1, ~
## $ flipper_length_mm <int> 181, 186, 195, NA, 193, 190, 181, 195, 193, 190, 186~
## $ body_mass_g   <int> 3750, 3800, 3250, NA, 3450, 3650, 3625, 4675, 3475, ~
## $ sex           <fct> male, female, female, NA, female, male, female, male~
## $ year          <int> 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007, 2007~
```

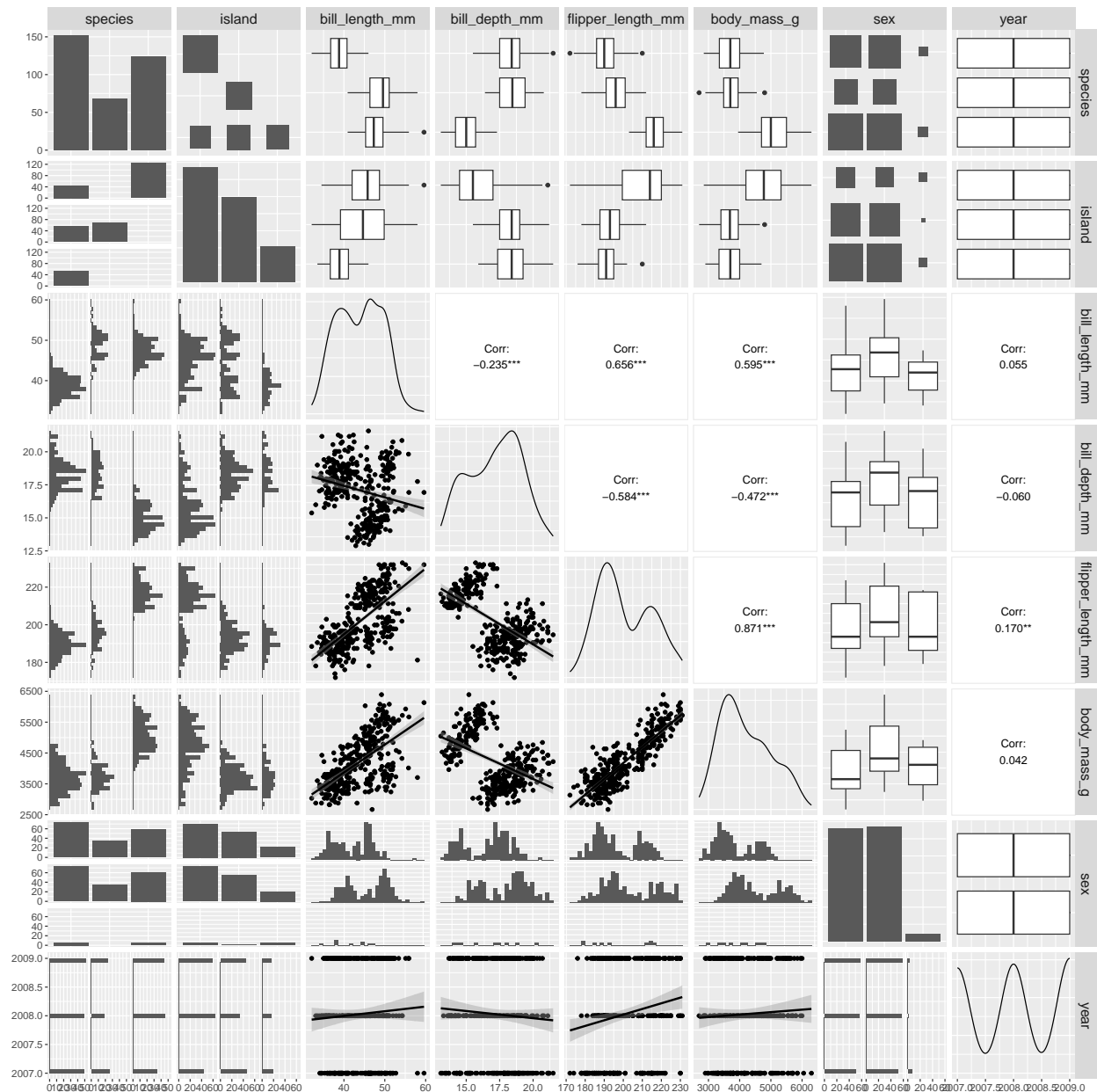
Make a correlation plot of all the variables we are looking at. The correlation coefficient r can be interpreted as follows:

- measure of *linear* relationship
- between -1.0 (perfectly *negative*) and $+1.0$ (perfectly *positive*)
- 0 means *no* (linear) relationship

correlation is not regression

- correlation = the extent to which x and y move together
- regression = the impact of a change of unit on x in y

```
data %>%
  ggpairs(lower=list(continuous=wrap("smooth", colour="black")),
          upper = list(continuous = wrap("cor", size=4, colour = "black"))) +
  # plot settings
  theme(strip.text = element_text(size = 14),
        axis.text = element_text(size = 10))
```



Linear regression

First, let's have a look at simple linear regression.

The Estimates can be manually read as follows. If in a hypothetical model, we are trying to predict the variable X based on the variables A , B , and sex , which have an estimate of 1, 3, and -1 (for male). And the intercept is 0.5. If a data point has the following values for the hypothetical variables encoded in the data.

- X (the predicted variable) = 370
- $A = 364$
- $B = 3$
- $sex = \text{male}$

Then, the predicted X would be = the intercept $0.5 + (1 * 364) + (3 * 3) + (-1 * 1) = 372.5$.

```
# arrange the data a bit
tmp <- data %>%
  # change the coding to numerical values if needed
  mutate(body_mass_g = as.numeric(body_mass_g)) %>%
  # rename the variables to be predicted
  rename(To_Predict = body_mass_g) %>%
  # remove the rows with NA values
  drop_na()

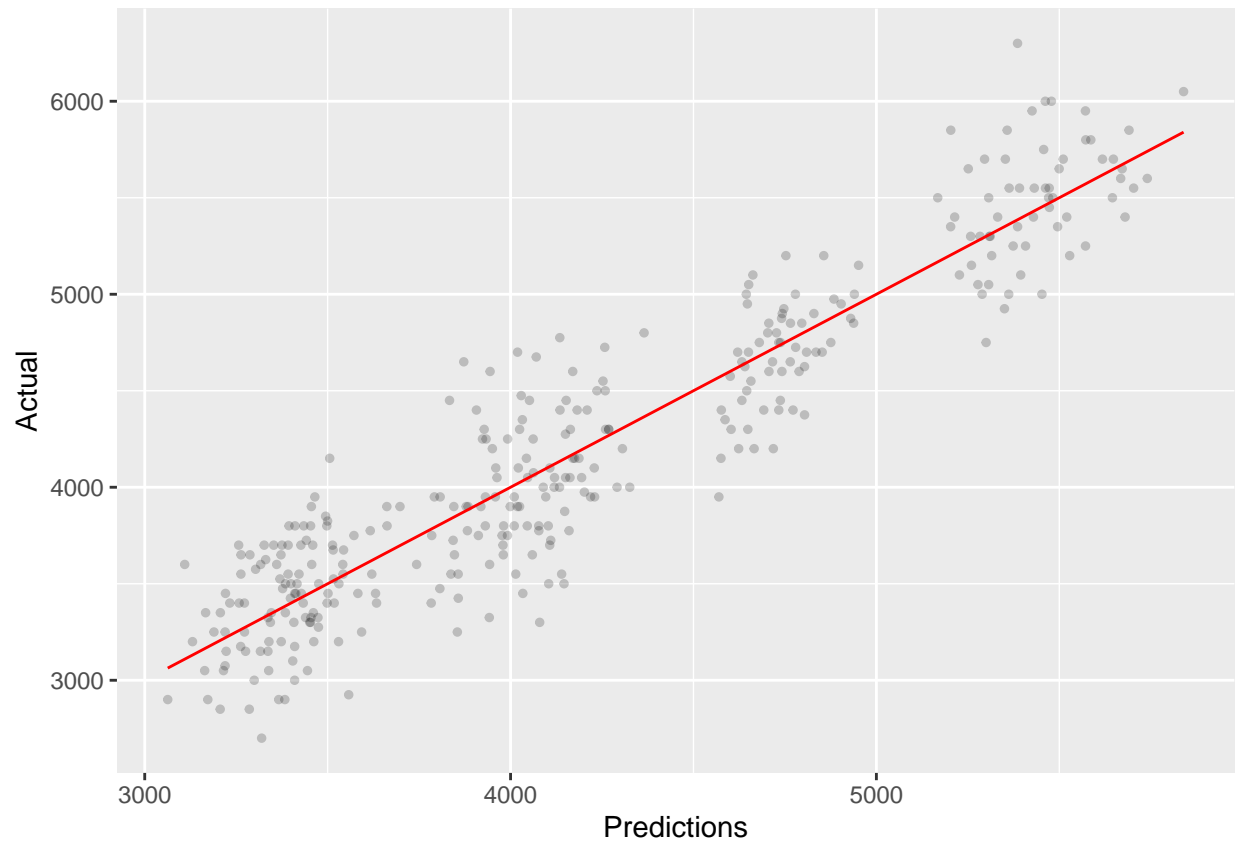
# creating the model
model <- lm(To_Predict ~ species + bill_length_mm + bill_depth_mm +
            flipper_length_mm + sex,
            data = tmp)

# visualizing the model
summary(model)
```

```
##
## Call:
## lm(formula = To_Predict ~ species + bill_length_mm + bill_depth_mm +
##     flipper_length_mm + sex, data = tmp)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -779.65 -173.18   -9.05   186.61   914.11
##
## Coefficients:
##              Estimate Std. Error t value      Pr(>|t|)
## (Intercept)   -1460.995     571.308  -2.557    0.011002 *
## speciesChinstrap  -251.477      81.079  -3.102    0.002093 **
## speciesGentoo     1014.627     129.561   7.831 0.00000000000006852 ***
## bill_length_mm      18.204       7.106   2.562    0.010864 *
## bill_depth_mm      67.218      19.742   3.405    0.000745 ***
## flipper_length_mm   15.950       2.910   5.482 0.000000008440786478 ***
## sexmale           389.892      47.848   8.148 0.00000000000000797 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 287.3 on 326 degrees of freedom
## Multiple R-squared:  0.875, Adjusted R-squared:  0.8727
## F-statistic: 380.2 on 6 and 326 DF, p-value: < 0.00000000000000022
```

We can visualize how the predictions align with the actual values.

```
# combine predictions and actual values
cbind(predict(model), tmp$To_Predict) %>%
  as.data.frame() %>%
  rename(Predictions = 1, Actual = 2) %>%
  # make a plot
  ggplot(aes(x = Predictions, y = Actual)) +
  geom_point(size=1, alpha = 0.2) +
  geom_line(aes(y = predict(model)), color = "red")
```



The R-square (R^2), which represents the correlation between the actual values and the predicted values. The higher the R^2 , the better the model (can interpret it as a correlation coefficient).

```
caret::R2(predict(model), tmp$To_Predict) %>% round(digits = 2)
```

```
## [1] 0.87
```

We can also extract the effect size of each variable by calculating how much the variance explained by the model drops when a given variable is removed.

```
# open an empty table to store the output
effect.table <- NULL %>% as.data.frame()

# extract a list with all the predictors, removing those we don't use
valid.names <- names(tmp)[!names(tmp) %in% c("To_Predict",
                                             "year",
                                             "island")]

# for each predictor
for(i in 0:length(valid.names)) {

  if(i == 0){
    # write the formula
    frm <- as.formula(paste("To_Predict ~ ",
                             paste(valid.names, collapse = "+")))
  }
}
```

```

# feed it to the model
model.tmp <- lm(frm, data = tmp)

#Determine R2:
effect.table <- rbind(effect.table,
                      c("all factors",
                        caret::R2(predict(model.tmp), tmp$To_Predict)))
}else{

  # write the formula
  frm <- as.formula(paste("To_Predict ~ ",
                          paste(valid.names[-i], collapse = "+")))

  # feed it to the model
  model.tmp <- lm(frm, data = tmp)

  #Determine R2:
  effect.table <- rbind(effect.table,
                        c(valid.names[i],
                          caret::R2(predict(model.tmp), tmp$To_Predict)))
}}

# adjust column names
colnames(effect.table) <- c("Factor", "Var.exp")

# add a column for effect size
effect.table <- effect.table %>%
  # make the column numeric
  mutate(Var.exp = as.numeric(Var.exp),
         # effect size = gap of variance explained with the main model
         Effect.size = as.numeric(effect.table$Var.exp[1]) - Var.exp) %>%
  filter(Factor != "all factors") %>%
  mutate(Effect.size = round(Effect.size, digits = 4))

# visual check
effect.table %>% arrange(desc(Effect.size))

```

```

##           Factor  Var.exp Effect.size
## 1      species 0.8230078      0.0520
## 2         sex 0.8494949      0.0255
## 3 flipper_length_mm 0.8634357      0.0115
## 4    bill_depth_mm 0.8705155      0.0044
## 5    bill_length_mm 0.8724449      0.0025

```

Linear mixed models

These models are similar to regression models. However, they include the distinction between fixed and random effects. Fixed effects are the variables that have a fixed type of interaction with the response variable while the random effects are the variables that do not. In short, mixed models consider that the **intercept and the slope can vary** across members of a group.

First, we compare which model is better. Typically, we look at the AIC (Akaike) score. The calculation of AIC not only regards the goodness of fit of a model, but also takes into account the simplicity of the model.

In this way, AIC deals with the trade-off between goodness of fit and complexity of the model, and as a result, discourages overfitting. A smaller AIC is preferred.

If you see the warning “boundary (singular) fit: see help(‘isSingular’)”, it means that the model can make too accurate predictions and it considers statistically unrealistic, i.e., the model overfits. If that happens, you can try removing some variables to avoid the warning. You can also report the model but highlighting that there is the possibility of overfitting.

```
# arrange the data a bit
tmp <- data %>%
  # change the coding to numerical values if needed
  mutate(body_mass_g = as.numeric(body_mass_g)) %>%
  # rename the variables to be predicted
  rename(To_Predict = body_mass_g) %>%
  # remove the rows with NA values
  drop_na()

# for random slope, e.g., (1+flipper_length_mm|island), data=base)
# This model, in addition to a random intercept for island, also contains a random slope in flipper length

# model with all factors, using year and island as random effects
model1 <- lmerTest::lmer(To_Predict ~ (1|year) + (1|island) +
  species + bill_length_mm + bill_depth_mm +
  flipper_length_mm + sex,
  # if you want to suppress the singularity warning
  #control=lmerControl(check.conv.singular =
  #.makeCC(action = "ignore", tol = 1e-4)),
  data = tmp)
```

boundary (singular) fit: see help(‘isSingular’)

```
# model with only measurements
model2 <- lmerTest::lmer(To_Predict ~ (1|year) + (1|island) +
  bill_length_mm + bill_depth_mm +
  flipper_length_mm,
  # if you want to suppress the singularity warning
  #control=lmerControl(check.conv.singular =
  #.makeCC(action = "ignore", tol = 1e-4)),
  data = tmp)

# model with only categorical variables
model3 <- lmerTest::lmer(To_Predict ~ (1|year) + (1|island) +
  species + sex,
  # if you want to suppress the singularity warning
  #control=lmerControl(check.conv.singular =
  #.makeCC(action = "ignore", tol = 1e-4)),
  data = tmp)
```

boundary (singular) fit: see help(‘isSingular’)

```
# null model
model4 <- lmerTest::lmer(To_Predict ~ (1|year) + (1|island),
  # if you want to suppress the singularity warning
```

```
#control=lmerControl(check.conv.singular =
#makeCC(action = "ignore", tol = 1e-4)),
data = tmp)
```

```
## boundary (singular) fit: see help('isSingular')
```

```
# comparing all the models
anova(model1, model2, model3, model4,
      # avoid that anova refits the models to ML
      refit = FALSE)
```

```
## Data: tmp
## Models:
## model4: To_Predict ~ (1 | year) + (1 | island)
## model2: To_Predict ~ (1 | year) + (1 | island) + bill_length_mm + bill_depth_mm + flipper_length_mm
## model3: To_Predict ~ (1 | year) + (1 | island) + species + sex
## model1: To_Predict ~ (1 | year) + (1 | island) + species + bill_length_mm + bill_depth_mm + flipper_length_mm
##      npar    AIC    BIC logLik deviance  Chisq Df      Pr(>Chisq)
## model4      4 5246.0 5261.3 -2619.0   5238.0
## model2      7 4864.1 4890.7 -2425.0   4850.1 387.965  3 < 0.00000000000000022 ***
## model3      7 4754.6 4781.3 -2370.3   4740.6 109.456  0
## model1     10 4675.9 4713.9 -2327.9   4655.9  84.745  3 < 0.00000000000000022 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can also test what happens when we remove one factor at a time.

```
drop1(model1, test="Chisq")
```

```
## Single term deletions using Satterthwaite's method:
##
## Model:
## To_Predict ~ (1 | year) + (1 | island) + species + bill_length_mm + bill_depth_mm + flipper_length_mm
##      Sum Sq Mean Sq NumDF  DenDF F value    Pr(>F)
## species    10318128 5159064      2 272.53 62.8403 < 0.00000000000000022
## bill_length_mm    555365  555365      1 324.66  6.7647    0.009723
## bill_depth_mm    854382  854382      1 316.55 10.4068    0.001387
## flipper_length_mm 2600439 2600439      1 159.41 31.6748 0.00000008017057969
## sex          5356973 5356973      1 325.92 65.2509 0.00000000000001294
##
## species      ***
## bill_length_mm **
## bill_depth_mm **
## flipper_length_mm ***
## sex          ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We visualize the output of the best model.

The **scaled residuals** should ideally be centered around 0, which would mean that the predictions are quite close to the real data points.

The Estimates can be manually read as follows. If in a hypothetical model, we are trying to predict the variable X based on the variables A, B, and sex, which have an estimate of 1,3, and -1 (for male). And the intercept is 0.5. If a data point has the following values for the hypothetical variables encoded in the data.

- X (the predicted variable) = 370
- A = 364
- B = 3
- sex = male

Then, the predicted X would be = the intercept 0.5 + (1 * 364) + (3 * 3) + (-1 * 1) = 372.5.

The random effect part tells you how much variance you find among levels of your grouping factors, plus the residual variance. If the variance is high, it means that the factors explain a lot of variation, e.g., we divide the variance of a factor by the total variance: $1/(1 + 3) = 0.25$. So the random effects explain 25% of the variance that is left over after the variance explained by the fixed effects.

Correlation of fixed effects is about the expected correlation of the regression coefficients. It is telling you that if you did the experiment again and it so happened that the coefficient for X got smaller, it is likely that the coefficient of Y would go up (or down).

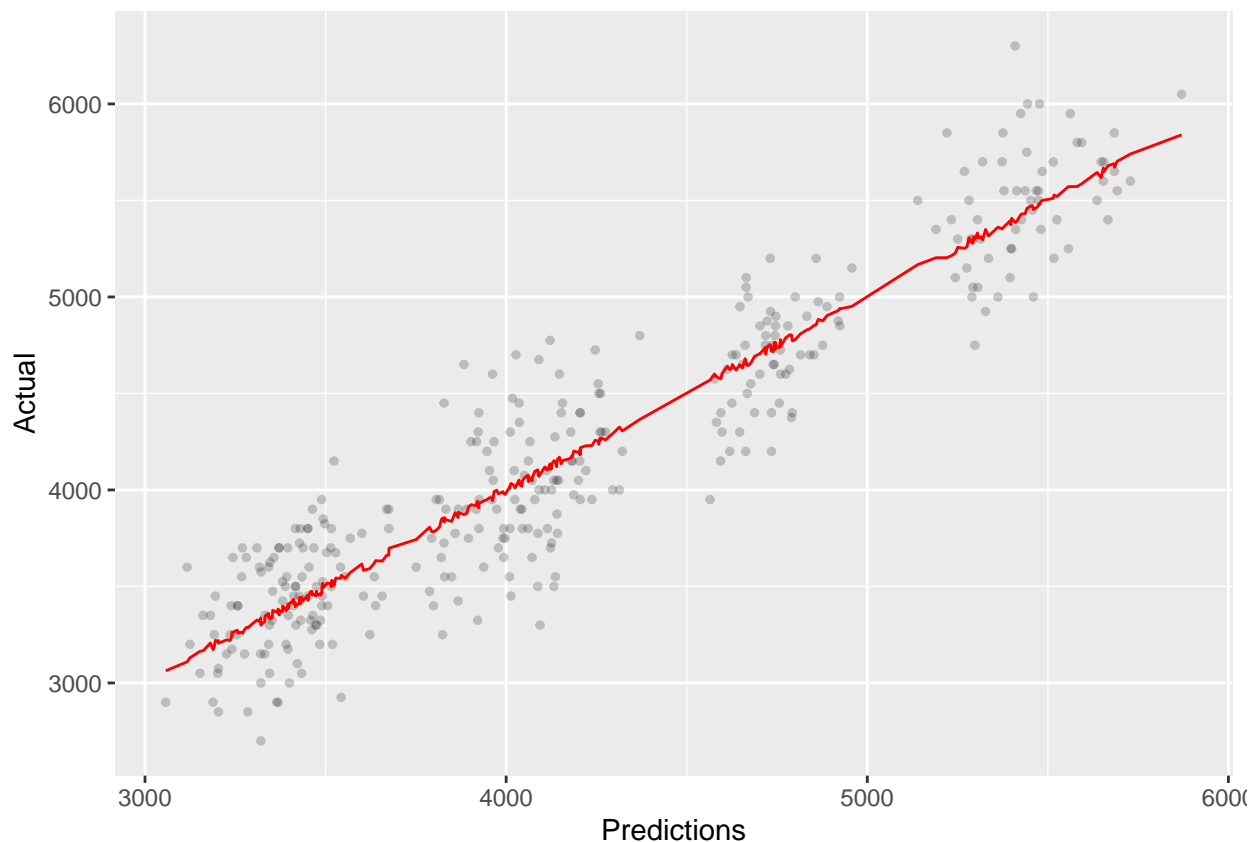
```
summary(model1)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: To_Predict ~ (1 | year) + (1 | island) + species + bill_length_mm +
##      bill_depth_mm + flipper_length_mm + sex
##      Data: tmp
##
## REML criterion at convergence: 4655.9
##
## Scaled residuals:
##      Min      1Q   Median      3Q      Max
## -2.77114 -0.60182 -0.01518  0.64234  3.10817
##
## Random effects:
##  Groups      Name                Variance Std.Dev.
##  year      (Intercept)          784.8    28.01
##  island    (Intercept)           0.0      0.00
##  Residual                        82098.0  286.53
## Number of obs: 333, groups:  year, 3; island, 3
##
## Fixed effects:
##              Estimate Std. Error      df t value      Pr(>|t|)
## (Intercept)   -1608.292    580.323   269.712  -2.771    0.00597 **
## speciesChinstrap  -262.185     81.304   324.311  -3.225    0.00139 **
## speciesGentoo     973.655    132.800   218.510   7.332  0.00000000000043681 ***
## bill_length_mm    18.533      7.126   324.662   2.601    0.00972 **
## bill_depth_mm     64.062     19.858   316.552   3.226    0.00139 **
## flipper_length_mm  16.978      3.017   159.406   5.628  0.0000000801705797 ***
## sexmale         386.365     47.830   325.917   8.078  0.0000000000000129 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
```

```
##          (Intr) spcsCh spcsGn bll_l_ bll_d_ flpp__
## spcsChnstrp  0.305
## speciesGent  0.364  0.536
## bll_lngth_m -0.170 -0.829 -0.409
## bll_dpth_mm -0.328  0.152  0.702 -0.138
## flppr_lngt_ -0.744 -0.033 -0.627 -0.207 -0.218
## sexmale      0.602  0.375  0.056 -0.375 -0.455 -0.182
## optimizer (nloptwrap) convergence code: 0 (OK)
## boundary (singular) fit: see help('isSingular')
```

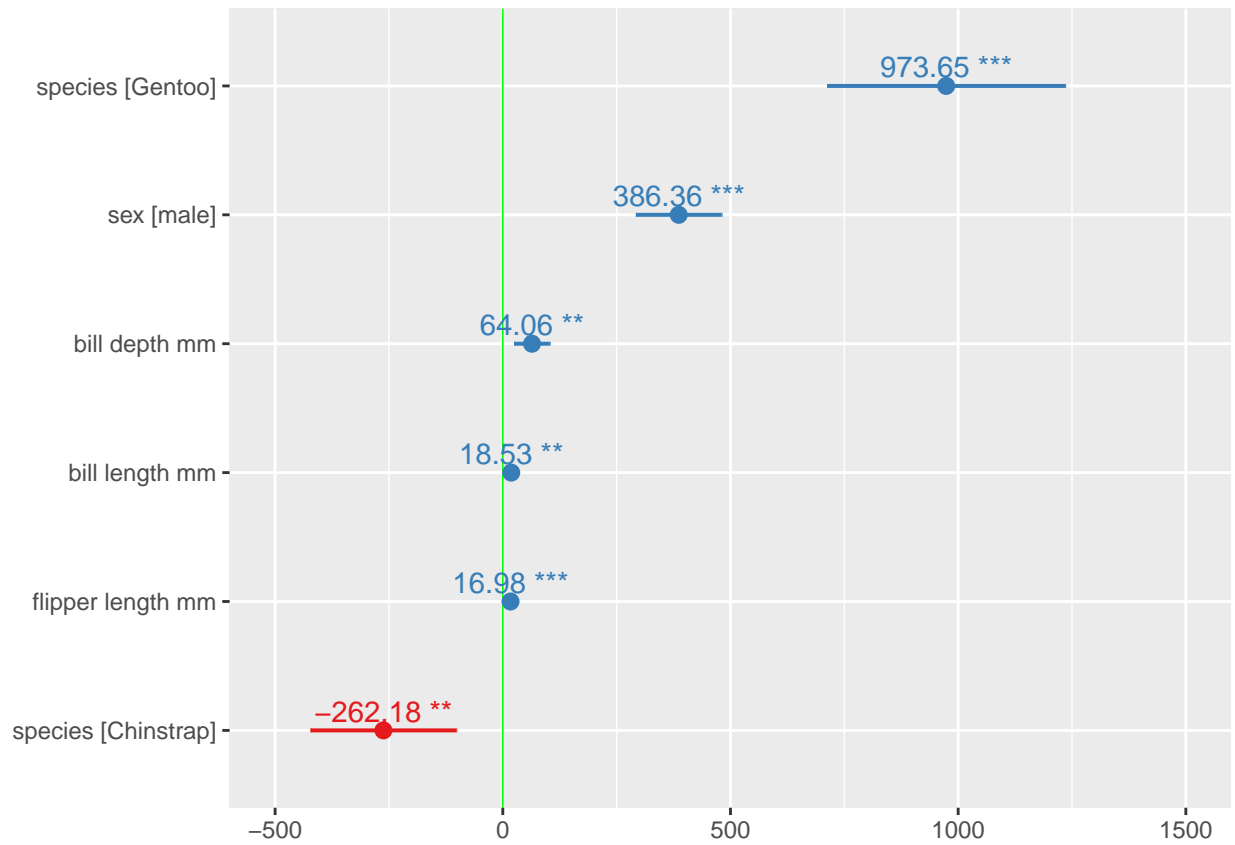
We can visualize how the predictions align with the actual values.

```
# combine predictions and actual values
cbind(predict(model1), tmp$To_Predict) %>%
  as.data.frame() %>%
  rename(Predictions = 1, Actual = 2) %>%
  # make a plot
ggplot(aes(x = Predictions, y = Actual)) +
  geom_point(size=1, alpha = 0.2) +
  geom_line(aes(y = predict(model)), color = "red")
```



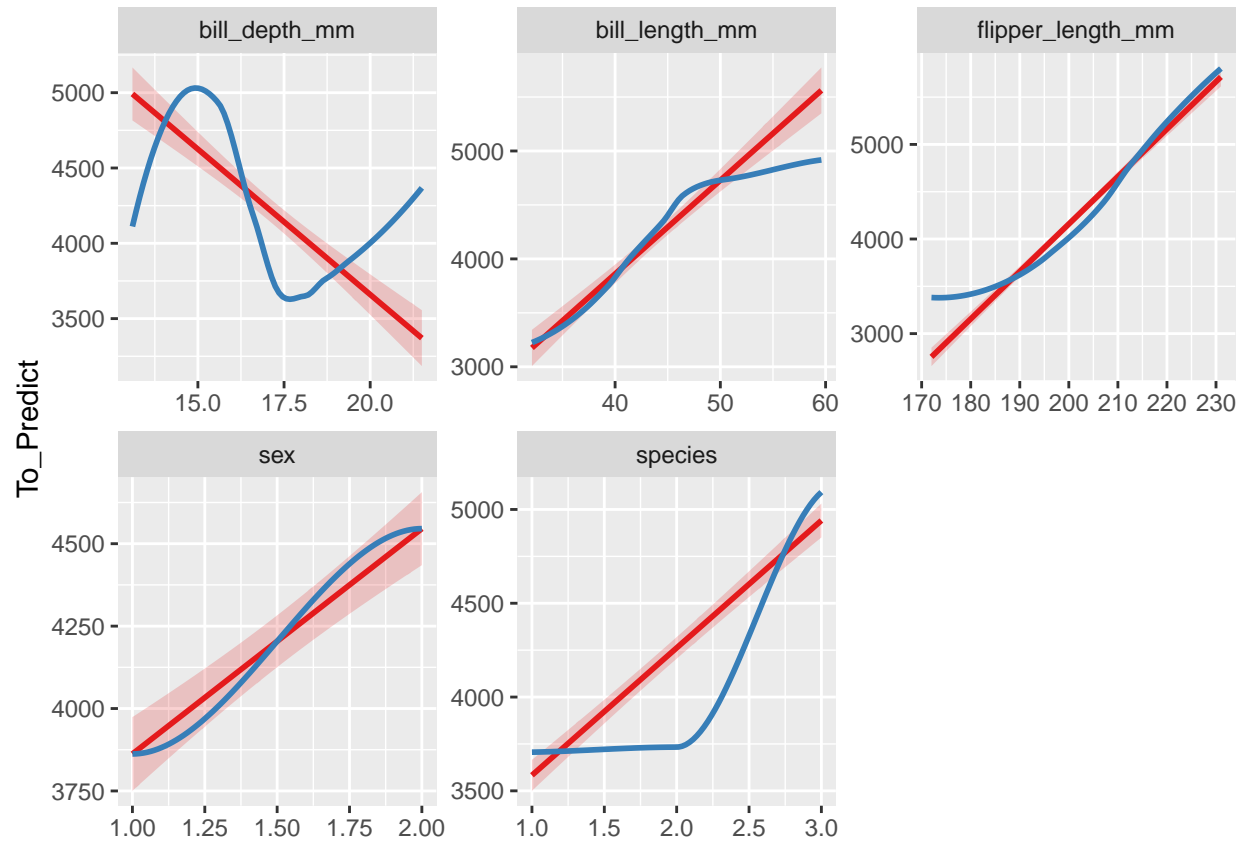
We visualize the fixed effects.

```
#png("plot.png", res=300, height=1100, width=1600)
sjPlot::plot_model(model1, type = "est", auto.label = FALSE,
  vline.color = "green", sort.est = TRUE, show.values = TRUE)
```



```
#dev.off()
#sjPlot::plot_model(model1, type = "pred")
sjPlot::plot_model(model1, type = "slope")
```

```
## 'geom_smooth()' using formula = 'y ~ x'
## 'geom_smooth()' using formula = 'y ~ x'
```

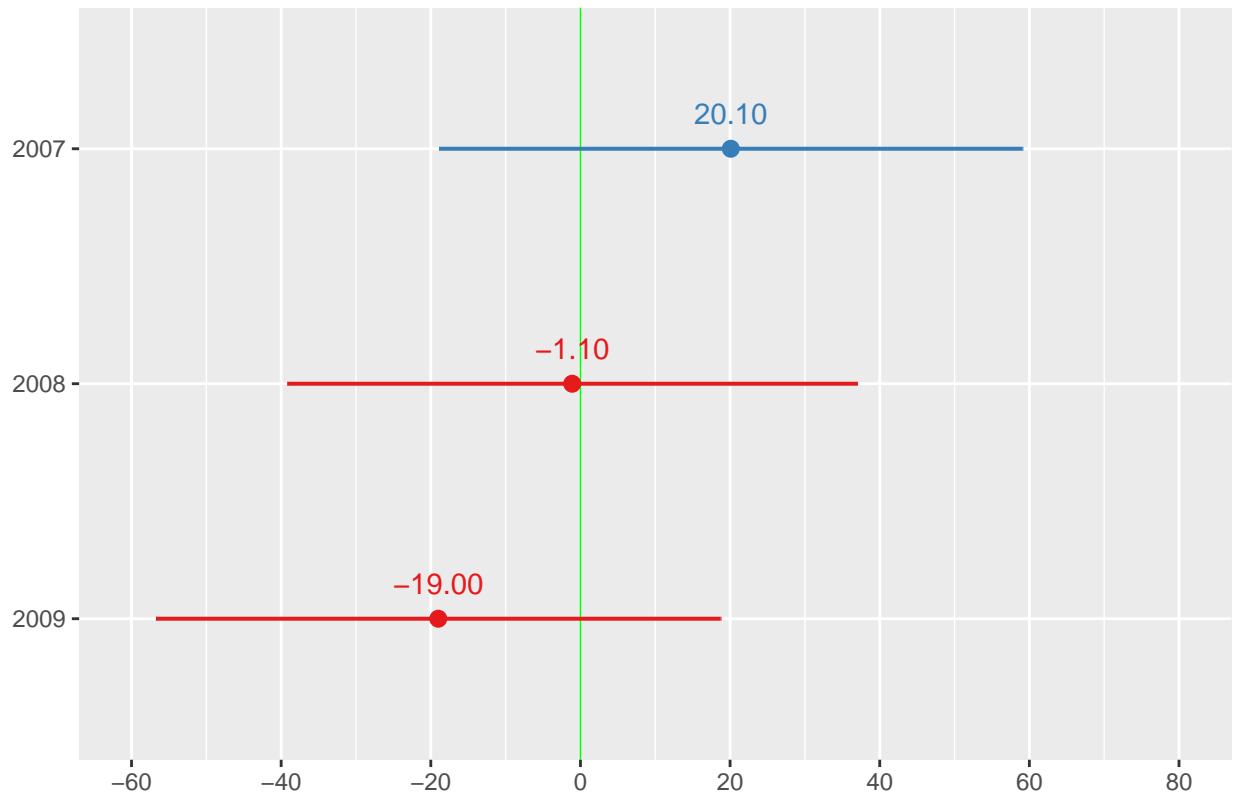


We also plot the random effects. Ideally, you want them centered around the middle line and crossing it, showing that there is no strong effect from these variables.

```
sjPlot::plot_model(model1, auto.label = FALSE, type = "re", # plots random effects
  vline.color = "green", sort.est = TRUE, show.values = TRUE,
  # add this to get the effects sorted for random effects
  grid = FALSE)
```

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

Random effects of year (Intercept)



```
##  
## [[2]]
```

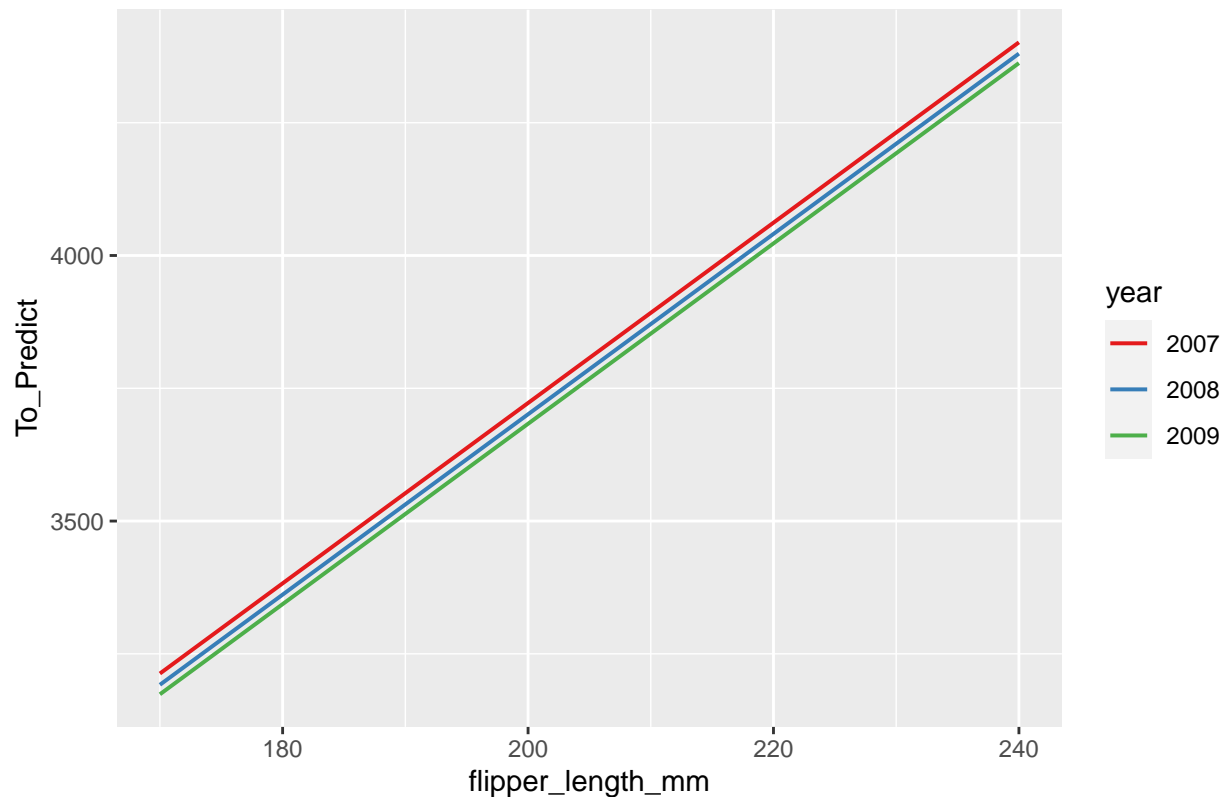
Random effects of island (Intercept)



We can plot the random intercepts and slopes

```
plot_model(model1, type="pred",  
            terms=c("flipper_length_mm", "year"),  
            pred.type="re", ci.lvl = NA)
```

Predicted values of To_Predict



We can extract the effect size of each variable by calculating how much the variance explained by the model drops when a given variable is removed.

```
# open an empty table to store the output
effect.table <- NULL %>% as.data.frame()

# extract a list with all the predictors
valid.names <- names(tmp)[!names(tmp) %in% c("To_Predict",
                                             "year",
                                             "island")]

# for each predictor
for(i in 0:length(valid.names)) {

  if(i == 0){
    # write the formula
    frm <- as.formula(paste("To_Predict ~ (1|year) + (1|island) +",
                           paste(valid.names, collapse = "+")))
    # feed it to the model
    model.tmp <- lmerTest::lmer(frm, data = tmp)

    #Determine R2:
    effect.table <- rbind(effect.table,
                          c("all factors",
                            caret::R2(predict(model.tmp), tmp$To_Predict)))
  }else{
```

```

# write the formula
frm <- as.formula(paste("To_Predict ~ (1|year) + (1|island) +",
                        paste(valid.names[-i], collapse = "+")))

# feed it to the model
model.tmp <- lmerTest::lmer(frm, data = tmp)

#Determine R2:
effect.table <- rbind(effect.table,
                      c(valid.names[i],
                        caret::R2(predict(model.tmp), tmp$To_Predict)))
}}

```

```

## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')
## boundary (singular) fit: see help('isSingular')

```

```

# adjust column names
colnames(effect.table) <- c("Factor", "Var.exp")

# add a column for effect size
effect.table <- effect.table %>%
  # make the column numeric
  mutate(Var.exp = as.numeric(Var.exp),
         # effect size = gap of variance explained with the main model
         Effect.size = as.numeric(effect.table$Var.exp[1]) - Var.exp) %>%
  filter(Factor != "all factors") %>%
  mutate(Effect.size = round(Effect.size, digits = 4))

# visual check
effect.table %>% arrange(desc(Effect.size))

```

```

##           Factor  Var.exp Effect.size
## 1      species 0.8478408     0.0282
## 2         sex 0.8518421     0.0242
## 3 flipper_length_mm 0.8634357     0.0126
## 4    bill_depth_mm 0.8727494     0.0033
## 5    bill_length_mm 0.8733327     0.0027

```

- The R-square (R2), which represents the correlation between the actual values and the predicted values. The higher the R2, the better the model (can interpret it as a correlation coefficient).

```

caret::R2(predict(model1), tmp$To_Predict) %>% round(digits = 2)

```

```

## [1] 0.88

```

```

# r2marginal represents the variance explained by the fixed effects
# r2conditional represents the variance explained by the entire model (fixed + random effects)
MuMIn::r.squaredGLMM(model1)

```



```
## Warning: 'r.squaredGLMM' now calculates a revised statistic. See the help page.
```

```
##           R2m           R2c
```

```
## [1,] 0.8728317 0.8740358
```