warmXtrophic Project: Plant Composition Diversity Data Analyses

Moriah Young, Pat Bills

January 12, 2022

Load in packages & data

```
# Clear all existing data
rm(list = ls())

# Load packages
library(tidyverse)
library(ggplot2)
library(lme4)
library(olsrr)
library(predictmeans)
library(car)
library(fitdistrplus)
library(ggpubr)
library(rstatix)
library(vegan)
library(interactions)
library(sjPlot)
library(effects)
library(glmmTMB)
library(labdsv)  # used with Vegan package, the matrify() and matrify2() functions
library(agricolae)  # HSD.test() function
library(bbmle)
library(jtools)  # summ() function

# Set working directory
Sys.getenv("L1DIR")
```

```
## [1] "/Volumes/GoogleDrive/Shared drives/SpaCE_Lab_warmXtrophic/data/L1"
```

```
L0_dir <- Sys.getenv("L0DIR")
L1_dir <- Sys.getenv("L1DIR")
L2_dir <- Sys.getenv("L2DIR")
list.files(L1_dir)
```

```
##  [1] "ANPP"              "climate_data"      "CN"
##  [4] "Greenness"         "herbivory"         "HOBO_data"
##  [7] "PAR"               "phenology"         "plant_composition"
## [10] "SLA"
```

```r
# read in plant comp data
comp <- read.csv(file.path(L1_dir, "plant_composition/final_plantcomp_L1.csv"))
comp <- comp %>% select(-X)  # get rid of 'X' column that shows up

# read in meta data
meta <- read.csv(file.path(L0_dir, "plot.csv"))  # dataframe above already has meta data in it
```

```r
# adding sequential year variable starting at 1: this is because the years (e.g.
# 2015, 2016, etc) are large numbers compared with other values in the dataset.
# We can always label axes with these real years.
comp$year_factor[comp$year == 2015] <- 1
comp$year_factor[comp$year == 2016] <- 2
comp$year_factor[comp$year == 2017] <- 3
comp$year_factor[comp$year == 2018] <- 4
comp$year_factor[comp$year == 2019] <- 5
comp$year_factor[comp$year == 2020] <- 6
comp$year_factor[comp$year == 2021] <- 7

# Remove non-plant data
comp <- comp[!(comp$species == "Bare_Ground" | comp$species == "Unknown" | comp$species ==
    "Brown" | comp$species == "Litter" | comp$species == "Vert_Litter" | comp$species ==
    "Animal_Disturbance"), ]
```

```r
# Function to get data in wide format to work in Vegan package - taken from link
# below
# https://stackoverflow.com/questions/50691393/transform-community-data-into-wide-format-for-vegan-pack

matrify2 <- function(data) {
    # Data must have columns: plot, SPEC, abundance measure,Year
    if (ncol(data) != 4)
        stop("data frame must have four column format")
    plt <- factor(data[, 1])
    spc <- factor(data[, 2])
    abu <- data[, 3]
    yrs <- factor(data[, 4])
    plt.codes <- sort(levels(factor(plt)))  ##object with sorted plot numbers
    spc.codes <- levels(factor(spc))  ##object with sorted SPEC names
    yrs.codes <- sort(levels(factor(yrs)))  ##object with sorted sampling Years
    taxa <- matrix(0, nrow = length(plt.codes) * length(yrs.codes), ncol = length(spc.codes))  ##Create
    plt.list <- rep(plt.codes, length(yrs.codes))  ##Create a list of all the plot numbers (in order of
    yrs.list <- rep(yrs.codes, each = length(plt.codes))  ##Create a list of all the Year numbers (in o
    col <- match(spc, spc.codes)  ##object that determines the alphabetical order ranking of each SPEC
    row.plt <- match(plt, plt.codes)  ##object that determines the rank order ranking of each plot of t
    row.yrs <- match(yrs, yrs.codes)  ##object that determines the rank order ranking of each Year of t
    for (i in 1:length(abu)) {
        row <- (row.plt[i]) + length(plt.codes) * (row.yrs[i] - 1)  ##Determine row number by assuming
        if (!is.na(abu[i])) {
            ## ONly use value if !is.na .. [ignore all is.NA values]
            taxa[row, col[i]] <- sum(taxa[row, col[i]], abu[i])  ##Add abundance measure of row i to th
        }
    }
    taxa <- data.frame(taxa)  ##Convert to data.frame for easier manipulation
    taxa <- cbind(yrs.list, plt.list, taxa)  ##Add ID columns for plot and Year to each row already rep
```

```
    names(taxa) <- c("Year", "Plot", spc.codes)
    taxa
}
```

```
# diversity_by_year <- function(comp, site, div_index = 'shannon'){
comp_kbs <- subset(comp, site == "kbs") %>% dplyr::select(plot, species, cover, year)
comp_umbs <- subset(comp, site == "umbs") %>% dplyr::select(plot, species, cover,
    year)


# convert the abundance (cover) data to wide format for each species in columns
# for the vegan package
comp_kbs$cover <- as.numeric(comp_kbs$cover)
comp_wide_kbs <- matrify2(comp_kbs)
comp_umbs$cover <- as.numeric(comp_umbs$cover)
```

```
## Warning: NAs introduced by coercion
```

```
comp_wide_umbs <- matrify2(comp_umbs)

# comp_wide_data is assumes to have columns Year, Plot, and columns for each
# species found, e.g. for Vegan

# first, split up the wide data into a list of years.  Each list item is a year
# of data
comp_wide_by_year_kbs <- dplyr::group_by(comp_wide_kbs, Year) %>% dplyr::group_split()
comp_wide_by_year_umbs <- dplyr::group_by(comp_wide_umbs, Year) %>% dplyr::group_split()

# we need to add plot names.  Get those Plot names by taking a column from any
# one of the years since we are assuming the Plot column is the exact same across
# years and IN THE SAME ORDER
plot_names <- comp_wide_by_year_kbs[[1]]$Plot
plot_names <- comp_wide_by_year_umbs[[1]]$Plot

# remove the plot and year columns from each item in the list so that Vegan will
# work.  This assumes row order is the exact same for all years (each row a plot)
comp_wide_by_year_kbs <- lapply(comp_wide_by_year_kbs, dplyr::select, c(-Year, -Plot))
comp_wide_by_year_umbs <- lapply(comp_wide_by_year_umbs, dplyr::select, c(-Year,
    -Plot))

# apply the diversity function to each year - in this case the main index is
# plot, each year considered separately
diversity_by_year_list_kbs <- lapply(comp_wide_by_year_kbs, vegan::diversity, index = "shannon")
diversity_by_year_list_umbs <- lapply(comp_wide_by_year_umbs, vegan::diversity, index = "shannon")

# each item in the list is a year of diversity, so name those with the years we
# know we have
names(diversity_by_year_list_kbs) <- as.character(2015:2021)
names(diversity_by_year_list_umbs) <- as.character(2015:2021)

# 'unlist' and create a new data frame, each year a column, each row a plot, and
# add a new row with the plot names
x_kbs <- do.call(cbind, diversity_by_year_list_kbs) %>% cbind(Plot = plot_names) %>%
```

```
    as.data.frame()
x_umbs <- do.call(cbind, diversity_by_year_list_umbs) %>% cbind(Plot = plot_names) %>%
    as.data.frame()
# an alternative tidyverse way x<- diversity_by_year(diversity_by_year_list)

## optional step!
x_kbs
```

```
##                2015             2016              2017             2018
## 1  1.74982308774477 1.89484825550382  1.46457885369235 1.79022612402249
## 2  1.90662337874857 1.84725215309623  1.13716030470414 1.36198444330393
## 3  1.98749299359934 1.61377593556905  1.71072749310064 1.67233412309495
## 4  1.76205160229741 1.77517711389533  1.32874743973109 1.03860433128686
## 5  1.75210718809553 1.66227443462069  1.69192914802754 1.90695858866473
## 6   1.6900850428406 1.43660622202068  1.39873211536317  1.8571841570528
## 7  1.49086543230336 1.47097168973964  1.21337363817196 1.28050329701639
## 8  1.80821328729531  1.9254070243331  1.14986934861213  1.2616299180373
## 9   2.1087845075131 1.80236101080512  1.56273408513826 1.83756938558579
## 10 1.98196568606134 1.90616309802705  1.91936048574866 1.96655841609062
## 11 1.84296460180767  1.8965200697777  1.25355451781611 1.26874311894322
## 12 1.76762986966693 1.34098707770158  1.28659495811355 1.56199705257429
## 13 1.43883353670948 1.44641730798802  1.26464841035831 1.49328461093835
## 14 1.75673653952927  1.6273811911906  1.59862659873349 1.57746303850457
## 15 1.93644517652882  1.9750616592266  1.60757066797302 1.53493741382157
## 16 1.58409303017981 1.79057866130084  1.84017946573311 1.96848518019582
## 17 1.60506299754593 1.91996670439262  1.83507553957834 1.61808366536974
## 18  1.4247004760432 1.27334741927488 0.988243173382032 1.46588240243596
## 19 1.72885447960798 1.50003663417278   0.7586338108316 1.21791312632341
## 20 2.14484946283764 1.92315942970564  1.01772685103368 1.68252998651447
## 21 2.10055052731924 1.81252205715579  1.57077556615872 1.86767543214417
## 22 1.89362827233848 1.86514077199398  2.13631841372597 2.17843467691095
## 23 1.73593769273967 2.12071598624058   1.7062450650214  1.9415443800649
## 24 2.03965289235968 1.56767987740433  1.60740329659913 1.52284566334068
##                2019             2020              2021 Plot
## 1   1.63141828218805  1.40619117164602 0.780282286358405   A1
## 2   1.03501044156779  1.30464181679929 0.761966098342232   A2
## 3   1.40564143802668  1.52023377614068  1.46818923776404   A3
## 4  0.807186457770002  1.10403362311013 0.522975735072191   A4
## 5   1.29530398546089  1.47867329613074  1.12613740316083   A5
## 6   1.56587400893309    1.776898556231  1.59272035502521   A6
## 7  0.966220584181466 0.903368616352641 0.787741797532107   B1
## 8    0.6001886754368  1.32028023954065 0.804089503088995   B2
## 9   1.55079429713458  1.75134927146181  1.50101684634471   B3
## 10   1.9076117928339  1.85066103391479  1.82018142958166   B4
## 11  1.07712923504737  1.51478024391315  1.63669387518496   B5
## 12  1.70747374757789  1.35162942534084  1.17907511874716   B6
## 13  1.46913677234012  1.55507473988238  1.50836959659867   C1
## 14  1.27508655116576  1.55732962234086 0.950411775293582   C2
## 15 0.883992796139038  1.72599963232697 0.996523451248442   C3
## 16  1.59775101771005  1.56782535025894  1.25369833009911   C4
## 17  1.72425066783732  1.70780555305176  1.47888650241859   C5
## 18  1.59762443325548  1.56141444142399  1.48024946913317   C6
## 19  1.25325132142148  1.78617517426128  1.13570812589948   D1
```

```
## 20  1.34286442427345  1.61533909948124  1.36264400396897   D2
## 21  1.94852371633076   1.7075032751265  1.86430302721765   D3
## 22  2.27772535378833  2.07864562723304   2.3134678596367   D4
## 23  1.98707494159271  2.36198520777061                 0   D5
## 24  1.02058088927649  1.28140861968835  1.14514426526423   D6
```

```
x_umbs
```

```
##                2015              2016              2017              2018
## 1   1.79750600753516  1.09884956681657  1.65427009762925  1.72338781506085
## 2   1.03103412684858   1.0510661759467  1.01847657461102 0.926553867333551
## 3   1.23108760738823 0.832986567666689  1.05822294906718   1.1308760877669
## 4   1.11379164827808  1.09649945437462  1.58403466753454  1.26981056014349
## 5   1.41499122544575  1.54066354734953  1.75670820671366  1.83315130300093
## 6    1.3233120828513  1.11575480079886 0.921037312752463  1.22171569425616
## 7   1.08343195035589  1.03279159363802 0.876057255954377  1.24419243607285
## 8   1.09147272846172  1.24329181475844  1.29855975285605  1.23947894561033
## 9    0.9085145936062 0.959769287477707 0.649404154675513  0.83871658655578
## 10    1.318924658555 0.450107654438585   1.5725829367254  1.57921265075159
## 11  1.26789311388011 0.770745734585134  1.18646068972002 0.773638703559191
## 12  1.30080601995319  1.20839564977282   1.4044265373131  1.38328133173653
## 13  1.36610720736298 0.842368707930785  1.26913274303141  1.11918161689174
## 14  1.37712670393695  1.50441888100153  1.69244837402528  1.45297259997684
## 15  1.50594110744263   1.4028839310817  1.48760307705478  1.44909606234782
## 16  0.80322481868332  0.83389395416706  1.14271822258295  1.51638798771447
## 17 0.627025432024711 0.606005414459172  1.16938338399709  1.09757402522846
## 18 0.973444495419564  1.35315156462733   1.1423956505593  1.23956352450531
## 19  1.07986059255333 0.879465952690604  1.24184224778392  1.31578039164066
## 20  1.44773174456628  1.00905352578159   1.4126219798077  1.40929473472627
## 21  1.69964939346275  1.46688269951968  1.45944936854917  1.75936201800645
## 22 0.486356026733479 0.588197569441821  1.56011268630929  1.49990768246823
## 23  1.03086631184147  1.07092560439311 0.907225455933764  1.10551205477812
## 24  1.21191539524654   1.1042823781343  1.21986566532071  1.35947699709629
##                2019              2020              2021 Plot
## 1    2.0145918118242  1.96581039743491  1.62531304044023   A1
## 2  0.691995913183193 0.974169862064642   1.2163860305222   A2
## 3    0.9724493832543  1.06768390565336 0.627765100792579   A3
## 4  0.915646292742865  1.33900551463997  1.50133902689179   A4
## 5   1.94764990159352  1.89835885910774  1.44706286625285   A5
## 6   1.45054061735534  1.37183773873638   0.9114708169835   A6
## 7   1.43250053727084  1.60608516148779  1.81536751087183   B1
## 8   1.11587149404342  1.18251733489018 0.699678505406374   B2
## 9  0.857420615656224 0.863774981807558 0.845161671782528   B3
## 10  1.50784697421321  1.49837008041475  1.26916283332234   B4
## 11 0.705571885627956 0.785670252458899 0.749093258816757   B5
## 12  1.51929284719405  1.26323616766899  1.48810325489543   B6
## 13  1.22897033192299  1.42548312428012  1.50648093715665   C1
## 14  1.55447238564657  1.84837077356116  1.53050384901037   C2
## 15  1.47245049389592  1.58772129323631  1.41124646634374   C3
## 16  1.66601534933355  1.58346400146614   1.6069039235169   C4
## 17  1.36933451043502  1.19760885543397  1.14167338748084   C5
## 18    1.430726166755   1.4500169349134  1.43036518013435   C6
## 19  1.47812095926711  1.58341388787888  1.48823193587079   D1
## 20   1.5069193093977  1.56477969632959   1.6541529758687   D2
```

```
## 21  1.86897728881615  1.68447529694085  1.35856632476605   D3
## 22  1.53228686591738  1.61076832230645  0.99499394631327   D4
## 23 0.830686156562969   1.0913363381619 0.773326936854788   D5
## 24  1.20925855258584  1.57780388206066  1.32789605853908   D6
```

```
# comp$cover <- as.numeric(comp$cover)

# this output has a column for each year 2015, 2016, and Plot, but if you need it
# narrow use 'melt' from reshape2:
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

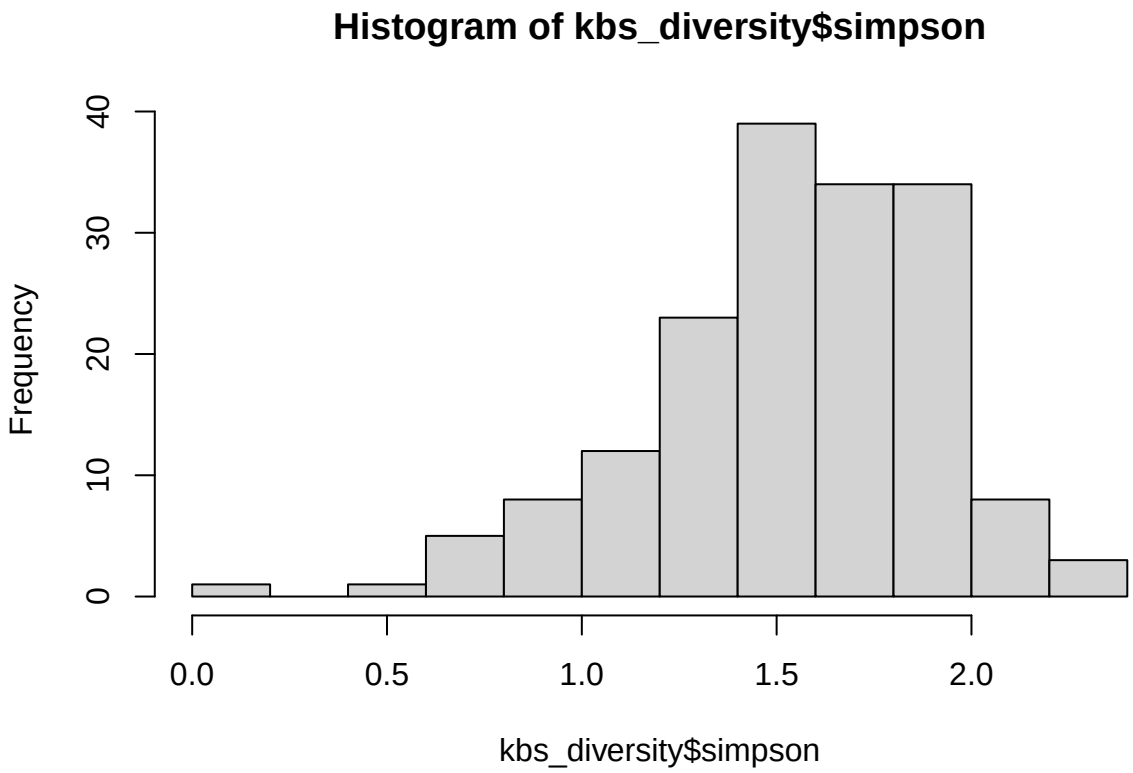
```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
# calculate shannon diversity
shannon_by_plot_year_kbs <- reshape2::melt(x_kbs, id = "Plot", variable.name = c("Year"),
    value.name = "shannon")
shannon_by_plot_year_kbs$site <- "kbs"  # adding site column
shannon_by_plot_year_umbs <- reshape2::melt(x_umbs, id = "Plot", variable.name = c("Year"),
    value.name = "shannon")
shannon_by_plot_year_umbs$site <- "umbs"  # adding site column

# calculate simpson diversity
simpson_by_plot_year_kbs <- reshape2::melt(x_kbs, id = "Plot", variable.name = c("Year"),
    value.name = "simpson")
simpson_by_plot_year_kbs$site <- "kbs"  # adding site column
simpson_by_plot_year_umbs <- reshape2::melt(x_umbs, id = "Plot", variable.name = c("Year"),
    value.name = "simpson")
simpson_by_plot_year_umbs$site <- "umbs"  # adding site column

# combine umbs and kbs shannon diversity measures into 1 dataframe
shannon_diversity <- full_join(shannon_by_plot_year_kbs, shannon_by_plot_year_umbs,
    by = c("Plot", "Year", "shannon", "site"))

# combine umbs and kbs simpson diversity measures into 1 dataframe
simpson_diversity <- full_join(simpson_by_plot_year_kbs, simpson_by_plot_year_umbs,
    by = c("Plot", "Year", "simpson", "site"))

# combine simpson and shannon diversity data frames into 1
comp_diversity <- full_join(simpson_diversity, shannon_diversity, by = c("Plot",
    "Year", "site"))
# Looks like diversity and simpson diveristy measures are the same?? Need to look
# into this

names(comp_diversity) <- tolower(names(comp_diversity))  # column names to lower case so I can combine

# merge meta data with comp_diversity
comp_diversity <- full_join(comp_diversity, meta, by = "plot")
```

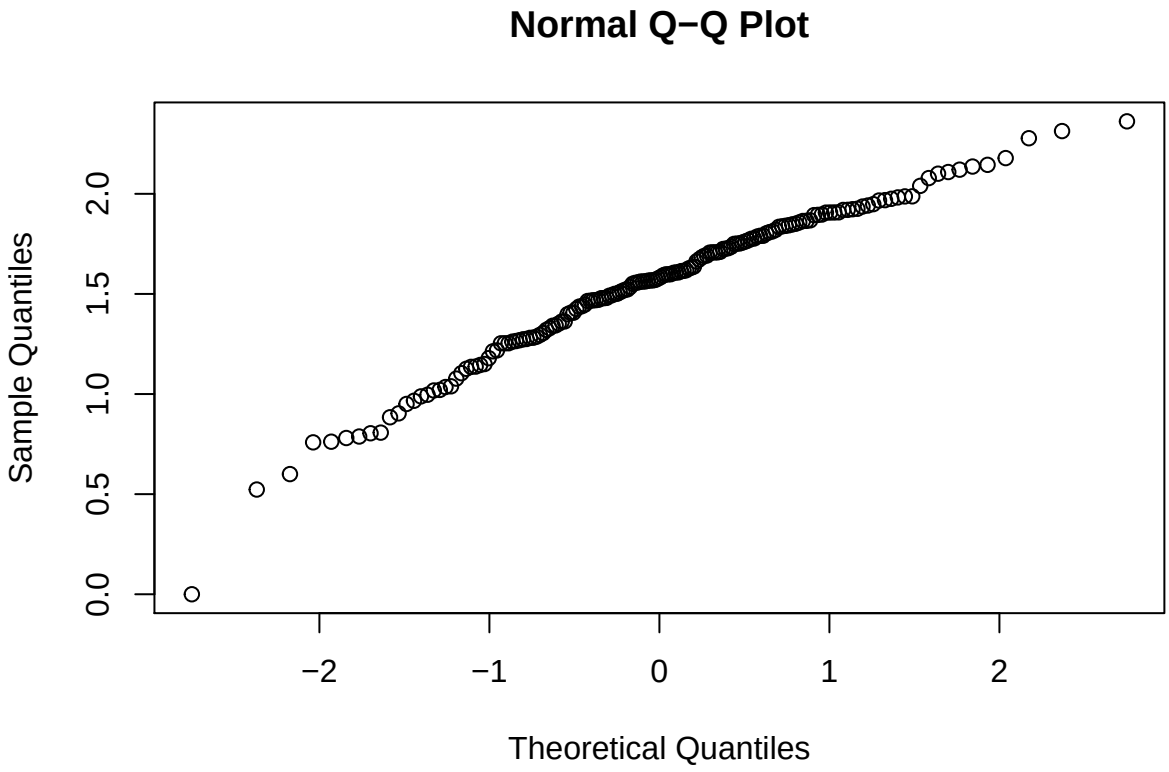
```
comp_diversity$simpson <- as.numeric(comp_diversity$simpson)
comp_diversity$shannon <- as.numeric(comp_diversity$shannon)

# write a new csv with diversity indices and upload to the shared google drive L2
# data folder
write.csv(comp_diversity, file.path(L2_dir, "plant_composition/final_plant_comp_diversity_L2.csv"))

# create separate data frames for kbs and umbs
kbs_diversity <- subset(comp_diversity, site == "kbs")
umbs_diversity <- subset(comp_diversity, site == "umbs")
```

KBS Simpson's Index

```
# Data exploration
hist(kbs_diversity$simpson)
```



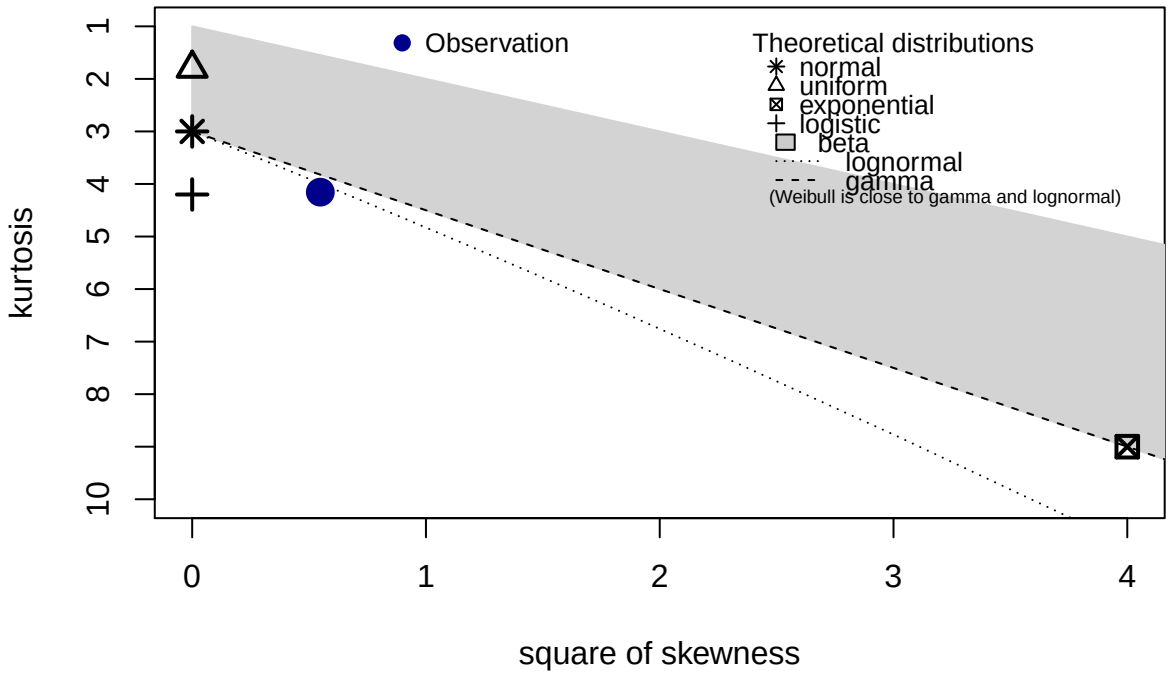

```
qqnorm(kbs_diversity$simpson)  # this looks good
```




```
shapiro.test(kbs_diversity$simpson)  # not normal
```

```
##
##  Shapiro-Wilk normality test
##
## data:  kbs_diversity$simpson
## W = 0.96985, p-value = 0.001019
```

```
# Exploring distributions for these slightly left-skewed data:
descdist(kbs_diversity$simpson, discrete = FALSE)  # log normal looks like a good fit
```

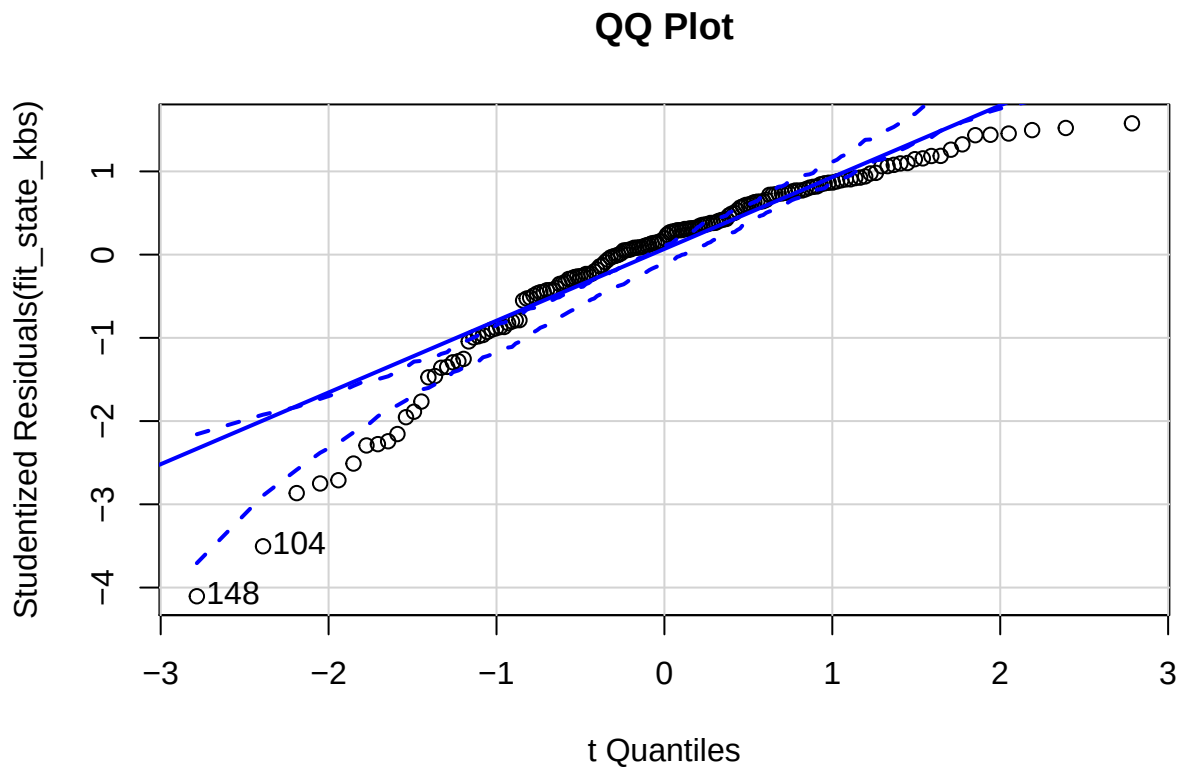



```
## summary statistics
## ------
## min:  0   max:  2.361985
## median:  1.580778
## mean:  1.546083
## estimated sd:  0.376504
## estimated skewness:  -0.7403559
## estimated kurtosis:  4.15561
```

```
# KBS State-only model
kbs_diversity <- kbs_diversity[-167, ]  # remove row with zero - won't be able to take the log of the s
fit_state_kbs <- lm(log(simpson) ~ state, data = kbs_diversity)
outlierTest(fit_state_kbs)  # yes row 148
```

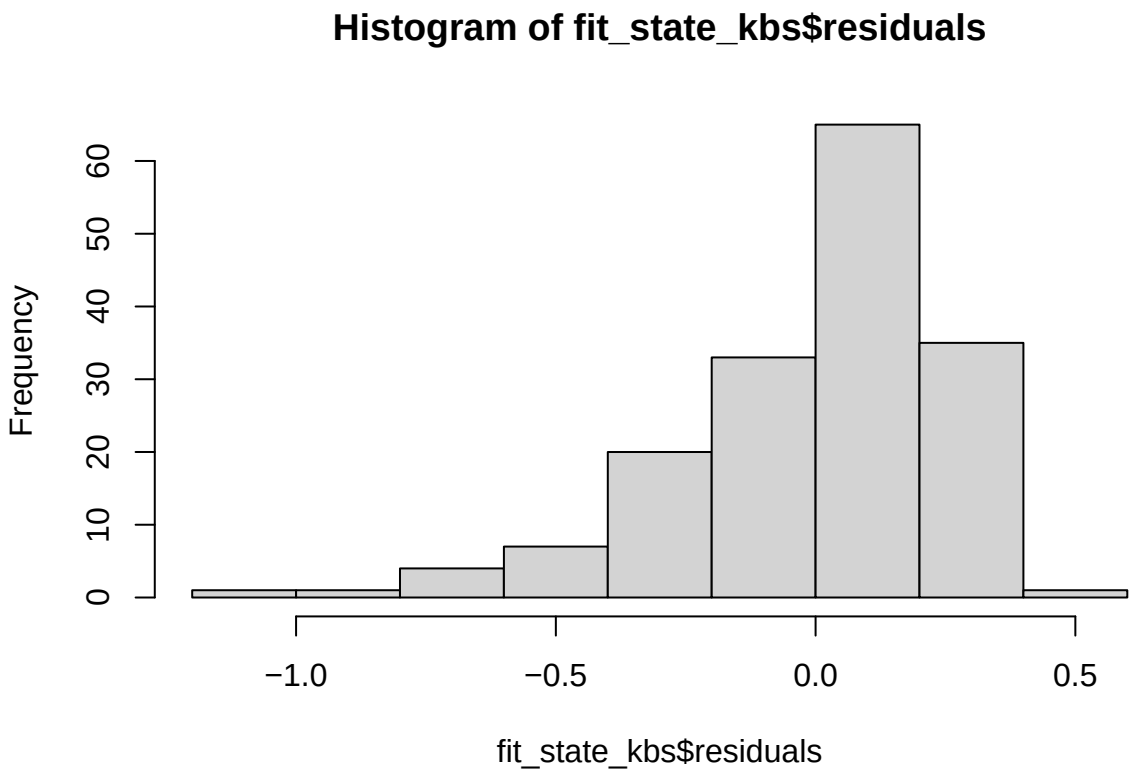
```
##      rstudent unadjusted p-value Bonferroni p
## 148 -4.104478         6.3743e-05     0.010645
```

```
qqPlot(fit_state_kbs, main = "QQ Plot")
```

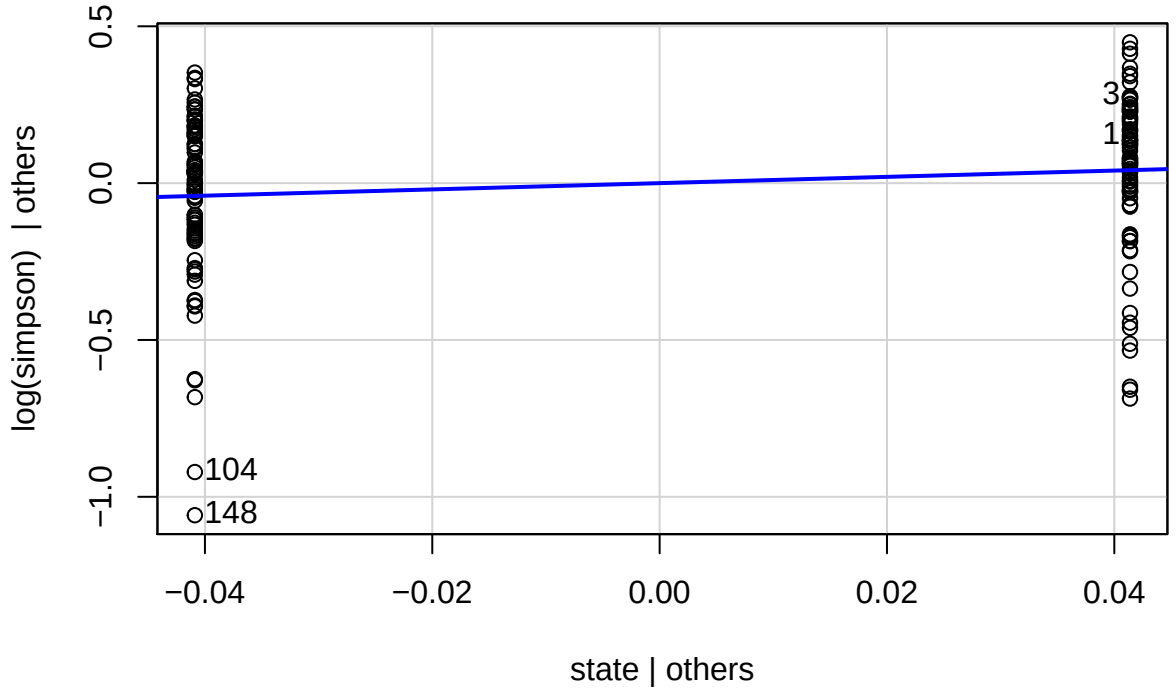



```
## [1] 104 148
```

```
hist(fit_state_kbs$residuals)
```



```r
leveragePlots(fit_state_kbs)
```



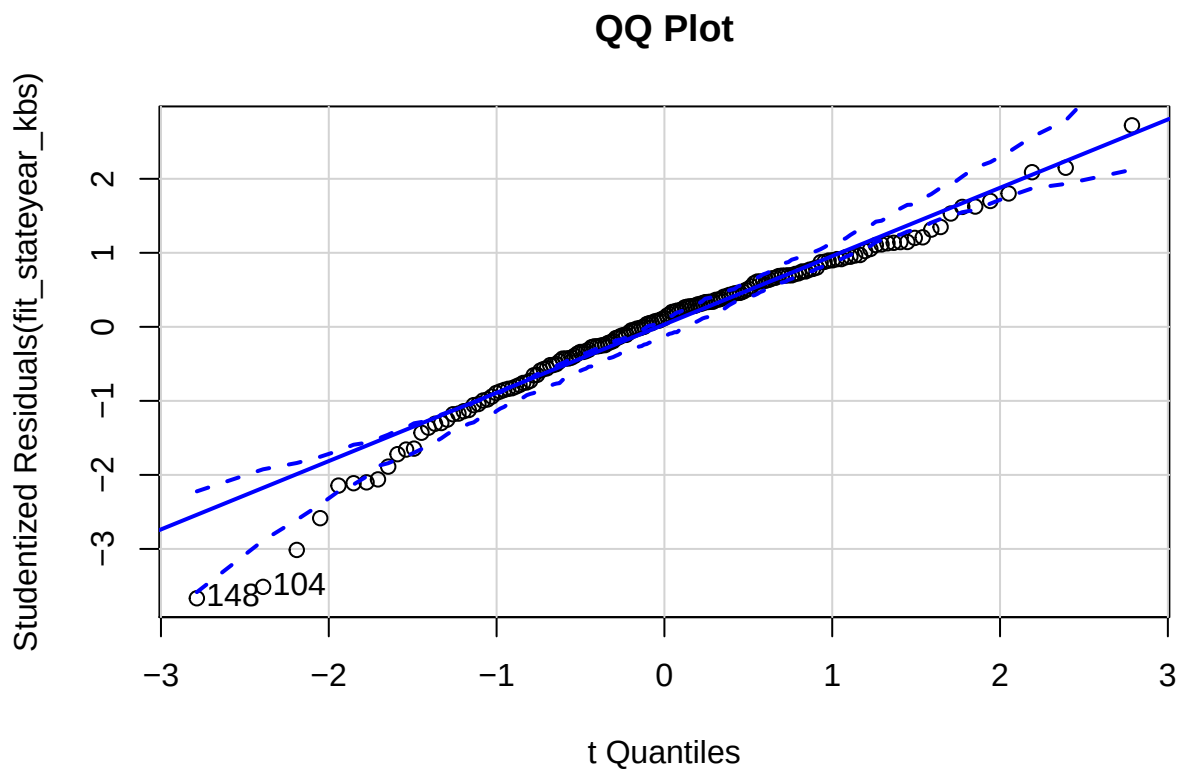

```r
ols_test_normality(fit_state_kbs)
```

```
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.911          0.0000
## Kolmogorov-Smirnov        0.1184         0.0185
## Cramer-von Mises         33.0384         0.0000
## Anderson-Darling          3.9793         0.0000
## -----------------------------------------------
```

```r
# KBS State and year model
fit_stateyear_kbs <- lm(log(simpson) ~ state + year, data = kbs_diversity)
outlierTest(fit_stateyear_kbs)  # no outliers
```

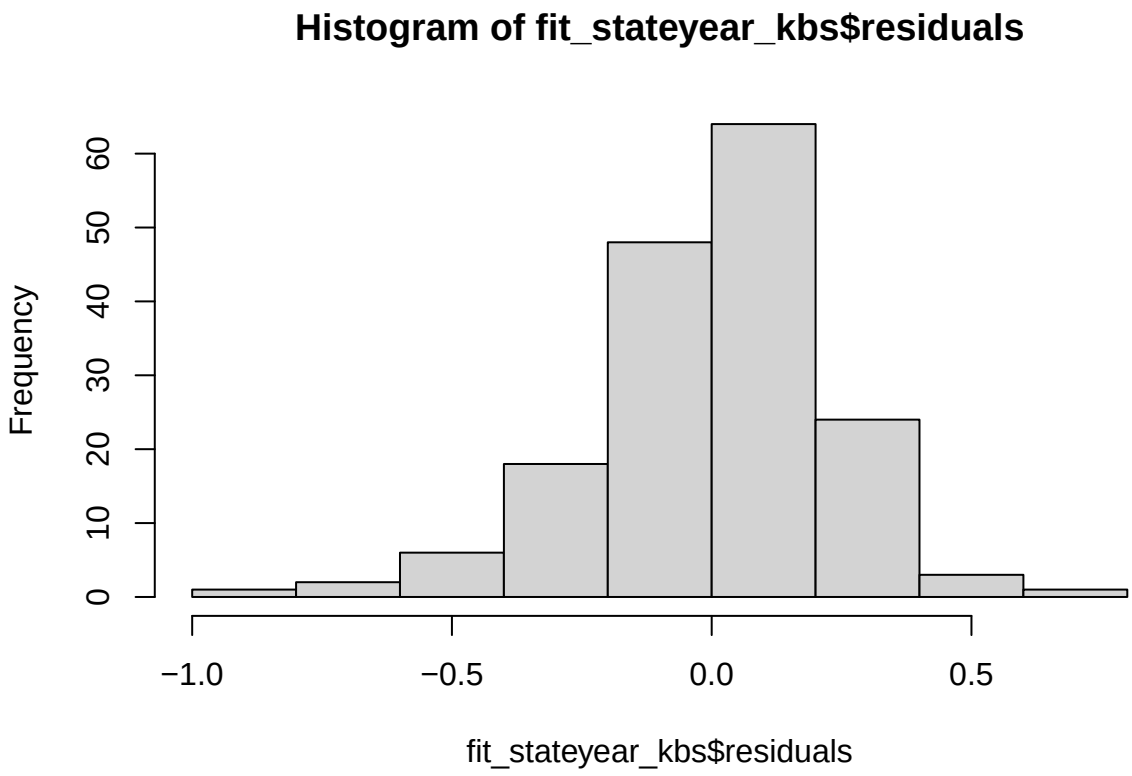
```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##      rstudent unadjusted p-value Bonferroni p
## 148 -3.666526         0.00033541     0.056013
```

```r
qqPlot(fit_stateyear_kbs, main = "QQ Plot")
```

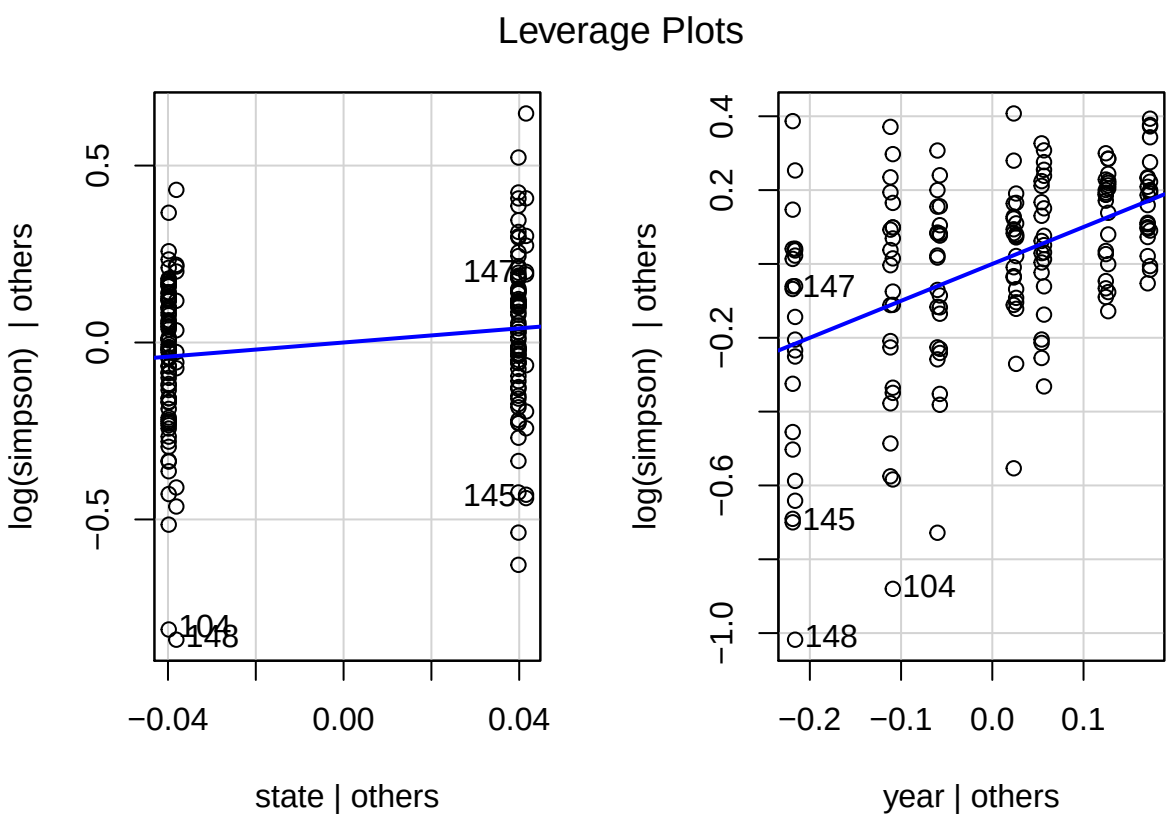



```
## [1] 104 148
```

```
hist(fit_stateyear_kbs$residuals)
```



```
leveragePlots(fit_stateyear_kbs)
```



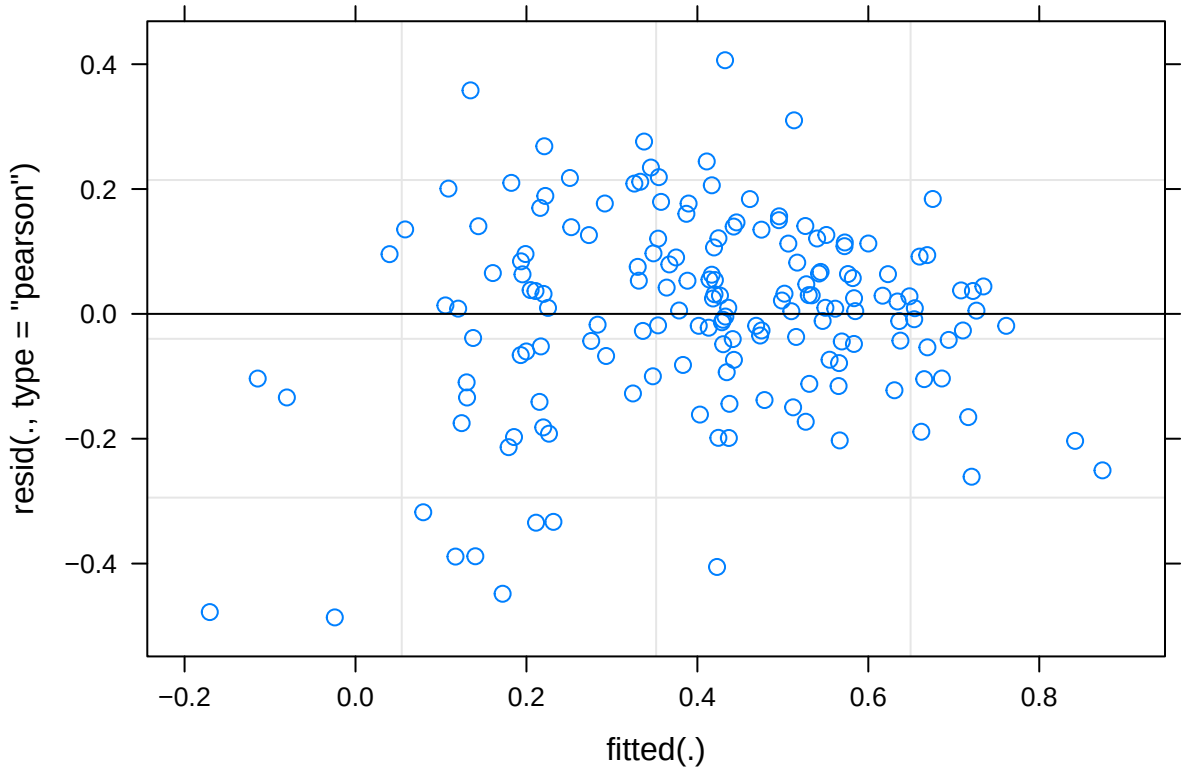

```
ols_test_normality(fit_stateyear_kbs)
```

```
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.9685          8e-04
## Kolmogorov-Smirnov        0.0646         0.4880
## Cramer-von Mises         35.0287         0.0000
## Anderson-Darling          1.3398         0.0017
## -----------------------------------------------
```

```
# Interaction plot (ignore for now the repeated measures with species); see:
# https://cran.r-project.org/web/packages/interactions/vignettes/interactions.html
# and: https://interactions.jacob-long.com/

# I can't get these to work fit3 <- lm(log(simpson) ~ state + year, data =
# kbs_diversity) interact_plot(fit3, pred = year, modx = state)
```

```
mod1 <- lmer(log(simpson) ~ state * year + insecticide * year + (1 | plot), kbs_diversity,
    REML = FALSE)

# Check Assumptions: (1) Linearity: if covariates are not categorical (year
# isn't) (2) Homogeneity: Need to Check by plotting residuals vs predicted
# values.
```

```r
par(mfrow = c(1, 2))
plot(mod1)
```




```r
# Homogeneity of variance is ok here (increasing variance in resids is not
# increasing with fitted values) Check for homogeneity of variances (true if
# p>0.05). If the result is not significant, the assumption of equal variances
# (homoscedasticity) is met (no significant difference between the group
# variances).  *****Levene's Test - tests whether or not the variance among two
# or more groups is equal - If the p-value is less than our chosen significance
# level, we can reject the null hypothesis and conclude that we have enough
# evidence to state that the variance among the groups is not equal (which we
# want).

leveneTest(residuals(mod1) ~ kbs_diversity$state)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.

## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1  0.0478 0.8271
##       165
```

```r
# Assumption not met
leveneTest(residuals(mod1) ~ kbs_diversity$insecticide)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1  2.5965  0.109
##       165
```

```
# Assumption not met
leveneTest(residuals(mod1) ~ kbs_diversity$plot)
```

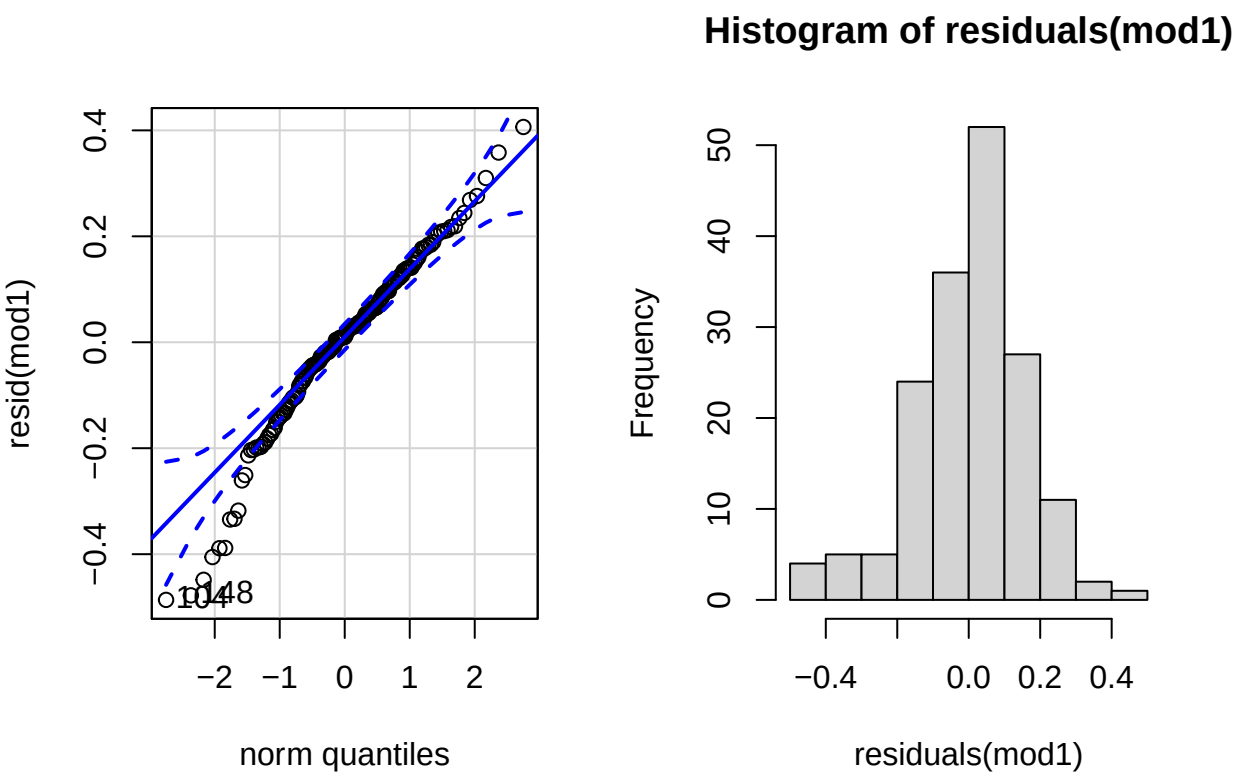
```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group  23  1.2707 0.1979
##       143
```

```
# Assumption not met

# (3) Normality of error term: need to check by histogram, QQplot of residuals,
# could do Kolmogorov-Smirnov test.  Check for normal residuals
qqPlot(resid(mod1))
```

```
## [1] 104 148
```

```
hist(residuals(mod1))
```



```r
shapiro.test(resid(mod1))  # ormally distributed resids bc p>0.05
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(mod1)
## W = 0.97032, p-value = 0.0012
```

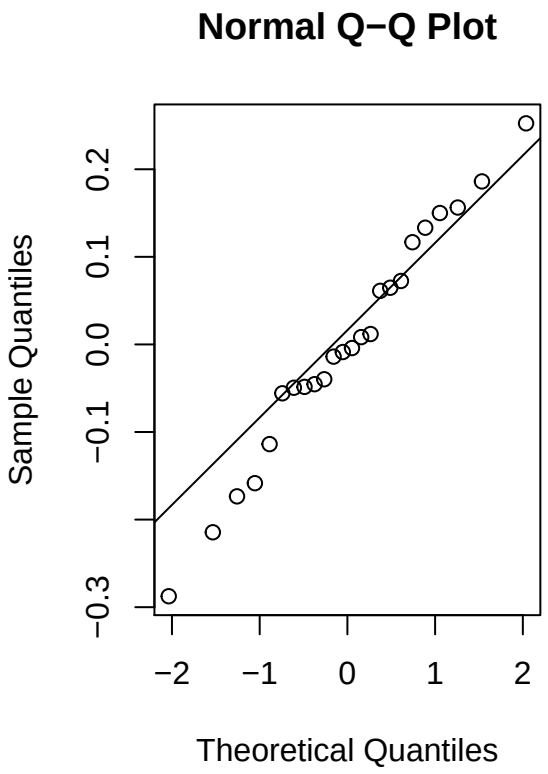
```r
outlierTest(mod1)  # no outliers
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##      rstudent unadjusted p-value Bonferroni p
## 104 -3.342862          0.0010585      0.17678
```

```r
# (4) Normality of random effect: Get the estimate of random effect (e.g., random
# intercepts), and check them as you would check the residual.
require(lme4)
r_int <- ranef(mod1)$plot$`(Intercept)`
qqnorm(r_int)
qqline(r_int)
shapiro.test(r_int)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  r_int
## W = 0.98308, p-value = 0.9451
```

```r
# Normally distributed random effect pvalue > 0.05
```




```
# Do we need to include plot as a random effect with the UMBS models?
mod1 <- lmer(log(simpson) ~ state * year + insecticide * year + (1 | plot), kbs_diversity,
    REML = FALSE)
mod2 <- lmer(log(simpson) ~ state * year + insecticide + year + (1 | plot), kbs_diversity,
    REML = FALSE)
# Run analysis of variance on each model (see this for more explanation on how
# anova on a linear mixed effects model is similar to an anove on a regular
# linear model: https://m-clark.github.io/docs/mixedModels/anovamixed.html)
anova(mod1)
```

```
## Analysis of Variance Table
##                  npar  Sum Sq Mean Sq F value
## state               1 0.04923 0.04923  1.7925
## year                6 2.56579 0.42763 15.5716
## insecticide         1 0.00417 0.00417  0.1517
## state:year          6 0.10694 0.01782  0.6490
## year:insecticide    6 0.62226 0.10371  3.7764
```

```
anova(mod2)
```

```
## Analysis of Variance Table
##             npar  Sum Sq Mean Sq F value
## state          1 0.05752 0.05752  1.8056
## year           6 2.56846 0.42808 13.4379
## insecticide    1 0.00485 0.00485  0.1522
## state:year     6 0.10668 0.01778  0.5581
```

```r
anova(mod1, mod2)  # Go with model 1 since pvalue <0.05, aka more complex model does have something in
```

```
## Data: kbs_diversity
## Models:
## mod2: log(simpson) ~ state * year + insecticide + year + (1 | plot)
## mod1: log(simpson) ~ state * year + insecticide * year + (1 | plot)
##      npar     AIC    BIC logLik deviance  Chisq Df Pr(>Chisq)
## mod2   17 -28.256 24.750 31.128  -62.256
## mod1   23 -37.258 34.456 41.629  -83.258 21.002  6   0.001833 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
summary(mod1)
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: log(simpson) ~ state * year + insecticide * year + (1 | plot)
##    Data: kbs_diversity
##
##      AIC      BIC   logLik deviance df.resid
##    -37.3     34.5     41.6    -83.3      144
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.93439 -0.46009  0.05937  0.58195  2.45210
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  plot     (Intercept) 0.01991  0.1411
##  Residual             0.02746  0.1657
## Number of obs: 167, groups:  plot, 24
##
## Fixed effects:
##                                 Estimate Std. Error t value
## (Intercept)                      0.56061    0.07695   7.285
## statewarmed                      0.01463    0.08885   0.165
## year2016                        -0.05695    0.08286  -0.687
## year2017                        -0.21499    0.08286  -2.595
## year2018                        -0.06251    0.08286  -0.754
## year2019                        -0.09392    0.08286  -1.134
## year2020                        -0.04152    0.08286  -0.501
## year2021                        -0.19385    0.08615  -2.250
## insecticideno_insects            0.02893    0.08885   0.326
## statewarmed:year2016            -0.06684    0.09568  -0.699
## statewarmed:year2017            -0.12632    0.09568  -1.320
## statewarmed:year2018            -0.06268    0.09568  -0.655
## statewarmed:year2019            -0.14113    0.09568  -1.475
## statewarmed:year2020            -0.13665    0.09568  -1.428
## statewarmed:year2021            -0.15061    0.09696  -1.553
## year2016:insecticideno_insects   0.08898    0.09568   0.930
## year2017:insecticideno_insects   0.09540    0.09568   0.997
## year2018:insecticideno_insects  -0.04480    0.09568  -0.468
## year2019:insecticideno_insects  -0.23494    0.09568  -2.456
```

```
## year2020:insecticideno_insects -0.07393    0.09568  -0.773
## year2021:insecticideno_insects -0.21572    0.09696  -2.225

##
## Correlation matrix not shown by default, as p = 21 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

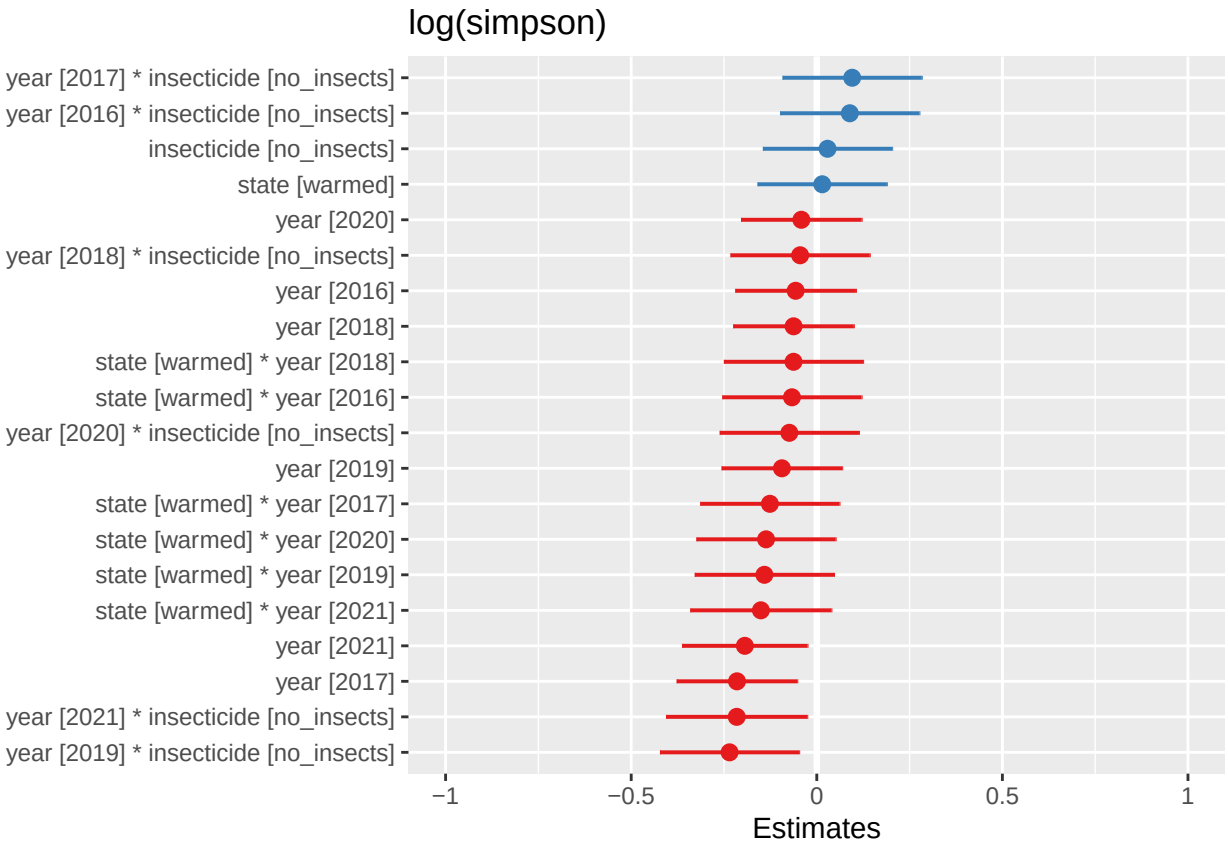
```
summary(mod2)
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: log(simpson) ~ state * year + insecticide + year + (1 | plot)
##    Data: kbs_diversity
##
##      AIC      BIC   logLik deviance df.resid
##    -28.3     24.7     31.1    -62.3      150
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.2592 -0.4062  0.0534  0.6898  1.9171
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  plot     (Intercept) 0.01906  0.1381
##  Residual             0.03186  0.1785
## Number of obs: 167, groups:  plot, 24
##
## Fixed effects:
##                       Estimate Std. Error t value
## (Intercept)            0.58749    0.07230   8.125
## statewarmed            0.01463    0.09212   0.159
## year2016              -0.01246    0.07287  -0.171
## year2017              -0.16729    0.07287  -2.296
## year2018              -0.08492    0.07287  -1.165
## year2019              -0.21139    0.07287  -2.901
## year2020              -0.07849    0.07287  -1.077
## year2021              -0.31052    0.07472  -4.156
## insecticideno_insects -0.02481    0.06277  -0.395
## statewarmed:year2016  -0.06684    0.10305  -0.649
## statewarmed:year2017  -0.12632    0.10305  -1.226
## statewarmed:year2018  -0.06268    0.10305  -0.608
## statewarmed:year2019  -0.14113    0.10305  -1.370
## statewarmed:year2020  -0.13665    0.10305  -1.326
## statewarmed:year2021  -0.14180    0.10436  -1.359

##
## Correlation matrix not shown by default, as p = 15 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```
AICctab(mod1, mod2, weights = T)  # model 1
```

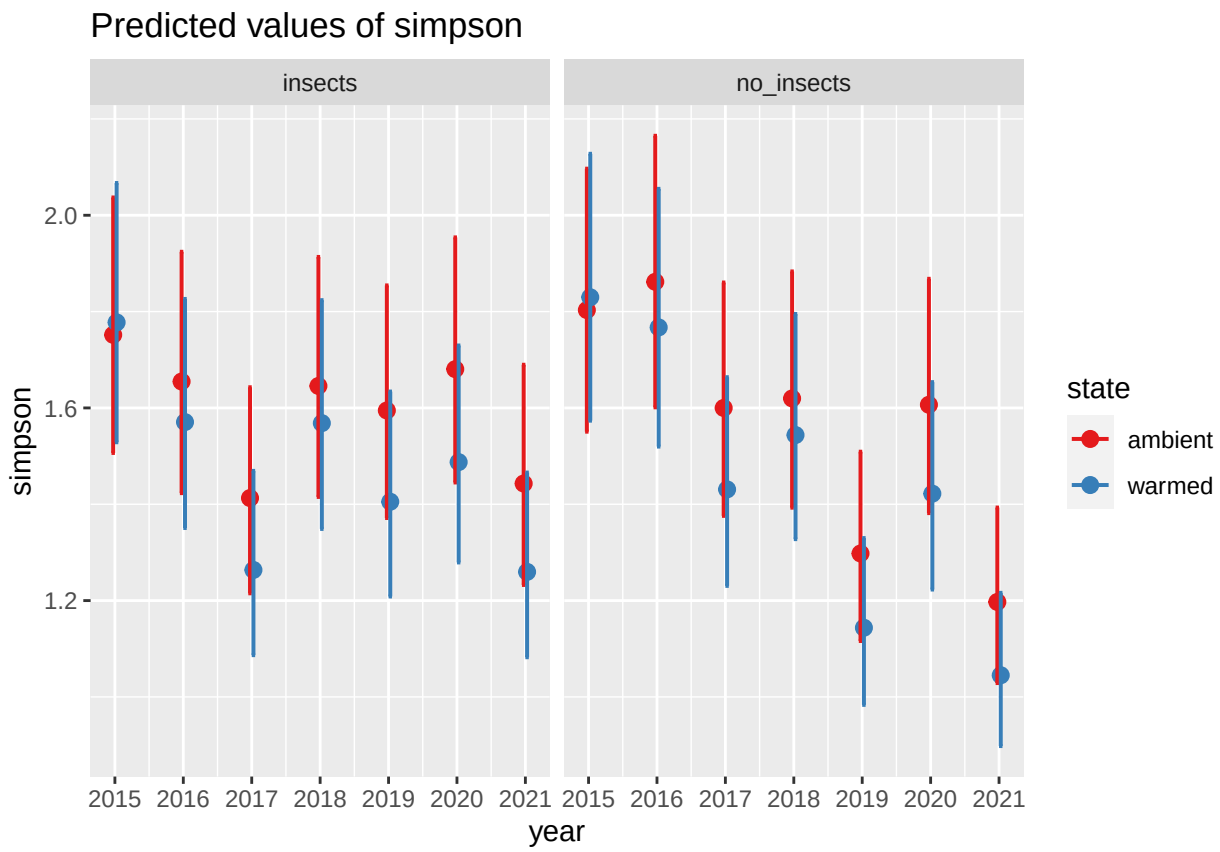
```
##      dAICc df weight
## mod1   0.0  23 0.937
## mod2   5.4  17 0.063
```

```
# Plot the fixed effects estimates for different models these are the fixed
# effects estimates from summary(mod1)
plot_model(mod1, sort.est = TRUE)
```

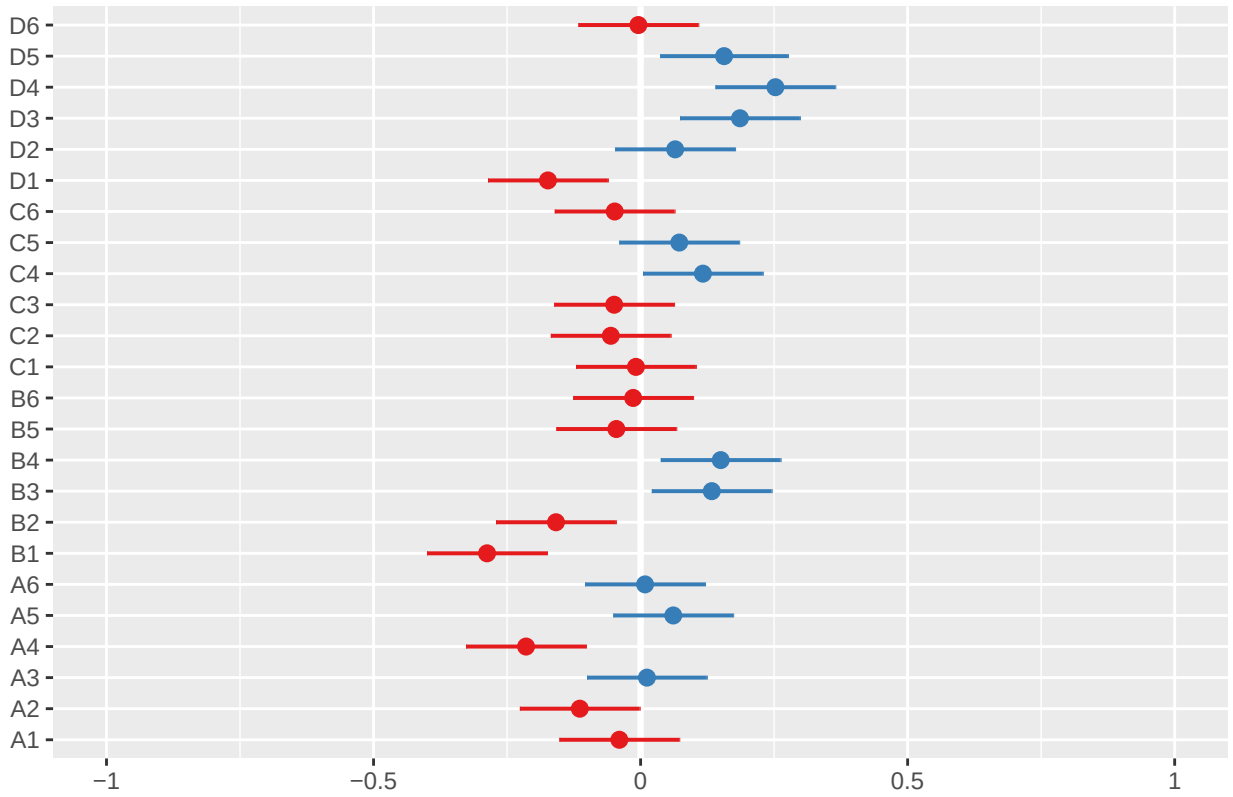



```
# these are the fixed predicted values:
plot_model(mod1, type = "pred", terms = c("year", "state", "insecticide"))
```

```
## Model has log-transformed response. Back-transforming predictions to original response scale. Standa
```




```
# these are the random effects estimates
plot_model(mod1, type = "re", terms = c("species"))
```




```
# Does year need to be interactive with state?
mod3 <- lmer(log(simpson) ~ state + year + insecticide * year + (1 | plot), kbs_diversity,
    REML = FALSE)
anova(mod2, mod3)
```

```
## Data: kbs_diversity
## Models:
## mod2: log(simpson) ~ state * year + insecticide + year + (1 | plot)
## mod3: log(simpson) ~ state + year + insecticide * year + (1 | plot)
##      npar     AIC     BIC logLik deviance  Chisq Df Pr(>Chisq)
## mod2   17 -28.256 24.7497 31.128  -62.256
## mod3   17 -45.253  7.7528 39.627  -79.253 16.997  0
```

```
AICctab(mod1, mod3, weights = T)  # going with mod3
```

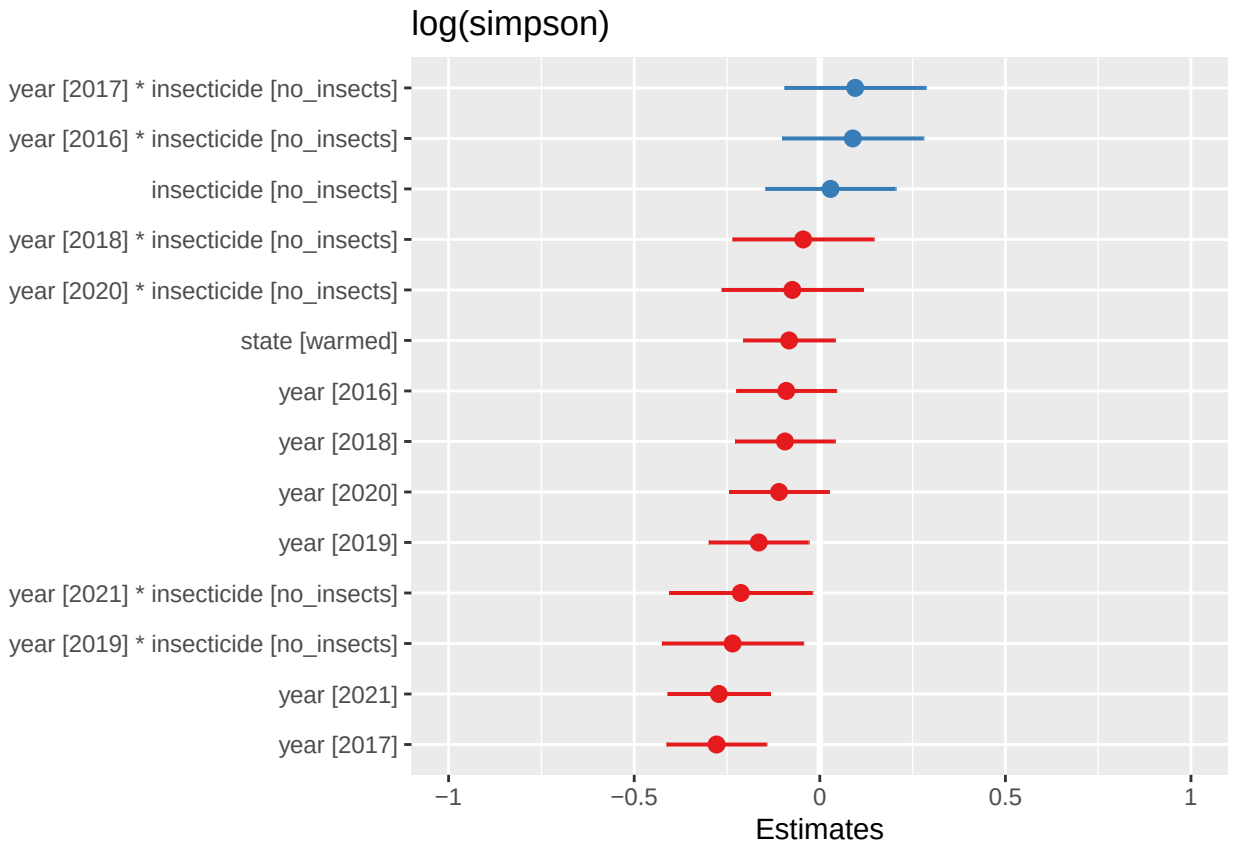
```
##      dAICc df weight
## mod3  0.0  17 0.997
## mod1 11.6  23 0.003
```

```
# Do we need to include insecticide? (dropping insecticide from the model)
mod5 <- lmer(log(simpson) ~ state + year + (1 | plot), kbs_diversity, REML = FALSE)
anova(mod3, mod5)
```

```
## Data: kbs_diversity
## Models:
```

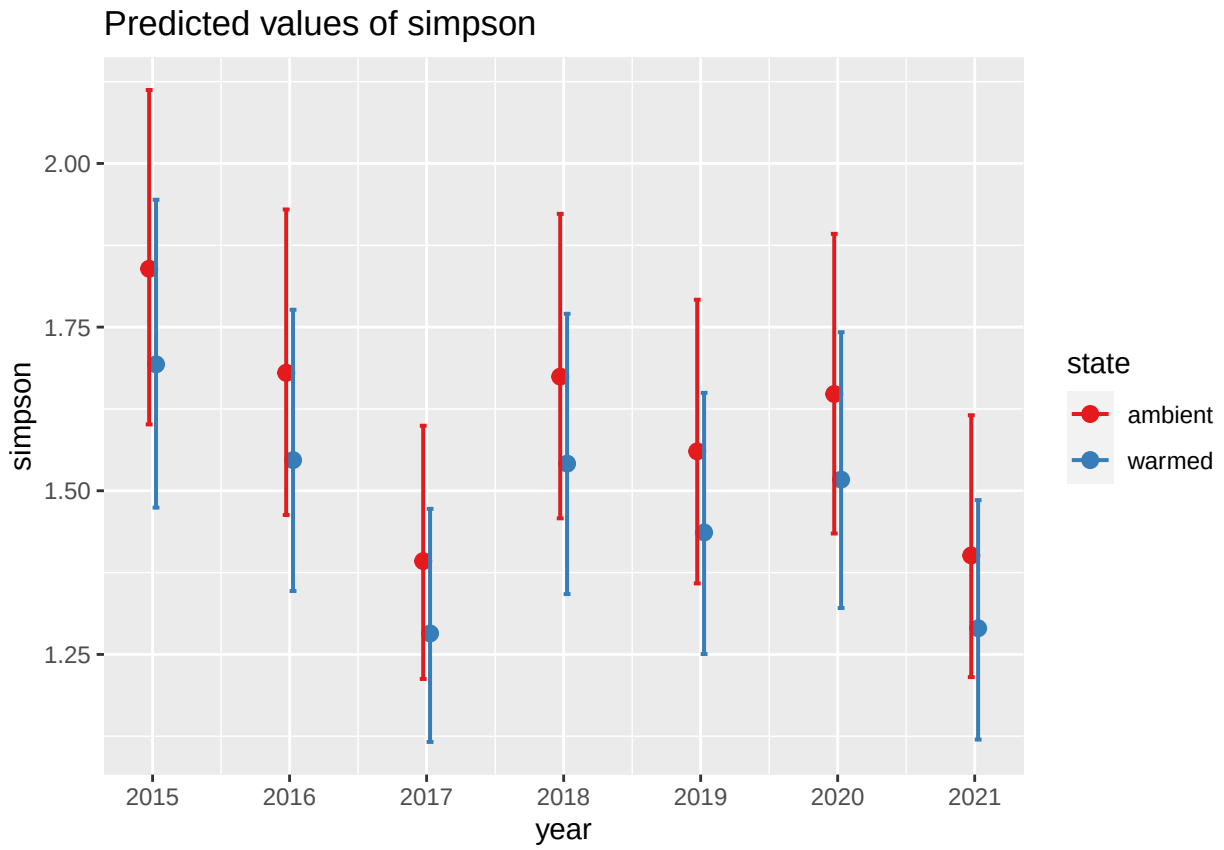
```
## mod5: log(simpson) ~ state + year + (1 | plot)
## mod3: log(simpson) ~ state + year + insecticide * year + (1 | plot)
##      npar     AIC     BIC logLik deviance  Chisq Df Pr(>Chisq)
## mod5   10 -38.795 -7.6155 29.398  -58.795
## mod3   17 -45.253  7.7528 39.627  -79.253 20.458  7   0.004662 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Yes, p<0.05 so insecticide*year does strongly improve model fit so we will
# stick with the more complex mod3

# Plot the fixed effects estimates for different models these are the fixed
# effects estimates from summary(mod5)
plot_model(mod3, sort.est = TRUE)
```

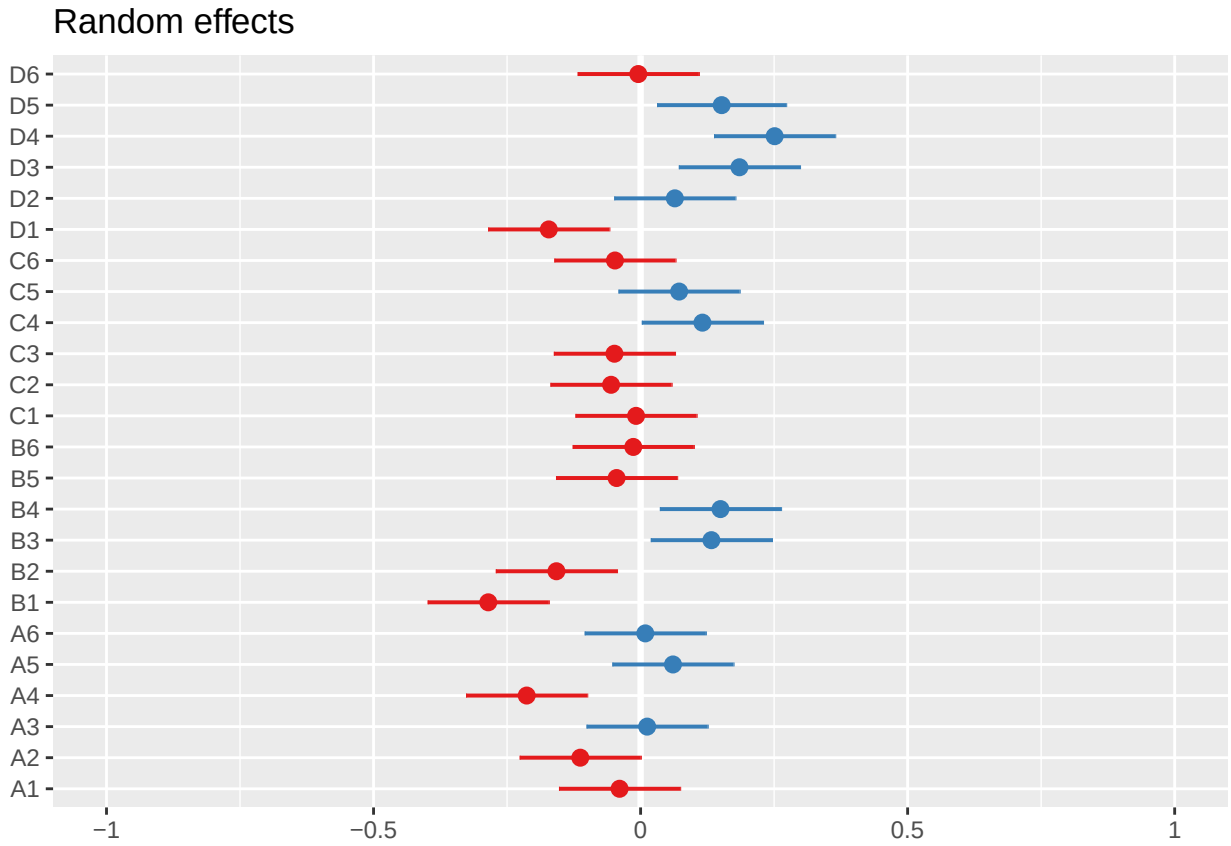



```r
# these are the fixed predicted values:
plot_model(mod3, type = "pred", terms = c("year", "state"))
```

```
## Model has log-transformed response. Back-transforming predictions to original response scale. Standa
```




```
# these are the random effects estimates
plot_model(mod3, type = "re", terms = c("species"))
```




```
# If we wanted to include plots nested within year it would look like this: mod6
# <- lmer(log(simpson) ~ state + year + insecticide*year + (1 + year|plot),
# kbs_diversity, REML=FALSE) anova(mod5, mod6) anova(mod5) cant get mod6 to work

# the best model fit appears to be = mod3 <- lmer(log(simpson) ~ state + year +
# insecticide*year + (1|plot), kbs_diversity, REML = FALSE)
summ(mod3)
```

Observations	167
Dependent variable	log(simpson)
Type	Mixed effects linear regression

AIC	-45.25
BIC	7.75
Pseudo-R^2 (fixed effects)	0.31
Pseudo-R^2 (total)	0.59

UMBS Simpson's Index

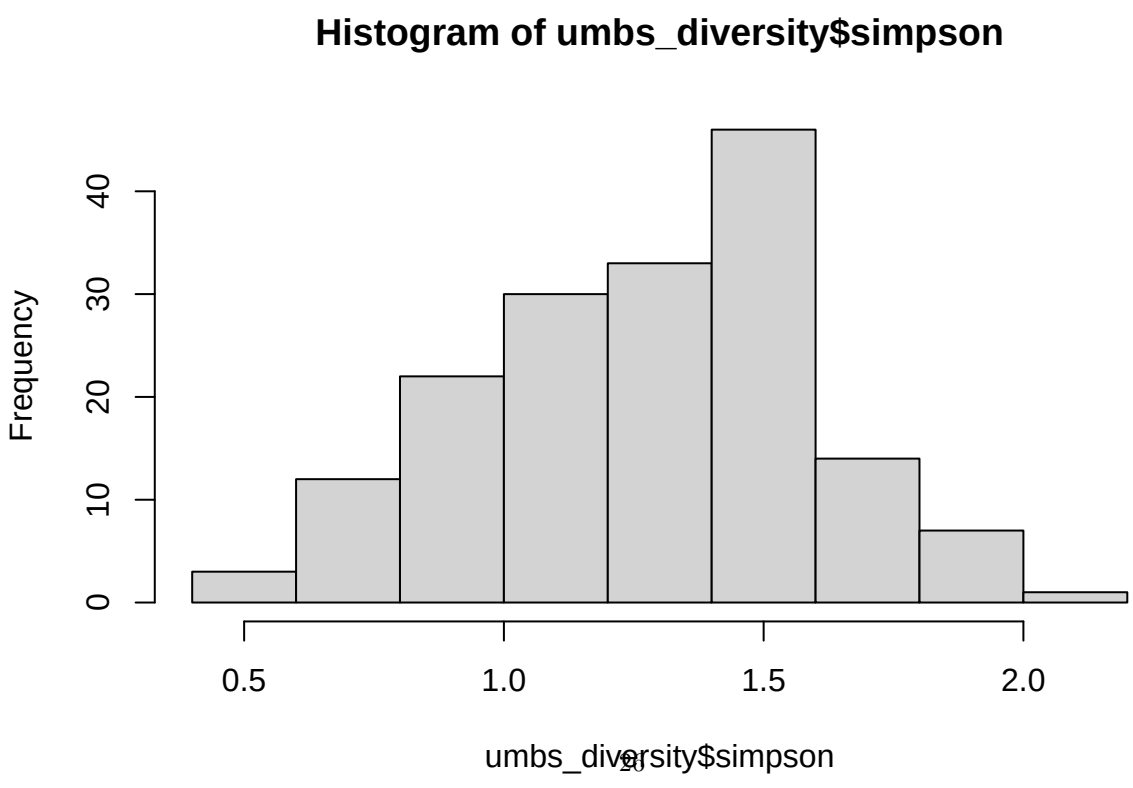
```
# Data exploration
hist(umbs_diversity$simpson)
```

Fixed Effects					
	Est.	S.E.	t val.	d.f.	p
(Intercept)	0.61	0.07	8.63	62.69	0.00
statewarmed	-0.08	0.06	-1.31	23.97	0.20
year2016	-0.09	0.07	-1.32	142.97	0.19
year2017	-0.28	0.07	-4.05	142.97	0.00
year2018	-0.09	0.07	-1.37	142.97	0.17
year2019	-0.16	0.07	-2.40	142.97	0.02
year2020	-0.11	0.07	-1.60	142.97	0.11
year2021	-0.27	0.07	-3.86	143.34	0.00
insecticideno_insects	0.03	0.09	0.32	83.14	0.75
year2016:insecticideno_insects	0.09	0.10	0.92	142.97	0.36
year2017:insecticideno_insects	0.10	0.10	0.98	142.97	0.33
year2018:insecticideno_insects	-0.04	0.10	-0.46	142.97	0.65
year2019:insecticideno_insects	-0.23	0.10	-2.42	142.97	0.02
year2020:insecticideno_insects	-0.07	0.10	-0.76	142.97	0.45
year2021:insecticideno_insects	-0.21	0.10	-2.17	143.16	0.03

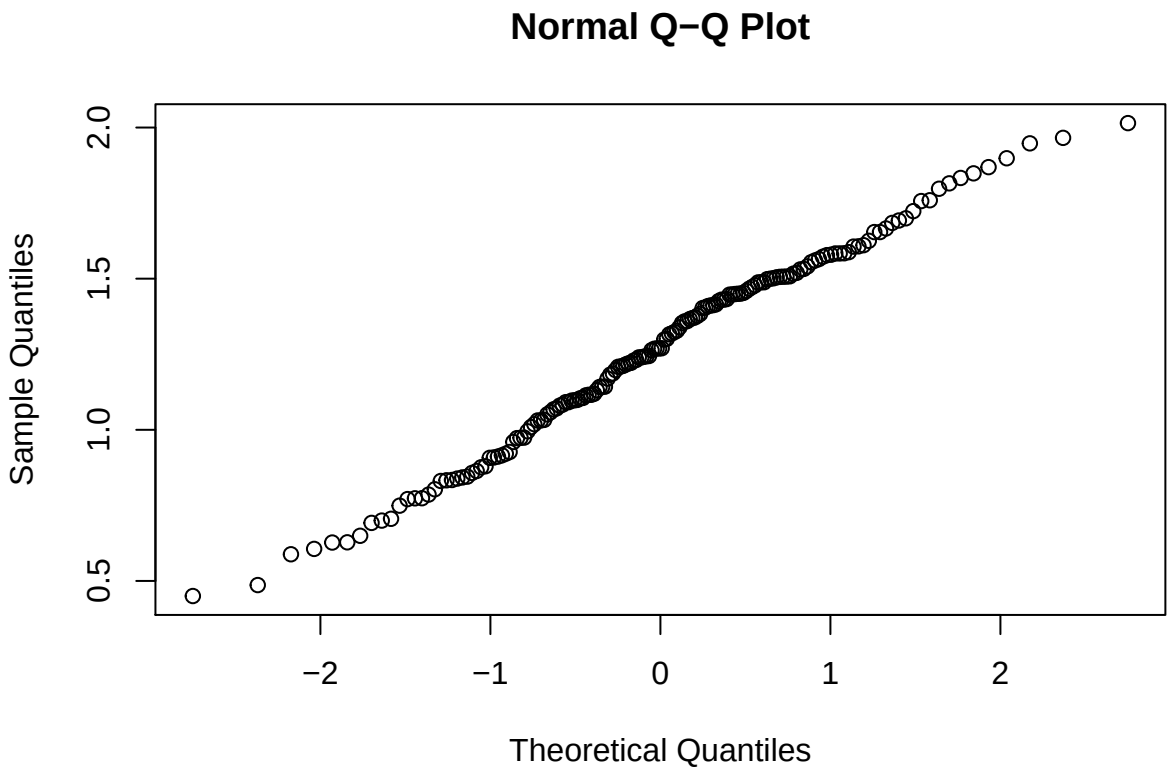
p values calculated using Satterthwaite d.f.

Random Effects		
Group	Parameter	Std. Dev.
plot	(Intercept)	0.14
Residual		0.17

Grouping Variables		
Group	# groups	ICC
plot	24	0.41



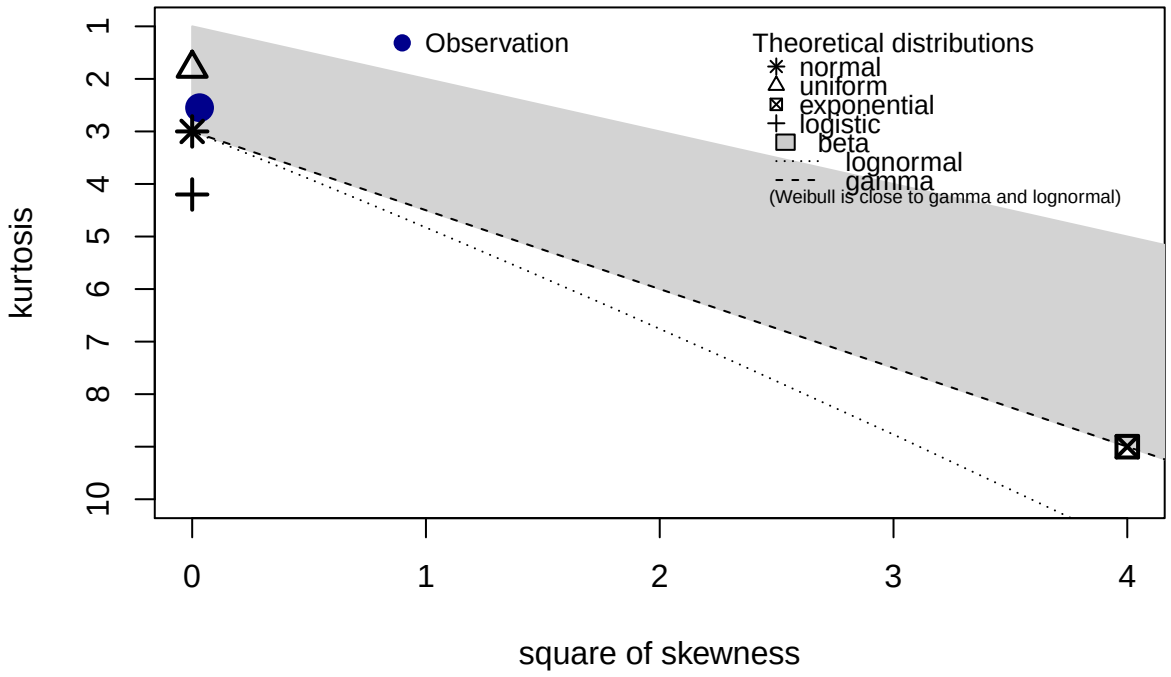
```r
qqnorm(umbs_diversity$simpson)  # this looks good
```




```r
shapiro.test(umbs_diversity$simpson)  # not normal
```

```
##
##  Shapiro-Wilk normality test
##
## data:  umbs_diversity$simpson
## W = 0.98934, p-value = 0.2377
```

```r
# Exploring distributions for these slightly left-skewed data:
descdist(umbs_diversity$simpson, discrete = FALSE)  # i think we can assume normality with umbs_diversi
```

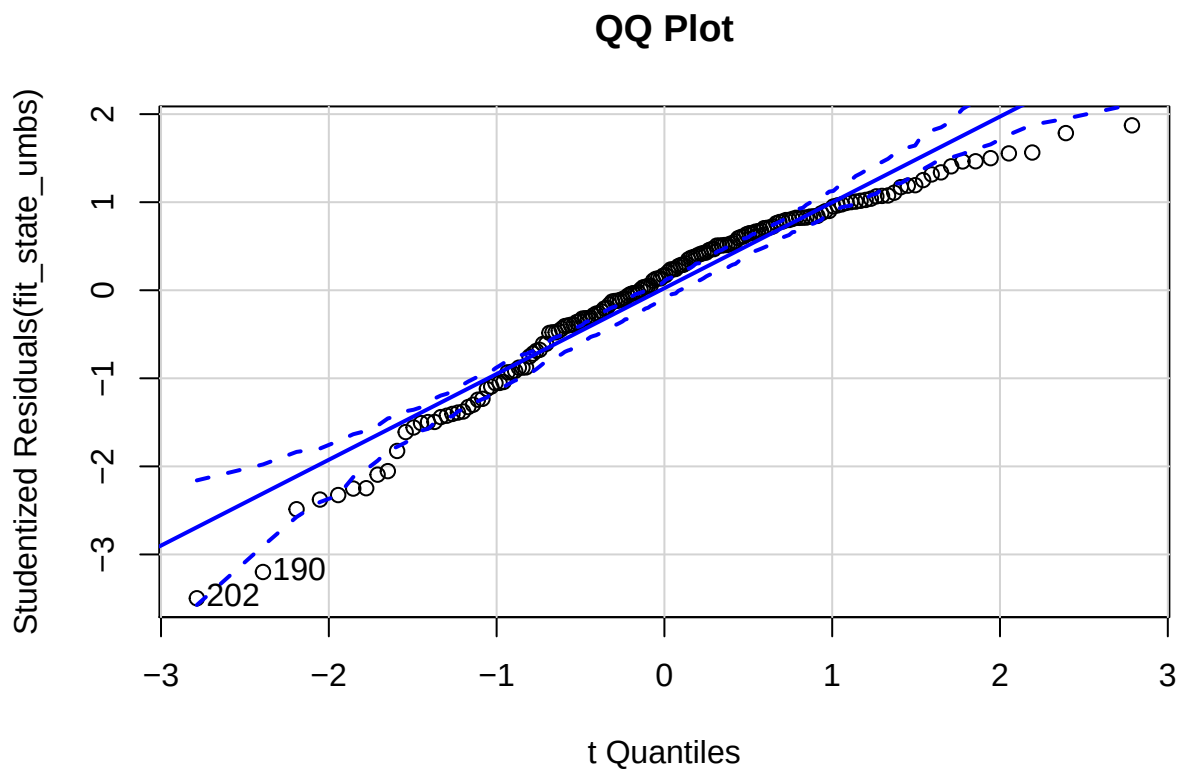



```
## summary statistics
## ------
## min:  0.4501077   max:  2.014592
## median:  1.269487
## mean:  1.264579
## estimated sd:  0.3273682
## estimated skewness:  -0.1779162
## estimated kurtosis:  2.548224
```

```
# UMBS State-only model
fit_state_umbs <- lm(log(simpson) ~ state, data = umbs_diversity)
outlierTest(fit_state_umbs)  # no outliers
```

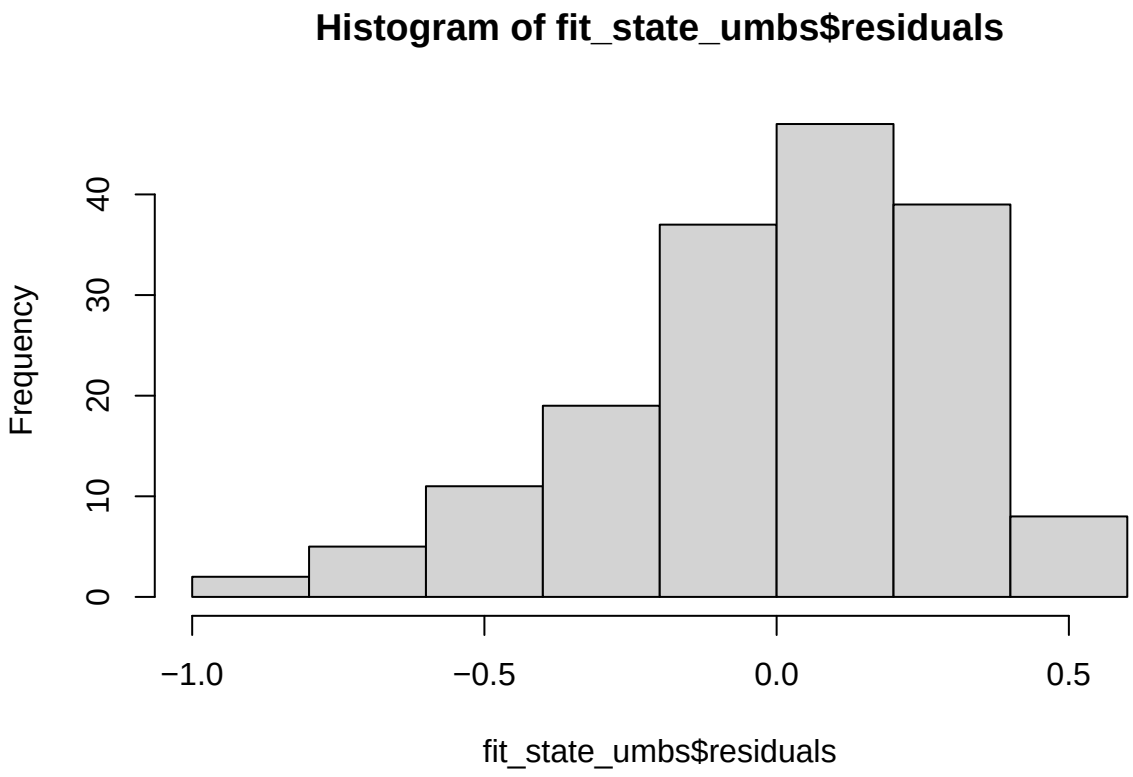
```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##     rstudent unadjusted p-value Bonferroni p
## 202 -3.49748         0.00060365      0.10141
```

```
qqPlot(fit_state_umbs, main = "QQ Plot")
```

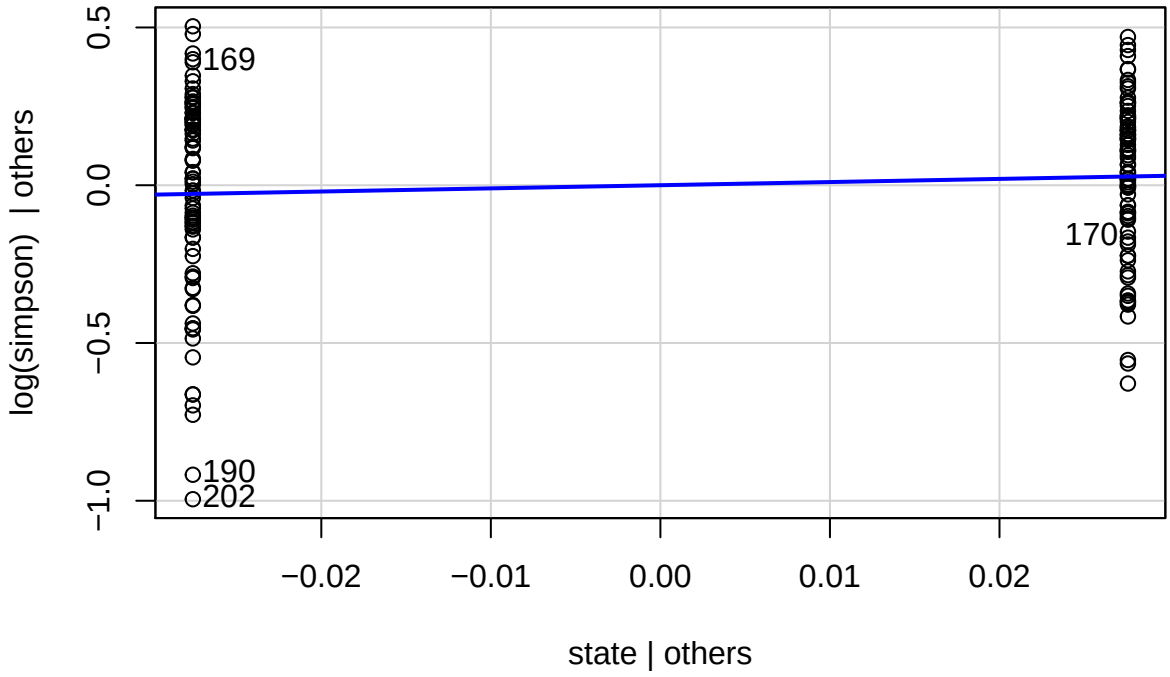



```
## 190 202
##  22  34
```

```
hist(fit_state_umbs$residuals)
```



```
leveragePlots(fit_state_umbs)
```




```
ols_test_normality(fit_state_umbs)
```

```
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.9546         0.0000
## Kolmogorov-Smirnov        0.0837         0.1897
## Cramer-von Mises         30.1203         0.0000
## Anderson-Darling          2.196          0.0000
## -----------------------------------------------
```

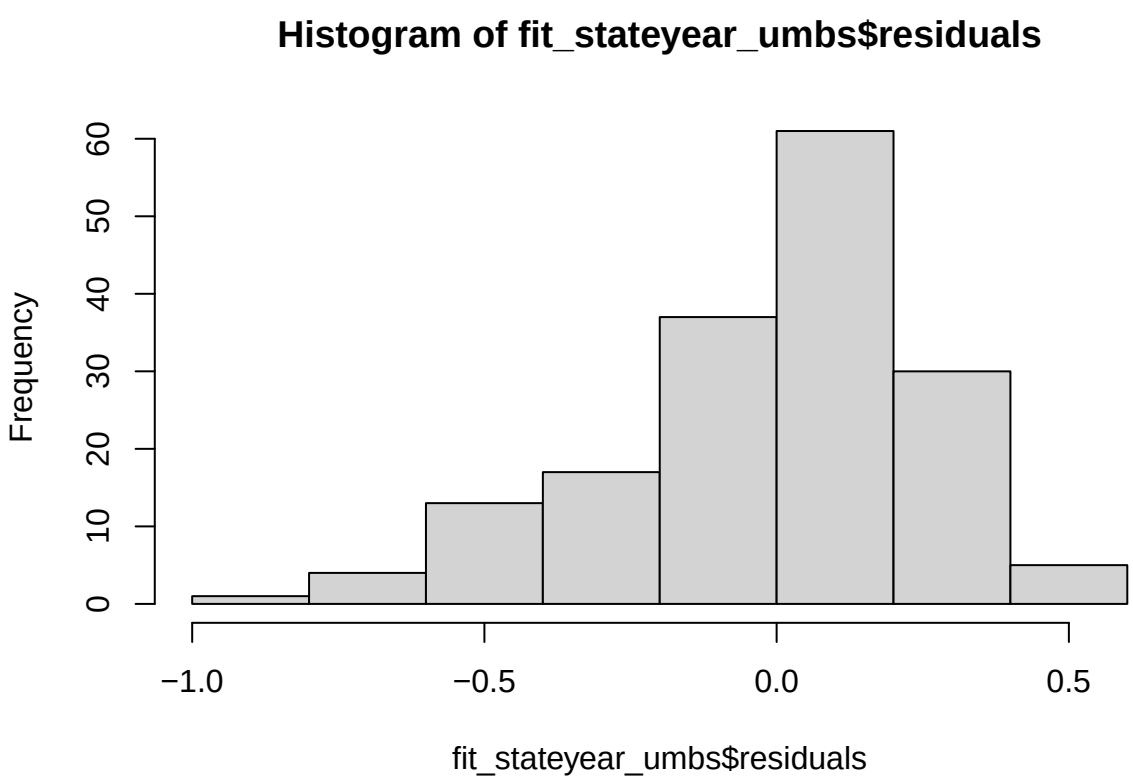
```
# UMBS State and year model
fit_stateyear_umbs <- lm(log(simpson) ~ state + year, data = umbs_diversity)
outlierTest(fit_stateyear_kbs)  # no outliers
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##      rstudent unadjusted p-value Bonferroni p
## 148 -3.666526         0.00033541     0.056013
```

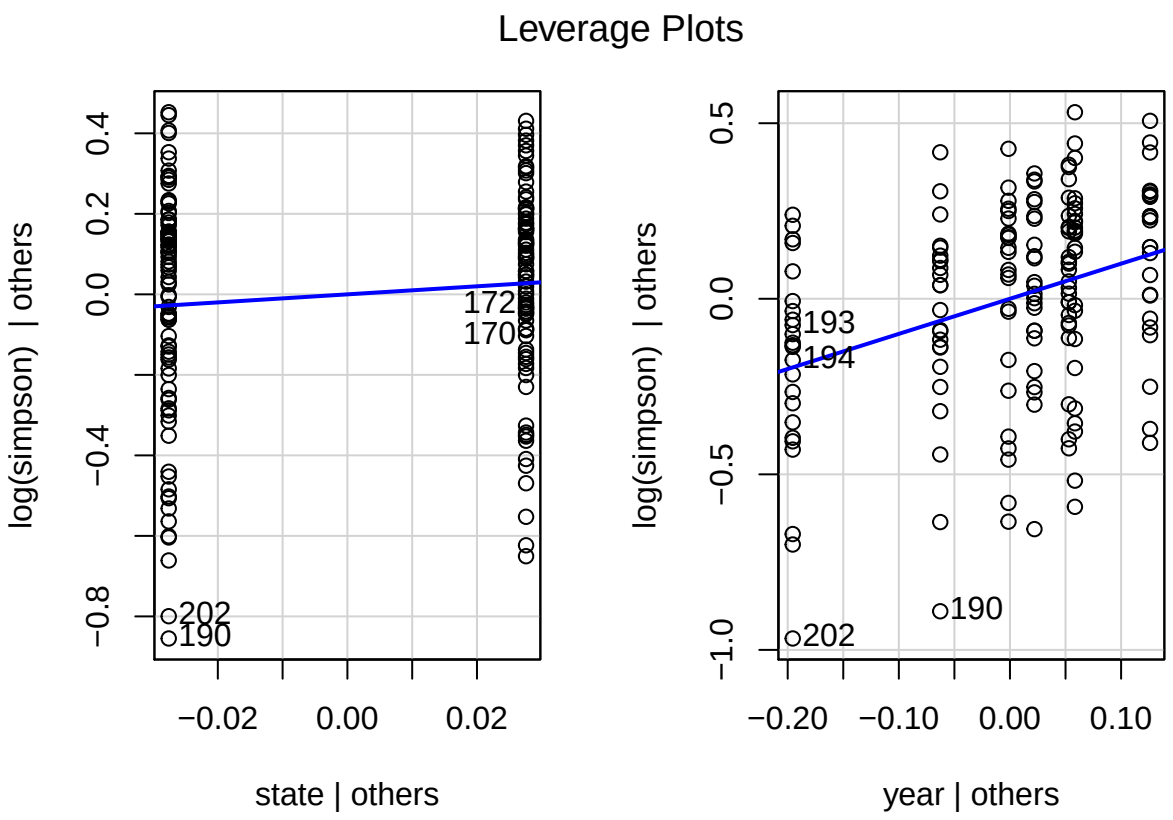
```
qqPlot(fit_stateyear_umbs, main = "QQ Plot")
```


```
## 190 202
##  22  34
```

```
hist(fit_stateyear_umbs$residuals)
```



```
leveragePlots(fit_stateyear_umbs)
```



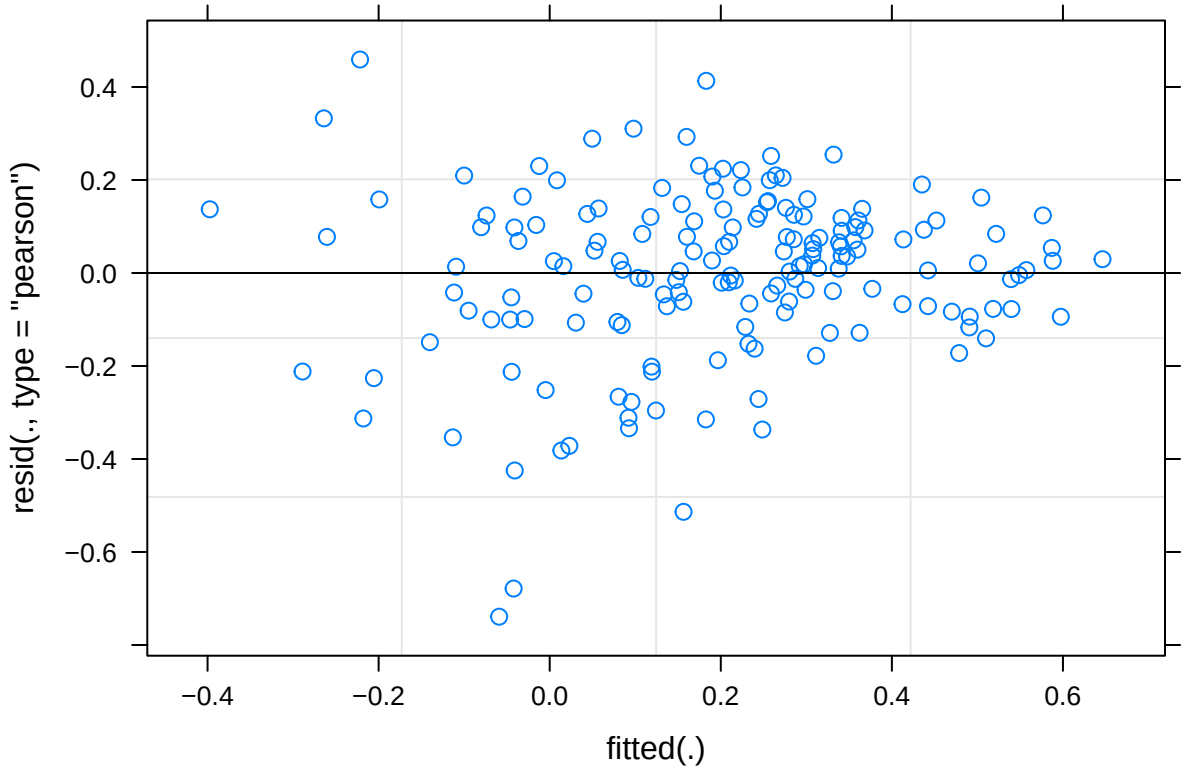

```
ols_test_normality(fit_stateyear_umbs)
```

```
## -----------------------------------------------
##        Test             Statistic       pvalue
## -----------------------------------------------
## Shapiro-Wilk              0.9505         0.0000
## Kolmogorov-Smirnov        0.0978         0.0806
## Cramer-von Mises         31.4482         0.0000
## Anderson-Darling          2.6352         0.0000
## -----------------------------------------------
```

```
# Interaction plot (ignore for now the repeated measures with species); see:
# https://cran.r-project.org/web/packages/interactions/vignettes/interactions.html
# and: https://interactions.jacob-long.com/

# I can't get these to work
fit3 <- lm(log(simpson) ~ state + year, data = umbs_diversity)
# interact_plot(fit3, pred = year, modx = state)
```

```
mod1u <- lmer(log(simpson) ~ state * year + insecticide * year + (1 | plot), umbs_diversity,
    REML = FALSE)

# Check Assumptions: (1) Linearity: if covariates are not categorical (year
# isn't) (2) Homogeneity: Need to Check by plotting residuals vs predicted
```

```
# values.
par(mfrow = c(1, 2))
plot(mod1u)
```




```
# Homogeneity of variance is ok here (increasing variance in resids is not
# increasing with fitted values) Check for homogeneity of variances (true if
# p>0.05). If the result is not significant, the assumption of equal variances
# (homoscedasticity) is met (no significant difference between the group
# variances).  *****Levene's Test - tests whether or not the variance among two
# or more groups is equal - If the p-value is less than our chosen significance
# level, we can reject the null hypothesis and conclude that we have enough
# evidence to state that the variance among the groups is not equal (which we
# want).

leveneTest(residuals(mod1u) ~ umbs_diversity$state)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value  Pr(>F)
## group   1  5.6029 0.01908 *
##       166
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Assumption
leveneTest(residuals(mod1u) ~ umbs_diversity$insecticide)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1  0.5045 0.4785
##       166
```

```r
# Assumption not met
leveneTest(residuals(mod1u) ~ umbs_diversity$plot)
```

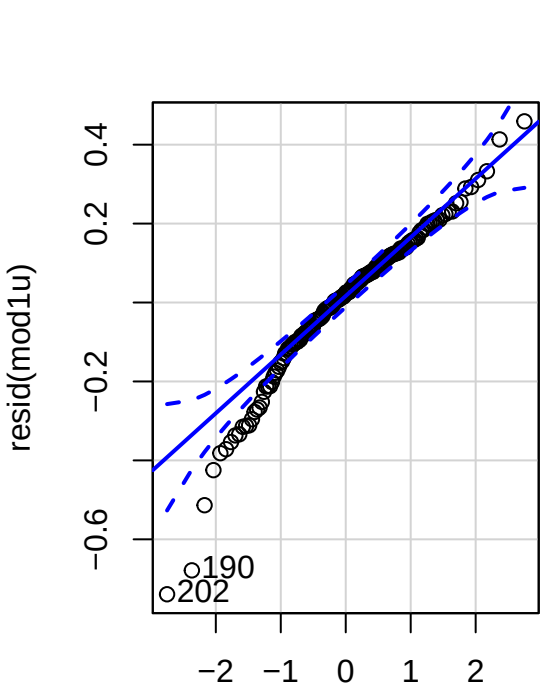
```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.
```

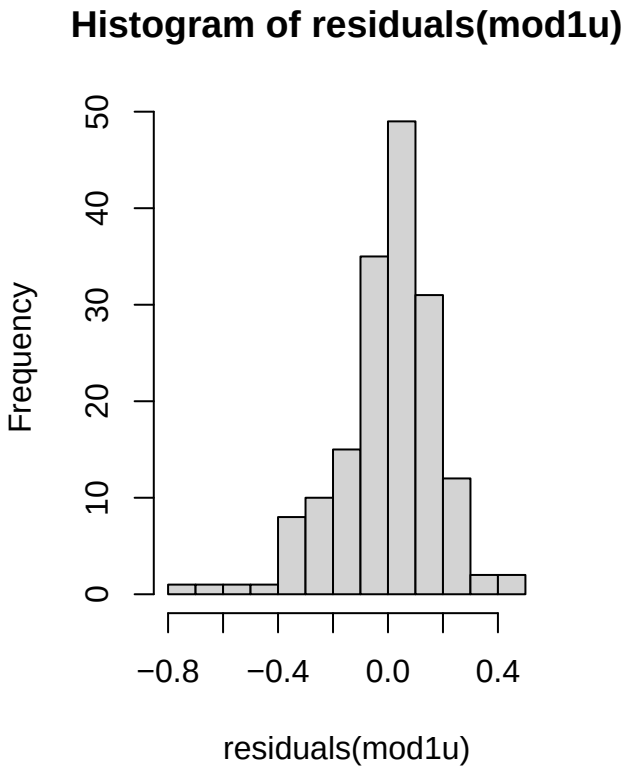
```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group  23   0.805 0.7209
##       144
```

```r
# Assumption not met

# (3) Normality of error term: need to check by histogram, QQplot of residuals,
# could do Kolmogorov-Smirnov test.  Check for normal residuals
qqPlot(resid(mod1u))
```

```
## 202 190
##  34  22
```

```r
hist(residuals(mod1u))
```






```
shapiro.test(resid(mod1u))  # not normally distributed resids bc p<0.05
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(mod1u)
## W = 0.95075, p-value = 1.313e-05
```

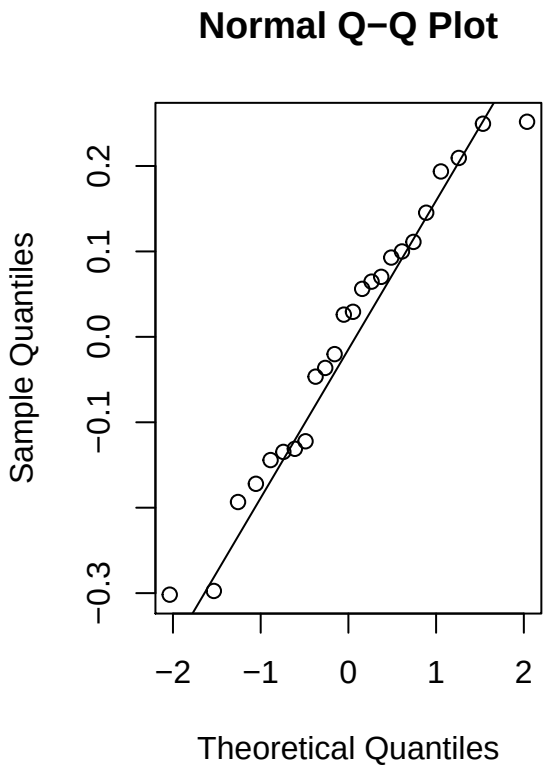
```
outlierTest(mod1u)  # no outliers
```

```
##      rstudent unadjusted p-value Bonferroni p
## 202 -4.372478         2.3404e-05    0.0039319
## 190 -4.014368         9.5531e-05    0.0160490
```

```
# (4) Normality of random effect: Get the estimate of random effect (e.g., random
# intercepts), and check them as you would check the residual.
require(lme4)
r_int_u <- ranef(mod1u)$plot$`(Intercept)`
qqnorm(r_int_u)
qqline(r_int_u)
shapiro.test(r_int_u)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  r_int_u
## W = 0.96292, p-value = 0.4999
```

```
# Normally distributed random effect pvalue > 0.05
```




```
# Do we need to include plot as a random effect with the UMBS models?
mod1u <- lmer(log(simpson) ~ state * year + insecticide * year + (1 | plot), umbs_diversity,
    REML = FALSE)
mod2u <- lmer(log(simpson) ~ state * year + insecticide + year + (1 | plot), umbs_diversity,
    REML = FALSE)
# Run analysis of variance on each model (see this for more explanation on how
# anova on a linear mixed effects model is similar to an anove on a regular
# linear model: https://m-clark.github.io/docs/mixedModels/anovamixed.html)
anova(mod1u)
```

```
## Analysis of Variance Table
##                  npar  Sum Sq  Mean Sq F value
## state               1 0.01962 0.019620  0.5281
## year                6 1.55405 0.259009  6.9714
## insecticide         1 0.05518 0.055184  1.4853
## state:year          6 0.32134 0.053557  1.4415
## year:insecticide    6 0.33277 0.055461  1.4928
```

```
anova(mod2u)
```

```
## Analysis of Variance Table
##             npar  Sum Sq  Mean Sq F value
## state          1 0.02084 0.020840  0.5281
## year           6 1.55405 0.259009  6.5632
## insecticide    1 0.05862 0.058617  1.4853
## state:year     6 0.32134 0.053557  1.3571
```

```r
anova(mod1u, mod2u)  # Go with model 2u since pvalue >0.05, aka more complex model does not have someth
```

```
## Data: umbs_diversity
## Models:
## mod2u: log(simpson) ~ state * year + insecticide + year + (1 | plot)
## mod1u: log(simpson) ~ state * year + insecticide * year + (1 | plot)
##       npar    AIC    BIC logLik deviance  Chisq Df Pr(>Chisq)
## mod2u   17 11.246 64.354 11.377  -22.754
## mod1u   23 14.557 86.408 15.721  -31.443 8.6891  6     0.1918
```

```r
summary(mod1u)
```

```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: log(simpson) ~ state * year + insecticide * year + (1 | plot)
##    Data: umbs_diversity
##
##      AIC      BIC   logLik deviance df.resid
##     14.6     86.4     15.7    -31.4      145
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.8348 -0.4337  0.1201  0.6064  2.3817
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  plot     (Intercept) 0.02926  0.1710
##  Residual             0.03715  0.1928
## Number of obs: 168, groups:  plot, 24
##
## Fixed effects:
##                                Estimate Std. Error t value
## (Intercept)                     0.13958    0.09111   1.532
## statewarmed                     0.04840    0.10521   0.460
## year2016                       -0.26894    0.09638  -2.791
## year2017                       -0.04981    0.09638  -0.517
## year2018                        0.04748    0.09638   0.493
## year2019                        0.07569    0.09638   0.785
## year2020                        0.12848    0.09638   1.333
## year2021                       -0.04939    0.09638  -0.513
## insecticideno_insects          -0.05952    0.10521  -0.566
## statewarmed:year2016            0.17915    0.11129   1.610
## statewarmed:year2017           -0.04630    0.11129  -0.416
## statewarmed:year2018           -0.03285    0.11129  -0.295
## statewarmed:year2019           -0.07803    0.11129  -0.701
## statewarmed:year2020           -0.06528    0.11129  -0.587
## statewarmed:year2021            0.09060    0.11129   0.814
## year2016:insecticideno_insects  0.09326    0.11129   0.838
## year2017:insecticideno_insects  0.31547    0.11129   2.835
## year2018:insecticideno_insects  0.16942    0.11129   1.522
## year2019:insecticideno_insects  0.16902    0.11129   1.519
## year2020:insecticideno_insects  0.18592    0.11129   1.671
## year2021:insecticideno_insects  0.13103    0.11129   1.177
```

```
##
## Correlation matrix not shown by default, as p = 21 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```
summary(mod2u)
```

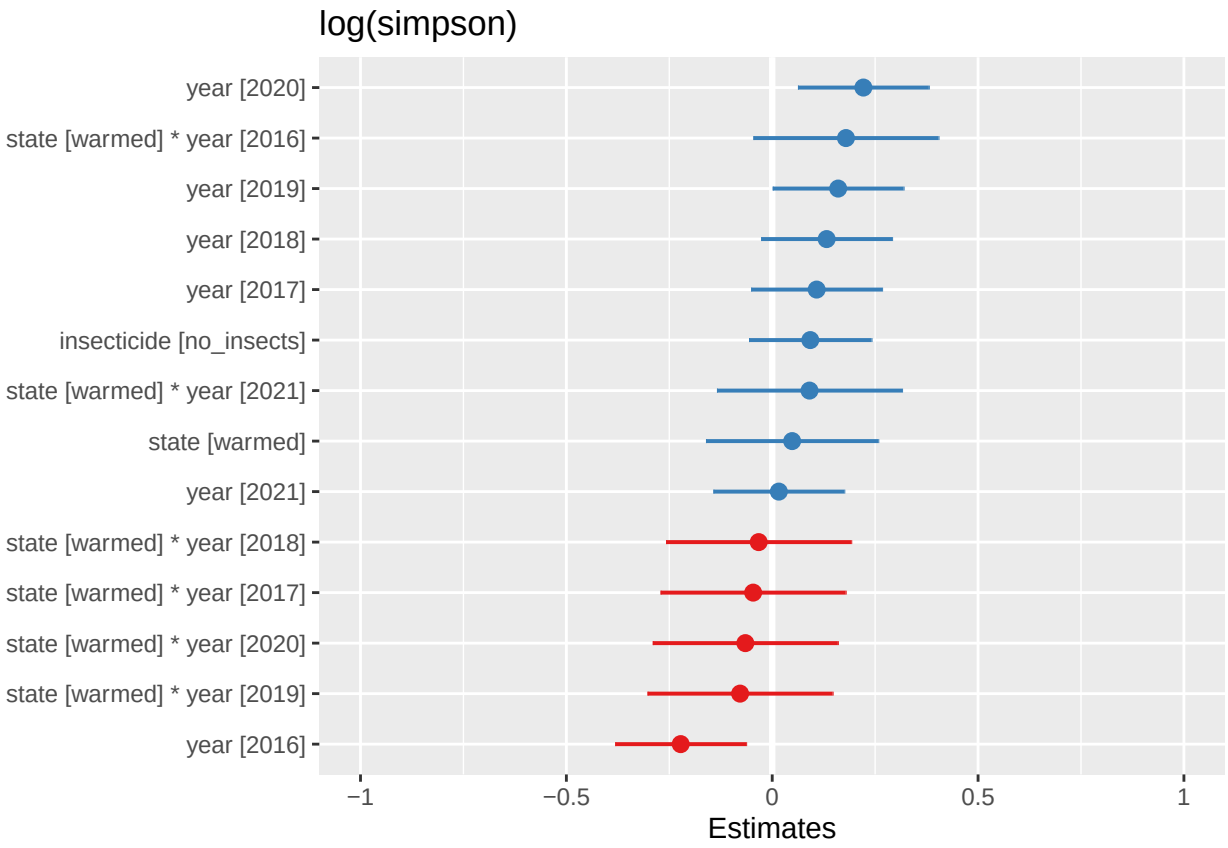
```
## Linear mixed model fit by maximum likelihood  ['lmerMod']
## Formula: log(simpson) ~ state * year + insecticide + year + (1 | plot)
##    Data: umbs_diversity
##
##      AIC      BIC   logLik deviance df.resid
##     11.2     64.4     11.4    -22.8      151
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -3.8057 -0.3803  0.1170  0.5764  2.1386
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  plot     (Intercept) 0.02893  0.1701
##  Residual             0.03946  0.1987
## Number of obs: 168, groups:  plot, 24
##
## Fixed effects:
##                       Estimate Std. Error t value
## (Intercept)            0.06357    0.08449   0.752
## statewarmed            0.04840    0.10676   0.453
## year2016              -0.22231    0.08110  -2.741
## year2017               0.10792    0.08110   1.331
## year2018               0.13219    0.08110   1.630
## year2019               0.16020    0.08110   1.975
## year2020               0.22144    0.08110   2.730
## year2021               0.01612    0.08110   0.199
## insecticideno_insects  0.09250    0.07590   1.219
## statewarmed:year2016   0.17915    0.11469   1.562
## statewarmed:year2017  -0.04630    0.11469  -0.404
## statewarmed:year2018  -0.03285    0.11469  -0.286
## statewarmed:year2019  -0.07803    0.11469  -0.680
## statewarmed:year2020  -0.06528    0.11469  -0.569
## statewarmed:year2021   0.09060    0.11469   0.790
```

```
##
## Correlation matrix not shown by default, as p = 15 > 12.
## Use print(x, correlation=TRUE)  or
##     vcov(x)        if you need it
```

```
AICctab(mod1u, mod2u, weights = T)  # model 2u
```

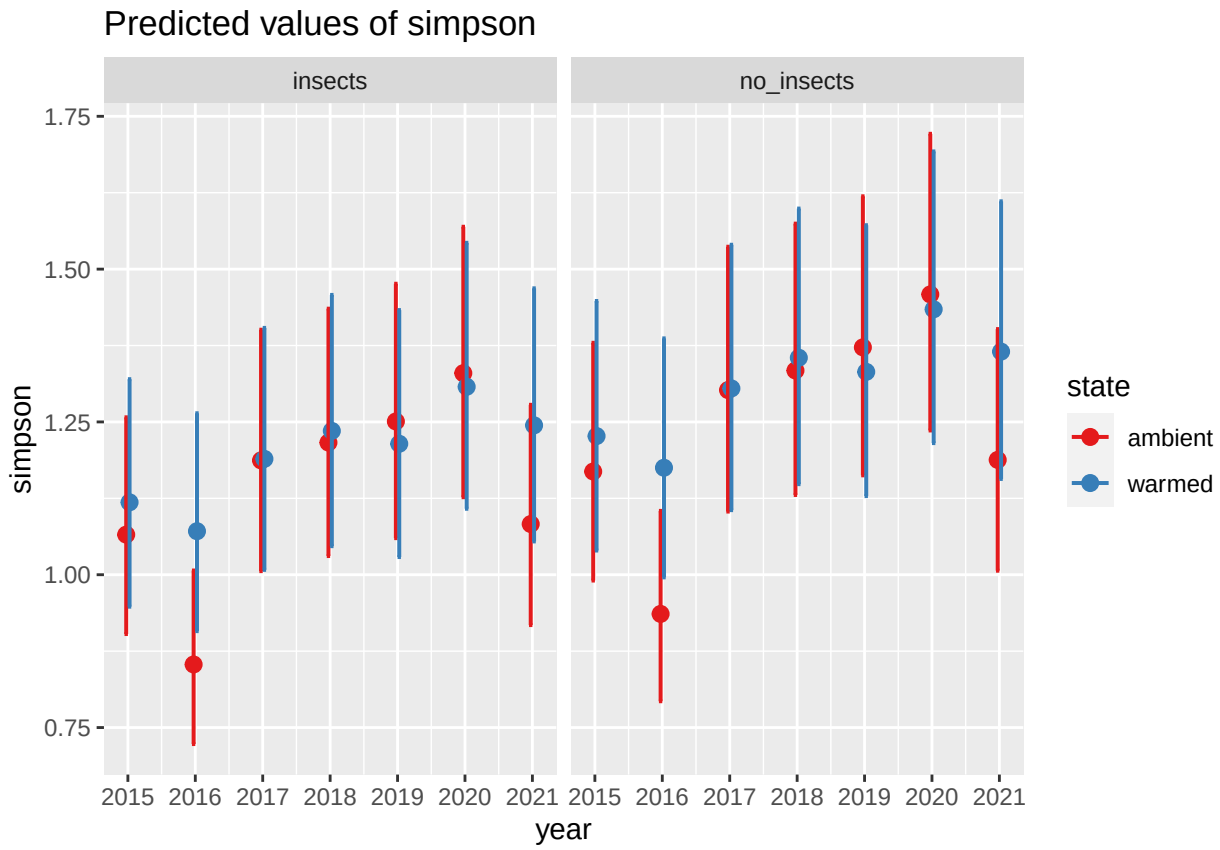
```
##       dAICc df weight
## mod2u  0.0  17 0.969
## mod1u  6.9  23 0.031
```

```
# Plot the fixed effects estimates for different models these are the fixed
# effects estimates from summary(mod1)
plot_model(mod2u, sort.est = TRUE)
```

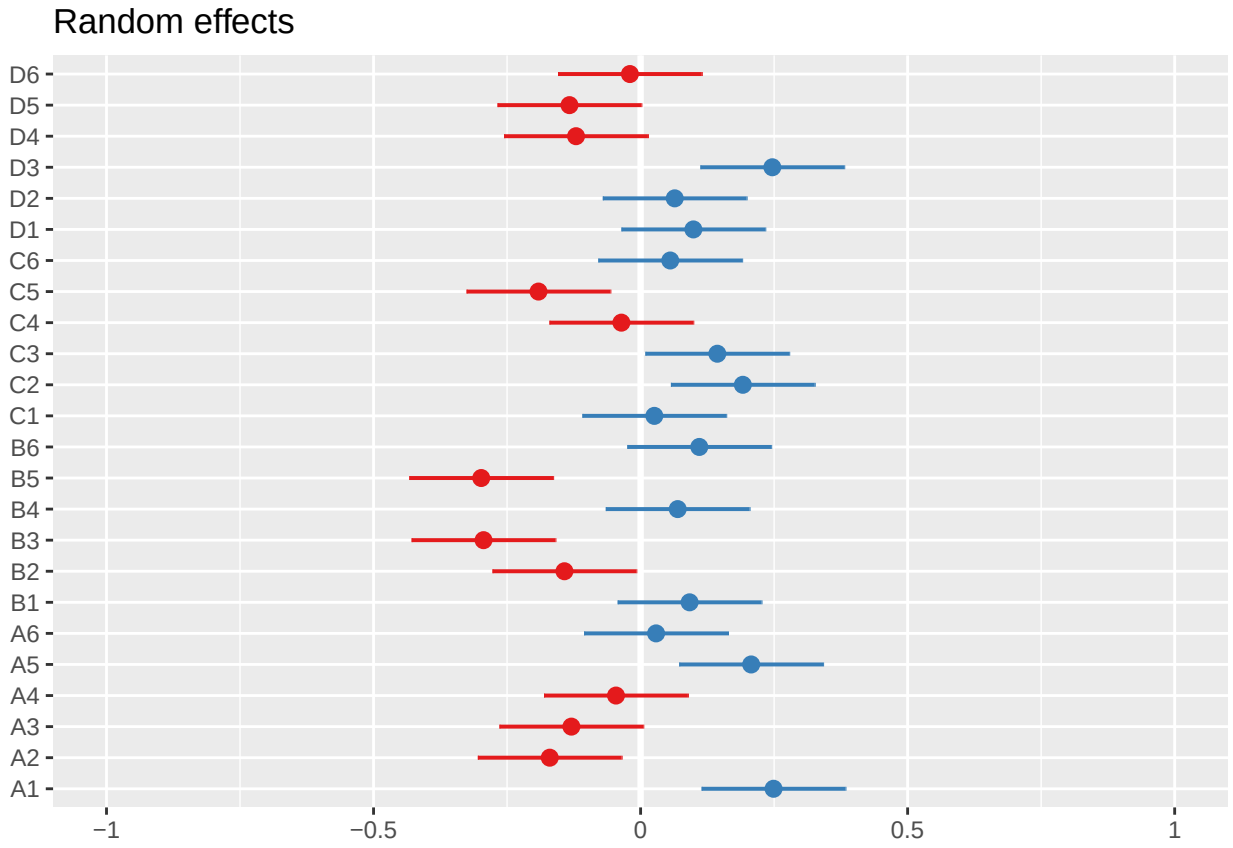



```
# these are the fixed predicted values:
plot_model(mod2u, type = "pred", terms = c("year", "state", "insecticide"))
```

```
## Model has log-transformed response. Back-transforming predictions to original response scale. Standa
```




```
# these are the random effects estimates
plot_model(mod2u, type = "re", terms = c("species"))
```




```
# Does year need to be interactive with state?
mod3u <- lmer(log(simpson) ~ state + year + insecticide * year + (1 | plot), umbs_diversity,
    REML = FALSE)
anova(mod2u, mod3u)
```

```
## Data: umbs_diversity
## Models:
## mod2u: log(simpson) ~ state * year + insecticide + year + (1 | plot)
## mod3u: log(simpson) ~ state + year + insecticide * year + (1 | plot)
##       npar    AIC    BIC logLik deviance  Chisq Df Pr(>Chisq)
## mod2u   17 11.246 64.354 11.377  -22.754
## mod3u   17 10.957 64.064 11.522  -23.043 0.2897  0
```

```
AICctab(mod1u, mod3u, weights = T)  # going with mod3u
```

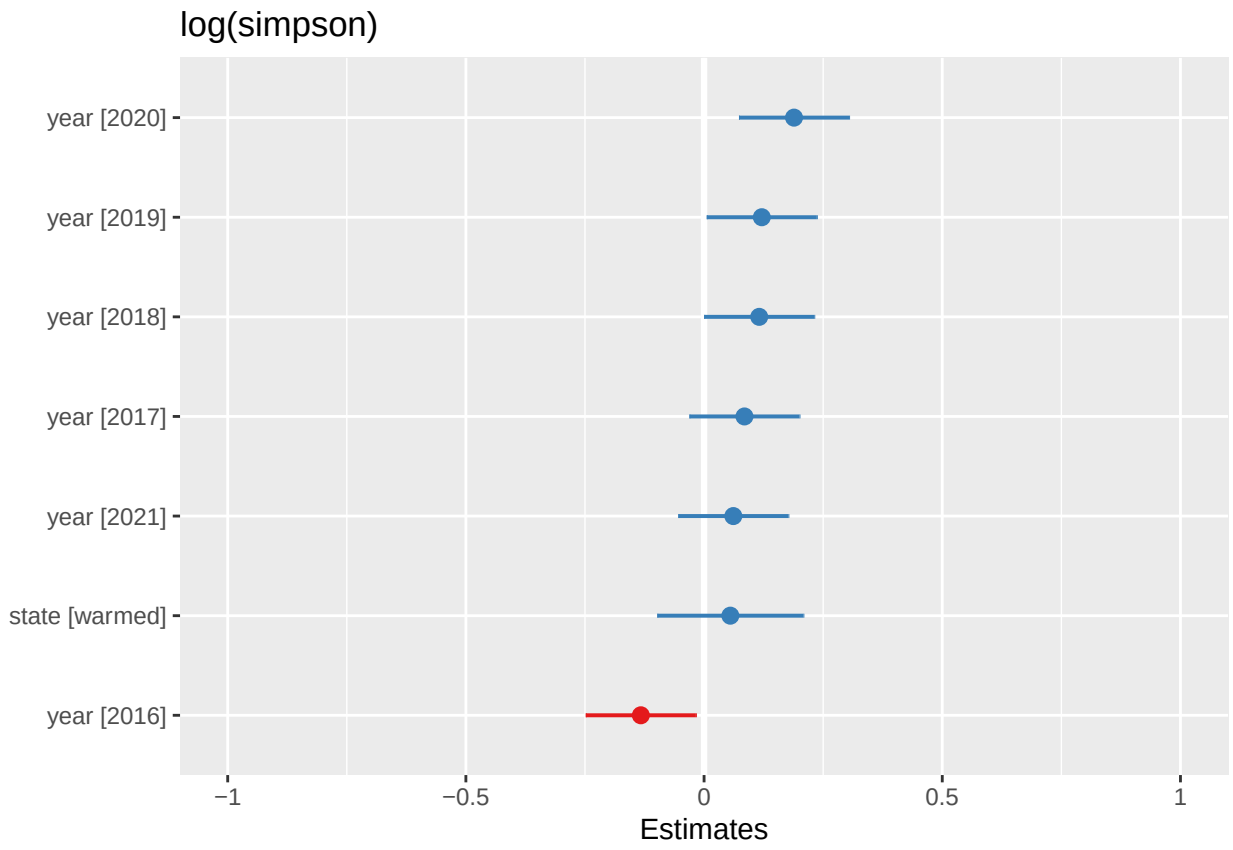
```
##       dAICc df weight
## mod3u  0.0  17 0.973
## mod1u  7.2  23 0.027
```

```
# Do we need to include insecticide? (dropping insecticide from the model)
mod5u <- lmer(log(simpson) ~ state + year + (1 | plot), umbs_diversity, REML = FALSE)
anova(mod3u, mod5u)
```

```
## Data: umbs_diversity
## Models:
```

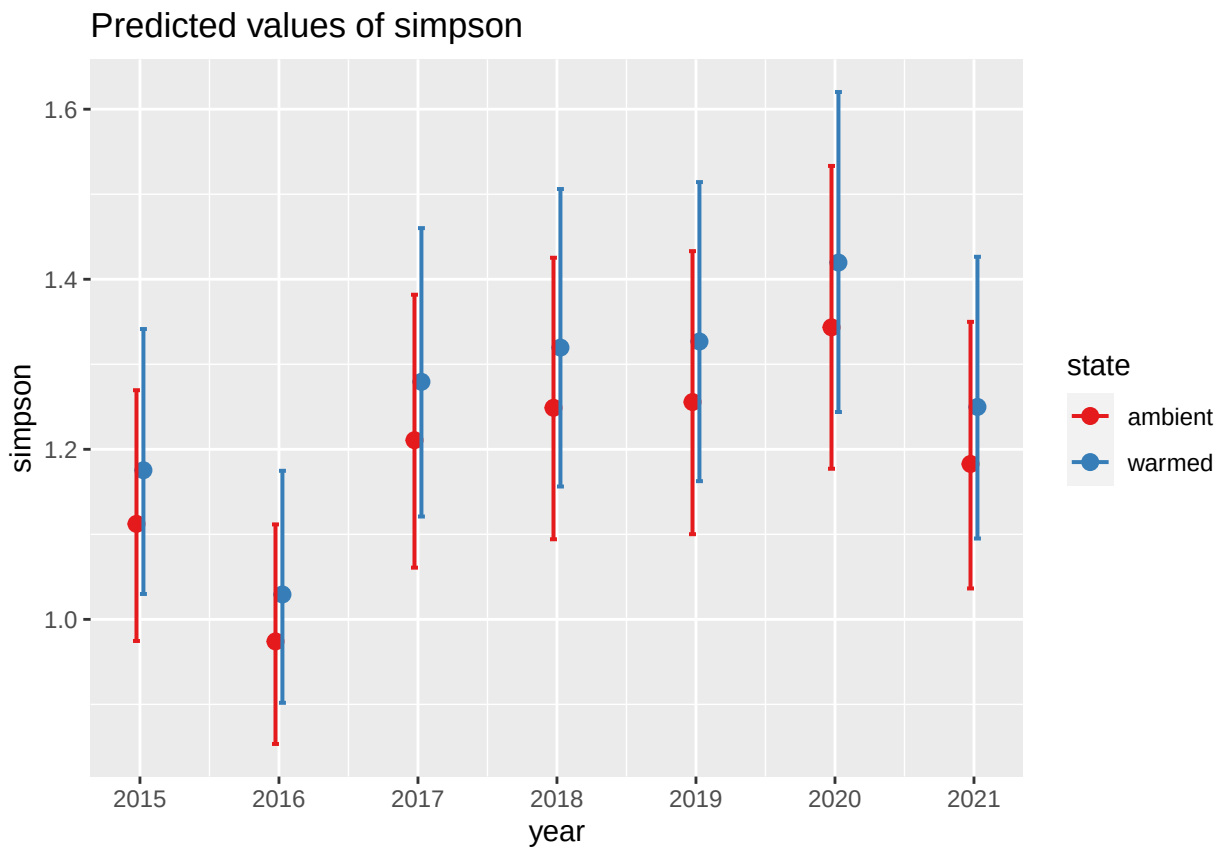
```
## mod5u: log(simpson) ~ state + year + (1 | plot)
## mod3u: log(simpson) ~ state + year + insecticide * year + (1 | plot)
##       npar     AIC    BIC  logLik deviance  Chisq Df Pr(>Chisq)
## mod5u   10  6.6083 37.848  6.6959  -13.392
## mod3u   17 10.9566 64.064 11.5217  -23.043 9.6517  7     0.2092
```

```
# No, p>0.05 so insecticide*year doesn't strongly improve model fit so we will go
# with the more simple model mod5u

# Plot the fixed effects estimates for different models these are the fixed
# effects estimates from summary(mod5u)
plot_model(mod5u, sort.est = TRUE)
```

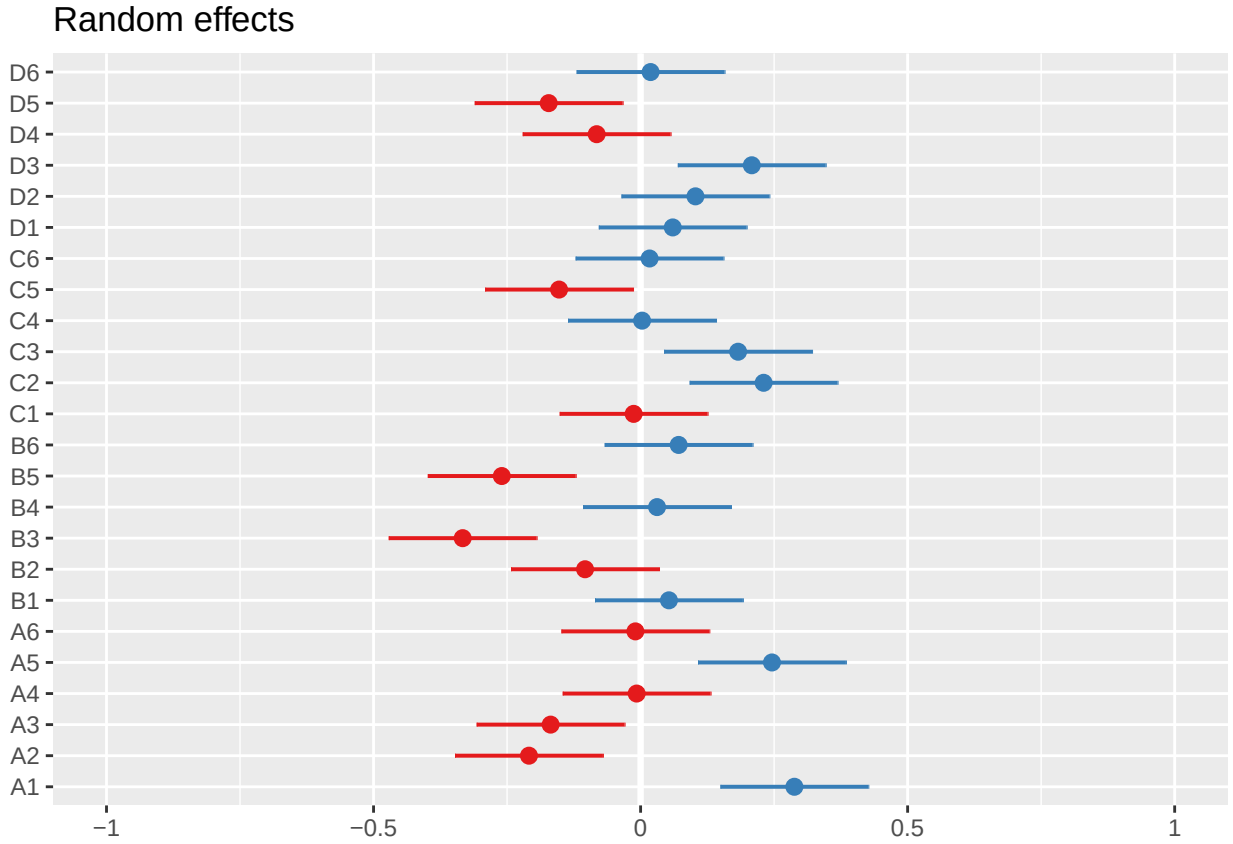



```
# these are the fixed predicted values:
plot_model(mod5u, type = "pred", terms = c("year", "state"))
```

```
## Model has log-transformed response. Back-transforming predictions to original response scale. Standa
```




```
# these are the random effects estimates
plot_model(mod5u, type = "re", terms = c("species"))
```




```
# If we wanted to include plots nested within year it would look like this: mod6
# <- lmer(log(simpson) ~ state + year + insecticide*year + (1 + year|plot),
# kbs_diversity, REML=FALSE) anova(mod5, mod6) anova(mod5) cant get mod6 to work

# the best model fit appears to be = mod5u <- lmer(log(simpson) ~ state + year +
# insecticide*year + (1|plot), umbs_diversity, REML = FALSE)
summ(mod5u)
```

Observations	168
Dependent variable	log(simpson)
Type	Mixed effects linear regression

AIC	6.61
BIC	37.85
Pseudo-R^2 (fixed effects)	0.12
Pseudo-R^2 (total)	0.49

Code below is a function written by Pat but unsuccessfully subsets sites so you get the same values for both kbs and umbs - above is a clumsy fix by Moriah (no function)

```
#' function to calculate annual diversity index for a specific site
#'
#' after reading a comp file, this function should do all that's needed to prep it and
#' run the diversity function on for each year.  diversity indexes are for the year only,
#' the diversity indexes use total abundances for a year,  do not sum/count/pool abundances in other ye
```

Fixed Effects					
	Est.	S.E.	t val.	d.f.	p
(Intercept)	0.11	0.07	1.58	51.04	0.12
statewarmed	0.06	0.08	0.71	24.00	0.49
year2016	-0.13	0.06	-2.25	144.00	0.03
year2017	0.08	0.06	1.44	144.00	0.15
year2018	0.12	0.06	1.96	144.00	0.05
year2019	0.12	0.06	2.06	144.00	0.04
year2020	0.19	0.06	3.20	144.00	0.00
year2021	0.06	0.06	1.04	144.00	0.30

p values calculated using Satterthwaite d.f.

Random Effects		
Group	Parameter	Std. Dev.
plot	(Intercept)	0.18
Residual		0.20

Grouping Variables		
Group	# groups	ICC
plot	24	0.42

```
#'
#' @param comp plant composition data as read from project folder
#' @param site one of kbs or umbs as coded in the comp data
#' @param div_index is the same as 'index' for vegan::diversity function 'shannon', 'simpson' or 'invsi
#'
#' @returns a matrix (data frame) of diversity indices for one site with years in the columns, and plot
#'
diversity_by_year <- function(comp, site, div_index = "shannon") {
    comp_site <- subset(comp, site == site) %>% dplyr::select(plot, species, cover,
        year)

    # remove non-species using 'not in' obs_to_exclude = c('Bare_Ground',
    # 'Unknown','Brown','Litter', 'Vert_Litter', 'Animal_Disturbance') comp_site
    # <-dplyr::filter(comp_site, !(species %in% obs_to_exclude))

    # convert the abundance data to abundance for each species in columns for the
    # vegan package
    comp_wide <- matrify2(comp_site)

    # comp_wide_data is assumes to have columns Year, Plot, and columns for each
    # species found, e.g. for Vegan

    # first, split up the wide data into a list of years.  Each list item is a year
    # of data
    comp_wide_by_year <- dplyr::group_by(comp_wide, Year) %>% dplyr::group_split()

    # we need to add plot names.  Get those Plot names by taking a column from any
```

```
    # one of the years since we are assuming the Plot column is the exact same across
    # years and IN THE SAME ORDER
    plot_names <- comp_wide_by_year[[1]]$Plot

    # remove the plot and year columns from each item in the list so that Vegan will
    # work.  This assumes row order is the exact same for all years (each row a plot)
    comp_wide_by_year <- lapply(comp_wide_by_year, dplyr::select, c(-Year, -Plot))

    # apply the diversity function to each year - in this case the main index is
    # plot, each year considered separately
    diversity_by_year_list <- lapply(comp_wide_by_year, vegan::diversity, index = div_index)

    # each item in the list is a year of diversity, so name those with the years we
    # know we have
    names(diversity_by_year_list) <- as.character(2015:2021)

    # 'unlist' and create a new data frame, each year a column, each row a plot, and
    # add a new row with the plot names
    x <- do.call(cbind, diversity_by_year_list) %>% cbind(Plot = plot_names) %>%
        as.data.frame()
    # an alternative tidyverse way x<- diversity_by_year(diversity_by_year_list)

    ## optional step!
    return(x)
}

comp$cover <- as.numeric(comp$cover)

# use the one function above to both matrify and calculate Shannon diversity
# index per year
diversity_by_year_kbs <- diversity_by_year(comp, site = "kbs", div_index = "shannon")
diversity_by_year_umbs <- diversity_by_year(comp, site = "umbs", div_index = "shannon")

# this output has a column for each year 2015, 2016, and Plot, but if you need it
# narrow use 'melt' from reshape2:
library(reshape2)
diversity_by_plot_year_kbs <- reshape2::melt(diversity_by_year_kbs, id = "Plot",
    variable.name = c("Year"), value.name = "shannon")
diversity_by_plot_year_umbs <- reshape2::melt(diversity_by_year_umbs, id = "Plot",
    variable.name = c("Year"), value.name = "shannon")

# To do just August (peak_comp):

peak_comp <- dplyr::filter(comp, month == 8)

peak_shannon_by_year_kbs <- diversity_by_year(peak_comp, site = "kbs", div_index = "shannon")
peak_shannon_by_year_umbs <- diversity_by_year(peak_comp, site = "umbs", div_index = "shannon")

peak_simpson_by_year_kbs <- diversity_by_year(peak_comp, site = "kbs", div_index = "simpson")
peak_simpson_by_year_umbs <- diversity_by_year(peak_comp, site = "umbs", div_index = "simpson")

# this output has a column for each year 2015, 2016, and Plot, but if you need it
# narrow use 'melt' from reshape2:
```

```
peak_shannon_by_plot_year_kbs <- reshape2::melt(peak_shannon_by_year_kbs, id = "Plot",
    variable.name = c("Year"), value.name = "shannon")
peak_shannon_by_plot_year_umbs <- reshape2::melt(peak_shannon_by_year_umbs, id = "Plot",
    variable.name = c("Year"), value.name = "shannon")

# this output has a column for each year 2015, 2016, and Plot, but if you need it
# narrow use 'melt' from reshape2:
peak_simpson_by_plot_year_kbs <- reshape2::melt(peak_simpson_by_year_kbs, id = "Plot",
    variable.name = c("Year"), value.name = "simpson")
peak_simpson_by_plot_year_umbs <- reshape2::melt(peak_simpson_by_year_umbs, id = "Plot",
    variable.name = c("Year"), value.name = "simpson")

diversity_kbs <- left_join(peak_shannon_by_plot_year_kbs, peak_simpson_by_plot_year_kbs)
diversity_kbs$site <- "kbs"  #add site column

diversity_umbs <- left_join(peak_shannon_by_plot_year_umbs, peak_simpson_by_plot_year_umbs)
diversity_umbs$site <- "umbs"  #add site column
```

Calculating Diversity Indices

```
# species richness
sppr <- specnumber(comp1_wide)

sppr_aov <- aov(sppr ~ state, data = meta)
summary(sppr_aov)
```