Regression: model development

ENVX2001 Applied Statistical Methods

Liana Pozza

Feb 2026

Variable selection

> "The hardest thing to learn in life is which bridge to cross and which to burn."
>
> – David Russell

Workflow

1. Model development
   - Explore: visualise, summarise
   - Transform predictors: linearise, reduce skewness/leverage
   - Model: fit, check assumptions, interpret, transform. Repeat.
2. Variable selection
   - VIF: remove predictors with high variance inflation factor
   - Model selection: stepwise selection, AIC, principle of parsimony, assumption checks
3. Predictive modelling
   - Predict: Use the model to predict new data
   - Validate: Evaluate the model's performance

Previously on ENVX2001...

We fitted a multiple linear regression model to the data.

```
full_fit ← lm(log(Ozone) ~ Temp + Solar.R + Wind, data = airquality)
summary(full_fit)
```

```

Call:
lm(formula = log(Ozone) ~ Temp + Solar.R + Wind, data = airquality)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.06193 -0.29970 -0.00231  0.30756  1.23578 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -0.2621323  0.5535669  -0.474 0.636798    
Temp         0.0491711  0.0060875   8.077 1.07e-12 ***
Solar.R      0.0025152  0.0005567   4.518 1.62e-05 ***
Wind        -0.0615625  0.0157130  -3.918 0.000158 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5086 on 107 degrees of freedom
  (42 observations deleted due to missingness)
Multiple R-squared:  0.6644,    Adjusted R-squared:  0.655
F-statistic: 70.62 on 3 and 107 DF,  p-value: < 2.2e-16
```

$$\widehat{log(Ozone)} = -0.262 + 0.0492 \cdot Temp + 0.00252 \cdot Solar.R - 0.0616 \cdot Wind$$

Question

$$\widehat{log(Ozone)} = -0.262 + 0.0492 \cdot Temp + 0.00252 \cdot Solar.R - 0.0616 \cdot Wind$$

Are all the variables/predictors needed?

Principles

A good model:

- Has only *useful* predictors: principle of parsimony
- Has *no redundant* predictors: principle of orthogonality (no multicollinearity)
- Is *interpretable* (principle of transparency; last week), or *predicts* well (principle of accuracy; next week)

On the principle of parsimony

- Ockham's razor: "Entities should not be multiplied unnecessarily."
- One should prefer the *simplest* explanation that fits the data if multiple explanations are equally good.

> "It is vain to do with more what can be done with fewer."

– William of Ockham (1287–1347)

What happens when we add more predictors to a model?

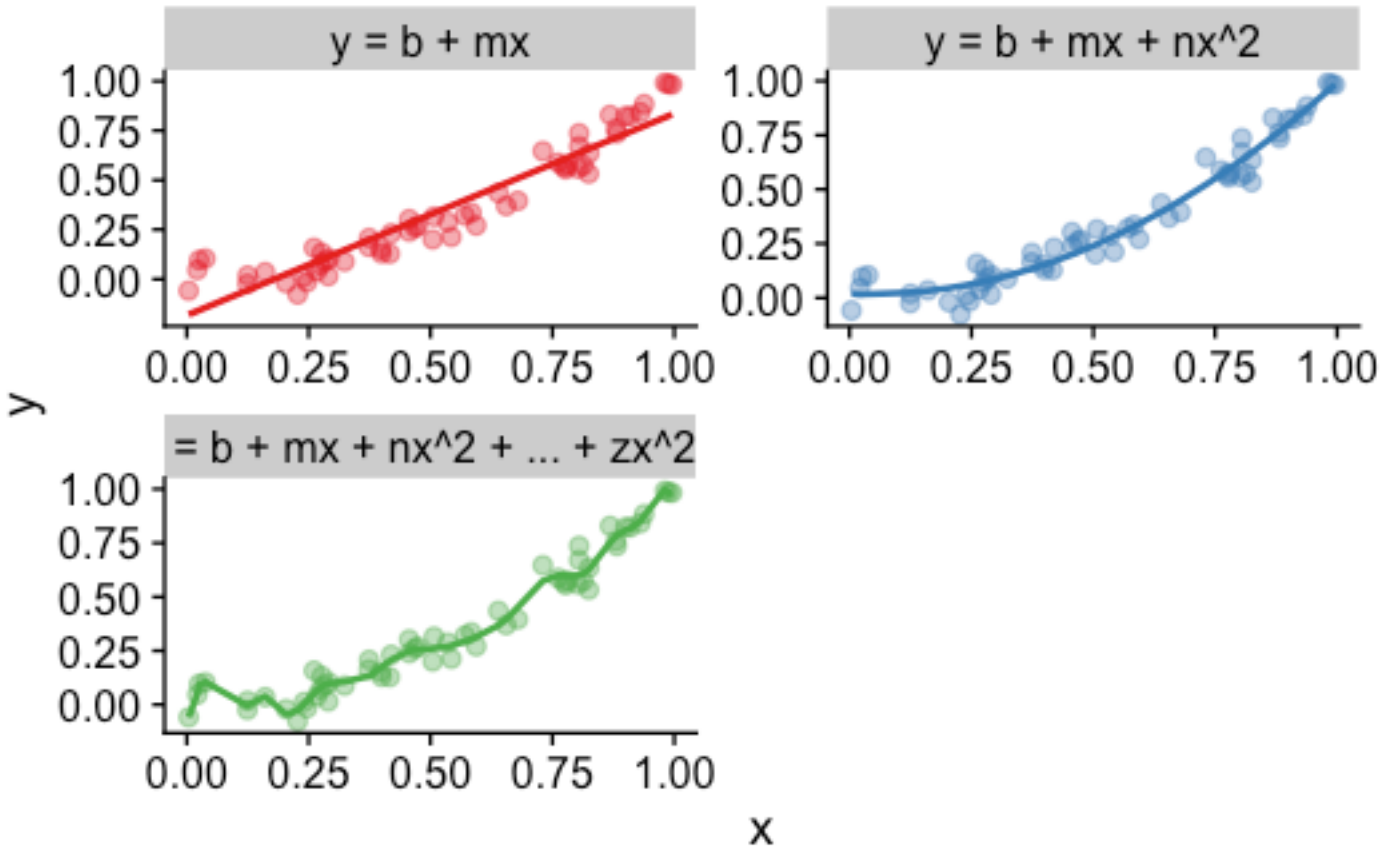
A simple example using polynomial regression.

- The more predictors we include, the more variance we can explain.
- However, the more predictors and complexity we include, the more overfitted the model becomes.

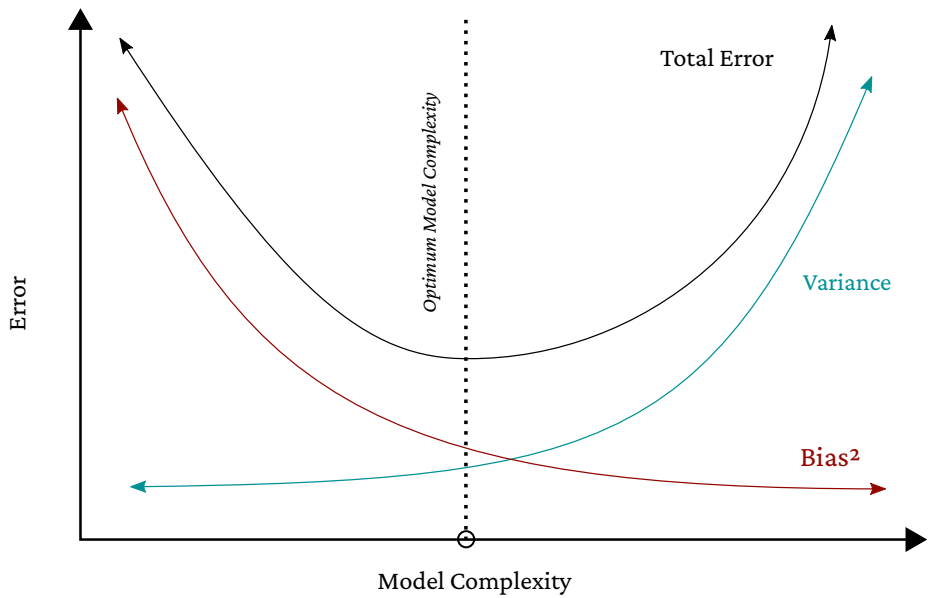
```r
set.seed(1030)
xsquared ← function(x) {
  x^2
}
# Generate xy data
sim_data ← function(xsquared, sample_size = 100) {
  x ← runif(n = sample_size, min = 0, max = 1)
  y ← rnorm(n = sample_size, mean = xsquared(x), sd = 0.05)
  data.frame(x, y)
}
# Generate predicted data (model)
df ← sim_data(xsquared, sample_size = 60)
fit ← lm(y ~ 1, data = df)
fit_1 ← lm(y ~ poly(x, degree = 1), data = df)
fit_2 ← lm(y ~ poly(x, degree = 2), data = df)
```

```r
fit_many ← lm(y ~ poly(x, degree = 20), data = df)
truth ← seq(from = 0, to = 1, by = 0.01)
# Combine the data and model fits into a single data frame
df ← data.frame(
  x = df$x,
  y = df$y,
  fit = predict(fit),
  fit_1 = predict(fit_1),
  fit_2 = predict(fit_2),
  fit_many = predict(fit_many)
)

# Reshape the data frame into long format
df_long ← pivot_longer(
  df,
  cols = starts_with("fit_"),
  names_to = "model",
  values_to = "value"
) %>%
  mutate(
    model = case_when(
```

```
      model == "fit" ~ "y = b",
      model == "fit_1" ~ "y = b + mx",
      model == "fit_2" ~ "y = b + mx + nx^2",
      model == "fit_many" ~ "y = b + mx + nx^2 + ... + zx^20",
      TRUE ~ model
    )
  )
# Plot
p ← ggplot(df_long, aes(x = x, y = value, color = model)) +
  facet_wrap(~model, ncol = 2, scales = "free") +
  geom_point(aes(y = y), alpha = .4, size = 2) +
  geom_line(linewidth = .9, linetype = 1) +
  scale_color_brewer(palette = "Set1") +
  theme(legend.position = "none") +
  geom_blank()
p
```


y = b + mx
y = b + mx + nx^2
= b + mx + nx^2 + ... + zx^2
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
y
x

Variance-bias trade-off



In concept...

- As complexity increases, bias ($\sum O - \bar{P}$) *decreases* (the mean of a model's predictions is closer to the true mean).
- As complexity increases, prediction variance ($\frac{\sum (P-\bar{P})^2}{n}$) increases.
- The **goal** is to find a model that isn't too simple or complex with a good balance between bias and variance.

$$\text{Mean Squared Error} = \text{Bias}^2 + \text{Variance} + \text{Immeasureable Error}$$

In practice, the math and relationships are a bit more irregular.

How do we determine the best model?

Some model quality measures you should be familiar with:

- **R^2**: variance explained by the model with a maximum value of 1 = 100%
- **Residual standard error**: the mean error of the observed values from the predicted/fitted values (i.e. line of best fit).
- **Partial F-test**: compare the full model to a reduced model, works well when the number of predictors is small and simple models.

Some other commonly used measures:

- **Information criteria**: AIC, BIC, etc. (more on this later).
- **Error measures**: useful when the aim for the model is to **predict**. The best model has the smallest residual error (or other similar metrics).

What models do we try?

- **Everything**: all possible combinations of predictors.
  - Each variable can either be included or excluded so the number of possible combinations is 2^n
  - e.g. 3 predictors (x_1, x_2, x_3) could have 8 models
    - No variables i.e. mean y the null hypothesis
    - 1 variable: x_1; x_2; x_3
    - 2 variables: $x_1 + x_2$; $x_1 + x_3$; $x_2 + x_3$
    - 3 variables: $x_1 + x_2 + x_3$
  - So...not recommended
- **Stepwise regression**: add/remove predictors one at a time until removing a variable makes the model worse.
- **Select meaningful predictors** based on domain knowledge, correlation, or significance.
- More complex approaches are available for big data and machine learning.

Air quality: can we reduce the number of predictors?

Full model:

```
summary(full_fit)
```

```
Call:
lm(formula = log(Ozone) ~ Temp + Solar.R + Wind, data = airquality)

Residuals:
     Min       1Q   Median       3Q      Max
-2.06193 -0.29970 -0.00231  0.30756  1.23578

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.2621323  0.5535669  -0.474 0.636798
Temp         0.0491711  0.0060875   8.077 1.07e-12 ***
Solar.R      0.0025152  0.0005567   4.518 1.62e-05 ***
Wind        -0.0615625  0.0157130  -3.918 0.000158 ***
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5086 on 107 degrees of freedom
  (42 observations deleted due to missingness)
Multiple R-squared:  0.6644,    Adjusted R-squared:  0.655
F-statistic: 70.62 on 3 and 107 DF,  p-value: < 2.2e-16
```

- `Wind` has the highest p-value, can we remove it?
- Full model: Multiple R-squared = 0.66, Adjusted R-squared = 0.66

- Full model: multiple R-squared = 0.66, adjusted R-squared = 0.66

Reduced model: take out `Wind`

```
reduced_fit ← lm(log(Ozone) ~ Temp + Solar.R, data = airquality)
summary(reduced_fit)
```

```

Call:
lm(formula = log(Ozone) ~ Temp + Solar.R, data = airquality)

Residuals:
     Min       1Q   Median       3Q      Max
-1.83864 -0.33727  0.03444  0.29877  1.38210

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.7646527  0.4249016  -4.153 6.58e-05 ***
Temp         0.0607386  0.0056663  10.719  < 2e-16 ***
Solar.R      0.0024651  0.0005924   4.161 6.38e-05 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5413 on 108 degrees of freedom
  (42 observations deleted due to missingness)
Multiple R-squared:  0.6163,    Adjusted R-squared:  0.6092
F-statistic: 86.73 on 2 and 108 DF,  p-value: < 2.2e-16
```

- Reduced model: multiple R-squared = 0.62, adjusted R-squared = 0.61
- **Adjusted R-squared is lower, but is a 4% difference "worth it"? Is it significant?**

The R^2 value

The R-squared value is the proportion of variance explained by the model.

$$R^2 = \frac{SS_{reg}}{SS_{tot}} = 1 - \frac{SS_{res}}{SS_{tot}}$$

The adjusted R-squared value is the proportion of variance explained by the model, adjusted for the number of predictors.

$$R^2_{adj} = 1 - \frac{SS_{res}}{SS_{tot}} \frac{n-1}{n-p-1}$$

where n is the number of observations and p is the number of predictors.

Partial F-test

Partial F-test

How much of an improvement in adjusted R^2 is worth having an extra variable / more complex model?

- We can perform a hypothesis test to determine whether the improvement is significant.
- The F-test measures the full model against an intercept only model in terms of explained variance (residual sum of squares).
- The **partial F-test** compares the full model to a reduced model in terms of the trade-off between model complexity and variance explained (i.e. **adjusted** R^2).
  - H_0: no significant difference between the full and reduced models
  - H_1: the full model is significantly better than the reduced model
  - Calculating the F-stat:

$$F = \left| \frac{SS_{reg,full} - SS_{reg,reduced}}{(df_{res,full} - df_{res,reduced})} \right| \div MS_{res,full}$$

Partial F-test: calculation

$$F = \left| \frac{SS_{reg,full} - SS_{reg,reduced}}{(df_{res,full} - df_{res,reduced})} \right| \div MS_{res,full}$$

where:

- $SS_{reg,full}$ is the sum of squares of the full model (total of predictors)
- $SS_{reg,reduced}$ is the sum of squares of the reduced model (total of predictors)
- $df_{res,full}$ is the degrees of freedom of the residuals of the full model
- $df_{res,reduced}$ is the degrees of freedom of the residuals of the reduced model
- $MS_{res,full}$ is the mean square of the residuals of the full model

```
full ← anova(full_fit) %>% broom::tidy()
full
```

```
# A tibble: 4 × 6
  term          df sumsq meansq statistic
p.value
  <chr>      <int> <dbl>  <dbl>     <dbl>
```

```
reduced ← anova(reduced_fit) %>% broom::tidy()
reduced
```

```
# A tibble: 3 × 6
  term          df sumsq meansq statistic
p.value
  <chr>      <int> <dbl>  <dbl>     <dbl>
```

```
<dbl>
1 Temp           1 45.8  45.8       177.
2.07e-24
2 Solar.R        1  5.07  5.07       19.6  2.29e-
5
3 Wind           1  3.97  3.97       15.4  1.58e-
4
4 Residuals    107 27.7   0.259      NA
NA
```

```
<dbl>
1 Temp           1 45.8  45.8       156.
1.05e-22
2 Solar.R        1  5.07  5.07       17.3  6.38e-
5
3 Residuals    108 31.6   0.293      NA
NA
```

Each row is the *individual effect* of each predictor on the response $log(Ozone)$ (whilst holding all other predictors constant).

By hand

$$F = \left| \frac{SS_{reg,full} - SS_{reg,reduced}}{(df_{res,full} - df_{res,reduced})} \right| \div MS_{res,full}$$

- $SS_{reg,full} = 45.8 + 5.07 + 3.97 = 54.84$
- $SS_{reg,reduced} = 45.8 + 5.07 = 50.87$
- $df_{res,full} = 107$
- $df_{res,reduced} = 108$
- $MS_{res,full} = 0.259$

$F = \left| \frac{54.84 - 50.87}{(107 - 108)} \right| \div 0.259 = 15.33$

In R (manually)

```
ss_full ← sum(full$sumsq[1:3])
ss_reduced ← sum(reduced$sumsq[1:2])
df_full ← full$df[4]
df_reduced ← reduced$df[3]
ms_full ← full$meansq[4]
F ← abs((ss_full - ss_reduced) / ((df_full - df_reduced))) / ms_full
F     # F-statistic
```

```
[1] 15.35026
```

```
pf(F, df1 = 1, df2 = df_full, lower.tail = FALSE) # corresponding p-value
```

```
[1] 0.0001576806
```

(There is a slight difference due to rounding)

In R with functions

```
anova(reduced_fit, full_fit) # reduced goes first else will see negative df
```

```
Analysis of Variance Table

Model 1: log(Ozone) ~ Temp + Solar.R
Model 2: log(Ozone) ~ Temp + Solar.R + Wind
  Res.Df    RSS Df Sum of Sq     F    Pr(>F)
1    108 31.645
2    107 27.675  1    3.9703 15.35 0.0001577 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The partial F-test is significant (p-value < 0.05), so we can reject the null hypothesis and conclude that the full model is significantly better, even if adjusted R^2 improves by 4%.

But wait...

Looking back at the original model, we can see that the partial regression coefficients are the *same* as the partial F-test results!

```
anova(reduced_fit, full_fit) # partial F-test
```

```
Analysis of Variance Table

Model 1: log(Ozone) ~ Temp + Solar.R
Model 2: log(Ozone) ~ Temp + Solar.R + Wind
  Res.Df    RSS Df Sum of Sq     F    Pr(>F)
1    108 31.645
2    107 27.675  1    3.9703 15.35 0.0001577 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(full_fit)$coefficients
```

```
               Estimate   Std. Error    t value     Pr(>|t|)
(Intercept) -0.262132313 0.5535668608 -0.4735332 6.367975e-01
Temp         0.049171124 0.0060875025  8.0773888 1.069103e-12
Solar.R      0.002515177 0.0005567301  4.5177673 1.616797e-05
Wind        -0.061562470 0.0157129654 -3.9179409 1.576806e-04
```

This is because the reduced model is *nested* within the full model so the partial F-test is equivalent to a partial regression coefficient test.

Nested models

- Previous example is a simple example of a **nested model**.
- A model is nested within another model if the predictors in the first model are a subset of the predictors in the second model.
- This makes comparing the two models easier, as we can compare the regression coefficients of the two models.

Example

- If the original model is y ~ a + b + c:
  - Nested: y ~ a + b
  - Nested: y ~ a
  - *Not* nested: y ~ a + b + **d** – because **d** is not in the full model

> ! Important
>
> **Partial F-tests will *only* make sense/work for nested models!**

Another example: Bird abundance

About

```
loyn ← read.csv("data/loyn.csv")
str(loyn)
```

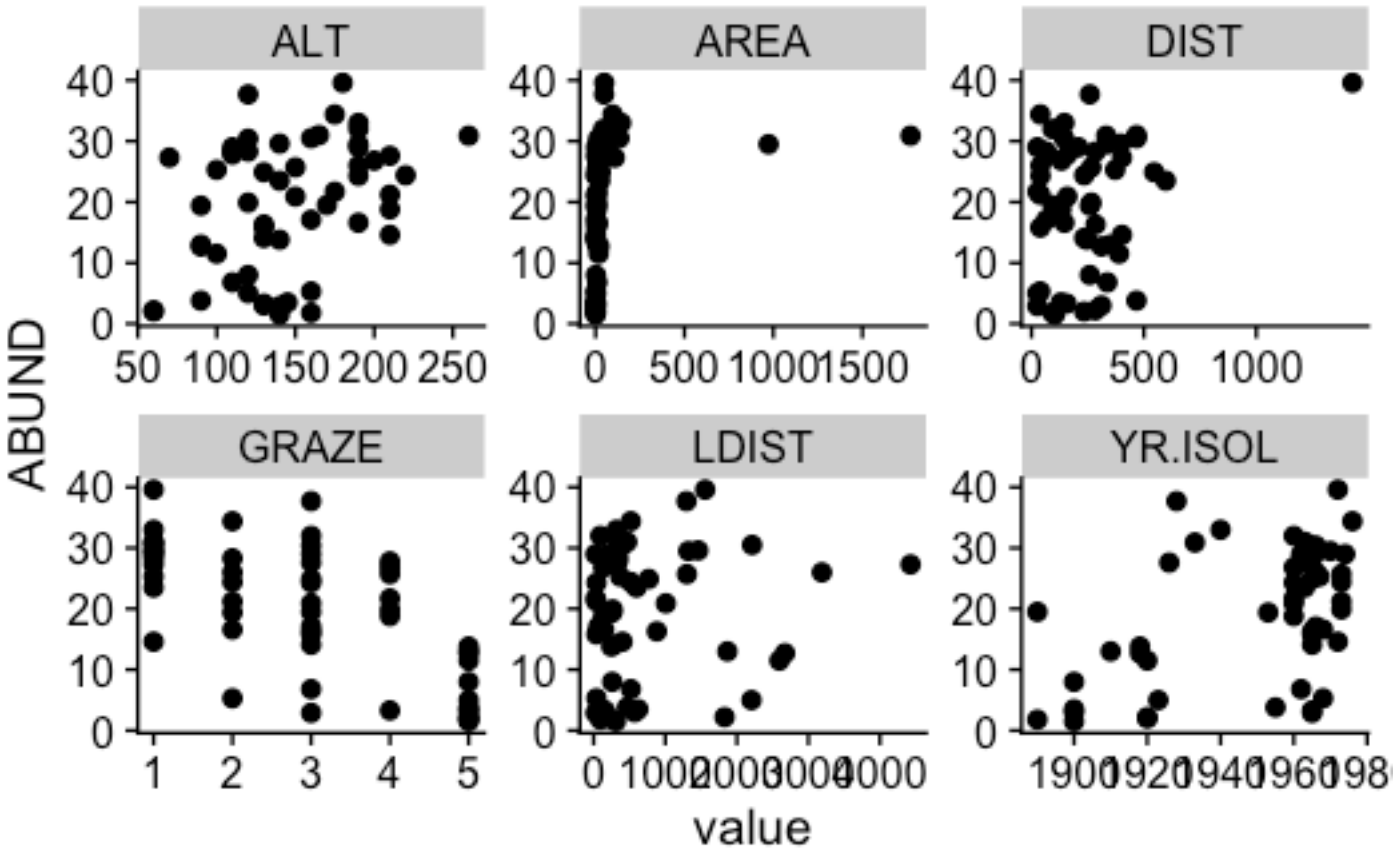
```
'data.frame':   56 obs. of  7 variables:
 $ ABUND  : num  5.3 2 1.5 17.1 13.8 14.1 3.8 2.2 3.3 3 ...
 $ AREA   : num  0.1 0.5 0.5 1 1 1 1 1 1 1 ...
 $ YR.ISOL: int  1968 1920 1900 1966 1918 1965 1955 1920 1965 1900 ...
 $ DIST   : int  39 234 104 66 246 234 467 284 156 311 ...
 $ LDIST  : int  39 234 311 66 246 285 467 1829 156 571 ...
 $ GRAZE  : int  2 5 5 3 5 3 5 5 4 5 ...
 $ ALT    : int  160 60 140 160 140 130 90 60 130 130 ...
```

- Can we predict the abundance of birds in forest patches cleared for agriculture, based on patch size, area, grazing and other variables?
- Loyn (1987)
  - DIST: Distance to nearest patch (km)
  - LDIST: Distance to a larger patch (km)
  - AREA: Patch area (ha)

- GRAZE: Grazing pressure 1 (light) – 5 (heavy) – ALT: Altitude (m)
- YR.ISOL: Years since isolation (years)
- ABUND: Density of forest birds in a forest patch (birds/patch)

Data exploration

```
loyn %>%
  pivot_longer(-ABUND) %>%
  ggplot(aes(x = value, y = ABUND)) +
  geom_point() +
  facet_wrap(~name, scales = "free") +
  labs(y = "ABUND")
```

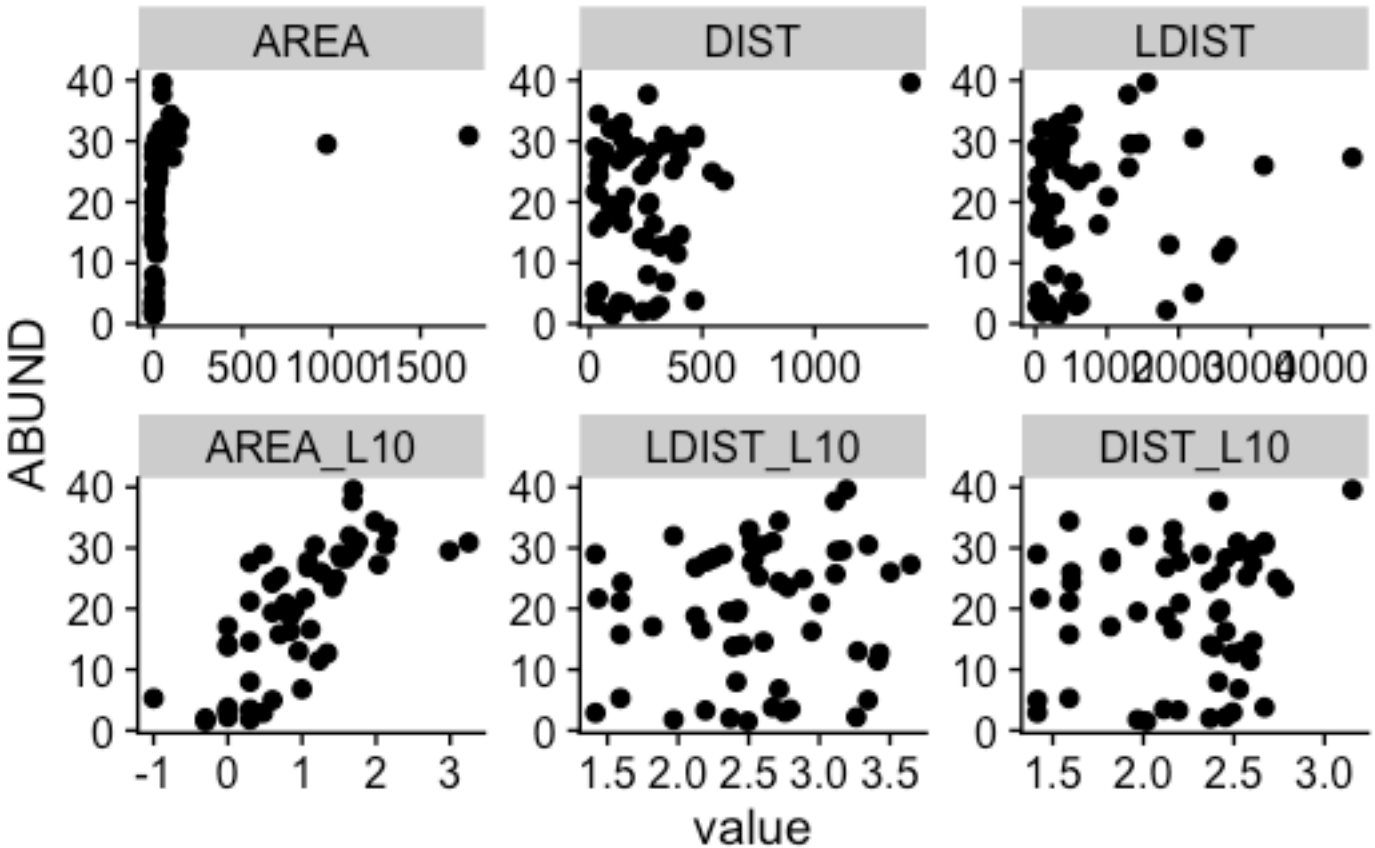


- The predictors are on very different scales, which can cause problems for the model.
- The relationships don't look particularly linear...and outliers.

- We will perform $\log_{10}$ transforms of `AREA`, `LDIST`, and `DIST`.

Log_{10} transformation

```
# perform transformations
loyn ← loyn %>%
  mutate(AREA_L10 = log10(AREA),
         LDIST_L10 = log10(LDIST),
         DIST_L10 = log10(DIST))

# View distributions again
loyn %>%
  select(-ALT, -GRAZE, -YR.ISOL) %>%
  pivot_longer(-ABUND) %>%
  mutate(name = factor(name, levels = unique(name))) %>%  # Preserve original order
  ggplot(aes(x = value, y = ABUND)) +
  geom_point() +
  facet_wrap(~name, scales = "free") +
  labs(y = "ABUND")
```


AREA
DIST
LDIST
AREA_L10
LDIST_L10
DIST_L10
ABUND
value
0
10
20
30
40
0
500
1000
1500
-1
1.5
2.0
2.5
3.0
3.5

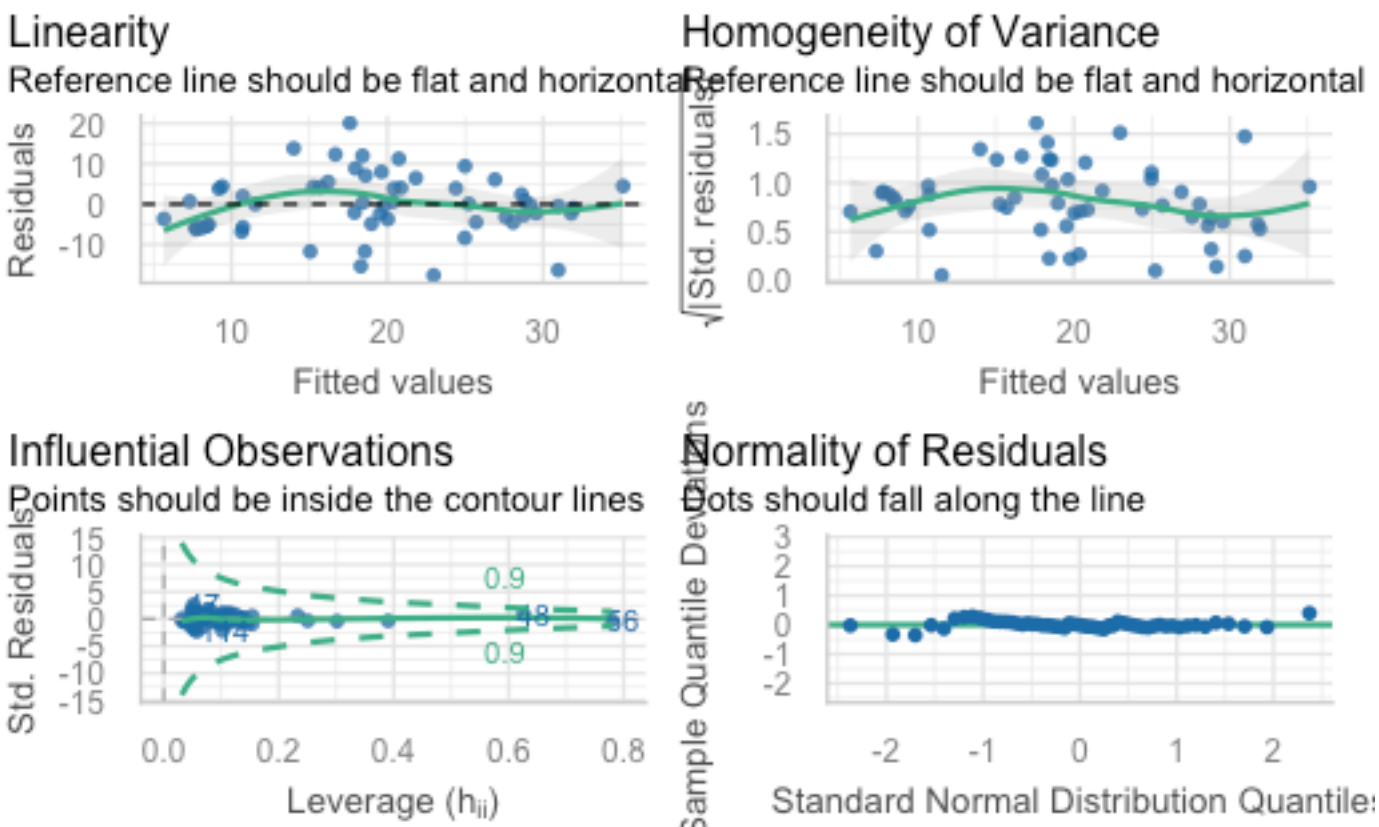
Checking assumptions - no transformation

```
loyn_fit ← lm(ABUND ~ YR.ISOL + GRAZE + ALT + AREA + LDIST + DIST, data = loyn)
summary(loyn_fit)
```

```

Call:
lm(formula = ABUND ~ YR.ISOL + GRAZE + ALT + AREA + LDIST + DIST,
    data = loyn)

Residuals:
     Min       1Q   Median       3Q      Max
-17.6638  -4.6409  -0.0883   4.2858  20.1042

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.097e+02  1.133e+02  -0.968  0.33791
YR.ISOL      6.693e-02  5.684e-02   1.177  0.24472
GRAZE       -3.447e+00  1.107e+00  -3.114  0.00308 **
ALT          4.772e-02  3.089e-02   1.545  0.12878
```

```
AREA         8.866e-04  4.657e-03   0.190  0.84980
LDIST        1.418e-03  1.310e-03   1.082  0.28451
DIST         3.811e-03  5.418e-03   0.703  0.48514
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.947 on 49 degrees of freedom
Multiple R-squared:  0.5118,    Adjusted R-squared:  0.452
F-statistic: 8.561 on 6 and 49 DF,  p-value: 2.24e-06
```

```r
performance::check_model(loyn_fit, check = c("linearity", "qq", "homogeneity", "outliers")) # check specific assumptions
```


Linearity
Reference line should be flat and horizontal
Residuals
20
10
0
-10
10
20
30
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
1.5
1.0
0.5
0.0
10
20
30
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
15
10
5
0
-5
-10
-15
0.9
0.9
0.0
0.2
0.4
0.6
0.8
Leverage (hii)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
3
2
1
0
-1
-2
-2
-1
0
1
2
Standard Normal Distribution Quantiles

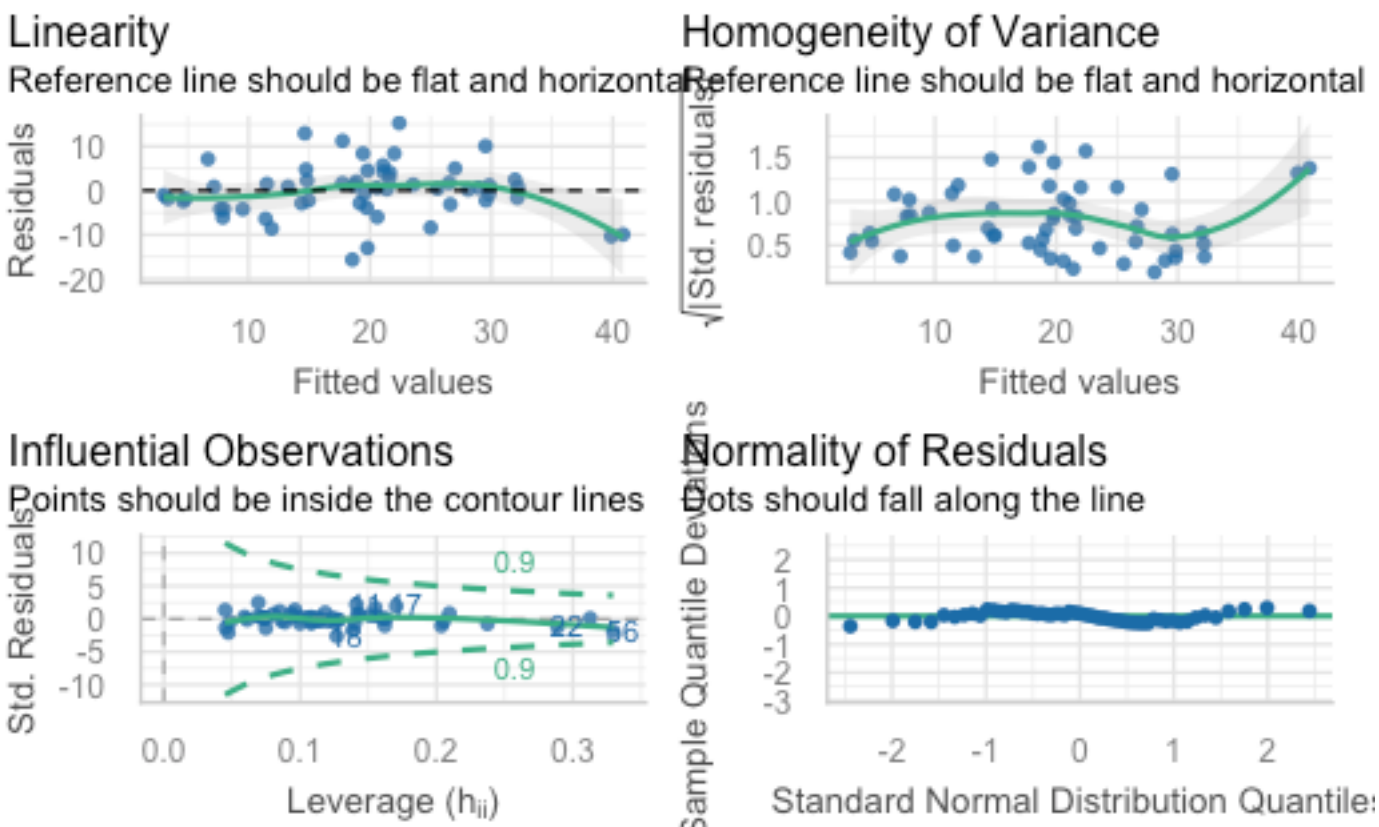
Checking assumptions - transformation

```
loyn_fit ← lm(ABUND ~ YR.ISOL + GRAZE + ALT + AREA_L10 + LDIST_L10 + DIST_L10, data = loyn)
summary(loyn_fit)
```

```

Call:
lm(formula = ABUND ~ YR.ISOL + GRAZE + ALT + AREA_L10 + LDIST_L10 +
    DIST_L10, data = loyn)

Residuals:
     Min       1Q   Median       3Q      Max
-15.6506  -2.9390   0.5289   2.5353  15.2842

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -125.69725   91.69228  -1.371   0.1767
YR.ISOL        0.07387    0.04520   1.634   0.1086
GRAZE         -1.66774    0.92993  -1.793   0.0791 .
ALT            0.01951    0.02396   0.814   0.4195
```

```
AREA_L10       7.47023    1.46489   5.099 5.49e-06 ***
LDIST_L10     -0.64842    2.12270  -0.305   0.7613
DIST_L10      -0.90696    2.67572  -0.339   0.7361
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.384 on 49 degrees of freedom
Multiple R-squared:  0.6849,    Adjusted R-squared:  0.6464
F-statistic: 17.75 on 6 and 49 DF,  p-value: 8.443e-11
```

```r
performance::check_model(loyn_fit, check = c("linearity", "qq", "homogeneity", "outliers")) # check specific assumptions
```


Linearity
Reference line should be flat and horizontal
Residuals
10
0
-10
-20
10
20
30
40
Fitted values
Homogeneity of Variance
Reference line should be flat and horizontal
√|Std. residuals|
1.5
1.0
0.5
10
20
30
40
Fitted values
Influential Observations
Points should be inside the contour lines
Std. Residuals
10
5
0
-5
-10
0.9
0.9
0.0
0.1
0.2
0.3
Leverage (hii)
Normality of Residuals
Dots should fall along the line
Sample Quantile Deviations
2
1
0
-1
-2
-3
-2
-1
0
1
2
Standard Normal Distribution Quantiles

Other Assumptions

- LINE... + leverage + collinearity

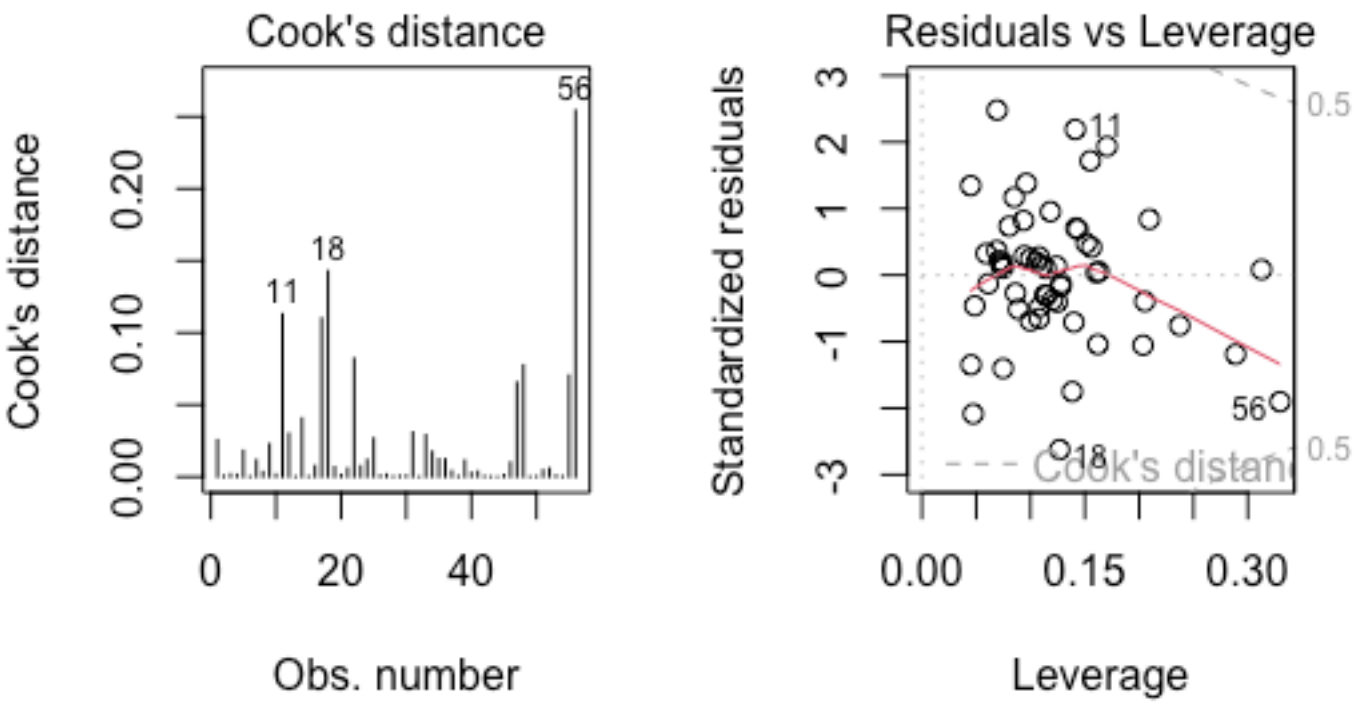
Leverage

- The leverage plot shows the influence of each observation (i.e. point) on the model.
- Points with high leverage can have a large effect on the model when removed.
- Identified by the Cook's distance statistic – named after the American statistician R. Dennis Cook, who introduced the concept in 1977.

> **Tip**
>
> The leverage plot is a useful tool for identifying outliers and influential points, but can also be used to check for other issues such as heteroskedasticity (equal variances) and non-linearity!

Reading the leverage plot

```
par(mfrow = c(1,2))
plot(loyn_fit, which = c(4,5))
```

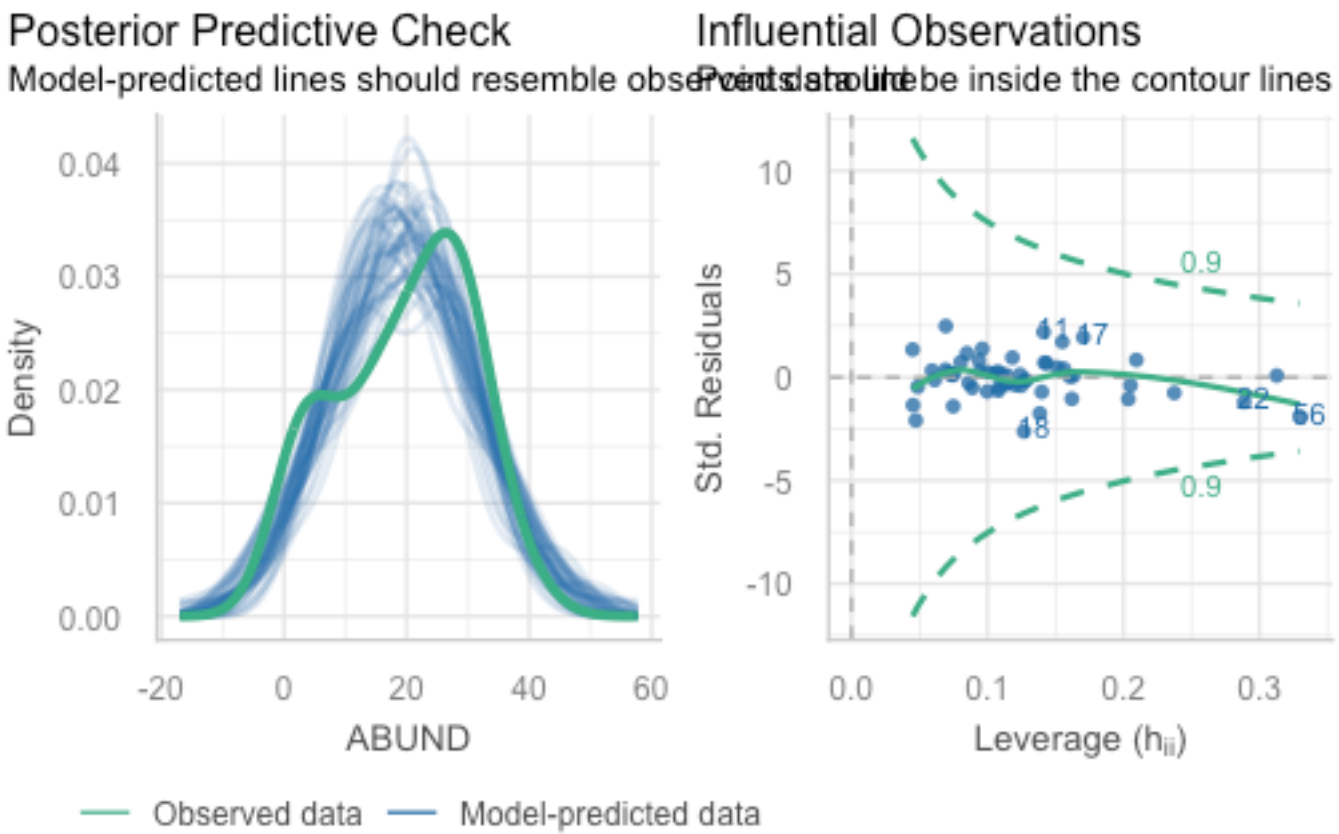


- Visually, points with Cook's distance > 0.5 are considered influential by default, but this is a somewhat arbitrary threshold.

- In practice, you should use a threshold that is appropriate for your data and model.

Outlier detection using `performance`

```
performance::check_model(loyn_fit, check = c("outliers", "pp_check"))
```



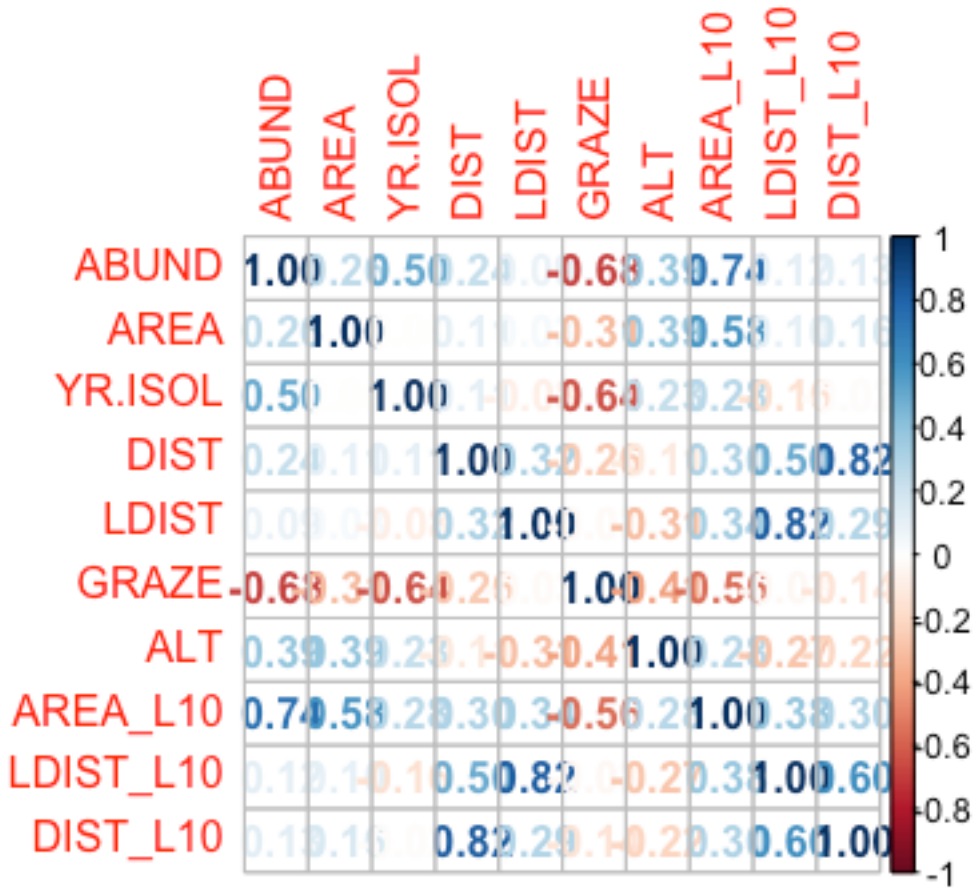

```
performance::check_outliers(loyn_fit)
```

```
OK: No outliers detected.
- Based on the following method and threshold: cook (0.919).
- For variable: (Whole model)
```

Collinearity

- Two predictors that have a *perfect* linear relationship (i.e. $r = 1$ or -1) breaks the assumption of collinearity
- Even strong correlations between predictors can lead to unstable estimates and large standard errors.
- Variance inflation factors (VIFs) are a measure of collinearity in the model.

```
corrplot::corrplot(cor(loyn), method = "number")
```

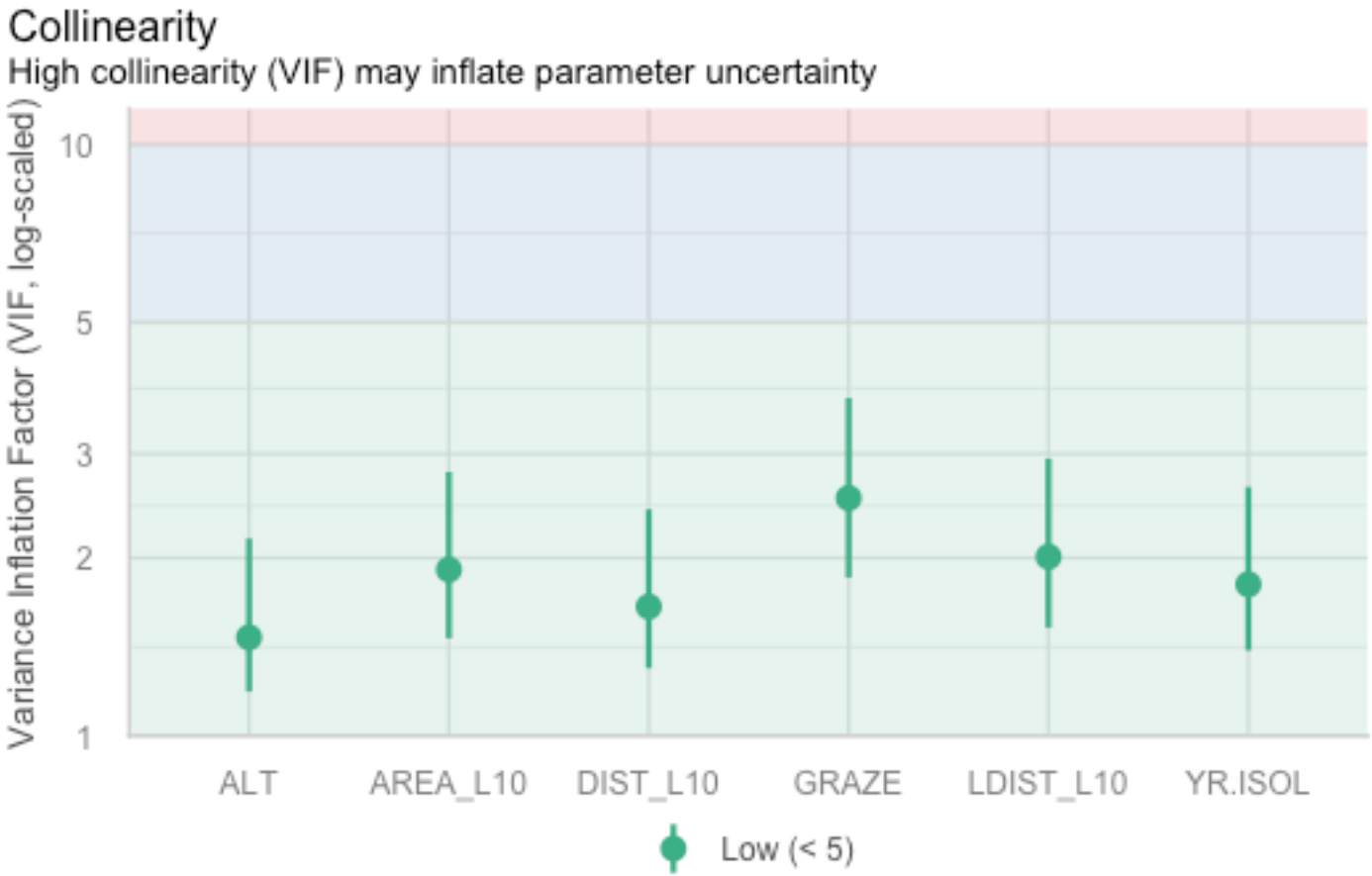

ABUND
AREA
YR.ISOL
DIST
LDIST
GRAZE
ALT
AREA_L10
LDIST_L10
DIST_L10
1.00
0.74
0.50
-0.63
1.00
0.58
0.50
-0.64
1.00
0.82
1.00
-0.64
1.00
-0.56
1.00
0.74
0.58
-0.56
1.00
0.60
0.82
0.60
1.00
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1

Calculating VIF

```
car::vif(loyn_fit) |> round(2) # numbers
```

```
  YR.ISOL     GRAZE       ALT  AREA_L10 LDIST_L10  DIST_L10
     1.80      2.52      1.47      1.91      2.01      1.65
```

```
plot(performance::check_collinearity(loyn_fit)) # visual
```



- 1 = no correlation with other predictors.

- > 10 is a sign for high, not tolerable correlation of model predictors (which need to be removed and the model refitted).

The best model?

If we remove the least significant variable...

```
full6 ← loyn_fit
part5 ← update(full6, . ~ . - LDIST_L10)
part4 ← update(part5, . ~ . - DIST_L10)
part3 ← update(part4, . ~ . - ALT)
part2 ← update(part3, . ~ . - YR.ISOL)
part1 ← update(part2, . ~ . - GRAZE)

formulas ← c(part1$call$formula,
             part2$call$formula,
             part3$call$formula,
             part4$call$formula,
             part5$call$formula,
             loyn_fit$call$formula)
formulas ←
  c("ABUND ~ AREA_L10",
    "ABUND ~ AREA_L10 + GRAZE",
    "ABUND ~ AREA_L10 + GRAZE + YR.ISOL",
```

```
    "ABUND ~ AREA_L10 + GRAZE + YR.ISOL + ALT",
    "ABUND ~ AREA_L10 + GRAZE + YR.ISOL + ALT + DIST_L10",
    "ABUND ~ AREA_L10 + GRAZE + YR.ISOL + ALT + DIST_L10 + LDIST_L10")

rs ← bind_rows(glance(part1),
          glance(part2),
          glance(part3),
          glance(part4),
          glance(part5),
          glance(full6)) %>%
        mutate(Model = formulas) %>%
        select(Model, r.squared, adj.r.squared)

knitr::kable(rs, digits = 2)
```

Model	r.squared	adj.r.squared
ABUND ~ AREA_L10	0.55	0.54
ABUND ~ AREA_L10 + GRAZE	0.65	0.64
ABUND ~ AREA_L10 + GRAZE + YR.ISOL	0.67	0.65
ABUND ~ AREA_L10 + GRAZE + YR.ISOL + ALT	0.68	0.66

Model	r.squared	adj.r.squared
ABUND ~ AREA_L10 + GRAZE + YR.ISOL + ALT + DIST_L10	0.68	0.65
ABUND ~ AREA_L10 + GRAZE + YR.ISOL + ALT + DIST_L10 + LDIST_L10	0.68	0.65

- R-squared increases with addition of predictors.
- Adj. R-squared *varies* with addition of predictors.

The problem

- Other combinations of predictors exist but are not shown.
- Need *automated way* to select the best model – 6 predictors gives us 2^6 = **64 models** to choose from!
- Options:
  - Backward elimination
  - Forward selection
  - Combination (R default)

Backward elimination

Steps for backward elimination

1. Start with full model.
2. For each predictor, test the effect of its removal on the model fit.
3. Remove the predictor that has the *least* effect on the model fit i.e. the **least informative** predictor, unless it is nonetheless supplying significant information about the response.
4. Repeat steps 2 and 3 until no predictors can be removed without significantly affecting the model fit.

In backward selection, the model fit is assessed using the Akaike Information Criterion (AIC) or the Bayesian Information Criterion (BIC). **Here we focus on the AIC.**

About AIC

- Most **popular** model selection criterion (can be used for non-nested models)
- Developed by Hirotsugu Akaike under the name of "an information criterion" (AIC)
- Founded on information theory which is concerned with the transmission, processing, utilization, and extraction of information.

$$AIC = 2k - 2\ln(L)$$

About AIC

$$AIC = 2k - 2\ln(L)$$

- k is the number of parameters in the model (predictors + intercept)
- L is the maximum value of the likelihood function
  - If we predict using the (current) model, what is the probability density of the prediction compared to the original distribution?
  - $ln(L)$ = goodness of fit (higher is better)
- For the number of parameters in the model $(2k)$ subtract the goodness of fit $2ln(L)$
  - **The smaller the AIC, the better the model fits the data.**
  - A *relative* measure and *unitless*, so it is not worth trying to interpret alone.
- Can be calculated with the `AIC()` function in R

About AIC - FYI

In linear regression, AIC is sometimes calculated as:

$$AIC = n \log\left(\frac{RSS}{n}\right) + 2k$$

where RSS is the residual sum of squares, $2k$ is the number of parameters in the model, and n is the number of observations.

- The difference between this equation and the previous equation is a constant, or scaling
- The `step()` function in R uses this equation (hence you may notice a difference in AIC values!)

Back to our example

```
back_step ← step(loyn_fit, direction = "backward")
```

```
Start:  AIC=214.14
ABUND ~ YR.ISOL + GRAZE + ALT + AREA_L10 + LDIST_L10 + DIST_L10

             Df Sum of Sq    RSS    AIC
- LDIST_L10  1      3.80 2000.7 212.25
- DIST_L10   1      4.68 2001.5 212.27
- ALT        1     27.02 2023.9 212.90
<none>                   1996.8 214.14
- YR.ISOL    1    108.83 2105.7 215.11
- GRAZE      1    131.07 2127.9 215.70
- AREA_L10   1   1059.75 3056.6 235.98

Step:  AIC=212.25
ABUND ~ YR.ISOL + GRAZE + ALT + AREA_L10 + DIST_L10

            Df Sum of Sq    RSS    AIC
```

```
- DIST_L10  1     12.64 2013.3 210.60
- ALT       1     35.12 2035.8 211.22
<none>                  2000.7 212.25
- YR.ISOL   1    121.64 2122.3 213.55
- GRAZE     1    132.44 2133.1 213.84
- AREA_L10  1   1193.04 3193.7 236.44

Step:  AIC=210.6
ABUND ~ YR.ISOL + GRAZE + ALT + AREA_L10

           Df Sum of Sq    RSS    AIC
- ALT       1     57.84 2071.1 210.19
<none>                  2013.3 210.60
- GRAZE     1    123.48 2136.8 211.94
- YR.ISOL   1    134.89 2148.2 212.23
- AREA_L10  1   1227.11 3240.4 235.25

Step:  AIC=210.19
ABUND ~ YR.ISOL + GRAZE + AREA_L10

           Df Sum of Sq    RSS    AIC
```

```
<none>                  2071.1 210.19
- YR.ISOL   1    129.81 2200.9 211.59
- GRAZE     1    188.45 2259.6 213.06
- AREA_L10  1   1262.97 3334.1 234.85
```

Printing `back_step` reveals the final model:

```
back_step
```

```

Call:
lm(formula = ABUND ~ YR.ISOL + GRAZE + AREA_L10, data = loyn)

Coefficients:
(Intercept)      YR.ISOL        GRAZE     AREA_L10
 -134.26065      0.07835     -1.90216      7.16617
```

Backward elimination: coefficients

Full model

```
sjPlot::tab_model(
  loyn_fit, back_step,
  show.ci = FALSE,
  show.aic = TRUE,
  dv.labels = c("Full model",
                "Reduced model")
)
```

	Full model		Reduced model	
Predictors	*Estimates*	*p*	*Estimates*	*p*
(Intercept)	−125.70	0.177	−134.26	0.126
YR ISOL	0.07	0.109	0.08	0.077
GRAZE	−1.67	0.079	−1.90	**0.034**
ALT	0.02	0.419		
AREA L10	7.47	**<0.001**	7.17	**<0.001**
LDIST L10	−0.65	0.761		

DIST L10	−0.91	0.736		
Observations	56		56	
R^2 / R^2 adjusted	0.685 / 0.646		0.673 / 0.654	
AIC	375.064		371.109	

The reduced model retains more explanatory power than the full model!

Summary

Model selection

Model development

1. Start with full model and check assumptions (e.g. normality, homoscedasticity, linearity, etc.).
2. Look for additional issues (e.g. multicollinearity, outliers, etc.) – correlations, leverage, VIF plots.
3. Consider transformations (e.g. log, sqrt, etc.).
4. Test assumptions again.

Model selection

5. Use VIF as an initial step to get rid of highly correlated predictors.
6. Perform variable selection using backward elimination (good and fast), because:
   - Using R^2 as a criterion is *not* recommended (it is not a good measure of model fit, only a good measure of variance explained).
   - Using partial F-test is good, but slow.

Next lecture

Next lecture: model training and prediction

- How to incorporate calibration and validation into your workflow
- Determining prediction intervals and performance metrics

Thanks!

Questions? Comments?

Slides made with Quarto