MB590 Microbiome Analysis - Regression

Christine V. Hawkes

March 23, 2022

Contents


```
We will explore two general approaches to regression with microbial community data today.
1. Ordinary least squares regression
  + Examine relationship between Ordination axis 1 and putative explanatory variables
  + ‘olsrr‘ package
2. Lasso regression
  + Identify "features" (ASVs) that explain plant traits or other dependent vars
  + ‘glmnet‘ package

Reference: Emmert-Streib & Dehmer (2019) High-dimensional LASSO-based computational regression models: i
```

```
Data: Erlandson et al. (2018) Soil abiotic variables are more important than Salicaceae phylogeny or hal
    + DRYAD entry: https://datadryad.org/stash/dataset/doi:10.5061/dryad.5f24ks4
    + wk11 includes bacteria data w/ ps object, otu_table, tax_table, and sample_data files
```

Libraries and Data

Install and load R packages

```
#install.packages("olsrr")
#install.packages("corrplot")
#install.packages("glmnet")

library(tidyverse)
library(phyloseq)
library(DESeq2)
library(vegan)
library(olsrr)
library(corrplot)
library(glmnet)
library(ggplot2)
library(ggpubr)
```

Load and prepare data

Open data file

```
#download.file(githuburl, "wk11_data.Rdata")
load("wk11_data.RData")
```

VST transform otu data

```
ps_ds <- phyloseq::phyloseq_to_deseq2(ps_bac, ~Treatment + Ecology)

ps_ds <- DESeq2::estimateSizeFactors(ps_ds)
ps_ds = DESeq2::estimateDispersions(ps_ds, fitType = "parametric")

otu_vst<-DESeq2::getVarianceStabilizedData(ps_ds)
otu_vst<- t(otu_vst)
# min(otu_vst)
# ps_vst_pos <- transform_sample_counts(ps_vst, function(x) x+1.98)

ps_vst <- ps_bac
phyloseq::otu_table(ps_vst) <- phyloseq::otu_table(otu_vst, taxa_are_rows = FALSE)
ps_vst
```

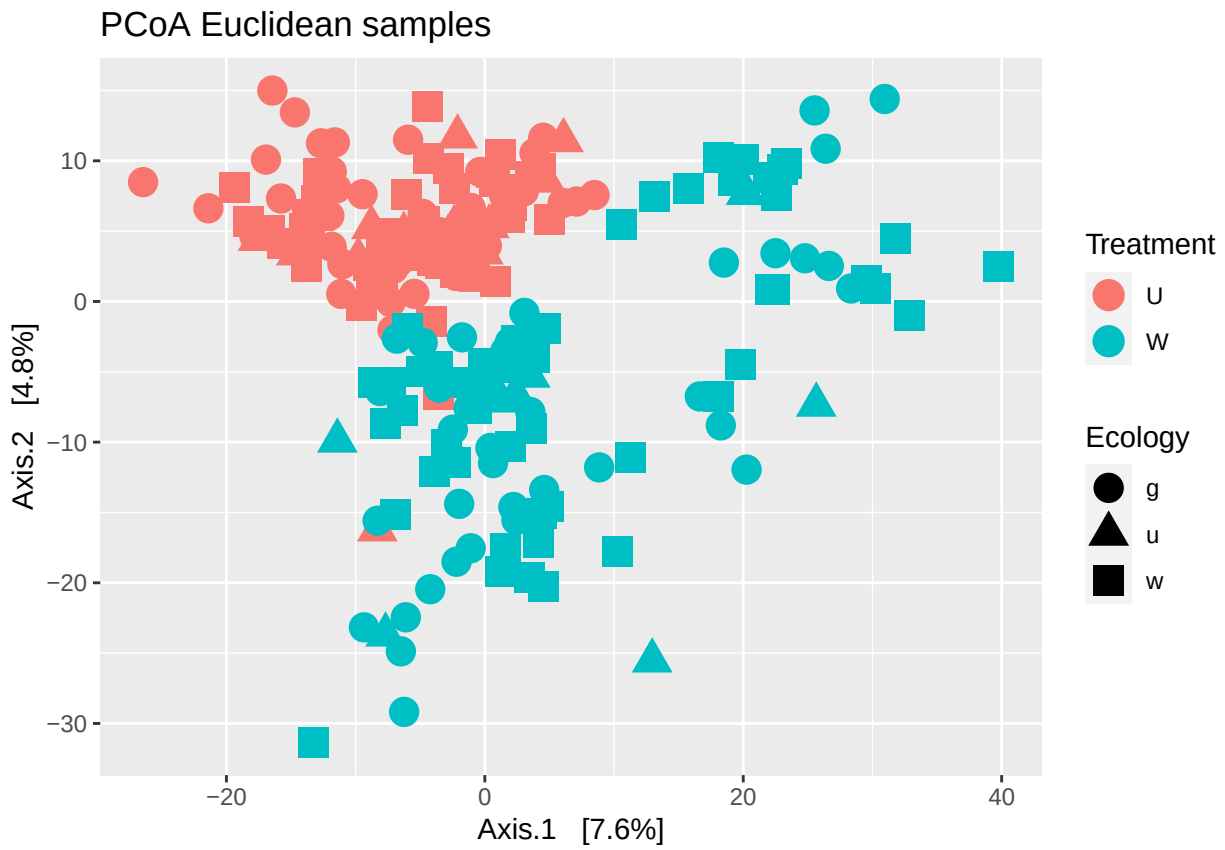
```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 6758 taxa and 215 samples ]
## sample_data() Sample Data:       [ 215 samples by 38 sample variables ]
## tax_table()   Taxonomy Table:    [ 6758 taxa by 7 taxonomic ranks ]
```

Linear regression analysis of environmental vars on ordination axes

Ordination

PCoA on vst-transformed data

```
# get ord scores for dependent vars (example with PCoA, but choice depends on data)
ord_vst <- phyloseq::ordinate(ps_vst, "PCoA", "euclidean")

phyloseq::plot_ordination(ps_vst, ord_vst,
                          type="samples",
                          color="Treatment",
                          shape = "Ecology",
                          title="PCoA Euclidean samples") +
                          ggplot2::geom_point(size=5)
```



```
# save sample scores from ord and set up for merger
sampleScores <- (ord_vst)$vectors
sampleScores <- tibble::rownames_to_column(as.data.frame(sampleScores)) %>%
                dplyr::select(rowname, Axis.1, Axis.2) %>%
                tibble::column_to_rownames(var="rowname")
```

Merge ordination samples scores into new env data

```
# get environmental data
env_vst <- phyloseq::sample_data(ps_vst)

# check that rownames match
all(rownames(sampleScores) == rownames(env_vst))
```

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

```
# merge and replace rownames lost in merge
env_new <- merge(env_vst, sampleScores, by="row.names") %>%
          tibble::column_to_rownames(var = "Row.names")
env_new[1,]
```

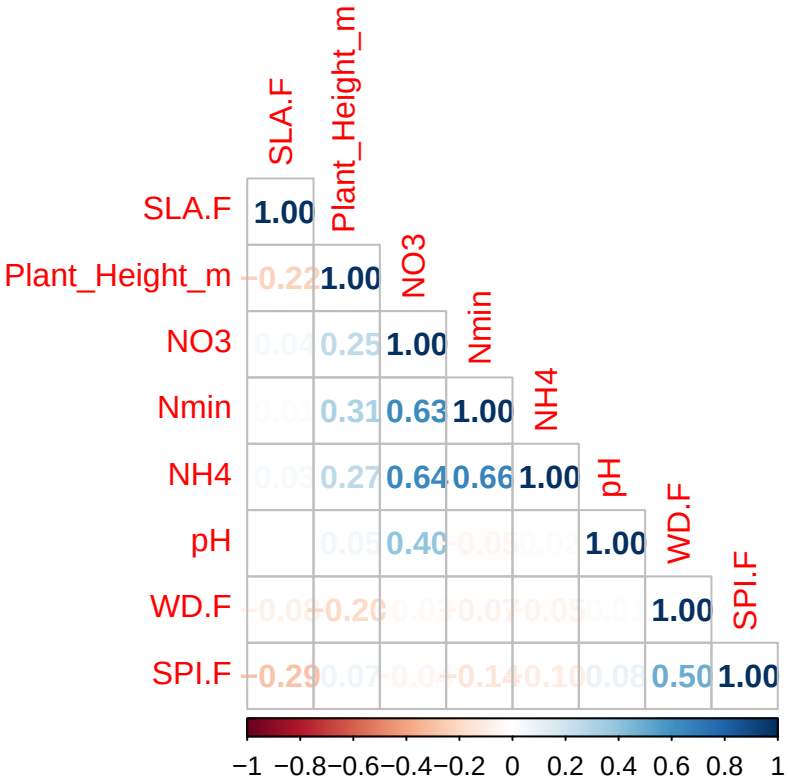
```
##         GardenID Garden.Location Number Treatment  June  July   Aug  Mean
## 11DAMYE      11D  Lawrence Strip      9         U -1.19 -1.39 -1.59 -1.39
##               Nmin    NO3    NH4   pH Spp Ecology  Sample Genotype
## 11DAMYE 0.02300651 0.0351 0.3768 5.41 AMY       g 11DAMYE
##         Caged.E..Not.Caged Plant_Height_m Date_Sampled extraction_date      Lat
## 11DAMYE                  E           0.21      7/11/13        10/24/13 45.40912
##              Long Plot       Dist1       Dist2     Dist3 order   TLP WD SPI LSV
## 11DAMYE -93.18967   11 -0.03877426 -0.01062293 0.1306906   157 -2.07 NA  NA   5
##          RER SLA   RGR TLP.F  WD.F SPI.F SLA.F    Axis.1   Axis.2
## 11DAMYE 0.15  NA 0.082 -1.86 0.432 0.086 157.7 -12.67706 11.24489
```

```
phyloseq::sample_data(ps_vst) <- phyloseq::sample_data(env_new)
```

Linear model design

Examine env var correlations

```
# first limit file to quantitative vars with no NAs
env_corr <- subset(env_vst, select=c(Plant_Height_m, pH, Nmin, NO3, NH4, WD.F, SPI.F, SLA.F))
# visualize the correlations and limit model design to avoid highly correlated vars
corrplot::corrplot(cor(env_corr), method="number", type="lower", order="hclust")
```



Set model independent and dependent vars

```
model1 <- lm(Axis.1 ~ Plant_Height_m + pH + Nmin + WD.F + SPI.F + SLA.F, data = env_new)
model1
```

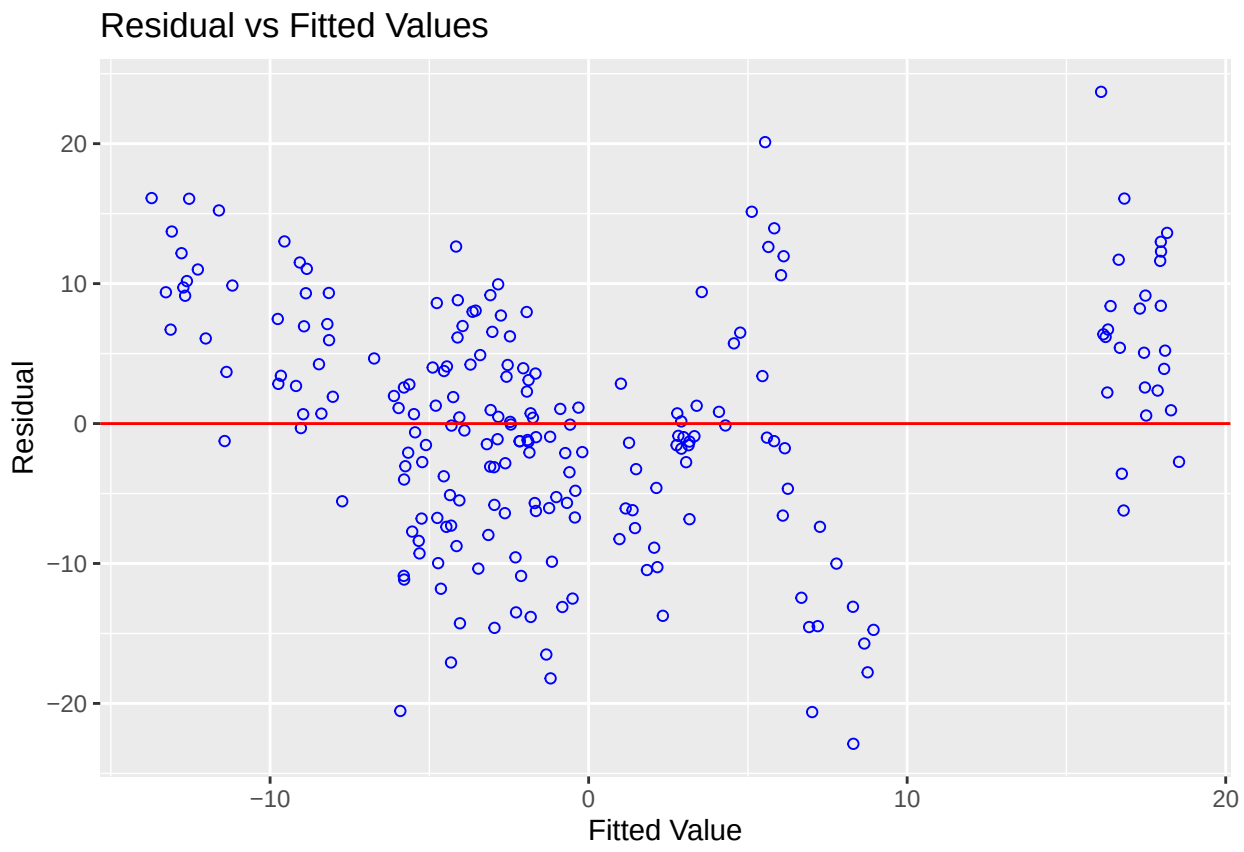
```
##
## Call:
## lm(formula = Axis.1 ~ Plant_Height_m + pH + Nmin + WD.F + SPI.F +
##     SLA.F, data = env_new)
##
## Coefficients:
##    (Intercept)  Plant_Height_m              pH            Nmin            WD.F
##     -139.30663         0.44056        25.39786        51.95849       -20.95408
##          SPI.F           SLA.F
##        2.57662        -0.01255
```

Evaluate assumptions of linear regression

For help, see

- https://cran.r-project.org/web/packages/olsrr/vignettes/residual_diagnostics.html
- https://cran.r-project.org/web/packages/olsrr/vignettes/regression_diagnostics.html
- https://cran.r-project.org/web/packages/olsrr/vignettes/heteroskedasticity.html

```
# "eyeball" test of non-linearity, unequal error variance, and outliers
# residuals should spread randomly around the zero line to indicate linearity
# residuals should also be in a horizontal band around 0 line for homogeneity of variance
# no residuals should be far away from the random pattern (= outliers)
olsrr::ols_plot_resid_fit(model1)
```




```
# Shapiro-Wilk test of normality outperforms all others, followed by Anderson-Darling
olsrr::ols_test_normality(model1)
```

```
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.9952         0.7372
## Kolmogorov-Smirnov        0.0525         0.5934
## Cramer-von Mises         15.6461         0.0000
## Anderson-Darling          0.3309         0.5114
## -----------------------------------------------
```

```
# heteroskedasticity X2 test for constant error variance
olsrr::ols_test_breusch_pagan(model1, rhs=TRUE, multiple=TRUE)
```

```
##
##  Breusch Pagan Test for Heteroskedasticity
##  -----------------------------------------
```

```
##  Ho: the variance is constant
##  Ha: the variance is not constant
##
##                       Data
##  ---------------------------------------------
##  Response : Axis.1
##  Variables: Plant_Height_m pH Nmin WD.F SPI.F SLA.F
##
##           Test Summary (Unadjusted p values)
##  ---------------------------------------------
##   Variable             chi2        df        p
##  ---------------------------------------------
##   Plant_Height_m     3.92071109     1    0.04769470
##   pH                 2.08949916     1    0.14831469
##   Nmin               1.44443616     1    0.22942282
##   WD.F               1.03708786     1    0.30849973
##   SPI.F              0.64776154     1    0.42091412
##   SLA.F              0.07430864     1    0.78516371
##  ---------------------------------------------
##   simultaneous      12.21690161     6    0.05730127
##  ---------------------------------------------
```

```
#test of collinearity; vif=1 indicates no correlations
olsrr::ols_coll_diag(model1)
```

```
## Tolerance and Variance Inflation Factor
## ---------------------------------------
##        Variables Tolerance      VIF
## 1 Plant_Height_m 0.7834913 1.276338
## 2             pH 0.9853261 1.014892
## 3           Nmin 0.8649843 1.156090
## 4           WD.F 0.6900416 1.449188
## 5          SPI.F 0.6493897 1.539907
## 6          SLA.F 0.8726329 1.145957
##
##
## Eigenvalue and Condition Index
## ------------------------------
##     Eigenvalue Condition Index    intercept Plant_Height_m           pH
## 1 6.5239500160        1.000000 1.477676e-05    0.002587152 1.858571e-05
## 2 0.2968959073        4.687629 5.662320e-05    0.008727644 7.375400e-05
## 3 0.1259518087        7.197027 6.510467e-05    0.817102753 7.453284e-05
## 4 0.0421487928       12.441213 4.422786e-04    0.023560882 5.299633e-04
## 5 0.0083199935       28.002307 8.501847e-03    0.028457915 1.152380e-02
## 6 0.0023589245       52.589409 2.754894e-02    0.116165957 8.078691e-02
## 7 0.0003745572      131.976394 9.633704e-01    0.003397697 9.069925e-01
##          Nmin         WD.F        SPI.F        SLA.F
## 1 0.004958291 7.406839e-05 0.0007099397 0.0003064333
## 2 0.758197949 3.558022e-04 0.0066973135 0.0012387661
## 3 0.162483671 5.952657e-04 0.0004204574 0.0049728173
## 4 0.061678231 7.781851e-06 0.5272746132 0.0719421168
## 5 0.001713595 5.493247e-02 0.3029609301 0.8792942196
## 6 0.009133509 8.703929e-01 0.1479974750 0.0313593725
## 7 0.001834753 7.364173e-02 0.0139392711 0.0108862744
```

Run linear regression

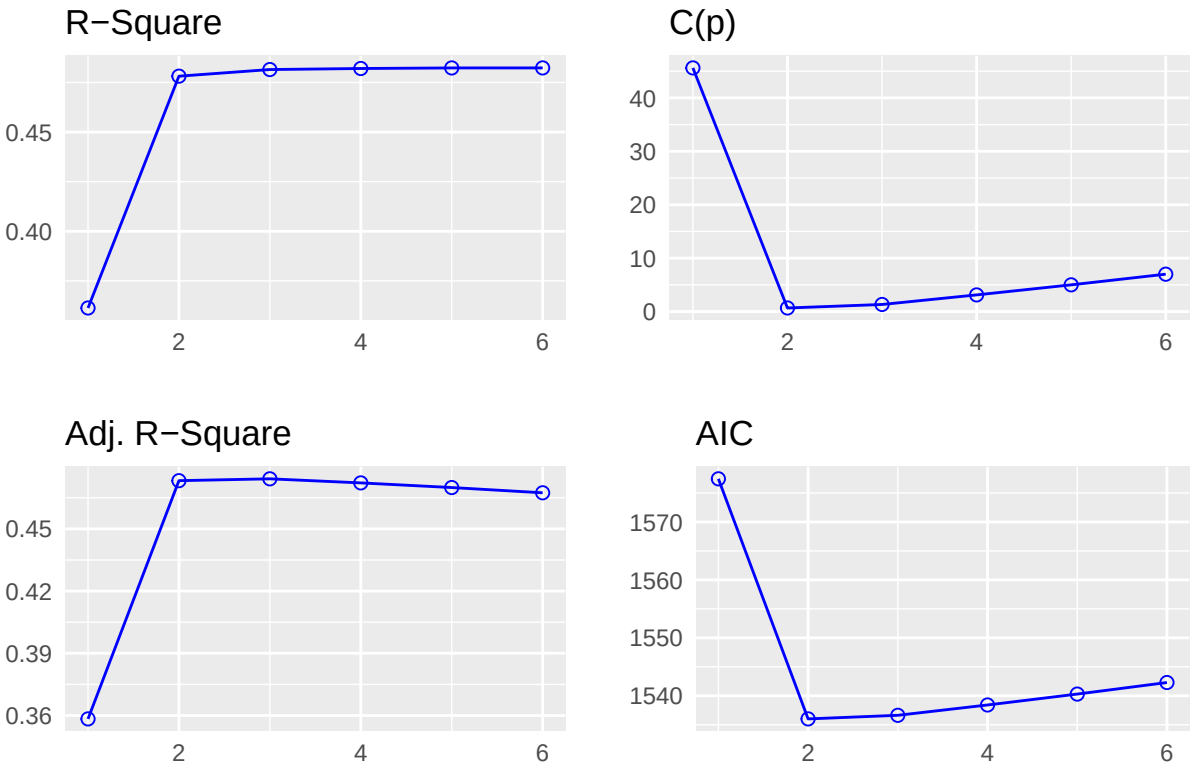
For help with `olsrr` see https://cran.r-project.org/web/packages/olsrr/vignettes/intro.html

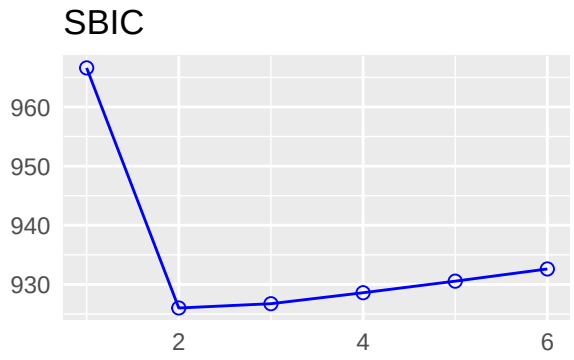
Best subsets
Examines all possible models and provides R2 and information criteria to select best model
For other options, see https://cran.r-project.org/web/packages/olsrr/vignettes/variable_selection.html

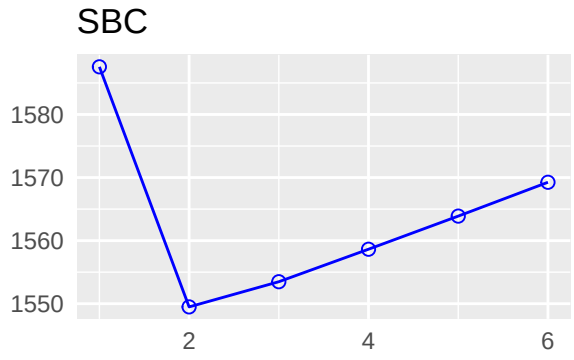
```
M1_best <- olsrr::ols_step_best_subset(model1)
M1_best
```

```
##                Best Subsets Regression
## ---------------------------------------------------
## Model Index    Predictors
## ---------------------------------------------------
##      1         Nmin
##      2         pH Nmin
##      3         pH Nmin WD.F
##      4         pH Nmin WD.F SLA.F
##      5         Plant_Height_m pH Nmin WD.F SLA.F
##      6         Plant_Height_m pH Nmin WD.F SPI.F SLA.F
## ---------------------------------------------------
##
##                                                    Subsets Regression Summary
## ------------------------------------------------------------------------------------------------------------
##                        Adj.        Pred
## Model    R-Square    R-Square    R-Square     C(p)         AIC         SBIC         SBC          MSE
## ------------------------------------------------------------------------------------------------------------
##   1        0.3614      0.3584      0.3467    45.6112    1577.4438    966.5991    1587.5558    18980.
##   2        0.4782      0.4733      0.4629     0.6717    1536.0098    926.0174    1549.4923    15581.
##   3        0.4815      0.4741      0.4612     1.3297    1536.6293    926.7398    1553.4824    15555.
##   4        0.4820      0.4722      0.4565     3.1187    1538.4114    928.5963    1558.6352    15614.
##   5        0.4823      0.4699      0.4508     5.0051    1540.2940    930.5517    1563.8885    15680.
##   6        0.4823      0.4674       0.446     7.0000    1542.2887    932.6140    1569.2538    15756.
## ------------------------------------------------------------------------------------------------------------
## AIC: Akaike Information Criteria
##  SBIC: Sawa's Bayesian Information Criteria
##  SBC: Schwarz Bayesian Criteria
##  MSEP: Estimated error of prediction, assuming multivariate normality
##  FPE: Final Prediction Error
##  HSP: Hocking's Sp
##  APC: Amemiya Prediction Criteria
```

```
plot(M1_best)
```


R−Square
0.45
0.40
2
4
6
C(p)
40
30
20
10
0
2
4
6
Adj. R−Square
0.45
0.42
0.39
0.36
2
4
6
AIC
1570
1560
1550
1540
2
4
6





Can include interaction terms in regression

```
model2 <- lm(Axis.1 ~ pH*Nmin, data = env_new)
M2_best <- olsrr::ols_step_best_subset(model2)
M2_best
```

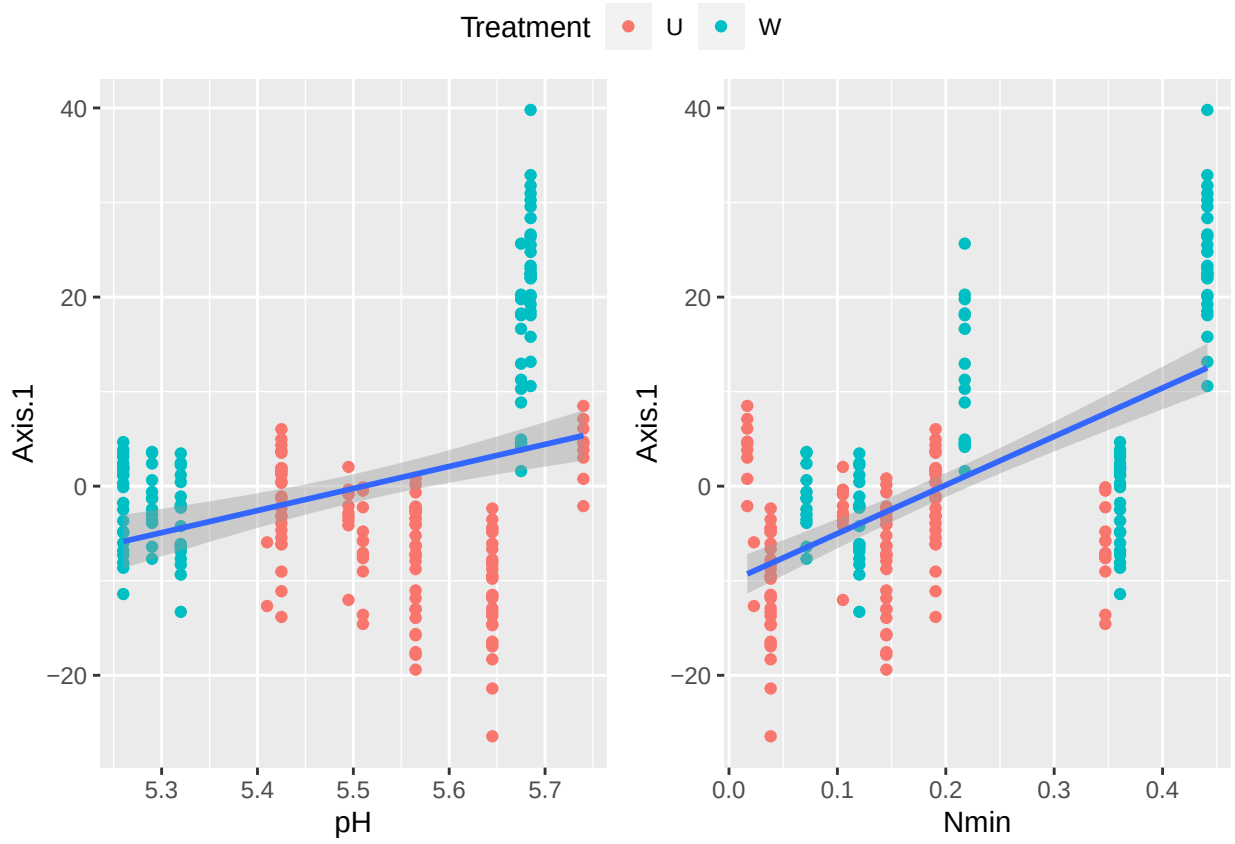
```
##    Best Subsets Regression
## ----------------------------
## Model Index    Predictors
## ----------------------------
##      1         pH:Nmin
##      2         Nmin pH:Nmin
##      3         pH Nmin pH:Nmin
## ----------------------------
##
##                                                    Subsets Regression Summary
## ---------------------------------------------------------------------------------------------------------
##                        Adj.        Pred
## Model    R-Square    R-Square    R-Square     C(p)         AIC          SBIC          SBC           MS
## ---------------------------------------------------------------------------------------------------------
##   1        0.3914      0.3885      0.3776    91.1863    1567.0973    -1116.0356    1577.2092    18088
##   2        0.5697      0.5656      0.5587     4.6546    1494.5604    -3322.0927    1508.0430    12849
##   3        0.5750      0.5690      0.5594     4.0000    1493.8724    -3339.2634    1510.7255    12750
## ---------------------------------------------------------------------------------------------------------
## AIC: Akaike Information Criteria
##  SBIC: Sawa's Bayesian Information Criteria
```

```
##  SBC: Schwarz Bayesian Criteria
##  MSEP: Estimated error of prediction, assuming multivariate normality
##  FPE: Final Prediction Error
##  HSP: Hocking's Sp
##  APC: Amemiya Prediction Criteria
```

```
# Note: can also include a specific interaction term as x1:x2
```

Graph main regression results

```
# plot PC1 vs. pH
p1 <- ggplot(env_new, aes(y=Axis.1, x=pH)) +
   geom_point(aes(colour=Treatment)) +
  geom_smooth(method="lm")

# plot PC1 vs. Nmin
p2 <- ggplot(env_new, aes(y=Axis.1, x=Nmin)) +
  geom_point(aes(colour=Treatment)) +
  geom_smooth(method="lm")

#arrange graphs
ggpubr::ggarrange(p1, p2,
          ncol=2, nrow = 1,
          common.legend=TRUE)
```



LASSO regression of OTUs on plant traits

Reduce OTU pool via differential analysis

Use DeSeq2 to identify taxa with differential "abundances" in the two main treatments

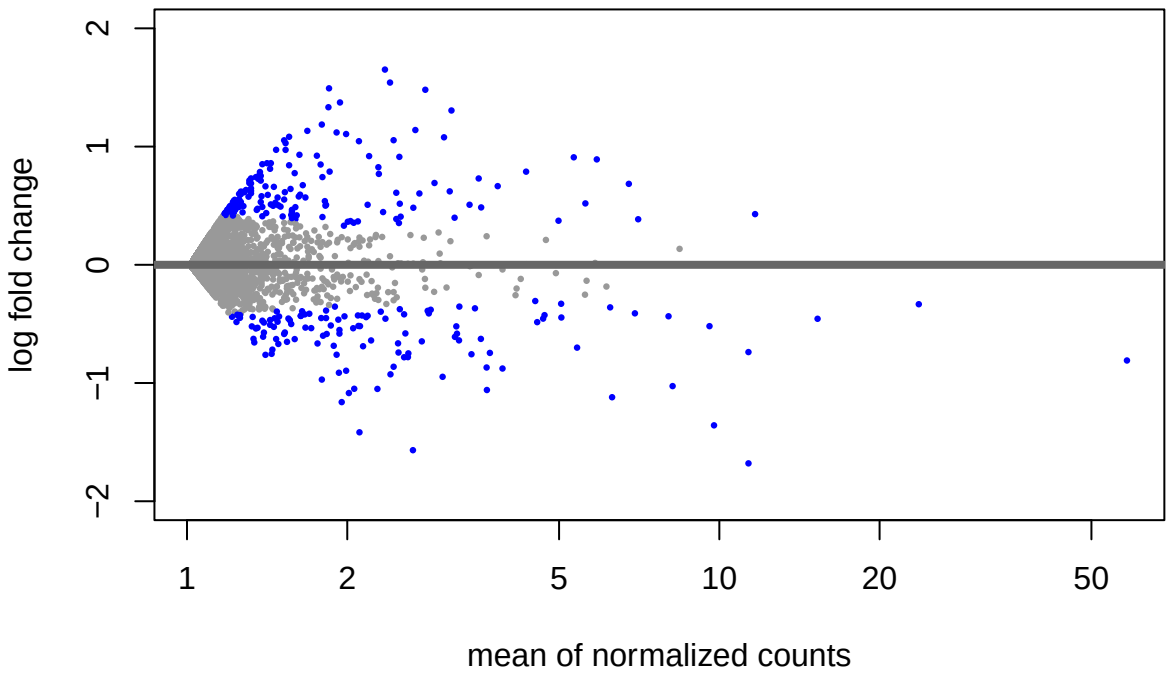
```
#assign otu table with pseudocount to remove zeros
otu <- otu_table(ps_bac)+1
env <- sample_data(ps_bac)
all(rownames(env) == colnames(otu))
```

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

```
otu_t <- t(otu) %>% as.data.frame()
all(rownames(env) == colnames(otu_t))
```

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

```
# for Deseq
dds <- DESeq2::DESeqDataSetFromMatrix(countData = round(otu_t), colData=env, design= ~ Treatment)
dds <- DESeq2::DESeq(dds)
# DESeq2::counts(dds)

# order and subset results by P value
res <- DESeq2::results(dds)
DESeq2::plotMA(res, ylim=c(-2,2)) # colored points are OTUs with P < 0.1
```




```
resOrdered <- res[order(res$pvalue),]
resSig <- subset(resOrdered, pvalue < 0.001)
resSig
```

```
## log2 fold change (MLE): Treatment W vs U
## Wald test p-value: Treatment W vs U
## DataFrame with 159 rows and 6 columns
##            baseMean log2FoldChange     lfcSE      stat      pvalue        padj
##           <numeric>      <numeric> <numeric> <numeric>   <numeric>   <numeric>
## OTU_4      11.34273       -1.67956  0.125869 -13.34367 1.28959e-40 1.61843e-37
## OTU_64      2.35414        1.65132  0.167341   9.86799 5.72987e-23 3.59549e-20
## OTU_5       9.77258       -1.35799  0.139091  -9.76335 1.61717e-22 6.76517e-20
## OTU_1928    2.65713       -1.56821  0.163273  -9.60484 7.62747e-22 2.39312e-19
## OTU_37      3.13894        1.30530  0.140246   9.30721 1.31231e-20 3.29391e-18
## ...             ...            ...       ...       ...         ...         ...
## OTU_223     3.18154       0.398261  0.119209   3.34085 0.000835212  0.00676252
## OTU_359     1.82011       0.500183  0.150871   3.31531 0.000915421  0.00736444
## OTU_123     4.66640      -0.455294  0.138122  -3.29631 0.000979628  0.00778361
## OTU_15324   1.66899      -0.531353  0.161200  -3.29623 0.000979929  0.00778361
## OTU_314     1.30477       0.576163  0.175044   3.29153 0.000996445  0.00786502
```

Subset data for LASSO based on DeSeq2 results

```
# subset ps_vst by diff exp taxa
names <- rownames(resSig)

ps_vst
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 6758 taxa and 215 samples ]
## sample_data() Sample Data:       [ 215 samples by 40 sample variables ]
## tax_table()   Taxonomy Table:    [ 6758 taxa by 7 taxonomic ranks ]
```

```
allTaxa <- taxa_names(ps_vst)
allTaxa <- allTaxa[(allTaxa %in% names)]
ps_sig <- prune_taxa(allTaxa, ps_vst)
ps_sig
```

```
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 159 taxa and 215 samples ]
## sample_data() Sample Data:       [ 215 samples by 40 sample variables ]
## tax_table()   Taxonomy Table:    [ 159 taxa by 7 taxonomic ranks ]
```

Prepare subsetted data for LASSO

Retrieve env and otu data, autoscale otus, then merge

```
# dependent vars are in the env matrix
# independent vars are in the otu matrix

# env
env_sig <- sample_data(ps_sig)
colnames(env_sig)
```

```
##  [1] "GardenID"           "Garden.Location"    "Number"
##  [4] "Treatment"          "June"               "July"
##  [7] "Aug"                "Mean"               "Nmin"
## [10] "NO3"                "NH4"                "pH"
## [13] "Spp"                "Ecology"            "Sample"
## [16] "Genotype"           "Caged.E..Not.Caged" "Plant_Height_m"
## [19] "Date_Sampled"       "extraction_date"    "Lat"
## [22] "Long"               "Plot"               "Dist1"
## [25] "Dist2"              "Dist3"              "order"
## [28] "TLP"                "WD"                 "SPI"
## [31] "LSV"                "RER"                "SLA"
## [34] "RGR"                "TLP.F"              "WD.F"
## [37] "SPI.F"              "SLA.F"              "Axis.1"
## [40] "Axis.2"
```

```
# otu - retrieve and autoscale (center by subtracting mean and scale by /sd)
otu_sig <- otu_table(ps_sig)
otu_sig_as <- scale(otu_sig, center=TRUE, scale=TRUE) %>% as.data.frame()

# check
all(rownames(env) == rownames(otu_sig_as))
```

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

```
# merge
ENVOTU <- merge(env_sig, otu_sig_as, by = "row.names", all=TRUE) %>% as.data.frame()
```

Create train and test data

```
set.seed(20220316)
train <- ENVOTU %>% dplyr::sample_frac(0.8) # use 80% of data for training and 20% for testing
test <- dplyr::anti_join(ENVOTU, train, by = "Row.names")
train <- column_to_rownames(train, "Row.names")
test <- column_to_rownames(test, "Row.names")

x_train <- train %>% dplyr::select(starts_with("OTU"))
x_train <- as.matrix(x_train)
x_test <- test %>% dplyr::select(starts_with("OTU"))
x_test <- as.matrix(x_test)
```

LASSO on Plant height (H)

Assign x and y for train and test data

```
y_train <- train$Plant_Height_m
y_test <- test$Plant_Height_m

anyNA(y_test)
```

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

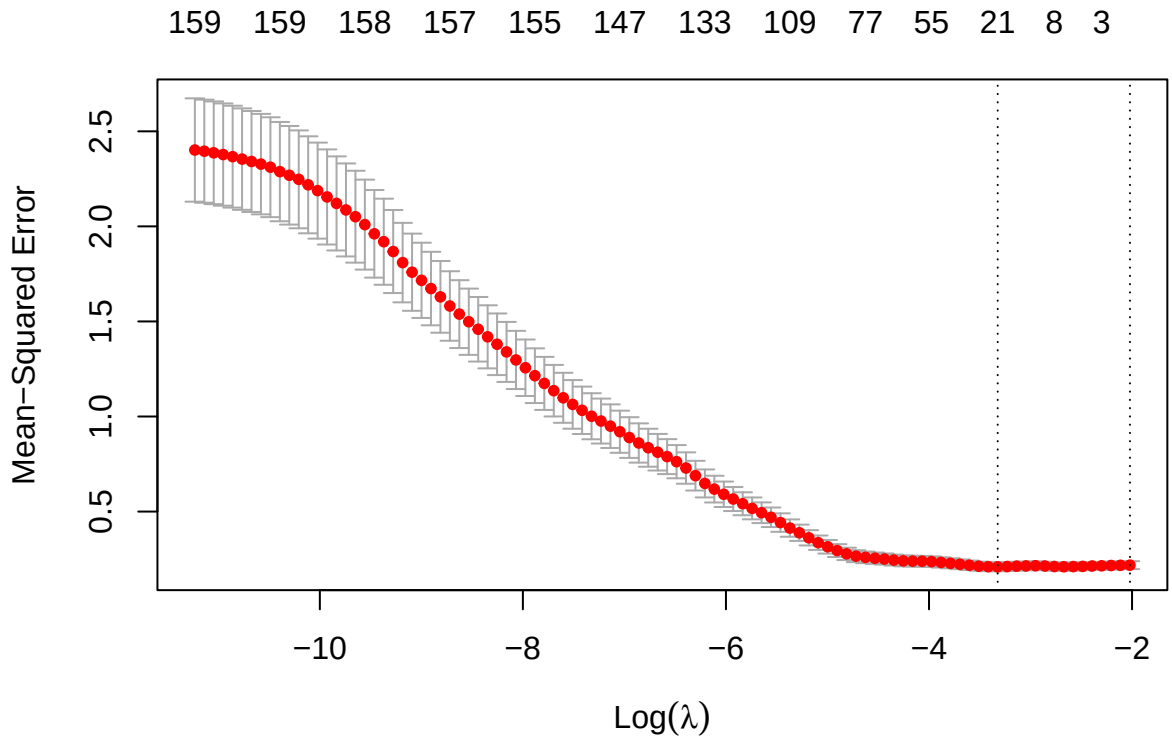
```
anyNA(y_train)
```

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

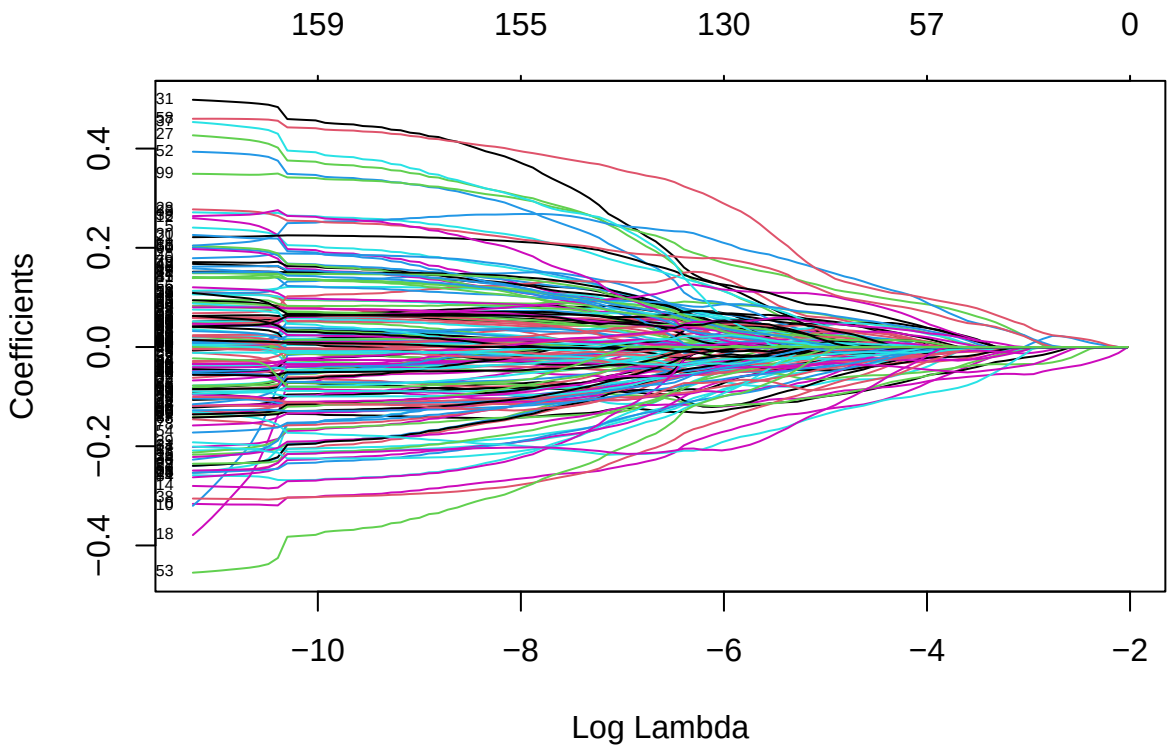
```
set.seed(20220317)
```

Use cross-validation (leave-one-out) to select lambda min

```
# can take a few minutes to run
cvfit.H <- glmnet::cv.glmnet(x_train, y_train,
                        alpha = 1, nfolds = nrow(x_train), grouped=FALSE)

# plot lambda vs. MSE to examine where error is minimized
plot(cvfit.H)
```



```
# plot coefficients for each OTU against lambda
plot(cvfit.H$glmnet.fit, xvar="lambda", label=TRUE)
```




```
# lambda minimum
cvfit.H$lambda.min
```

```
## [1] 0.03603819
```

```
lam_min <- cvfit.H$lambda.min
```

Retrieve model coefficients and fit parameters at lambda min

```
# model coefficients at lambda min
coef(cvfit.H, s=lam_min)
```

```
## 160 x 1 sparse Matrix of class "dgCMatrix"
##                          s1
## (Intercept)  1.0888593419
## OTU_22       .
## OTU_87      -0.0097569159
## OTU_37       .
## OTU_845      .
## OTU_188      .
## OTU_130     -0.0174419494
```

```
## OTU_441     -0.0302163968
## OTU_7        .
## OTU_9076     .
## OTU_2        .
## OTU_3        .
## OTU_1        .
## OTU_186      .
## OTU_292     -0.0051508779
## OTU_192      .
## OTU_224      .
## OTU_223      .
## OTU_21       .
## OTU_5444     .
## OTU_215      .
## OTU_81       .
## OTU_632      .
## OTU_5632     .
## OTU_9        .
## OTU_106      .
## OTU_312      .
## OTU_105      .
## OTU_494      .
## OTU_299     -0.0055865555
## OTU_568      .
## OTU_5        .
## OTU_4407     .
## OTU_184      .
## OTU_124      .
## OTU_25       .
## OTU_54      -0.0549623759
## OTU_10       .
## OTU_283      .
## OTU_1670    -0.0345473262
## OTU_4292     .
## OTU_27       .
## OTU_107      .
## OTU_242      .
## OTU_61      -0.0062323717
## OTU_954      .
## OTU_589      .
## OTU_151      .
## OTU_32       .
## OTU_12296    .
## OTU_30       .
## OTU_169      .
## OTU_154      .
## OTU_180      .
## OTU_213      .
## OTU_29       .
## OTU_100      .
## OTU_359      .
## OTU_23       .
## OTU_13769    .
## OTU_15324   -0.0130220956
```

```
## OTU_187      .
## OTU_8        .
## OTU_209     -0.0173102228
## OTU_195      .
## OTU_24       .
## OTU_9274     .
## OTU_93       .
## OTU_14100    .
## OTU_52       .
## OTU_41       .
## OTU_50       .
## OTU_828      .
## OTU_9471    -0.0081125343
## OTU_238      .
## OTU_64       .
## OTU_8024     .
## OTU_94       .
## OTU_4        .
## OTU_13       .
## OTU_95       .
## OTU_49       .
## OTU_260      .
## OTU_1370     .
## OTU_800     -0.0092949319
## OTU_170      .
## OTU_45       .
## OTU_314      .
## OTU_5807     .
## OTU_69       .
## OTU_11754   -0.0195371094
## OTU_60       .
## OTU_230      .
## OTU_254      .
## OTU_33       .
## OTU_440      .
## OTU_56       .
## OTU_18       .
## OTU_159      .
## OTU_263      0.0496997660
## OTU_74       .
## OTU_220      .
## OTU_1826     .
## OTU_96       .
## OTU_79       .
## OTU_5154     .
## OTU_285      .
## OTU_123     -0.0505618624
## OTU_78       .
## OTU_534      .
## OTU_39       .
## OTU_357      .
## OTU_117      .
## OTU_1928     .
## OTU_237      .
```

```
## OTU_2441     .
## OTU_55       .
## OTU_110      .
## OTU_26       0.0389445312
## OTU_11921    .
## OTU_34       .
## OTU_104      0.0008882859
## OTU_957      .
## OTU_91       .
## OTU_306      .
## OTU_477      .
## OTU_307      .
## OTU_160      .
## OTU_652      0.0404960349
## OTU_454      .
## OTU_119      .
## OTU_1218     .
## OTU_210     -0.0453856684
## OTU_836      .
## OTU_284      .
## OTU_167      .
## OTU_90       .
## OTU_278      .
## OTU_3372     .
## OTU_362      .
## OTU_11153    .
## OTU_86       .
## OTU_360      .
## OTU_162      .
## OTU_145      .
## OTU_1390     .
## OTU_279      .
## OTU_236      .
## OTU_290      .
## OTU_11241    .
## OTU_212      0.0085385052
## OTU_705      .
## OTU_443      .
## OTU_514      .
## OTU_12450    .
## OTU_348      .
## OTU_571      .
## OTU_3294     .
## OTU_538      0.0647348743
## OTU_217      .
```

```
# fit model with lambda minimum
best.H <- glmnet::glmnet(x_train, y_train, alpha = 1, lambda = lam_min)
best.H
```

```
##
## Call:  glmnet::glmnet(x = x_train, y = y_train, alpha = 1, lambda = lam_min)
##
##   Df  %Dev  Lambda
```

```
## 1 21 28.63 0.03604
```

```
# assess model fit on test data
assess.H <- glmnet::assess.glmnet(best.H, newx = x_test, newy = y_test)
assess.H
```

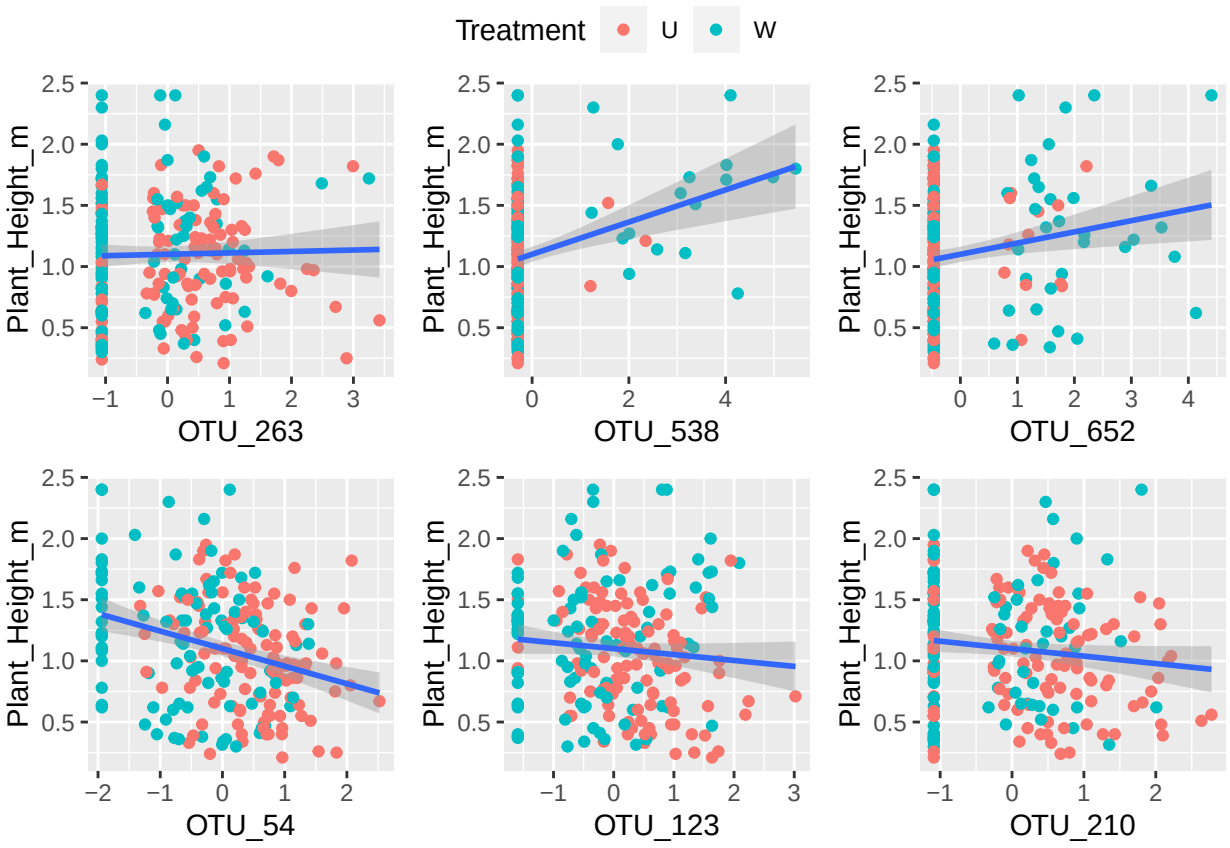
```
## $mse
##       s0
## 0.247926
## attr(,"measure")
## [1] "Mean-Squared Error"
##
## $mae
##        s0
## 0.4300754
## attr(,"measure")
## [1] "Mean Absolute Error"
```

```
# if you want to retrieve r2 values for any linear relationship use
lm54 <- lm(ENVOTU$Plant_Height_m ~ ENVOTU$OTU_54)
summary(lm54)$adj.r.squared
```

```
## [1] 0.08585157
```

Examine results: plot otu abundances vs. plant height

```
# plots for subset of otus with largest + and - coefficients
p263 <- ggplot2::ggplot(ENVOTU, aes(y=Plant_Height_m, x=OTU_263)) +
  geom_point(aes(colour=Treatment)) +
  geom_smooth(method="lm")

p538 <- ggplot2::ggplot(ENVOTU, aes(y=Plant_Height_m, x=OTU_538)) +
  geom_point(aes(colour=Treatment)) +
  geom_smooth(method="lm")

p652 <- ggplot2::ggplot(ENVOTU, aes(y=Plant_Height_m, x=OTU_652)) +
  geom_point(aes(colour=Treatment)) +
  geom_smooth(method="lm")

p54 <- ggplot2::ggplot(ENVOTU, aes(y=Plant_Height_m, x=OTU_54)) +
  geom_point(aes(colour=Treatment)) +
  geom_smooth(method="lm")

p123 <- ggplot2::ggplot(ENVOTU, aes(y=Plant_Height_m, x=OTU_123)) +
  geom_point(aes(colour=Treatment)) +
  geom_smooth(method="lm")

p210 <- ggplot2::ggplot(ENVOTU, aes(y=Plant_Height_m, x=OTU_210)) +
  geom_point(aes(colour=Treatment)) +
  geom_smooth(method="lm")

ggpubr::ggarrange(p263, p538, p652, p54, p123, p210, ncol=3, nrow=2, common.legend = TRUE)
```


Treatment
U
W
Plant_Height_m
OTU_263
OTU_538
OTU_652
OTU_54
OTU_123
OTU_210

Coding Exercises

1. Use the gls::nlme function to re-run the regression on PC Axis1 (as if homoscedasticity was violated)

- print the model info using the "summary(model)" command
- manual: https://cran.r-project.org/web/packages/nlme/nlme.pdf
- function: https://rdrr.io/cran/nlme/man/gls.html

2. Examine alpha diversity as function of soil and plant traits

- calculate observed richness (you know how!)
  - calculate on the untransformed data
  - note that if you use `phyloseq::estimate_richness` it may add an "X" prefix to your row names
  - replace row names before merging richness into the sample_data file
  - merge observed richness into the `env_new`
- define a model that includes plant and soil variables you think will be important
  - if needed, transform richness (for heteroskedastic or non-normal data)
  - if needed, remove highly collinear variables (or combine them into a single variable)
- select a regression approach and explain your choice
- plot and interpret results
- save ps_vst to use next week once you add richness to the sample data
  - add the updated `env_new` to `sample_data(ps_vst)`
  - use grid view in the environment pane, select ps_vst, and hit save icon
  - save as `wk11_data.Rdata`

3. Rerun the LASSO regression on a different plant trait

- change the P value cutoff (e.g., 0.05 or 0.0001) and subset the data
- select a new plant trait (see `Wk11_README.txt`)
  - use `tidyr::drop_na` to remove rows with NA in your chosen variable
- rerun `glmnet::lasso`
- plot top results
- explain what you found

Session Info

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats4    stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] ggpubr_0.4.0               glmnet_4.1-3
##  [3] Matrix_1.4-0               corrplot_0.92
##  [5] olsrr_0.5.3                vegan_2.5-7
##  [7] lattice_0.20-45            permute_0.9-7
##  [9] DESeq2_1.34.0              SummarizedExperiment_1.24.0
## [11] Biobase_2.54.0             MatrixGenerics_1.6.0
## [13] matrixStats_0.61.0         GenomicRanges_1.46.1
## [15] GenomeInfoDb_1.30.1        IRanges_2.28.0
## [17] S4Vectors_0.32.3           BiocGenerics_0.40.0
## [19] phyloseq_1.38.0            forcats_0.5.1
## [21] stringr_1.4.0              dplyr_1.0.8
## [23] purrr_0.3.4                readr_2.1.2
## [25] tidyr_1.2.0                tibble_3.1.6
## [27] ggplot2_3.3.5              tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##   [1] colorspace_2.0-3      ggsignif_0.6.3        ellipsis_0.3.2
##   [4] XVector_0.34.0        fs_1.5.2              rstudioapi_0.13
##   [7] farver_2.1.0          bit64_4.0.5           AnnotationDbi_1.56.2
##  [10] fansi_1.0.2           lubridate_1.8.0       xml2_1.3.3
##  [13] codetools_0.2-18      splines_4.1.2         cachem_1.0.6
##  [16] geneplotter_1.72.0    knitr_1.37            ade4_1.7-18
##  [19] jsonlite_1.8.0        broom_0.7.12          annotate_1.72.0
##  [22] cluster_2.1.2         dbplyr_2.1.1          png_0.1-7
##  [25] compiler_4.1.2        httr_1.4.2            backports_1.4.1
##  [28] assertthat_0.2.1      fastmap_1.1.0         cli_3.2.0
##  [31] htmltools_0.5.2       tools_4.1.2           igraph_1.2.11
##  [34] gtable_0.3.0          glue_1.6.2            GenomeInfoDbData_1.2.7
##  [37] reshape2_1.4.4        Rcpp_1.0.8.3          carData_3.0-5
##  [40] cellranger_1.1.0      vctrs_0.3.8           Biostrings_2.62.0
```

```
##  [43] rhdf5filters_1.6.0     multtest_2.50.0        ape_5.6-2
##  [46] nlme_3.1-155           iterators_1.0.14       xfun_0.29
##  [49] rvest_1.0.2            lifecycle_1.0.1        rstatix_0.7.0
##  [52] goftest_1.2-3          XML_3.99-0.9           zlibbioc_1.40.0
##  [55] MASS_7.3-54            scales_1.1.1           hms_1.1.1
##  [58] parallel_4.1.2         biomformat_1.22.0      rhdf5_2.38.0
##  [61] RColorBrewer_1.1-2     yaml_2.3.5             gridExtra_2.3
##  [64] memoise_2.0.1          stringi_1.7.6          RSQLite_2.2.10
##  [67] highr_0.9              genefilter_1.76.0      nortest_1.0-4
##  [70] foreach_1.5.2          BiocParallel_1.28.3    shape_1.4.6
##  [73] rlang_1.0.2            pkgconfig_2.0.3        bitops_1.0-7
##  [76] evaluate_0.15          Rhdf5lib_1.16.0        labeling_0.4.2
##  [79] cowplot_1.1.1          bit_4.0.4              tidyselect_1.1.2
##  [82] plyr_1.8.6             magrittr_2.0.2         R6_2.5.1
##  [85] generics_0.1.2         DelayedArray_0.20.0    DBI_1.1.2
##  [88] pillar_1.7.0           haven_2.4.3            withr_2.5.0
##  [91] mgcv_1.8-39            abind_1.4-5            KEGGREST_1.34.0
##  [94] survival_3.2-13        RCurl_1.98-1.6         car_3.0-12
##  [97] modelr_0.1.8           crayon_1.5.0           utf8_1.2.2
## [100] tzdb_0.2.0             rmarkdown_2.11         locfit_1.5-9.4
## [103] grid_4.1.2             readxl_1.3.1           data.table_1.14.2
## [106] blob_1.2.2             reprex_2.0.1           digest_0.6.29
## [109] xtable_1.8-4           munsell_0.5.0
```