Week 5 Lab

Sarah Roberts

2023-02-01

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.3.2
```

```
library(interactions)
```

```
## Warning: package 'interactions' was built under R version 4.3.2
```

```
library(car)
```

```
## Warning: package 'car' was built under R version 4.3.2
```

```
## Loading required package: carData
```

```
## Warning: package 'carData' was built under R version 4.3.2
```

```
library(lmtest)
```

```
## Warning: package 'lmtest' was built under R version 4.3.2
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:car':
##
##     recode
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
library(GGally)
```

```
## Warning: package 'GGally' was built under R version 4.3.2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
library(gvlma)
library(rstatix)
```

```
## Warning: package 'rstatix' was built under R version 4.3.2
```

```
## Registered S3 methods overwritten by 'broom':
##   method            from
##   tidy.glht         jtools
##   tidy.summary.glht jtools
```

```
##
## Attaching package: 'rstatix'
```

```
## The following object is masked from 'package:stats':
##
##     filter
```

```
library(tidyverse)
```

```
## Warning: package 'readr' was built under R version 4.3.2
```

```
## Warning: package 'stringr' was built under R version 4.3.2
```

```
## Warning: package 'lubridate' was built under R version 4.3.2
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats   1.0.0     v readr     2.1.5
## v lubridate 1.9.3     v stringr   1.5.1
## v purrr     1.0.2     v tibble    3.2.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x rstatix::filter() masks dplyr::filter(), stats::filter()
## x dplyr::lag()      masks stats::lag()
## x dplyr::recode()   masks car::recode()
## x purrr::some()     masks car::some()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
theme_Publication <- function(base_size=10, base_family="Arial") {

  (theme_foundation(base_size=base_size, base_family=base_family)
   + theme(plot.title = element_text(hjust = 0.5),
           text = element_text(),
           panel.background = element_rect(colour = NA),
           plot.background = element_rect(colour = NA),
           panel.border = element_rect(colour = NA),
           axis.title = element_text(size = rel(1)),
           axis.title.y = element_text(angle=90,vjust =2),
           axis.text = element_text(),
           axis.line = element_line(colour="black"),
           axis.ticks = element_line(),
           panel.grid.major = element_line(colour="#f0f0f0"),
           panel.grid.minor = element_blank(),
           legend.key = element_rect(colour = NA),
           legend.position = "right",
```

```
            legend.title = element_text(face="italic"),
            strip.background=element_rect(colour="#f0f0f0",fill="#f0f0f0")
    ))

}
```

Class notes

- Multicollinearity: Adding more independent variables creates more relationships among them – when independent variables are correlated with eachother, they aren't adding much information
- Partial regression coefficients
- VIF: multicollinearity => higher VIF, higher multicollinearity; >10 is too much; high VIF is from 5-10
- Model goals:
- Better predict response variable
- Figure out what influences
- Best model understanding var: AIC
- Best model predicting var: Cross validation

Read in the data

```
antdat <- read.csv("ant_density.csv")
```

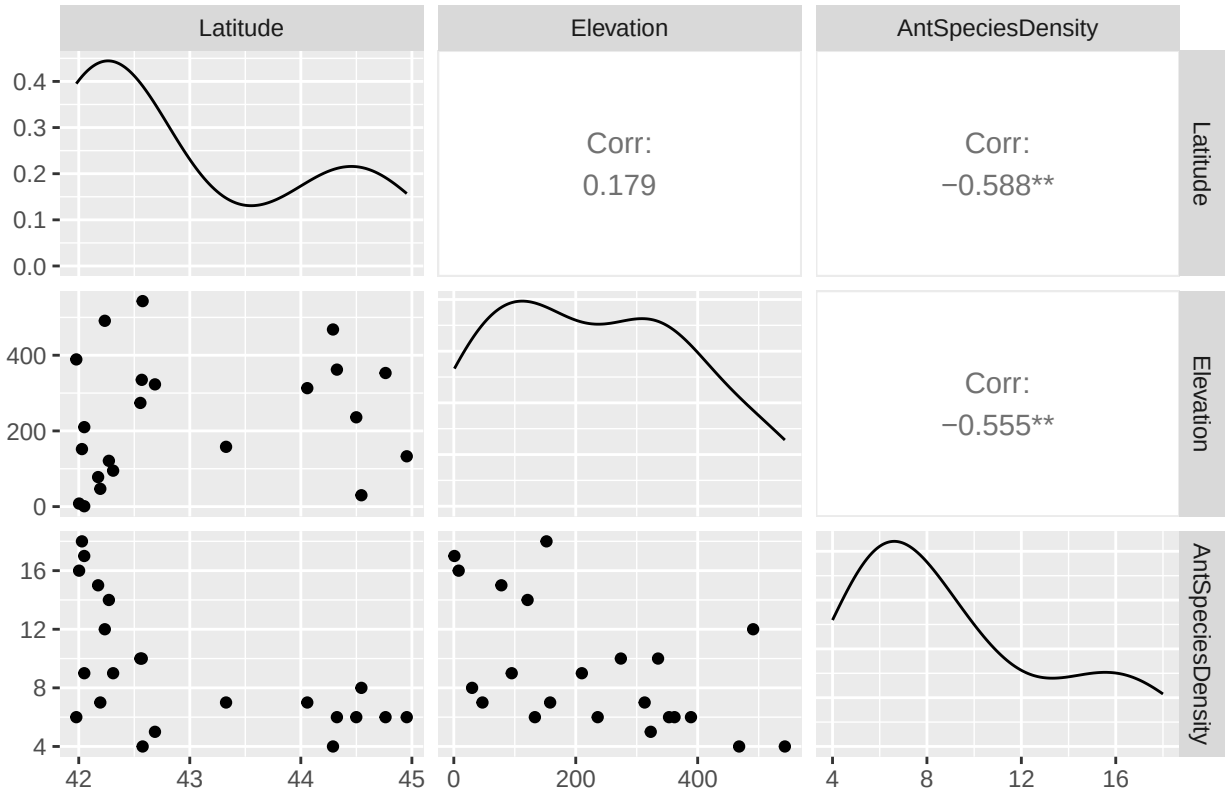
As always, I start with looking at the data visually. We can do a scatterplot matrix in R. http://www.sthda.com/english/wiki/ggally-r-package-extension-to-ggplot2-for-correlation-matrix-and-survival-plots-r-software-and-data-visualization. To do so, we will use a new package, GGally.

Let's make a scatterplot matrix. Let's first look at few variables at once.

Through the pairs plot, we can get general feeling for:

1. Any collinearity between predictors
2. Predictors that have a weak relationship and may be phased out in variable selection
3. The general strength of the relationship between the response and the predictors

```
ggpairs(antdat, columns = 1:3, title = "",
  axisLabels = "show")
```



Assumptions ## Checking for Multicollinearity Based on our analysis of the pairs plot, you probably now have some idea of what multicollinearity in a linear model is (but not why it is bad). More specifically, multicollinearity is a linear or near linear dependence between the predictors in the model. A linear dependence means that one variable is a linear combination of another (they are collinear), and are therefore not giving us any new information. Strong correlation between two variables indicates collinearity. Strong correlation between multiple predictors in our MLR model means that multicollinearity may be present.

Let's go ahead and build an initial model. Similar to simple linear regression, we use the lm() function. We place the response variable (Y) first followed by a ~. After the ~ we place the explanatory variables seperated by a +. We can deal with missing values by the argument na.action=na.exclude. We then use the summary() function to see the model results.

```
lm1 <- lm(AntSpeciesDensity ~ Latitude + Elevation, data = antdat)
summary(lm1)
```

```
##
## Call:
## lm(formula = AntSpeciesDensity ~ Latitude + Elevation, data = antdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.1180 -2.3759  0.3218  1.9070  5.8369
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 98.49651   26.50701   3.716  0.00147 **
## Latitude    -2.00981    0.61956  -3.244  0.00427 **
## Elevation   -0.01226    0.00411  -2.983  0.00765 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.022 on 19 degrees of freedom
## Multiple R-squared:  0.5543, Adjusted R-squared:  0.5074
## F-statistic: 11.82 on 2 and 19 DF,  p-value: 0.000463
```

The major symptoms of multicollinearity are:

1. Estimated coefficients which have the wrong sign (for example, if the sign on Elevation was positive instead of negative this would be counter-intuitive from our pairs plot)
2. Large (inflated) standard errors and variance for the estimated coefficients. This in turn causes a lack of significance for the predictor when it seems like it should be significant

If the predictors are highly correlated with each other, it becomes more difficult to determine what variance is explained by the response. As a result, the standard error of the coefficient becomes large.

Luckily, we don't see these issues present in our model, so multicollinearity is likely not an issue. To make sure, we can use a measure called the VIF (Variance Inflation Factor)

```
vif(lm1)
```

```
##  Latitude Elevation
##  1.033004  1.033004
```

The VIF command shows that multicollinearity is not an issue in our model. Note that (as a general rule of thumb) a VIF of 10 or greater is a cause for concern.

Just to illustrate some of the points I made, and to show you how multicollinearity can occur, I will add the predictor "Elevation_2". The idea here is that the elevation was measured in two ways

```
# Adding the secondary measurement for elevation
antdat$Elevation_2 <- antdat$Elevation + rnorm(nrow(antdat), sd=3)
# Construct the new model
lm2 <- lm(AntSpeciesDensity ~ ., data = antdat) # this is a trick to run a regression on all of the var
summary(lm2)
```

```
##
## Call:
## lm(formula = AntSpeciesDensity ~ ., data = antdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.4417 -1.9953  0.5612  1.4294  4.8218
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 101.0339    25.3495   3.986 0.000867 ***
## Latitude     -2.0563     0.5921  -3.473 0.002715 **
## Elevation     0.3558     0.2181   1.631 0.120227
## Elevation_2  -0.3702     0.2194  -1.688 0.108733
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.885 on 18 degrees of freedom
## Multiple R-squared:  0.6152, Adjusted R-squared:  0.5511
## F-statistic: 9.593 on 3 and 18 DF,  p-value: 0.0005267
```

```
vif(lm2)
```

```
##    Latitude   Elevation Elevation_2
##    1.035245 3193.272520 3192.285767
```

We can now see what impact this has on our model. The coefficient on Elevation is positive when it should be negative. It also no longer has a significant linear relationship. The VIF is also massive. This problem common to many data sets where one or more variables are essentially measuring the same object. The simple solution is to remove the redundant variables.

Other Assumptions

Five important assumptions need to hold so that an OLS regression model can be useful for hypothesis testing, confidence intervals, and predication. These are:

1. The relationship between the response y and the regression is linear (at least approximately).

Why? This one is fairly obvious. If we are trying to explain some data using a linear model, then in order for it to be effective, there needs to be a linear relation present. This is required for prediction under a MLR model.

2. The error term has zero mean.

Why? It should be equally probable that the errors fall above and below the regression line.

3. The error term has constant variance ^{2}.

Why? This is also known as homoscedasticity or homogeneity of variance. When there is unequal variance for different levels of the predictors, the parameter estimates and standard errors will not be consistent (biased) across different samples.

4. The errors are uncorrelated.

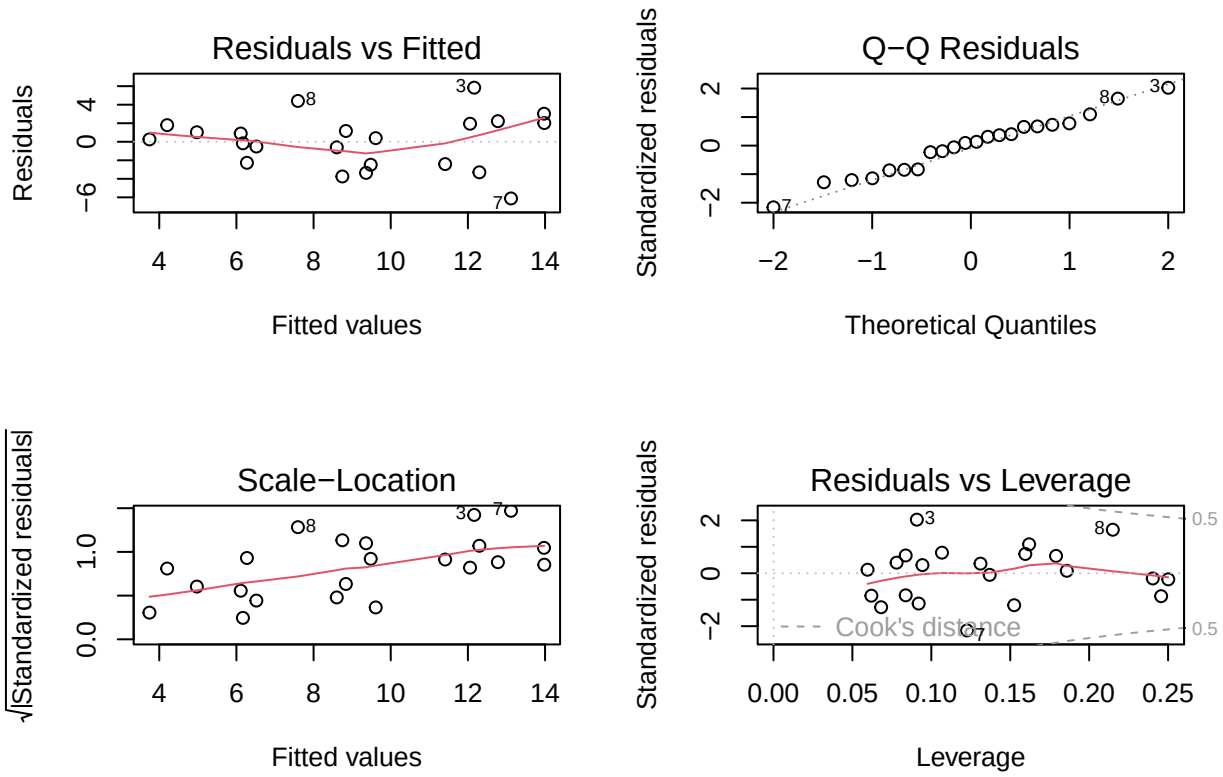
Why? The computation of standard errors relies on statistical independence.

5. The errors are normally distributed.

Why? Confidence intervals, hypothesis testing, and the estimated parameters are all computed under the assumption of a normal distribution.

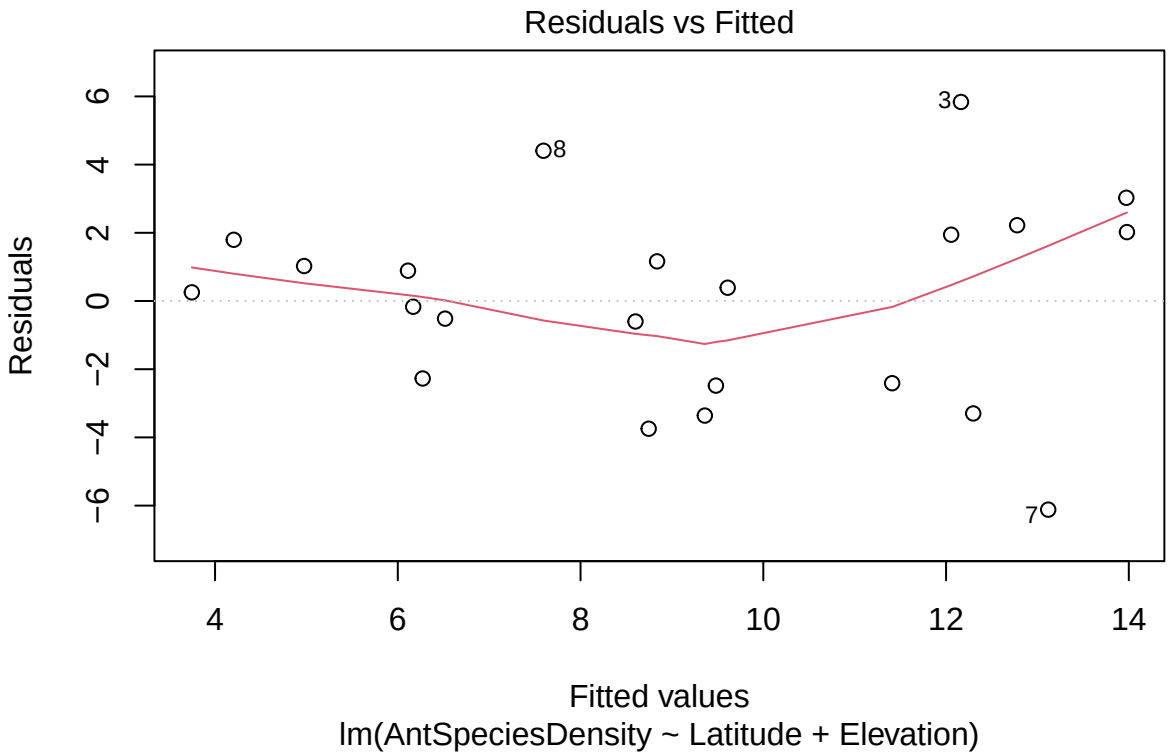
Assumptions 2 through 5 can be summarized as "The errors are i.i.d. normally distributed, with zero mean and constant variance". Note that 4 and 5 are required for hypothesis testing and interval estimation. Probably the best way to check if these assumptions hold is to plot the standardized residuals and compare them to useful elements of you model.Overall, I like to think of this stage as tuning the model to satisfy these assumptions.

```
par(mfrow=c(2,2))
plot(lm1)
```



The relationship between the response y and the regression is linear:

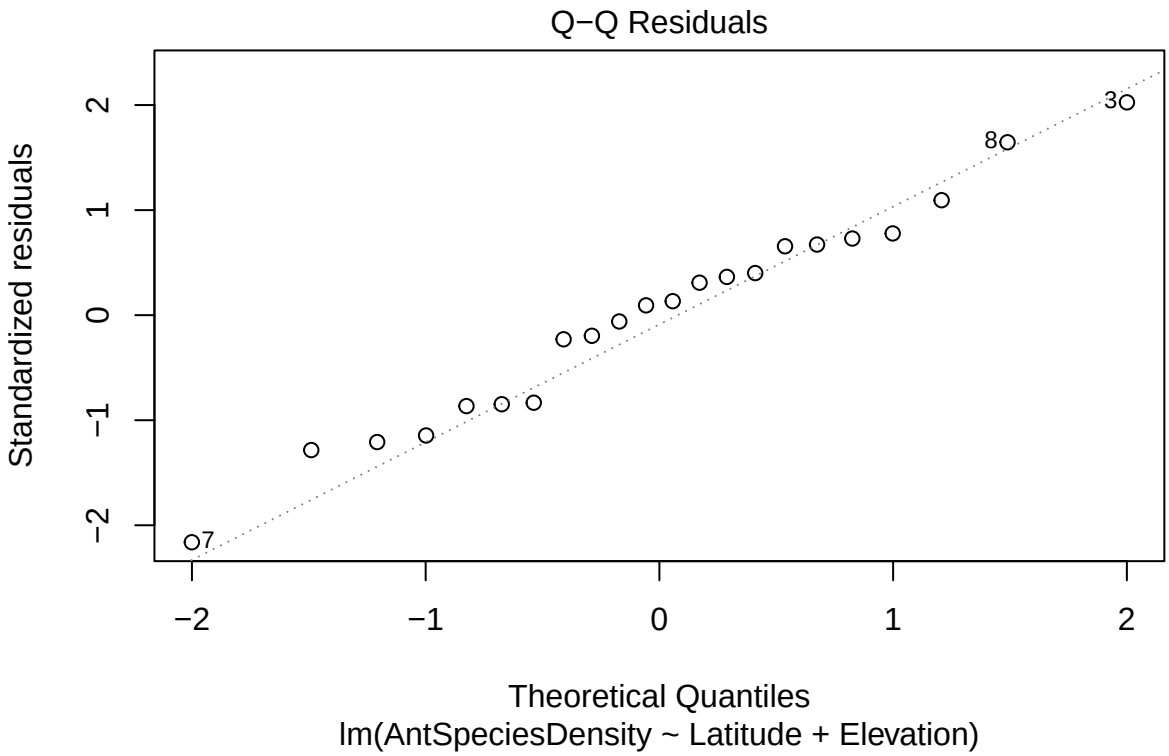
```
plot(lm1, which = 1)
```



You can think of the Residuals vs Fitted plot as the 2-d representation of our fitted model . A relatively straight line through the fitted values is what we are looking for here. If a proper fit is not possible through any kind of transformation, then a linear model is not appropriate for the given data set.

The errors are normally distributed

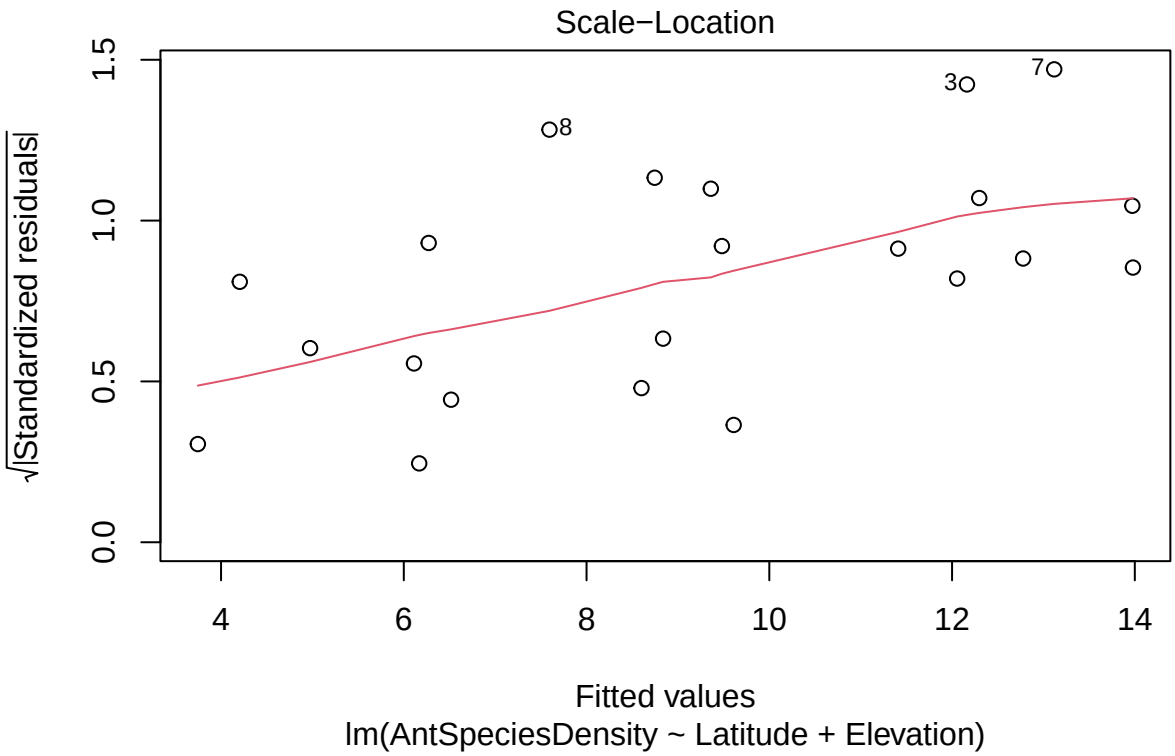
```
plot(lm1, which = 2)
```



The Normal Q-Q plot compares the standardized residuals of the model with a standard normal distribution (the diagonal line). Hence, any deviation from the diagonal means that the residuals are deviating from the standard normal distribution. Even though the residuals do not line up perfectly in this case (i.e. doesn't exhibit perfect symmetry), the current state is more than acceptable. As long as the majority of the points lie along the diagonal, this should be fine.

The error term has constant variance ^{2}

```
plot(lm1, which = 3)
```



The Scale-Location is similar to the Residuals vs Fitted plot, except we are looking at the magnitude of the residuals. Again, we are looking for a straight line through the center to show that the average of the residuals does not change much. We can also roughly estimate the upper and lower bound of the data points to determine if the overall variability changes.

The error term has zero mean

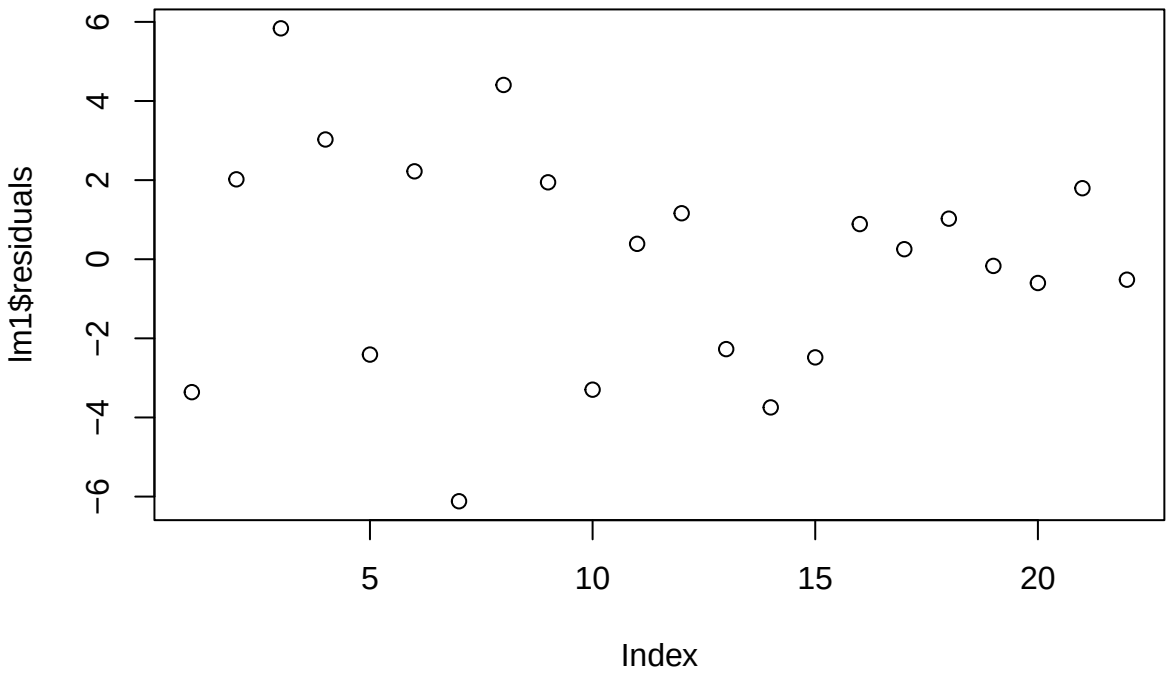
```
mean(resid(lm1))
```

```
## [1] 7.571674e-17
```

This should be evident through most of the plots. In particular, the Residuals vs Index plot illustrates this well. Calculating the mean of the residuals shows that it is practically zero.

The errors are uncorrelated

```
plot(lm1$residuals)
```



The Residual vs. Index plot shows the observations index on the x-axis and its residual on the y-axis. We want a random scattering of residuals around =0 (i.e. no correlation of the errors).

##hypothesis tests for assumptions Going over these plots can be a little bit exhausting, but I feel like its important for people new to MLR or regression in general to analyze these plots and get the intuition. Once you have done this enough times, I feel like it is fine to validate the model assumptions using the hypothesis tests available in R commands ("car" package) like ncvTest (tests homoscedasticity) or durbinWatsonTest (tests correlation between the errors). gvlma is a command which will test all the model assumptions in one go. A word of caution though, even though these commands give a yes or no answer, they are still subjective and open to interpretation. For example, if you choose an =0.05, and the p-value ends up being 0.04, you should still investigate and correct this issue if possible. You should also not "cherry-pick" and change your after the fact.

```
gvlma(lm1)
```

```
##
## Call:
## lm(formula = AntSpeciesDensity ~ Latitude + Elevation, data = antdat)
##
## Coefficients:
## (Intercept)     Latitude    Elevation
##    98.49651     -2.00981     -0.01226
##
##
## ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
## USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
## Level of Significance =  0.05
```

```
##
## Call:
##  gvlma(x = lm1)
##
##                       Value p-value                   Decision
## Global Stat        6.47787 0.16619    Assumptions acceptable.
## Skewness           0.03681 0.84785    Assumptions acceptable.
## Kurtosis           0.10885 0.74146    Assumptions acceptable.
## Link Function      1.48976 0.22225    Assumptions acceptable.
## Heteroscedasticity 4.84245 0.02777 Assumptions NOT satisfied!
```

Global Stat checks whether the relationship between the dependent and independent relationship roughly linear. We can see that the assumption is met. Skewness and kurtosis assumptions show that the distribution of the residuals are normal. Link function checks to see if the dependent variable is continuous or categorical. Our variable is continuous. Heteroskedasticity assumption means the error variance is equally random and we have homoskedasticity!

To me this is a little bit too "black-bocky". Here is their paper on what test they used for each: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2820257/

Let's at least use some tests that we know from before

linearity:

```
cor(antdat$Latitude, log(antdat$AntSpeciesDensity))
```

```
## [1] -0.5777397
```

```
cor(antdat$Elevation, log(antdat$AntSpeciesDensity))
```

```
## [1] -0.5942384
```

These correlations tell me that there is a linear relationship between these variables

normality of residuals

```
resid(lm1) %>% shapiro_test()
```

```
## # A tibble: 1 x 3
##   variable statistic p.value
##   <chr>        <dbl>   <dbl>
## 1 .            0.983   0.956
```

homogeneity of variances

```
bptest(lm1)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm1
## BP = 5.9769, df = 2, p-value = 0.05037
```

No high leverage points Cooks distance is a function of both leverage and standardized residuals.

```
cooks.distance(lm1) > 3*mean(cooks.distance(lm1))
```

```
##     1     2     3     4     5     6     7     8     9    10    11    12    13
## FALSE FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
##    14    15    16    17    18    19    20    21    22
## FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
```

it looks like there are a few influential points here that may be influencing the results of our model. We can run the regression with and without these points and report both results.

Also, from these tests and the above plots, our homoscedasticity assumption may be violated (Bp test says we can barely fail to reject the null, and gvlma tells us there is heteroskedasticity present). We might try taking the log of species density

```
lm3 <- lm(log(AntSpeciesDensity) ~ Latitude + Elevation, data = antdat )
bptest(lm3)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  lm3
## BP = 5.9909, df = 2, p-value = 0.05001
```

```
gvlma(lm3)
```

```
##
## Call:
## lm(formula = log(AntSpeciesDensity) ~ Latitude + Elevation, data = antdat)
##
## Coefficients:
## (Intercept)     Latitude    Elevation
##   11.235491    -0.204313    -0.001411
##
##
## ASSESSMENT OF THE LINEAR MODEL ASSUMPTIONS
## USING THE GLOBAL TEST ON 4 DEGREES-OF-FREEDOM:
## Level of Significance =  0.05
##
## Call:
##  gvlma(x = lm3)
##
##                     Value p-value                Decision
## Global Stat        2.5496  0.6358 Assumptions acceptable.
## Skewness           0.1208  0.7282 Assumptions acceptable.
## Kurtosis           0.2086  0.6478 Assumptions acceptable.
## Link Function      0.1477  0.7007 Assumptions acceptable.
## Heteroscedasticity 2.0725  0.1500 Assumptions acceptable.
```

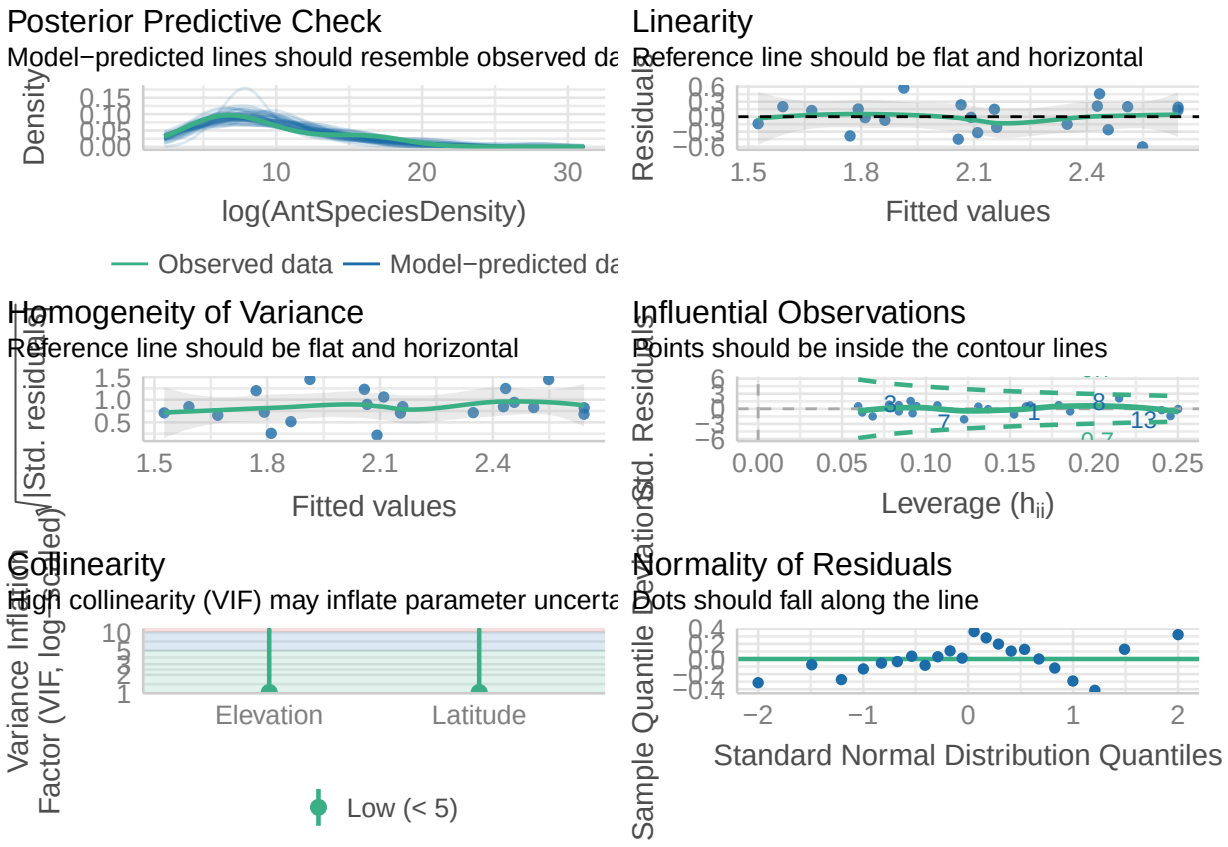
It is hard to tell if this makes it better (as the Bp test and gvlma give conflicting results), we will stick with the logged data for now (so I can show you how to interpret it, and because our residuals vs. fitted looks pretty flat now)

If you want to get fancy, you can use the check_model function in the performance package

```
library(performance)
```

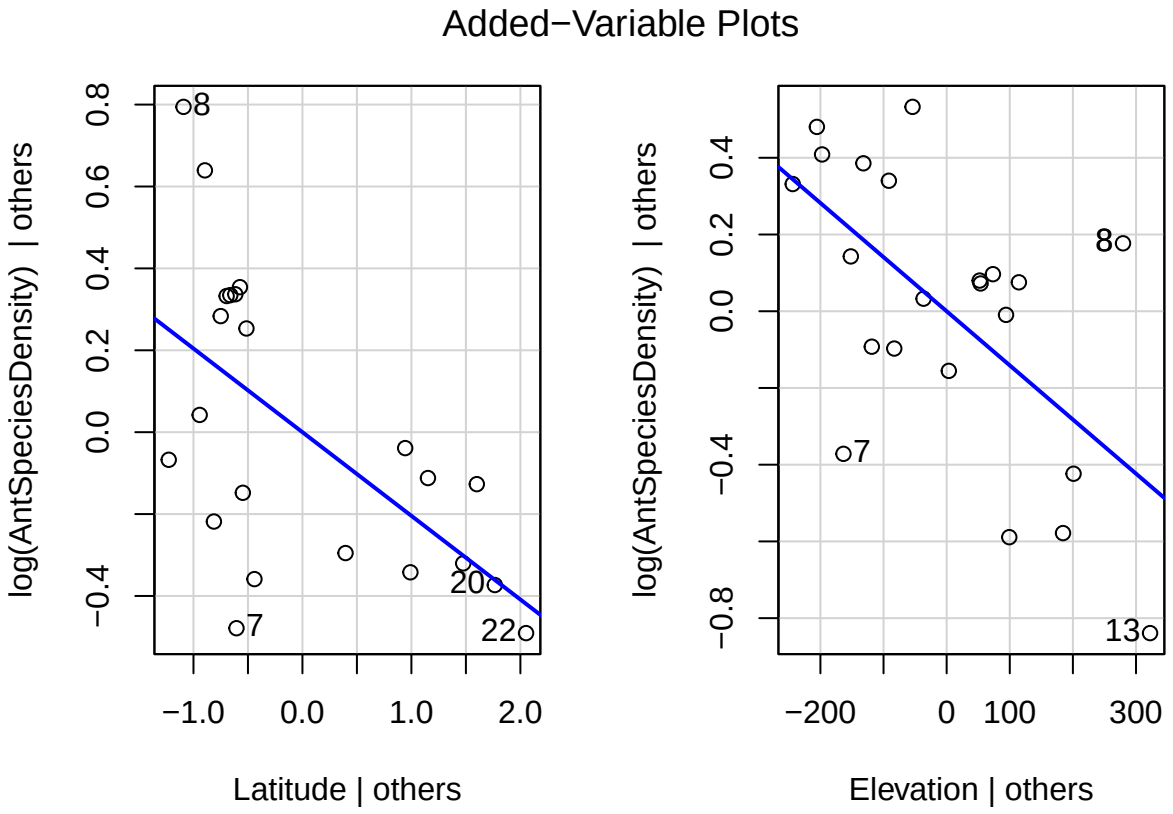
```
## Warning: package 'performance' was built under R version 4.3.2
```

```
check_model(lm3)
```



Now that we have checked our assumptions, we can plot the added variable plots to look at the effect of Latitude and Elevation on logged Ant Species Density. These type of plots allow us to observe the relationship between each individual predictor variable and the response variable in a model while holding other predictor variables constant.

```
avPlots(lm3)
```



The x-axis displays a single predictor variable and the y-axis displays the response variable. The blue line shows the association between the predictor variable and the response variable, while holding the value of all other predictor variables constant. The points that are labelled in each plot represent the two observations with the largest residuals and the two observations with the largest partial leverage. Note that the angle of the line in each plot matches the sign of the coefficient from the estimated regression equation.

#Interpretation we can also interpret the coefficients from our above linear model

```
summary(lm3)
```

```
##
## Call:
## lm(formula = log(AntSpeciesDensity) ~ Latitude + Elevation, data = antdat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.60243 -0.19846  0.05538  0.19715  0.57141
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.2354914  2.7028400   4.157 0.000535 ***
## Latitude    -0.2043130  0.0631747  -3.234 0.004367 **
## Elevation   -0.0014112  0.0004191  -3.367 0.003234 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3081 on 19 degrees of freedom
## Multiple R-squared:  0.5828, Adjusted R-squared:  0.5389
```

```
## F-statistic: 13.27 on 2 and 19 DF,  p-value: 0.0002474
```

Remember our interpretation for a logged dependent variable - Exponentiate the coefficient, subtract one from this number, and multiply by 100. This gives the percent increase (or decrease) in the response for every one-unit increase in the independent variable.

Here is how I would report these results.

Multiple linear regression was used to predict ant species density from latitude and elevation. The fitted regression model (Density ~ Latitude + Elevation) was statistically significant (Adjusted R^2 = .54, F = 13.27(19), p = .0002). For every one degree increase in latitude, ant species diversity decreases by 18.4% holding elevation constant (p = .004, df = 19). For everyone one unit increase in elevation, ant species diversity decreases by .14% holding latitude constant (p = .003, df = 19). The response variable was logged due to potential violations of homoscedasticity (Breusch=Pagan test p < .05 however gvlma heteroskedasticity test p <.05). Using logged Ant diversity, there was no evidence of violations of linearity (cor = -.57 (Latitude), -.59(Elevation)), homoscedasticity (Breusch=Pagan test, p = 0.05001, gvlma heteroskedasticity test p >.05), normality of residuals (shapiro test, p = 0.73), leverage points (cooks distance), or multicollinearity of predictors (VIF = 1.033).

check out this handy trick to report your results: but make sure that you are able to interpret the logged coefficients correctly

```
library(report)
```

```
## Warning: package 'report' was built under R version 4.3.2
```

```
report(lm3)
```

```
## Formula contains log- or sqrt-terms.
##   See help("standardize") for how such terms are standardized.
## Formula contains log- or sqrt-terms.
##   See help("standardize") for how such terms are standardized.
```

```
## We fitted a linear model (estimated using OLS) to predict AntSpeciesDensity
## with Latitude and Elevation (formula: log(AntSpeciesDensity) ~ Latitude +
## Elevation). The model explains a statistically significant and substantial
## proportion of variance (R2 = 0.58, F(2, 19) = 13.27, p < .001, adj. R2 = 0.54).
## The model's intercept, corresponding to Latitude = 0 and Elevation = 0, is at
## 11.24 (95% CI [5.58, 16.89], t(19) = 4.16, p < .001). Within this model:
##
##   - The effect of Latitude is statistically significant and negative (beta =
## -0.20, 95% CI [-0.34, -0.07], t(19) = -3.23, p = 0.004; Std. beta = -0.21, 95%
## CI [-0.35, -0.08])
##   - The effect of Elevation is statistically significant and negative (beta =
## -1.41e-03, 95% CI [-2.29e-03, -5.34e-04], t(19) = -3.37, p = 0.003; Std. beta =
## -0.22, 95% CI [-0.36, -0.08])
##
## Standardized parameters were obtained by fitting the model on a standardized
## version of the dataset. 95% Confidence Intervals (CIs) and p-values were
## computed using a Wald t-distribution approximation.
```

##plot your results In addition to your added variable plots above, you can plot all of the coefficients in one go as well

```
library(ggstance)
```

```
## Warning: package 'ggstance' was built under R version 4.3.2
```

```
##
## Attaching package: 'ggstance'
```

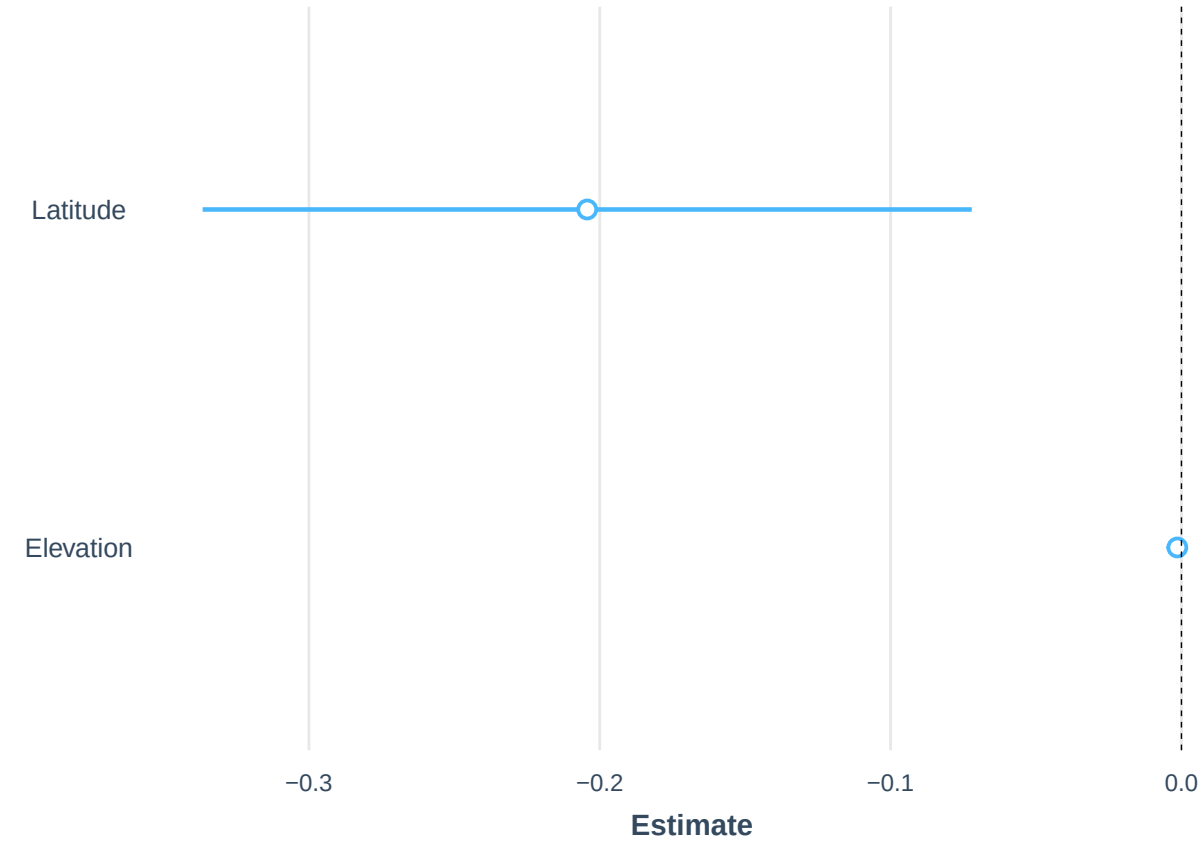
```
## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh
```

```
library(jtools)
```

```
## Warning: package 'jtools' was built under R version 4.3.2
```

```
plot_coefs(lm3)
```

```
## Loading required namespace: broom.mixed
```



Cool! when the solid line does not cross the vertical dashed line, the estimates is significantly different from 0 at the 5% significance level (i.e., p-value < 0.05) furthermore, a point to the right (left) of the vertical dashed line means that there is a positive (negative) relationship between the two variables the more extreme the point, the stronger the relationship!

#Prediction Recall, we can make predictions just like we did in the simple linear regression

```
pred <- predict(lm3, list(Latitude = 41, Elevation = 50), interval = "prediction")
pred
```

```
##        fit      lwr     upr
## 1 2.788099 2.069408 3.50679
```

```
L <- 41
E <- 50
lm3$coefficients[1] + lm3$coefficients[2]*L +
    lm3$coefficients[3]*E
```

```
## (Intercept)
```

```
##    2.788099
```

Choosing the "best" model

##backward selection Lets pretend that we have a few more variables in our dataset, note - these are fake variables I am adding to a real ecological dataset. I am doing this to show you that sometimes selection procedures are not the best way to go as they can leave in independent variables that are not actually important for predicting our response variable. But, alas it is fun to show you this anyway.

lets just add some random variables in there to show you that stepAIC isn't always the best

```
antdat$sarah <- rep(c("sarah", "roberts"), each= 11)

antdat$randomx <- rnorm(22, mean = 22)
```

Now pretend we want to see which combination of these variables is best for predicting ant density. We will use AIC, which selects models based on minimizing Residual Sum of Squares and penalizes added parameters.

In backward selection, we start with the maximal model - lets ignore interactions for now, we will talk about that a little further down.

```
#lets add in the logged data for Density
antdat$log_density <- log(antdat$AntSpeciesDensity)
full_mod <- lm(log_density ~ Elevation + Latitude + sarah + randomx, data = antdat)
mod_back_aic = step(full_mod, direction = "backward")
```

```
## Start:  AIC=-50.25
## log_density ~ Elevation + Latitude + sarah + randomx
##
##             Df Sum of Sq    RSS     AIC
## - sarah      1   0.04130 1.4637 -51.621
## - Latitude   1   0.10561 1.5280 -50.675
## <none>                   1.4224 -50.251
## - randomx    1   0.16942 1.5919 -49.775
## - Elevation  1   0.68430 2.1067 -43.610
##
## Step:  AIC=-51.62
## log_density ~ Elevation + Latitude + randomx
##
##             Df Sum of Sq    RSS     AIC
## <none>                   1.4637 -51.621
## - randomx    1   0.34005 1.8038 -49.025
## - Latitude   1   0.77964 2.2434 -44.227
## - Elevation  1   1.16931 2.6330 -40.704
```

R will then repeatedly attempt to delete a predictor until it stops, or reaches the model density ~ 1, which contains no predictors.

At each "step," R reports the current model, its AIC, and the possible steps with their RSS and more importantly, AIC.

Returning to the first step, R then gives us a row which shows the effect of deleting each of the current predictors. The - signs at the beginning of each row indicate we are considering removing a predictor. There is also a row with which is a row for keeping the current model. Notice that this row has the smallest RSS, as it is the largest model. We see that every row above has a smaller AIC than the row for with the one at the top, randomx, giving the lowest AIC. Thus we remove randomx from the model, and continue the process.

We continue the process until we reach the model log_density ~ Elevation + sarah. At this step, the row for tops the list, as removing any additional variable will not improve the AIC. This is the model which is stored in mod_back_aic

```
coef(mod_back_aic)
```

```
##  (Intercept)    Elevation     Latitude      randomx
##  7.941406208 -0.001475546 -0.183708723  0.110638462
```

##forward selection Forward selection is the exact opposite of backwards selection. Here we tell R to start with a model using no predictors, that is log_density ~ 1, then at each step R will attempt to add a predictor until it finds a good model or reaches the maximal model. Again, by default R uses AIC as its quality metric when using the step() function. Also note that now the rows begin with a + which indicates addition of predictors to the current model from any step. scope tells us where to stop.

```
mod_start = lm(log_density ~ 1, data = antdat)
mod_forw_aic = step(
  mod_start,
  scope = log_density ~ Elevation + Latitude + sarah + randomx,
  direction = "forward")
```

```
## Start:  AIC=-33.79
## log_density ~ 1
##
##             Df Sum of Sq    RSS     AIC
## + sarah      1   2.18924 2.1342 -47.325
## + Elevation  1   1.52670 2.7968 -41.377
## + Latitude   1   1.44310 2.8804 -40.729
## + randomx    1   0.46339 3.8601 -34.288
## <none>                   4.3235 -33.794
##
## Step:  AIC=-47.32
## log_density ~ sarah
##
##             Df Sum of Sq    RSS     AIC
## + Elevation  1   0.50058 1.6336 -51.205
## <none>                   2.1342 -47.325
## + randomx    1   0.02455 2.1097 -45.579
## + Latitude   1   0.00021 2.1340 -45.327
##
## Step:  AIC=-51.21
## log_density ~ sarah + Elevation
##
##            Df Sum of Sq    RSS     AIC
## <none>                  1.6336 -51.205
## + randomx   1  0.105590 1.5280 -50.675
## + Latitude  1  0.041784 1.5919 -49.775
```

this lands on the same model of Elevation + sarah

##bidirectional selection Also called stepwise selection We will either add a new variable to our model or delete an existing variable from our model at each stage of this new method, taking the action that results in the greatest reduction in AIC.

We eventually come to a halt when there is no variable that can be added or removed to reduce AIC.

```
mod_start = lm(log_density ~ 1, data = antdat)
mod_forw_aic = step(
```

```
  mod_start,
  scope = log_density ~ Elevation + Latitude + sarah + randomx,
  direction = "both")
```

```
## Start:  AIC=-33.79
## log_density ~ 1
##
##             Df Sum of Sq    RSS     AIC
## + sarah      1   2.18924 2.1342 -47.325
## + Elevation  1   1.52670 2.7968 -41.377
## + Latitude   1   1.44310 2.8804 -40.729
## + randomx    1   0.46339 3.8601 -34.288
## <none>                   4.3235 -33.794
##
## Step:  AIC=-47.32
## log_density ~ sarah
##
##             Df Sum of Sq    RSS     AIC
## + Elevation  1   0.50058 1.6336 -51.205
## <none>                   2.1342 -47.325
## + randomx    1   0.02455 2.1097 -45.579
## + Latitude   1   0.00021 2.1340 -45.327
## - sarah      1   2.18924 4.3235 -33.794
##
## Step:  AIC=-51.21
## log_density ~ sarah + Elevation
##
##             Df Sum of Sq    RSS     AIC
## <none>                   1.6336 -51.205
## + randomx    1   0.10559 1.5280 -50.675
## + Latitude   1   0.04178 1.5919 -49.775
## - Elevation  1   0.50058 2.1342 -47.325
## - sarah      1   1.16312 2.7968 -41.377
```

What is wrong with this picture? sarah is a completely random variable that probably doesn't actually influence species density. Also, we know from our linear model that Latitude is probably important for species density. Lets now look at cross validation to examine which of these variables predicts ant density the best.

#Cross validation Cross-validation refers to a set of methods for measuring the performance of a given predictive model on new test data sets.

The basic idea, behind cross-validation techniques, consists of dividing the data into two sets:

The training set, used to train (i.e. build) the model; and the testing set (or validation set), used to test (i.e. validate) the model by estimating the prediction error. Cross-validation is also known as a resampling method because it involves fitting the same statistical method multiple times using different subsets of the data.

Here you will learn: the most commonly used statistical metrics for measuring the performance of a regression model in predicting the outcome of new test data.

The different cross-validation methods for assessing model performance. I will cover the following approaches:

Validation set approach (or data split) Leave One Out Cross Validation k-fold Cross Validation Repeated k-fold Cross Validation Each of these methods has their advantages and drawbacks. Use the method that best suits your problem. Generally, the (repeated) k-fold cross validation is recommended.

```r
library(caret) #this package is really useful for cross validation
```

```
## Warning: package 'caret' was built under R version 4.3.2
```

```
## Loading required package: lattice
```

```
## Warning: package 'lattice' was built under R version 4.3.2
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

Model performance metrics

After building a model, we are interested in determining the accuracy of this model on predicting the outcome for new unseen observations not used to build the model. Put in other words, we want to estimate the prediction error.

To do so, the basic strategy is to:

Build the model on a training data set Apply the model on a new test data set to make predictions Compute the prediction errors We will use several statistical metrics for quantifying the overall quality of regression models. These include:

R-squared (R2), representing the squared correlation between the observed outcome values and the predicted values by the model. The higher the adjusted R2, the better the model.

Root Mean Squared Error (RMSE), which measures the average prediction error made by the model in predicting the outcome for an observation. That is, the average difference between the observed known outcome values and the values predicted by the model. The lower the RMSE, the better the model.

Mean Absolute Error (MAE), an alternative to the RMSE that is less sensitive to outliers. It corresponds to the average absolute difference between observed and predicted outcomes. The lower the MAE, the better the model

In classification setting, the prediction error rate is estimated as the proportion of misclassified observations.

Cross-validation methods Briefly, cross-validation algorithms can be summarized as :

1. Reserve a small sample of the data set
2. Build (or train) the model using the remaining part of the data set
3. Test the effectiveness of the model on the the reserved sample of the data set. If the model works well on the test data set, then it's good.

##Validation set up The validation set approach consists of randomly splitting the data into two sets: one set is used to train the model and the remaining other set sis used to test the model.

The process works as follow:

Build (train) the model on the training data set Apply the model to the test data set to predict the outcome of new unseen observations Quantify the prediction error as the mean squared difference between the observed and the predicted outcome values. The example below splits the data set so that 70% is used for training a linear regression model and 30% is used to evaluate the model performance.

```r
# Split the data into training and test set
set.seed(123)
training.samples <- antdat$log_density %>%
  createDataPartition(p = 0.7, list = FALSE)
train.data  <- antdat[training.samples, ]
test.data <- antdat[-training.samples, ]
```

```r
# Build the model
lm_full <- lm(log_density ~ Elevation + Latitude + sarah + randomx, data = antdat)
# Make predictions and compute the R2, RMSE and MAE
predictions_full <- lm_full %>% predict(test.data)
data.frame( R2 = R2(predictions_full, test.data$log_density),
            RMSE = RMSE(predictions_full, test.data$log_density),
            MAE = MAE(predictions_full, test.data$log_density))
```

```
##          R2      RMSE       MAE
## 1 0.7472895 0.1857941 0.1570727
```

```r
#compare that to a simpler model
lm_simp <- lm(log_density ~ Elevation + Latitude, data = antdat)
# Make predictions and compute the R2, RMSE and MAE
predictions_simp <- lm_simp %>% predict(test.data)
data.frame( R2 = R2(predictions_simp, test.data$log_density),
            RMSE = RMSE(predictions_simp, test.data$log_density),
            MAE = MAE(predictions_simp, test.data$log_density))
```

```
##         R2      RMSE       MAE
## 1 0.663139 0.2138176 0.2001537
```

When comparing two models, the one that produces the lowest test sample RMSE is the preferred model.

the RMSE and the MAE are measured in the same scale as the outcome variable. Dividing the RMSE by the average value of the outcome variable will give you the prediction error rate, which should be as small as possible:

```r
RMSE(predictions_full, test.data$log_density)/mean(test.data$log_density)
```

```
## [1] 0.08717448
```

```r
RMSE(predictions_simp, test.data$log_density)/mean(test.data$log_density)
```

```
## [1] 0.1003231
```

Note that, the validation set method is only useful when you have a large data set that can be partitioned. A disadvantage is that we build a model on a fraction of the data set only, possibly leaving out some interesting information about data, leading to higher bias. Therefore, the test error rate can be highly variable, depending on which observations are included in the training set and which observations are included in the validation set.

Leave one out cross validation - LOOCV

This method works as follow:

Leave out one data point and build the model on the rest of the data set Test the model against the data point that is left out at step 1 and record the test error associated with the prediction Repeat the process for all data points Compute the overall prediction error by taking the average of all these test error estimates recorded at step 2.

```r
# Define training control
train.control <- trainControl(method = "LOOCV")
# Train the model
model_full <- train(log_density ~ Elevation + Latitude + sarah + randomx, data = antdat, method = "lm",
               trControl = train.control)
# Summarize the results
print(model_full)
```

```
## Linear Regression
##
## 22 samples
##  4 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 21, 21, 21, 21, 21, 21, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3336473  0.4558807  0.2588082
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
#simpler model
# Define training control
train.control <- trainControl(method = "LOOCV")
# Train the model
model_simp <- train(log_density ~ Elevation + Latitude, data = antdat, method = "lm",
               trControl = train.control)
# Summarize the results
print(model_simp)
```

```
## Linear Regression
##
## 22 samples
##  2 predictor
##
## No pre-processing
## Resampling: Leave-One-Out Cross-Validation
## Summary of sample sizes: 21, 21, 21, 21, 21, 21, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3346335  0.4410209  0.2759947
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

The advantage of the LOOCV method is that we make use all data points reducing potential bias.

However, the process is repeated as many times as there are data points, resulting to a higher execution time when n is extremely large.

Additionally, we test the model performance against one data point at each iteration. This might result to higher variation in the prediction error, if some data points are outliers. So, we need a good ratio of testing data points, a solution provided by the k-fold cross-validation method.

K-fold cross-validation

The k-fold cross-validation method evaluates the model performance on different subset of the training data and then calculate the average prediction error rate. The algorithm is as follow:

Randomly split the data set into k-subsets (or k-fold) (for example 5 subsets) Reserve one subset and train the model on all other subsets Test the model on the reserved subset and record the prediction error Repeat this process until each of the k subsets has served as the test set. Compute the average of the k recorded

errors. This is called the cross-validation error serving as the performance metric for the model. K-fold cross-validation (CV) is a robust method for estimating the accuracy of a model.

The most obvious advantage of k-fold CV compared to LOOCV is computational. A less obvious but potentially more important advantage of k-fold CV is that it often gives more accurate estimates of the test error rate than does LOOCV (James et al. 2014).

Typical question, is how to choose right value of k?

Lower value of K is more biased and hence undesirable. On the other hand, higher value of K is less biased, but can suffer from large variability. It is not hard to see that a smaller value of k (say k = 2) always takes us towards validation set approach, whereas a higher value of k (say k = number of data points) leads us to LOOCV approach.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2014. An Introduction to Statistical Learning: With Applications in R. Springer Publishing Company, Incorporated.

In practice, one typically performs k-fold cross-validation using k = 5 or k = 10, as these values have been shown empirically to yield test error rate estimates that suffer neither from excessively high bias nor from very high variance.

The following example uses 10-fold cross validation to estimate the prediction error. Make sure to set seed for reproducibility.

```
# Define training control
set.seed(123)
train.control <- trainControl(method = "cv", number = 10)
# Train the full model
model_full <- train(log_density ~ Elevation + Latitude + sarah + randomx, data = antdat, method = "lm",
               trControl = train.control)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```
# Summarize the results
print(model_full)
```

```
## Linear Regression
##
## 22 samples
##  4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 19, 20, 20, 20, 20, 20, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3075862  0.9374741  0.2676076
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# Train the simpler model
model_simp <- train(log_density ~ Elevation + Latitude, data = antdat, method = "lm",
               trControl = train.control)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo,
## : There were missing values in resampled performance measures.
```

```r
# Summarize the results
print(model_simp)
```

```
## Linear Regression
##
## 22 samples
##  2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 20, 20, 20, 19, 20, 20, ...
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.3046901  0.9013132  0.2738229
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Repeated K-fold cross-validation

The process of splitting the data into k-folds can be repeated a number of times, this is called repeated k-fold cross validation.

The final model error is taken as the mean error from the number of repeats.

Because we have such a small dataset, I want to try using a smaller number of folds. The following example uses 3-fold cross validation with 5 repeats:

```r
# Define training control
set.seed(123)
train.control <- trainControl(method = "repeatedcv", number = 3, repeats = 5)
# Train the full model
model_full <- train(log_density ~ Elevation + Latitude + sarah + randomx, data = antdat, method = "lm",
               trControl = train.control)
# Summarize the results
print(model_full)
```

```
## Linear Regression
##
## 22 samples
##  4 predictor
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 5 times)
## Summary of sample sizes: 14, 15, 15, 15, 14, 15, ...
## Resampling results:
##
##   RMSE      Rsquared  MAE
##   0.377026  0.489883  0.3017988
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```r
# Train the simpler model
model_simp <- train(log_density ~ Elevation + Latitude, data = antdat, method = "lm",
               trControl = train.control)
# Summarize the results
```

```
print(model_simp)
```

```
## Linear Regression
##
## 22 samples
##  2 predictor
##
## No pre-processing
## Resampling: Cross-Validated (3 fold, repeated 5 times)
## Summary of sample sizes: 15, 14, 15, 16, 14, 14, ...
## Resampling results:
##
##   RMSE     Rsquared   MAE
##   0.30382  0.5608188  0.2506857
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

From all of these results I would say that the model that just includes latitude and elevation is better at out-of-sample prediction that the model that includes sarah and randomx.

##looping You can loop through different combinations of predictor variables in k fold cross validation. This is a function I wrote to get out all of the possible combinations of our linear models - be very very careful with this

```
library(gtools)
```

```
## Warning: package 'gtools' was built under R version 4.3.2
```

```
##
## Attaching package: 'gtools'
```

```
## The following object is masked from 'package:car':
##
##     logit
```

```
pastePerm<- function(row, names){
  keep<- which(row==1)
  if(length(keep)==0){
    return('1')
  }else{
    return(paste(names[keep],collapse='+'))
  }
}
dredgeform<- function(pred, covars, alwaysIn=''){
  p<- length(covars)
  perm.tab<- permutations(2, p, v=c(0,1), repeats.allowed=T)
  myforms<- NULL
  for(j in 1:nrow(perm.tab)){
    myforms[j]<- pastePerm(perm.tab[j,], covars)
  }
  myforms<- paste0(pred, '~ 1', alwaysIn,'+', myforms)
  return(myforms)
}

allformulas<- dredgeform(pred = "log_density", covars = c("Elevation", "Latitude", "sarah", "randomx"))
allformulas <- allformulas[2:length(allformulas)] #i dont want the intercept only one
```

now pass these formulas through a for loop in a cross validation - lets say we want this to repeat 100 times

```
set.seed(123)
compare_var <- as.data.frame(matrix(ncol = 4, nrow = 0))
colnames(compare_var) <- c("formula", "RMSE", "R2", "MAE")

for ( i in 1:length(allformulas)) {

train.control <- trainControl(method = "repeatedcv", number = 3, repeats = 100)
# Train the full model
model_full <- train(as.formula(allformulas[i]), data = antdat, method = "lm",
               trControl = train.control)
# Summarize the results
compare_var[i, 1] <- allformulas[i]
compare_var[i, 2] <- mean(model_full$resample$RMSE)
compare_var[i, 3] <- mean(model_full$resample$Rsquared, na.rm = T)
compare_var[i, 4] <- mean(model_full$resample$MAE)


}

compare_var$prediction_error_rate <- compare_var$RMSE/mean(antdat$log_density)

compare_var %>% arrange(prediction_error_rate)
```

```
##                                           formula      RMSE        R2
## 1                    log_density~ 1+Elevation+sarah 0.3121811 0.5885891
## 2         log_density~ 1+Elevation+Latitude+randomx 0.3242515 0.5839693
## 3                 log_density~ 1+Elevation+Latitude 0.3243645 0.5559310
## 4           log_density~ 1+Elevation+sarah+randomx 0.3270392 0.5696257
## 5          log_density~ 1+Elevation+Latitude+sarah 0.3288865 0.5475511
## 6                              log_density~ 1+sarah 0.3294814 0.5557290
## 7  log_density~ 1+Elevation+Latitude+sarah+randomx 0.3369308 0.5579627
## 8                      log_density~ 1+sarah+randomx 0.3480844 0.5268454
## 9                    log_density~ 1+Latitude+sarah 0.3578717 0.4894346
## 10           log_density~ 1+Latitude+sarah+randomx 0.3704350 0.4544699
## 11                          log_density~ 1+Latitude 0.3784386 0.3945685
## 12                log_density~ 1+Elevation+randomx 0.3786485 0.4248544
## 13                 log_density~ 1+Latitude+randomx 0.3838907 0.4179896
## 14                       log_density~ 1+Elevation 0.3914144 0.4197541
## 15                         log_density~ 1+randomx 0.4377794 0.1772593
##          MAE prediction_error_rate
## 1  0.2518662             0.1474266
## 2  0.2584176             0.1531268
## 3  0.2713005             0.1531801
## 4  0.2608912             0.1544432
## 5  0.2686359             0.1553156
## 6  0.2738744             0.1555966
## 7  0.2707658             0.1591145
## 8  0.2936545             0.1643817
## 9  0.2943496             0.1690038
## 10 0.3108246             0.1749367
## 11 0.3057838             0.1787165
## 12 0.3039888             0.1788156
## 13 0.3210861             0.1812912
```

```
## 14 0.3316630             0.1848442
## 15 0.3730294             0.2067400
```

```
compare_var %>% arrange(RMSE)
```

```
##                                          formula      RMSE        R2
## 1                    log_density~ 1+Elevation+sarah 0.3121811 0.5885891
## 2         log_density~ 1+Elevation+Latitude+randomx 0.3242515 0.5839693
## 3                 log_density~ 1+Elevation+Latitude 0.3243645 0.5559310
## 4            log_density~ 1+Elevation+sarah+randomx 0.3270392 0.5696257
## 5           log_density~ 1+Elevation+Latitude+sarah 0.3288865 0.5475511
## 6                              log_density~ 1+sarah 0.3294814 0.5557290
## 7  log_density~ 1+Elevation+Latitude+sarah+randomx 0.3369308 0.5579627
## 8                       log_density~ 1+sarah+randomx 0.3480844 0.5268454
## 9                     log_density~ 1+Latitude+sarah 0.3578717 0.4894346
## 10           log_density~ 1+Latitude+sarah+randomx 0.3704350 0.4544699
## 11                          log_density~ 1+Latitude 0.3784386 0.3945685
## 12                log_density~ 1+Elevation+randomx 0.3786485 0.4248544
## 13                 log_density~ 1+Latitude+randomx 0.3838907 0.4179896
## 14                        log_density~ 1+Elevation 0.3914144 0.4197541
## 15                          log_density~ 1+randomx 0.4377794 0.1772593
##          MAE prediction_error_rate
## 1  0.2518662             0.1474266
## 2  0.2584176             0.1531268
## 3  0.2713005             0.1531801
## 4  0.2608912             0.1544432
## 5  0.2686359             0.1553156
## 6  0.2738744             0.1555966
## 7  0.2707658             0.1591145
## 8  0.2936545             0.1643817
## 9  0.2943496             0.1690038
## 10 0.3108246             0.1749367
## 11 0.3057838             0.1787165
## 12 0.3039888             0.1788156
## 13 0.3210861             0.1812912
## 14 0.3316630             0.1848442
## 15 0.3730294             0.2067400
```

If prediction is what we are after, it appears Elevation + Sarah or Elevation + Latitude is our best bet.

This is an important thing to consider, that just finding the "best" model isn't always appropriate. Sure, if we want to predict species diversity maybe we would use Elevation + Latitude + Sarah, but if we want to truly understand the effect of latitude or elevation on ant diversity, we would not use stepwise selection or cross-validation. I think a lot of people fall into these pitfalls, when you have to remember why you built your model in the first place. Is it to examine the effect of something on something else or is it only for out-of-sample prediction?

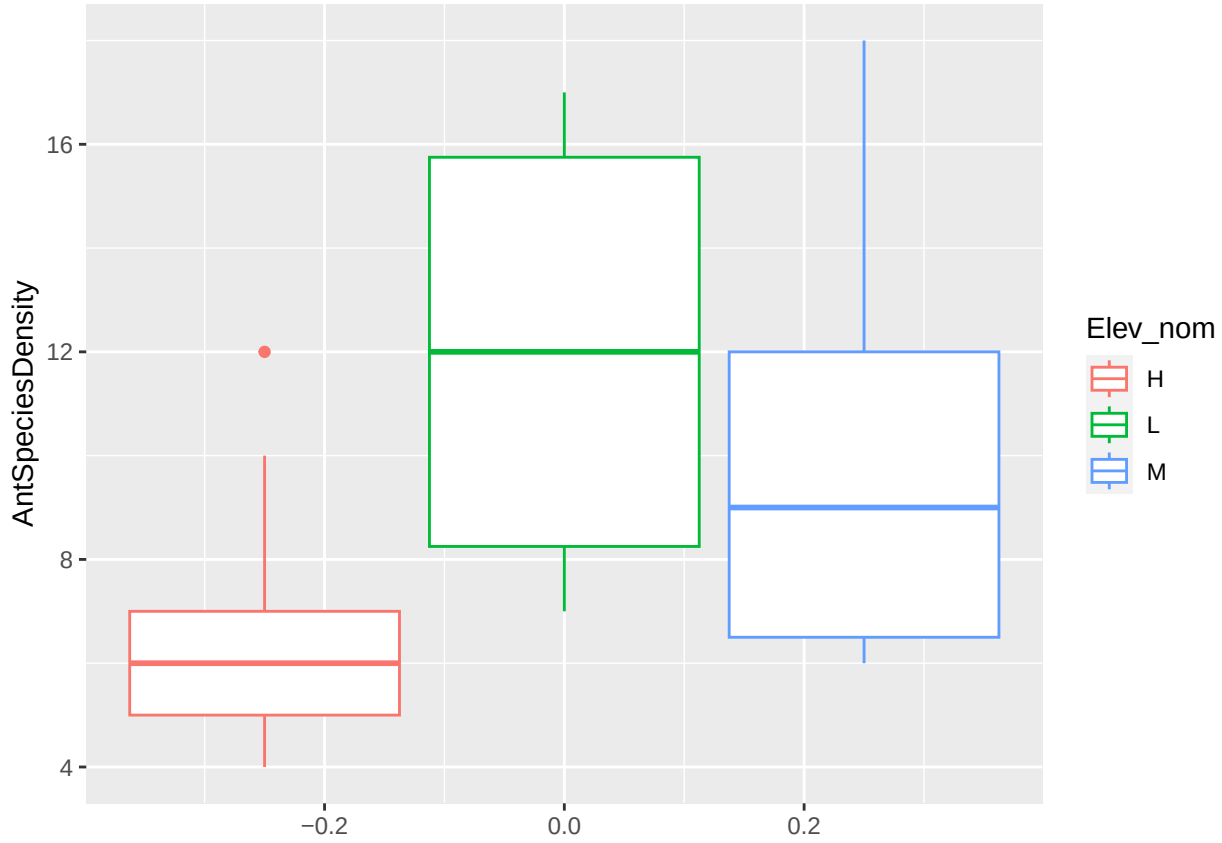
#nominal variables Let's show you how to interpret and plot nominal variables in R. Pretend we make Elevation into a nominal variable, High, Medium, Low

Note, I am going to run this on the non-logged transformed data for ease of interpretation (for your sake), in practice, you want to make sure your assumptions are met, and can log the data if not, then follow the log rules from the lecture.

```
antdat <- antdat %>% mutate(Elev_nom = ifelse(Elevation > 300, "H", ifelse(Elevation > 100 & Elevation <

antdat$Elev_nom <- as.factor(antdat$Elev_nom)
```

When building a regression model with categorical variables with more than two levels (ie High, Med, Low) R is doing internally some transformation to be able to compute regression coefficient. What R is doing is that it is turning your categorical variables into a set of contrasts, this number of contrasts is the number of level in the variable (3 in the example above) minus 1.

```
contrasts(antdat$Elev_nom) <- contr.treatment(n=levels(antdat$Elev_nom))

design <- model.matrix(~antdat$Elev_nom)
antdat_example <- cbind(antdat, design[,2:3])

antdat %>% ggplot(aes(y = AntSpeciesDensity, colour = Elev_nom)) + geom_boxplot()
```




```
lm4 <- lm(AntSpeciesDensity ~ Elev_nom, data = antdat)
summary(lm4)
```

```
##
## Call:
## lm(formula = AntSpeciesDensity ~ Elev_nom, data = antdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.0000 -2.9167 -0.6667  3.2500  8.0000
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    6.667      1.280   5.210 4.99e-05 ***
## Elev_nomL      5.333      2.023   2.636   0.0163 *
```

```
## Elev_nomM      3.333      1.935   1.723   0.1011
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.839 on 19 degrees of freedom
## Multiple R-squared:  0.2807, Adjusted R-squared:  0.205
## F-statistic: 3.707 on 2 and 19 DF,  p-value: 0.04371
```

```
antdat %>% anova_test(AntSpeciesDensity ~ Elev_nom, detailed = T)
```

```
## ANOVA Table (type II tests)
##
##     Effect     SSn SSd DFn DFd     F     p p<.05   ges
## 1 Elev_nom 109.273 280   2  19 3.707 0.044     * 0.281
```

```
antdat %>% tukey_hsd(AntSpeciesDensity ~ Elev_nom, detailed = T)
```

```
## # A tibble: 3 x 9
##   term  group1 group2 null.value estimate conf.low conf.high  p.adj p.adj.signif
## * <chr> <chr>  <chr>       <dbl>    <dbl>    <dbl>     <dbl>  <dbl> <chr>
## 1 Elev~ H      L               0     5.33    0.193     10.5  0.0412 *
## 2 Elev~ H      M               0     3.33   -1.58       8.25 0.223  ns
## 3 Elev~ L      M               0    -2.00   -7.43       3.43 0.625  ns
```

The first coefficient, the intercept, is the estimated average for the baseline group, which in our case is the high group (in the high elevation group the ant diversity is estimated to be on average 6.667). The second coefficient "low" is the estimated difference between the average in the baseline group and the average in the "low" group (this group has an increase in 5.33 diversity compared to the control group). Similarly the third coefficient is the difference between the average in the baseline and the "Med" group (this group has an increase of 3.33 diversity compared to the control group).

You can read a bit more about contrasts here: https://rpubs.com/monajhzhu/608609

What if we add in Latitude and keep in the categorical variable for elevation?

```
lm5 <- lm(AntSpeciesDensity ~ Latitude + Elev_nom, data = antdat)
summary(lm5)
```

```
##
## Call:
## lm(formula = AntSpeciesDensity ~ Latitude + Elev_nom, data = antdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.6876 -2.7513 -0.2331  2.2653  5.9112
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  91.2324    29.8857   3.053  0.00685 **
## Latitude     -1.9541     0.6901  -2.832  0.01106 *
## Elev_nomL     3.9083     1.8008   2.170  0.04360 *
## Elev_nomM     2.9866     1.6578   1.802  0.08839 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.281 on 18 degrees of freedom
## Multiple R-squared:  0.5024, Adjusted R-squared:  0.4194
## F-statistic: 6.057 on 3 and 18 DF,  p-value: 0.004897
```

When building linear model, there are different ways to encode categorical variables, known as contrast coding systems. The default option in R is to use the first level of the factor as a reference and interpret the remaining levels relative to this level.

Note that, ANOVA (analyse of variance) is just a special case of linear model where the predictors are categorical variables. And, because R understands the fact that ANOVA and regression are both examples of linear models, it lets you extract the classic ANOVA table from your regression model using the R base anova() function or the Anova() function [in car package]

The results of predicting density from using a multiple regression procedure are presented below.

```
anova(lm5)
```

```
## Analysis of Variance Table
##
## Response: AntSpeciesDensity
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## Latitude   1 134.562 134.562 12.5035 0.00236 **
## Elev_nom   2  60.996  30.498  2.8339 0.08512 .
## Residuals 18 193.715  10.762
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Taking other variables (latitude) into account, it can be seen that the categorical variable ekev is no longer significantly associated with the variation in density. Latitude is still significant.

When you put a factor variable into a regression, you're allowing a different intercept at every level of the factor.

How do we interpret the 2 elevation coefficients? For categorical variables, the interpretation is relative to the given baseline. The baseline is just whatever level comes first (here, "high"). E.g., the estimate of elev_l means that the estimated intercept is 3.9083 higher among "low" elevation compared to high elevation. Another way of putting it: among latitudes of the same level, density at low elevation is, on average 3.39 (density units) higher than high elevation.

We can now do some predictions

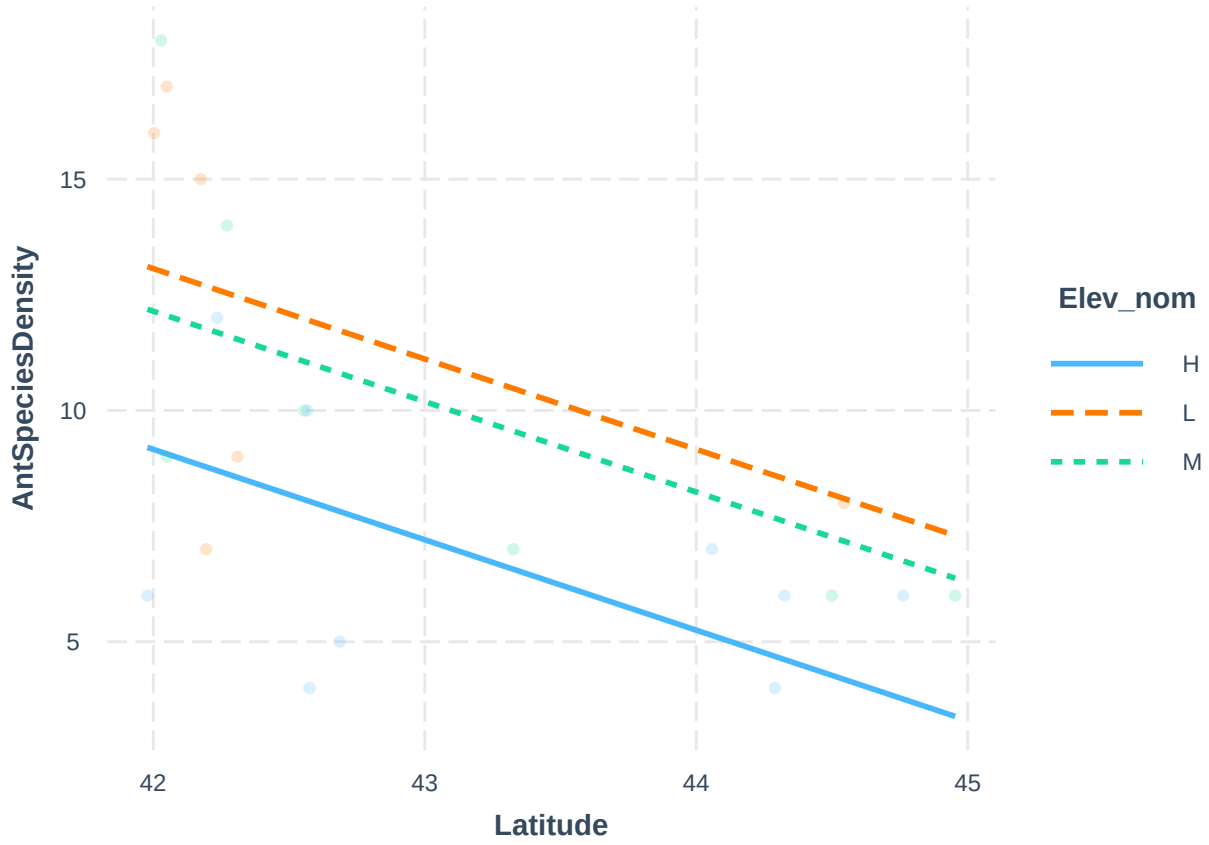
```
pred <- predict(lm5, list(Latitude = 43, Elev_nom = "L"), interval = "prediction")
pred
```

```
##        fit      lwr      upr
## 1 11.11292 3.639499 18.58634
```

Essentially you're saying that your data is broken down into 3 elevation groups, and you want to model your data as having the same slope governing how density changes with latitude, but potentially different intercepts. Here's a picture of what's happening.

```
interact_plot(lm5, pred = Latitude, modx = Elev_nom,
              plot.points = TRUE, point.alpha = 0.2)
```

```
## Warning: Latitude and Elev_nom are not included in an interaction with one another
## in the model.
```



In this case we have not only elevation-specific intercepts, but also elevation-specific slopes.

```
lm6 <- lm(AntSpeciesDensity ~ Latitude*Elev_nom, dat = antdat)
summary(lm6)
```

```
##
## Call:
## lm(formula = AntSpeciesDensity ~ Latitude * Elev_nom, data = antdat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8186 -2.1211  0.4751  2.0177  4.9798
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         46.4653    47.7999   0.972    0.345
## Latitude            -0.9197     1.1043  -0.833    0.417
## Elev_nomL           64.5213    80.0295   0.806    0.432
## Elev_nomM           85.3072    68.1937   1.251    0.229
## Latitude:Elev_nomL  -1.4069     1.8693  -0.753    0.463
## Latitude:Elev_nomM  -1.9058     1.5786  -1.207    0.245
##
## Residual standard error: 3.324 on 16 degrees of freedom
## Multiple R-squared:  0.5459, Adjusted R-squared:  0.404
## F-statistic: 3.847 on 5 and 16 DF,  p-value: 0.01763
```

okay, I know these aren't significant, but I want to show you how to interpret interactions

We now have new terms appearing. Terms like Latitude:Elev_nomL are deviations from the baseline slope (the coefficient of Latitude in the model) in the same way that terms like Elev_nomL are deviations from the baseline intercept. This models says that:

On average among high elevation, every additional increase in latitude is associated with a -0.91 unit decrease in the ant diversity.

the coefficient of Latitude:Elev_nomL (-1.4069) means that slope of lattiude is 1.4 units lower for low elevation compared to high elevation.

So, do the lines with different slopes fit the data significantly better than the common slope model? Let's compare the two with the anova() function.

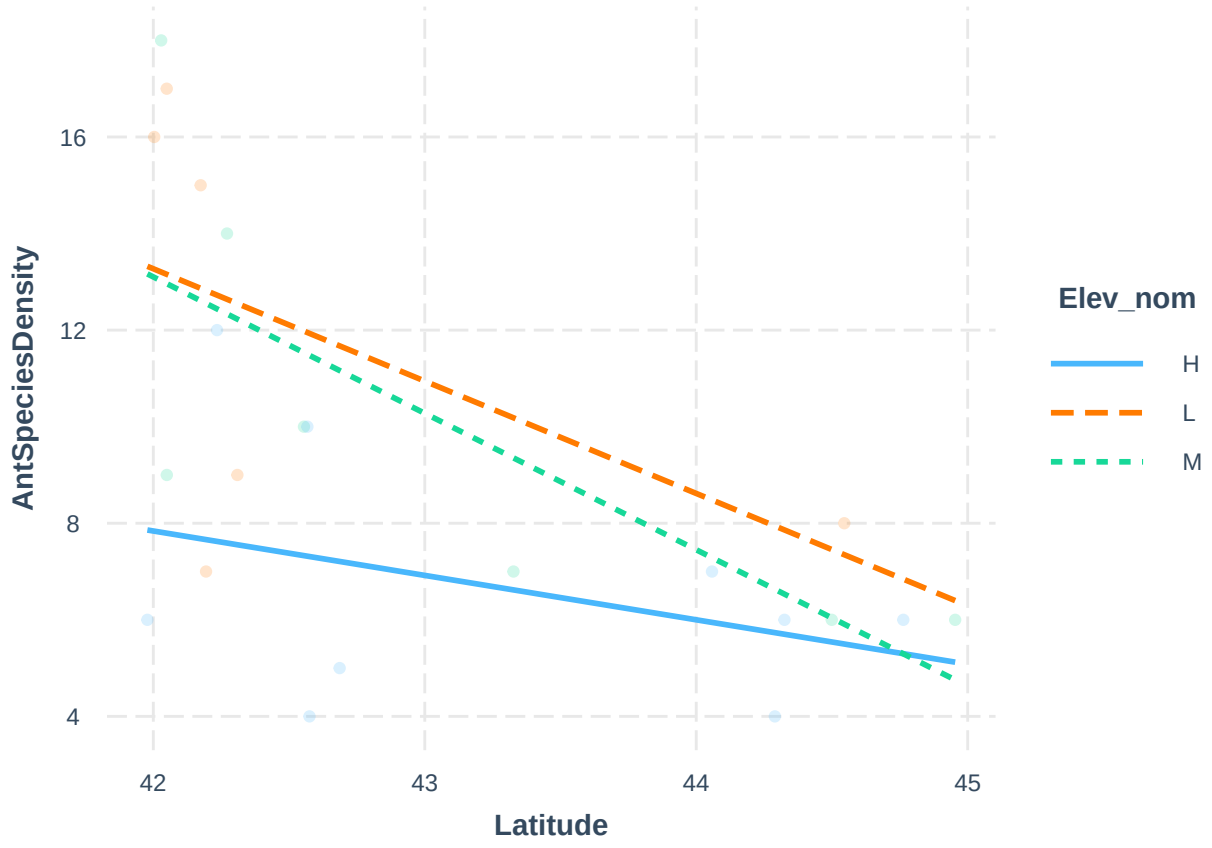
```
anova(lm5, lm6)
```

```
## Analysis of Variance Table
##
## Model 1: AntSpeciesDensity ~ Latitude + Elev_nom
## Model 2: AntSpeciesDensity ~ Latitude * Elev_nom
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1     18 193.72
## 2     16 176.76  2    16.959 0.7676 0.4805
```

This p-value turns out to not be statistically significant. So even though the estimated slopes in the interaction model look very different, our estimates are quite variable, so we don't have enough evidence to conclude that the interaction term (different slopes) is providing significant additional explanatory power over the simpler model

finally, you can plot these using interact_plot

```
#as well as interact plot
interact_plot(lm6, pred = Latitude, modx = Elev_nom,
              plot.points = TRUE, point.alpha = 0.2)
```



so beautiful

#in class assessment Make the above interaction plot using geom_point and stat_smooth. The colours don't exactly have to match, but the plot should look very similar. Upload a screenshot of your code and plot. Note, upload by Friday, February 17th at 5pm along with your problem set.

```
library(ggpmisc)
library(ggthemes)

 inclass_ass <- antdat %>% ggplot(aes(x = Latitude, y = AntSpeciesDensity, color = Elev_nom)) +
  geom_point()+
  labs(y="Ant Species Density", x="Latitude", title = "In class assignment 5") +
  theme_Publication()+
  stat_smooth(method=lm, alpha = 0.5, se=F) +
  scale_color_manual(values = c("red", "blue", "green"))


ggsave("NTAQ_In class Assignment 5.jpeg", inclass_ass, width = 5, height = 4, units = "in", dpi = 300 )
```