Handout: Temporal Diversity

Z620: Quantitative Biodiversity, Indiana University

February 10, 2017

OVERVIEW

Biological diversity varies through time in response to changes in the biotic and abiotic environment. Rising temperatures and CO_2, trends in precipitation and drought, and the dynamics of migration and colonization of species are just a few factors that influence biodiversity through time.
While some biologists focus on processes that occur on contemporary scales (minutes, days, weeks, years), other scientists are interested in the forces that drive patterns of biodiversity in the fossil record and over geologic time scales. In this handout, we will be adding a temporal dimension to the site-by-species matrix. You will be introduced to R data handling packages that will help you manage and manipulate this data structure. You will then learn the basics of time series analysis, repeated-measures analysis of variance, temporal beta diversity (i.e., turnover), as well as features relating to community stability.

1) SETUP

Retrieve and Set Your Working Directory

```
rm(list=ls())
getwd()
setwd("~/GitHub/QB-2017/Week5-Temporal/")
```

Install Packages

This module will require several R packages. We will describe them in greater details in the sections below. For now, let's just load them. The `require()` function in `R` returns `TRUE` if the package was successfully loaded or `FALSE` if the package failed to load. This `for` loop loads each package and installs the package when `require()` returns `FALSE`.

```
package.list <- c('vegan', 'tidyr', 'dplyr', 'codyn', 'ggplot2',
      'cowplot', 'MullerPlot', 'RColorBrewer', 'reshape2', 'lubridate',
      'TTR', 'xtable', 'multcomp', 'pander', 'png', 'grid', 'tseries', 'nlme',
      'forecast', 'lsmeans')
for (package in package.list) {
  if (!require(package, character.only = TRUE, quietly = TRUE)) {
    install.packages(package, repos='http://cran.us.r-project.org')
    library(package, character.only = TRUE)
  }
}
```

2) LOADING DATA

To learn about topics of temporal diversity, we will use the long-term rodent dataset from the Chihuahuan Desert ecosystem near Portal, Arizona. Known as the "Portal Project" http://portal.weecology.org/, this research site was initiated in 1977 by Jim Brown and colleagues to study species interactions, specifically competitive dynamics within and among species of rodents and ants. Once every month since its inception,

members of The Portal Project monitor 24 experimental plots, each 0.25 ha in area (50 X 50 m). In total, 4.8 kilometers of fencing is used to maintain the Portal experiment!

Let's take a look at the data:

```
portal <- read.table("data/combined.csv", sep = ",", header = TRUE)
```

Take a few minutes to explore the structure of this dataset. The version of the data that we are working with spans from 1977 - 2002. During this period, individual rodents were captured from plots ("plot_id"), identified to species ("species_id"), which led to the creation of a "record_id" with an associated date (day, month, and year). In addition, the sex, size ("hindfoot_length" and "weight") and taxonomic identity ("genus" and "species") of animals were recorded. All of this was done in five experimentally replicated treatments (see Figure):

1) Controls - fencing around plots does not exclude rodents (plot_id: 2, 4, 8, 11, 12, 14, 17, 22)
2) Long-term Krat - long-term exclusion of Kangaroo rats (*Dipodomys* spp.) (plot_id: 3, 15, 19, 21)
3) Short-term Krat - short-term exclusion of Kangaroo rats (*Dipodomys* spp.) (plot_id: 6, 13, 18, 20)
4) Rodent Exclosure - exclusions of all rodents (plot_id: 5, 7, 10, 16, 23, 24)
5) Spectab exclosure - exclusion of Banner-tailed kangaroo rat (*Dipodomys spectabilis*) (plot_id: 1, 9)

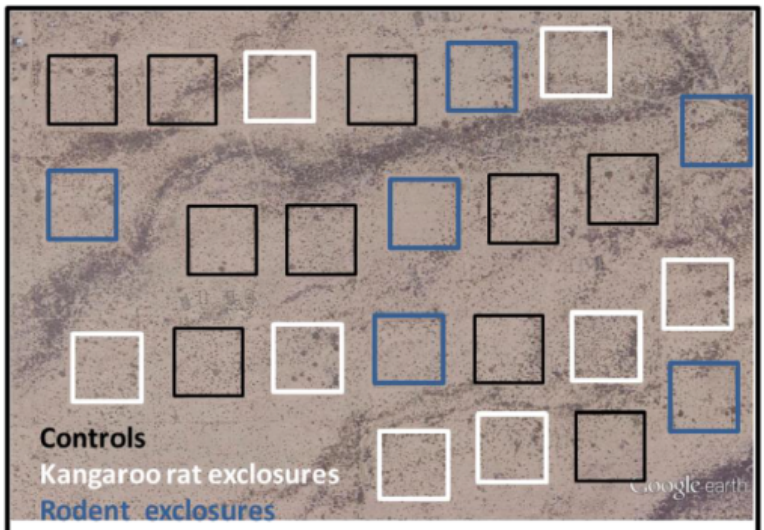


Fig 3. Experimental plots at the 20 ha Portal site. Black: Control plots where all rodents have access. White: Only kangaroo rats excluded. Blue: All rodents excluded.

3) DATA WRANGLING

One thing you may notice when looking at `portal` is that it has a lot of observations, almost 35,000 (and remember this is only 1977 - 2002). These data are entered in **long format**, which means that each row represents a unique observation (i.e., a trapped animal). The long format is a convenient way to enter data that reduces the probability of entry errors. However, manipulating long-format data for visualization and statistical analysis is often a challenge. For example, many R packages and functions require you to organize your data in **wide format** where different sampling time points would be represented in columns.

In the following sections, we will demonstrate how to manipulate the `portal` data set to carry out different types of analyses using functions from the `dplyr` and `tidyr` packages. These packages were especially designed for transforming, subsetting, and summarizing tabular data. The functions contained in these two packages (and others like `apply()` and `aggregate()`) are preferred for manipulating data in R because they are faster and more efficient than other control-flow statements (e.g., loops) that are commonly used in other programming languages.

First, let's use the `unite()` function from the `tidyr` package to create new columns that contain information in other columns. The "remove = FALSE" statement retains the original data vectors.

```
# Make a date vector that contains year, month, and day
portal <- unite(portal, col = date, c(year, month, day), sep = "-", remove = FALSE)

# Make a taxon vector that contains genus and species names
portal <- unite(portal, col = taxon, c(genus, species), sep = "_", remove = FALSE)
```

Now, we are going to use `dplyr` to create a time-by-species matrix. To do this, we are going to use a new operator referred to as a "pipe" (`%>%`). Pipes allow output from one function to be used as input for another function in the same line of code. When using pipes, the output from the function to the left becomes the first argument of the function on the right.

In the following R chunk, we will build a time-by-species matrix. First, we will use a `dplyr` function called `group_by()`, which splits the dataset up according the variable(s) supplied. We then use the `count()` function to sum the number of individuals belonging to each taxon while maintaining the year and plot_id grouping. The fill argument assigns a value of zero if there are no values for a given combination of groupings. Finally, we use the `spread()` function from `tidyr` to convert the long format output of the `count()` function into a wide format dataset. The `key =` argument specifies the new column names and the `value =` argument specifies the values that will become the elements of the wide form dataset.

```
time.by.species <- group_by(portal, year, plot_id) %>%
  count(taxon) %>% spread(key = taxon, value = n, fill = 0)
```

Take a moment to inspect the structure of the `time.by.species` object we just created. From this data structure, we can retrieve different different types of information. For example, if we want the site-by-species matrix for 1984 we can use the following code:

```
dplyr::filter(time.by.species, year == 1984) # return 1984 site-by-species
```

And if we want the species data from plot 5 for all years, we could use the following code:

```
dplyr::filter(time.by.species, plot_id == 5) # return plot5 time-by-species
```

Last, let's convert the `tidyr` object to a data frame so it can be used with functions from other packages:

```
time.by.species <- as.data.frame(time.by.species)
```

4) TIME SERIES ANALYSIS - A PRIMER

In this section, we introduce some basic concepts and tools used in time series analysis. One goal of time series analysis is to detect and decompose trends in temporal data that are independent from other sources of variability (e.g., seasonality, cycles, and error). Another goal of time series analysis is to make **forecasts** about a system into the future. We will demonstrate some of these tools using the abundance of rodents for a single site in the Portal Project. Throughout the year, total rodent abundance may fluctuate widely in response to multiple stimuli, one of which is likely precipitation. Because the Chihuahuan Desert sits in a rain shadow created by the Sierra Madre Mountains, the Portal site is an arid ecosystem with rain falling mostly during the months of June through October. This variability in precipitation provides an opportunity

for us to detect a signal of this **seasonality** using time series analysis while also testing for long-term trends and forecasting rodent densities into the future.

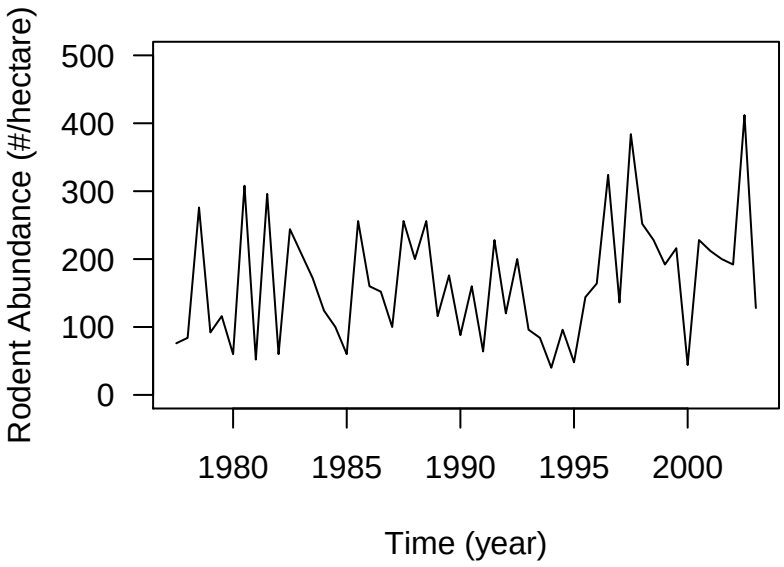
Data wrangling

In the following R chunk, we manipulate `portal` using some of the tools introduced above from the `dplyr` and `tidyr` packages. Again, we use `count()` to sum the number of individuals belonging to each taxon while maintaining the grouping by year, month, and plot_id. We retain month in this case so we can create a categorical variable called "season".

```
# Create a time-by-species matrix that includes year, month, and plot_id
# time.by.spec.2 <- group_by(portal, year, month, plot_id) %>% count(taxon)
# Before we were using all taxa, not just rodents

time.by.spec.2 <- filter(portal, taxa=="Rodent") %>%    # Select only rodents
  group_by(year, month, plot_id) %>%
  count(taxon)

# Create a seasonality variable using month number (6 = June; 10 = October)
time.by.spec.2$season <- NA
time.by.spec.2$season <- time.by.spec.2$month %in% c(6:10)

# Rainy seasons are June - October
time.by.spec.2$season <- ifelse(time.by.spec.2$season == TRUE, "rain", "norain")

# Group the data by year and season
group_by(time.by.spec.2, year, season)
```

Now we are going to use the `filter()` function in `dplyr` to pick a plot. Next, we calucalte rodent abundance per hectare (0.25 ha * 4 = 1 ha) for each season within a year. Then, we created a create a time series object that identifies the start time and seasonality of our sampling, which is determined by the frequency argument.

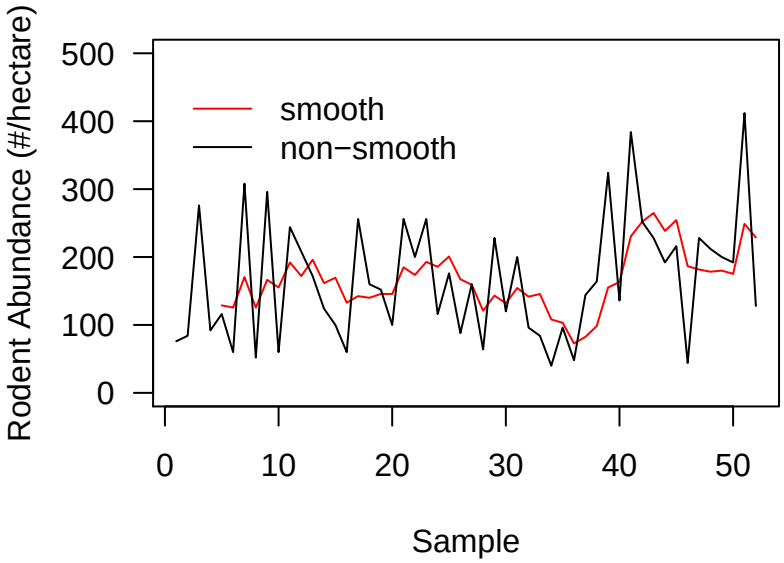
```
abund <- filter(time.by.spec.2, plot_id == 2) %>%
  group_by(year, season) %>%
  count(wt = n)

abund$nn <- abund$nn * 4

abund.ts <- ts(abund$nn, frequency = 2, start = c(1977, 2))

plot.ts(abund.ts, type = "l", ylab = "Rodent Abundance (#/hectare)",
        xlab = "Time (year)", las = 1, ylim = c(0, 500))
```



Smoothing

One technique that is commonly used, mostly for visualizing trends in a time series is smoothing. Smoothing techniques attempt to remove noise from a data set, and are used to help detect signal in a dataset. The simplest form of smoothing is the **simple moving average**. Simple moving average calculates an unweighted arithmetic mean of your observations across a window of past n observations.
Let's give this a try using the `SMA()` function, which is contained in the `TTR` package. (Note that this package does not accept a `ts` object) Play around with n to visualize the effect of the smoothing window.

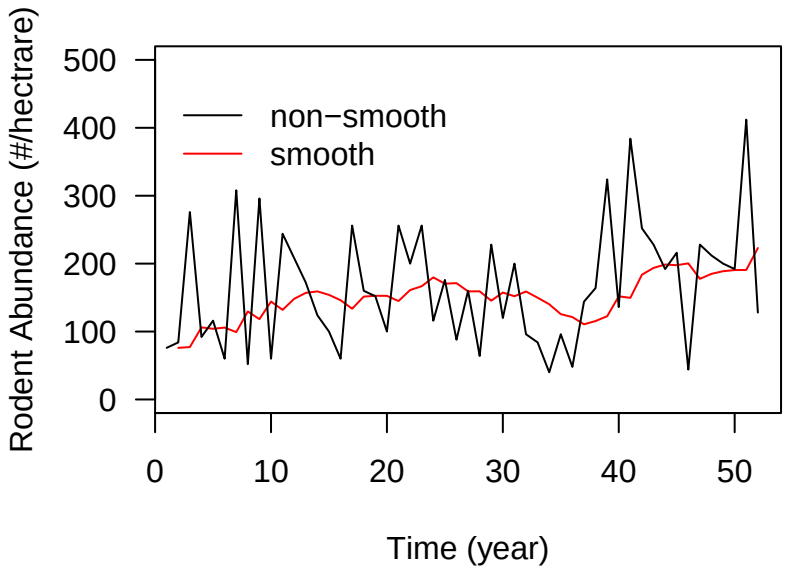
```
abund.sm <- SMA(abund$nn, n = 5)
plot(abund.sm, type = "l", col = "red", ylab = "Rodent Abundance (#/hectare)",
     xlab = "Sample", las = 1, ylim = c(0, 500))
lines(abund$nn, col = "black")
legend(0, 475, col = c("red", "black"), lty = c(1,1),
       c("smooth", "non-smooth"), bty = "n", cex = 1)
```



Exponential smoothing is another way to smooth a time series. This process places exponentially less

weight on past observations. The Holt-Winters filtering technique is commonly used for exponential smoothing. In R, the `HoltWinters()` function in the `stats` package allows one to specify two parameters. The beta argument when set to FALSE performs exponential smoothing, while the gamma argument is used to specify seasonality. In the output, you will get an alpha parameter, which ranges from 0 to 1. Values closer to 1 indicate that the smoothing is influenced by current observations, while values closer to 0 indicate that more weight is placed on past observations.

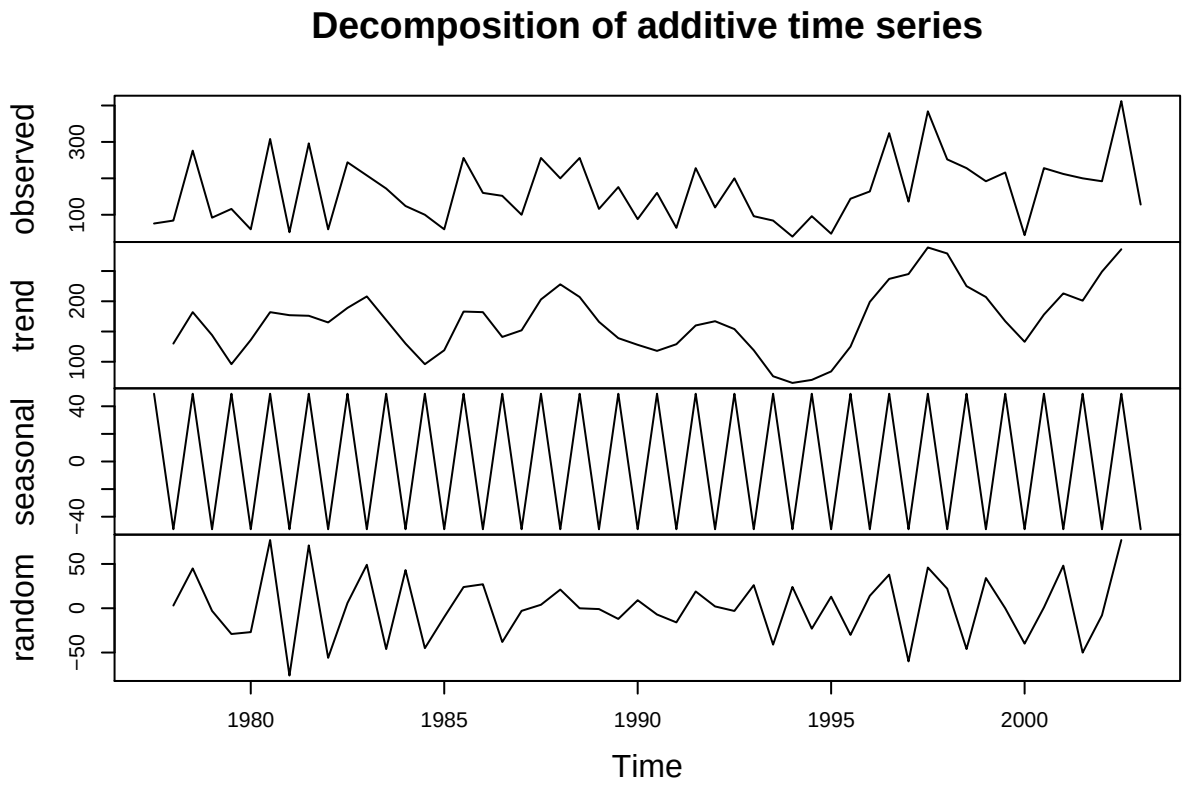
```
abund.hw <- HoltWinters(abund$nn, beta = FALSE, gamma = FALSE)
# abund.hw$fitted
plot(abund.hw, xlab = "Time (year)", ylim = c(0, 500),
     ylab = "Rodent Abundance (#/hectrare)", las = 1, main = NA)
legend(0, 475, col = c("black", "red"), lty = c(1,1),
       c("non-smooth", "smooth"), bty = "n", cex = 1)
```



Decomposition of a time series

A time series can be broken down into different "categories" to gain insight into what is giving rise to patterns. The common categories of a time series decomposition are the trend, seasonal, cycle, and residual error. As mentioned above, we subsampled the `portal` data set so that it included seasonal variation in precipitation. In the following section we will use the `decompose()` function to look at different decomposition categories. If seasonal trends are a nuisance in your study, there are ways to remove them, as done in the chunk below.

```
# moving average decomposition
abund.comp <- decompose(abund.ts)

# plot decomposition categories
plot(abund.comp)
```




```
# remove seasonality
abund.adj <- abund.ts - abund.comp$seasonal

# Turn this into a question
# # compare plots of seasonality corrected and uncorrected time series
# plot.ts(abund.adj, ylab = "Rodent Abundance (#/hectrare)",
#         xlab = "Time (year)", las = 1, col = "red", ylim = c(0, 500))
# lines(abund.ts, col = "black")
# legend(1977, 475, col = c("black", "red"), lty = c(1,1),
#        c("uncorrected", "corrected"), bty = "n", cex = 1)
```

Auto-regressive moving average (ARMA) models

Auto-regressive moving average (ARMA) models are commonly used to model a time series. ARMA approaches can identify trends in data and help to make forecasts about future observations. As such, ARMA is often used in biology and the environmental sciences, but also in business and economics. In this section we walk through the basic ways to implement ARMA in R.

The autoregressive component of an ARMA model uses regression to obtain *coefficients* that predict current observations using previous or "lagged" observations from a time series, which can be described with the following equation:

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \epsilon_t$$

where Y_t is an observation at time t, c is a constant, ϵ is the error term, and ϕ is the fitted parameter associated with the autoregressive order (or lag) p.

The moving average component of an ARMA model should not be confused with the moving average techniques that were described in the section above on smoothing. Rather, the moving average component of an ARMA model uses multiple regression to recover *error terms* of the current and prior observations, which can be described with the following equation:

$$Y_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-1}$$

where μ is the mean of the time series (Y), θ is fitted parameter, and ϵ refers to errors associated with order (or lag) q. The full ARMA model can then be expressed as follows

$$Y_t = c + \sum_{i=1}^{p} \phi_i Y_{t-i} + \epsilon_t + \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-1}$$

Assumptions of ARMA models: stationarity

If the mean, variance, or covariance in a time series is affected by time, then we are likely to be violating the assumption of **stationary**. If the assumption of stationarity is not met, corrective measures should be taken, which could involve transforming or differencing the data. In the following R chunk, we use the `adf.test()` function in the `tseries` package, which implements the Augmented Dickey-Fuller Test.
This statistic tests the stationarity of a time series where the null hypothesis is that the time series is non-stationary. Thus, we will conclude that a time series is stationary if the p-value < 0.05.

```
adf.raw <- adf.test(abund.ts, alternative = "stationary")
adf.raw$p.value
```

```
## [1] 0.3229509
```

The Dickey-Fuller test indicates that our time series does not meet the assumption of stationarity. Let's try differencing the time series using the `diff()` function, which will create a new time series where $Y_d = Y_t$ - Y_{t-1}. We will then re-run the Dickey-Fuller test and take a look at a plot of the differenced data.

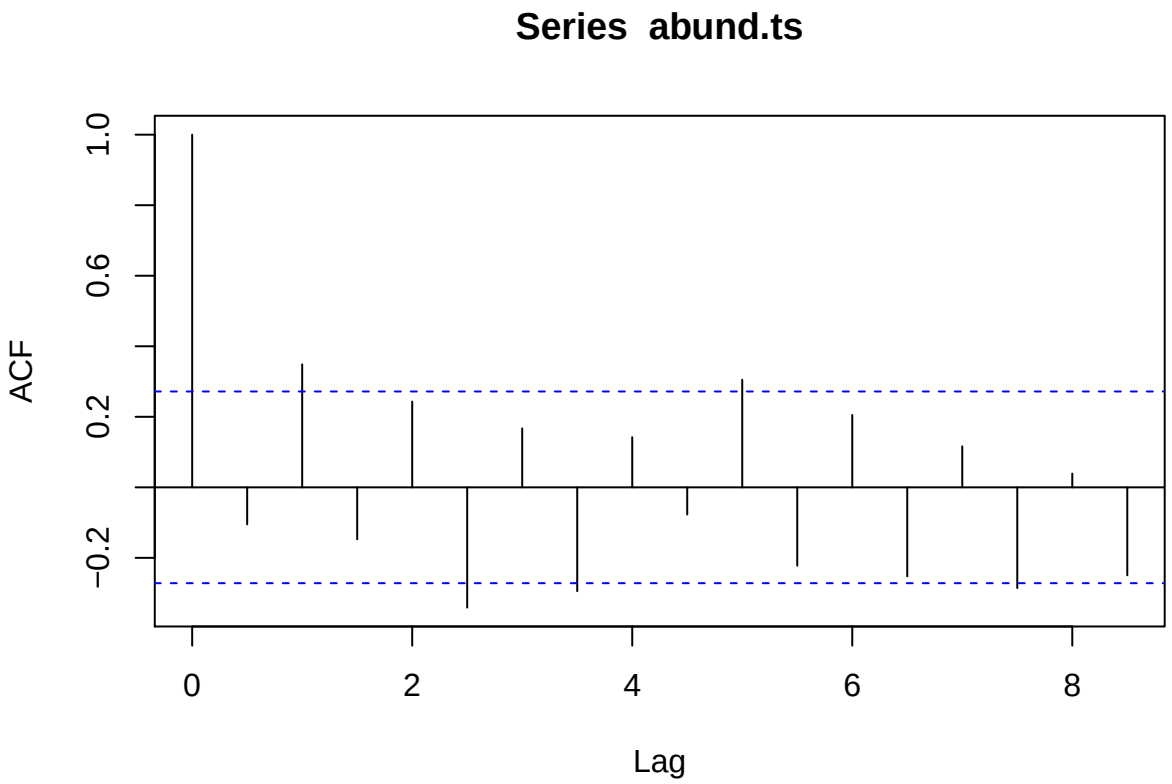
```
abund.ts.diff <- diff(abund.ts)
adf.diff <- adf.test(abund.ts.diff, alternative = "stationary")
adf.diff$p.value
```

```
## [1] 0.02186602
```

```
#plot.ts(abund.ts.diff, ylab = "Rodent Abundance (#/hectare)", xlab = "Time (year)", las = 1)
```

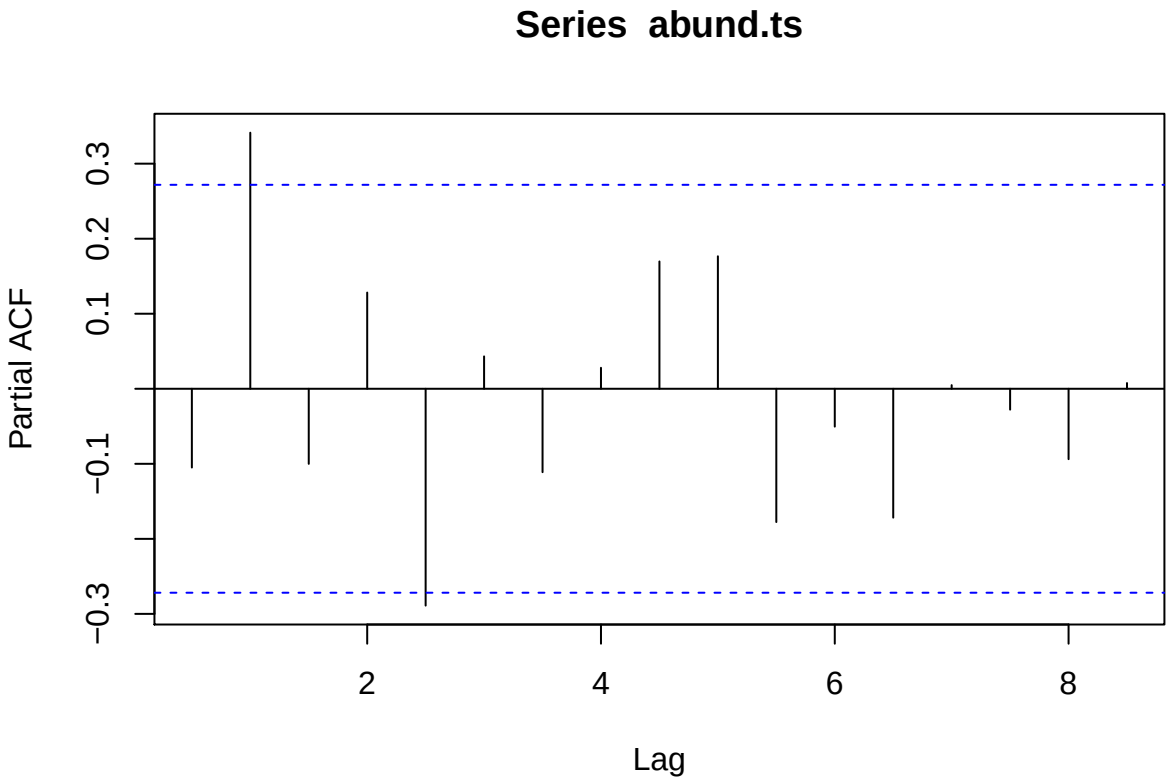
The next step in the ARMA pipline is to look at the lags in our time series using the **autocorrelation function (ACF)**. These plots help us visualize structure in our data, but also help inform the parameterization of our ARMA models. Specifically, the ACF tells us about the lags of the forecast errors and thus provides information for the MA component of the model. The ACF simply looks at the correlation between lagged intervals in the time series. When lag = 0, ACF will always equal 1. After that, we expect the ACF to decline with increasing time lag. If we see that there is a significant correlation at a given lag, this information can be used to parameterize our AMRA models. For example, in the non-differenced time series we see that there is a significant positive correlation at lag = 2. Recall, the frequency of our data divides the year up into rainy and non-rainy season. So the lag = 2 correlation could be reflecting autocorrelation of error terms on an annual cycle.

```
acf(abund.ts)
```



We will also look at the time series using the **partial autocorrelation function (PACF)**. PACF calculates correlations between two series after correcting for correlation that might exist with another lagged series. In contrast to the ACF, significant partial correlation coefficients tell us about lags that can be addressed with the autoregressive component of the ARMA model. In the following PACF, we can see that there is a significant partial correlation at lag = 2 and lag = 5.

```
pacf(abund.ts)
```

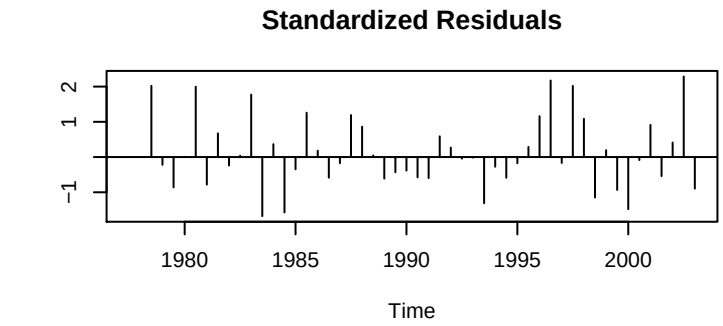


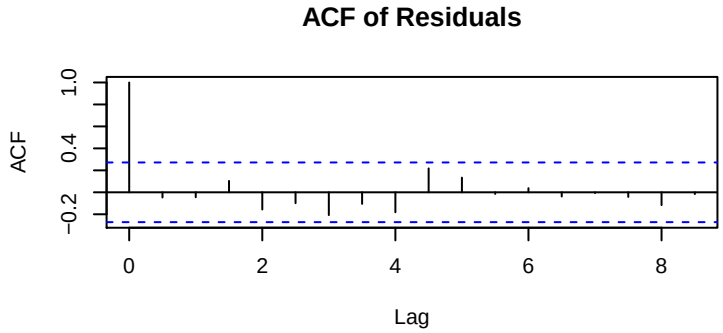
Now, we will try to arrive at our final ARMA model. Actually, because we found that differencing helped us meet the assumption of stationarity, we are going to use an auto-regressive *integrated* moving average (ARIMA) model. ARIMA models accept parameters (p, d, and q) that describe the number of autoregressive lags (inferred from PACF), differencing, and order of the moving-averge model (inferred from ACF), respectively. We are going to use the function `auto.arima` from the `forecast` package to identify the best ARIMA model based on information criteria (AIC, AICc, and BIC). This function is particularly useful because we have some complexities related to the effect of seasonality on our parameters. After identifying the ARIMA model, we can look at the diagnostics using `tsdiag` and then make forecasts about rodent densities into the future using the `predict()` function from the `arima` package.

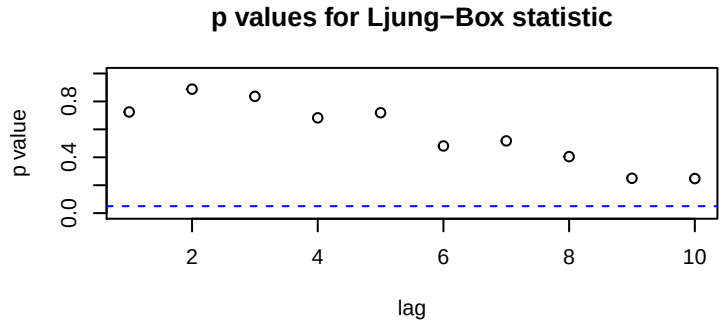
```
abund.arm <- auto.arima(abund.ts)
abund.arm <- arima((abund.ts), c(0, 0, 1), seasonal = list(order = c(2, 1, 0),
                  period = 2), include.mean = TRUE)
```

Let's look at some diagnostic plots for the ARIMA model

```
tsdiag(abund.arm)
```

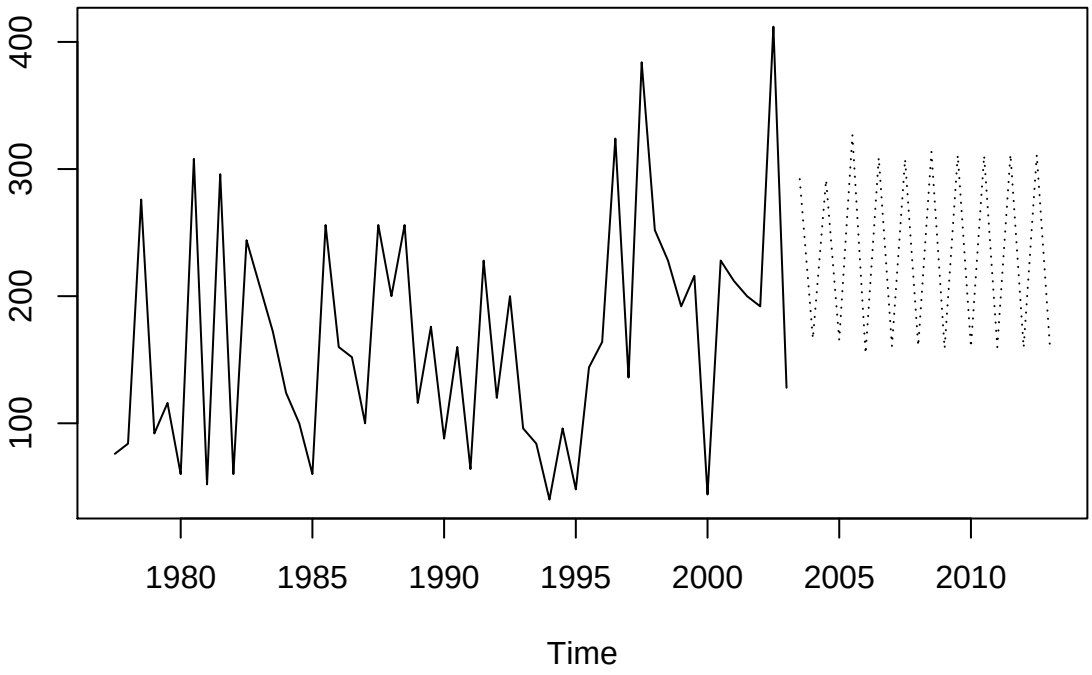






Finally, let's use predictions from the model to forecast future rodent abundances in the study plot:

```
pred.arm <- predict(abund.arm, n.ahead = 20)
ts.plot(abund.ts, pred.arm$pred, lty = c(1,3))
```



5) REPEATED MEASURES ANALYSIS OF VARIANCE (RM-ANOVA)

When scientists set up an experiment, they are inclined to take more than a single measurement on a subject (i.e., experimental unit). This is true for laboratory and field experiments in ecology and evolution, but is also common in other fields such as medicine and psychology. Studies that take repeated measurements on a subject are referred to as **longitudinal designs**. It is important to emphasize that repeated observations on a subject are non-independent. Independence is a critical assumption of most parametric statistics, like analysis of variance (ANOVA). RM-ANOVA explicitly accounts for this non-independence and allows one to test for the main effects of time and treatment along with the time x treatment interaction.
In the following section, we will show you how to perform and interpret an RM-ANOVA using richness data from the Portal Project.

Wrangle data for RM-ANOVA

In the following R chunk, we calculate richness over time from two of the Portal plots. The first treatment is the "Control" where rodents where free to move in and out of plots. The second treatment is the "Rodent Exclosure", which attempted to exclude all rodents. To create the necessary data frame, we use `dplyr` to construct a time-by-species matrix. We use `count()` to sum the number of individuals belonging to each taxon while maintaining the grouping by year, plot_id, and plot_type. We use the `spread()` function from `tidyr` on the split dataset to convert parts of a long format dataset to a wide format dataset. Last, we calculate richness (i.e., number of taxon per plot per year) for each replicate in the two treatments.

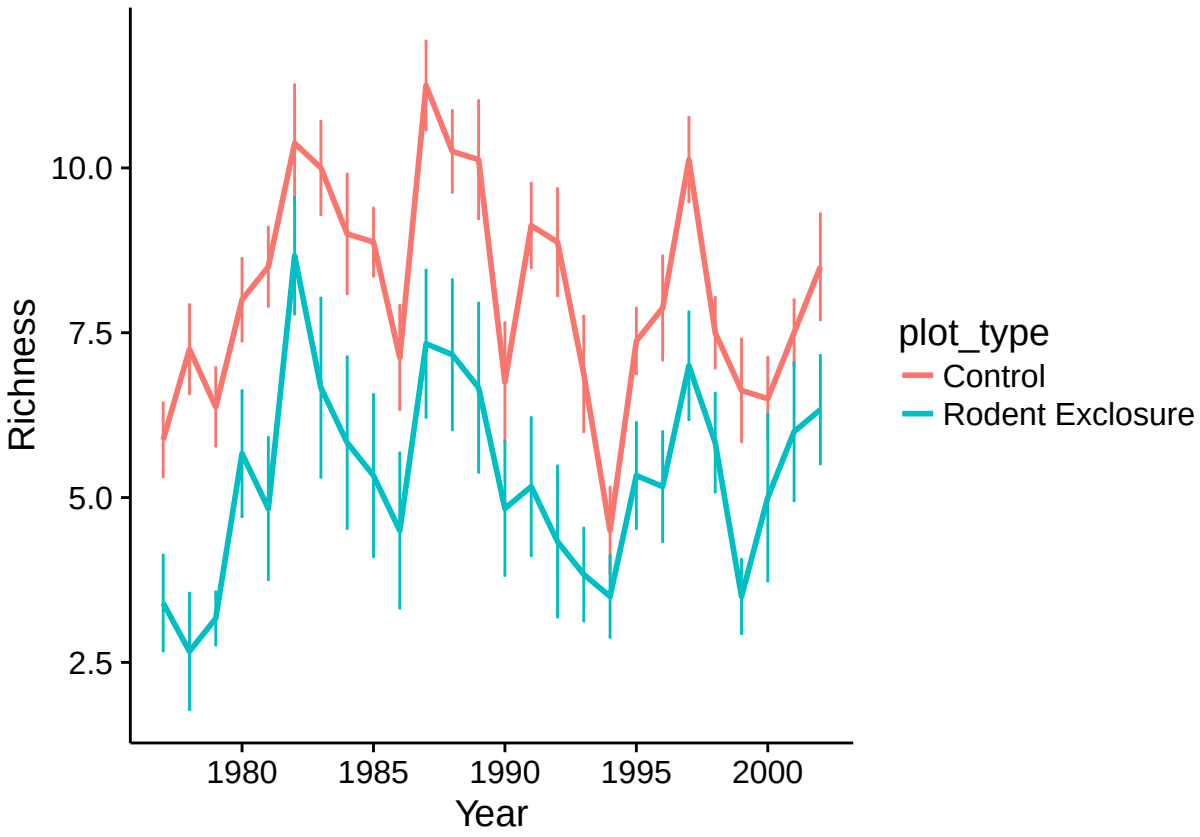
```
# Construct time-by-species matrix
time.by.species <- group_by(portal, year, plot_id,
      plot_type) %>% count(taxon) %>% spread(key = taxon, value = n, fill = 0)

# Calculate observed richness from time-by-species matrix
richness <- as.data.frame(rowSums(time.by.species[,-c(1:3)] > 0))

# Create data frame with experimental design and richness data
rich.all <- data.frame(time.by.species[,1:3,], richness)
```

```
# Rename column
names(rich.all)[4] <- "richness"

# Pull out two of the five Portal treatments
rich.treat <- rich.all[which(rich.all$plot_type ==
      "Control" | rich.all$plot_type == "Rodent Exclosure"), ]
```

Plot data

While the base package in R can make beautiful plots and has a tremendous flexibility, it can also require a lot of effort to generate figures. Another, way to make figures in R is with the package `ggplot`. As you will see below, with relatively few lines of code, we can make a nice-looking figure with `ggplot` showing the effects of rodent exlcosure on mammal richness. But first, we need to caculate some summary statistics for the plotting. Again, we use the `group_by` function from `dplyr` along with pipes to retrieve the mean and standard error of the mean (SEM) using the `summarise` function.

```
rich.treat.plot <- group_by(rich.treat, plot_type, year) %>%
  summarise(
    mean = mean(richness),    # avg. richness per group
    sd = sd(richness),        # stand. dev. per group
    n = n(),                  # num. obs. per group
    sem = sd/sqrt(n))         # calc. std. err. mean.

rich.plot <- ggplot(rich.treat.plot, aes(x = year, y = mean, color = plot_type)) +
  geom_line(size = 1, show.legend = T) +
  geom_errorbar(aes(ymin = mean - sem, ymax = mean + sem), width = .1) +
  xlim(1977, 2002) +
  xlab("Year") +
  ylab("Richness")

plot(rich.plot)
```



Perform RM-ANOVA

Based on our plot, it appears that rodent richness changes quite a bit from year to year. The plot also suggests that the enclosure treatment reduced rodent richness. Let's peform a RM-ANOVA to see whether statistics match up with our observations. We will implement RM-ANOVA using the "linear and non-linear mixed-effects models" package in R (i.e., `nlme`). **Mixed effects** refers to models that contain both fixed and random effects. The fixed effects in our analysis are time and treatment; these are aspects of the design that were under the researchers' control. In contrast, within plot (i.e., subject) variation is considered a random effect. RM-ANOVA quantifies model error based on the variance of observations within and between subjects. In the R chunk below, we implement RM-ANOVA with the `lme()` function to test for the effects of treatment (i.e., "plot_type"), time (i.e., year), and a treatment x time interaction on rodent richness. The random statement indicates that plot_id is a random variable that will be assinged its own intercept. In addition, the correlation statement specifies the covariance structure that is associated with the repeated measures. Here, we use an autoregressive process of order 1, which means that an observation at t will be most influenced by observations at t-1. Alternate covariance structures to consider are "unstructured" and "compound symmetry".

```
rich.rm <- lme(richness ~ plot_type * year, random = ~ 1 | plot_id,
               correlation = corAR1(form = ~ 1 | plot_id),
               data = rich.treat)

# Look at detailed output
summary(rich.rm)

# Obtain F-test
anova(rich.rm)
```

```r
# Make cleaner ANOVA table
set.caption("RMANOVA for Portal")
pander(anova(rich.rm))

# Use `lsmeans` package for time-corrected for marginal means
lsmeans(rich.rm, ~plot_type)
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

6) TEMPORAL BETA DIVERSITY

As we have learned this semester, β-diversity examines biodiversity between samples. In previous weeks, we have emphasized features of β-diversity that relate to the distribution of species and samples in space. Now we will explore temporal dimensions of β-diversity, which will examine changes in the abundance or presence-absence of species over time. Specifically, we introduce concepts and measures of turnover, rank shift, and rank change interval.

Let us revisit the figure of annual rodent richness over time. RM-ANOVA revealed that exclosure led to 70% reduction in rodent richness. Although there was no significant effect of time in the RM-ANOVA model, rodent richness fluctuates with discernable peaks in 1982, 1987, 1992, 1997, and 2002. Interestingly, the interannual trends in richness are similar across treatments. We will investigate these patterns in the context of temporal biodivesity in the following sections.

A. Turnover

The temporal pattern of annual richness could be explained in part by changes in species composition over time (i.e., species turnover). Turnover quantifies the similarity of species composition over time. Low turnover means that composition is relatively stable through time, while high turnover reflects dynamic community structure. Turnover can be driven by the introduction and establishment of new species or the loss of a resident species. These gains and losses of species can reflect a suite of eco-evolutionary processes including natural immigration, invasion by exotic species, disease outbreaks, and competitive exclusion. Here, we calculate the overall turnover with the follwing simple equation:

$$\text{Total turnover} = \frac{\text{species gained} + \text{species lost}}{\text{total species in both timepoints}}$$

In the following R chunks, we use the `codyn` package (Hallett et al. 2016), which calculates a number of metrics to analyze community dynamics.

```r
# First, calculate the species abundances from each site over time
portal.species.abunds <- group_by(portal, year, plot_type) %>% count(taxon)


# Calculate total turnover
portal.total <- turnover(df = portal.species.abunds,
                         time.var = "year",
                         species.var = "taxon",
                         abundance.var = "n",
                         replicate.var = "plot_type",
                         metric = "total")

# Calculate species gained
portal.appearance <- turnover(df = portal.species.abunds,
```

```
                              time.var = "year",
                              species.var = "taxon",
                              abundance.var = "n",
                              replicate.var = "plot_type",
                              metric = "appearance")

# Calculate species lost
portal.disappearance <- turnover(df = portal.species.abunds,
                              time.var = "year",
                              species.var = "taxon",
                              abundance.var = "n",
                              replicate.var = "plot_type",
                              metric = "disappearance")
```

Each of these objects contains a column for the value of the turnover metric (total, appearance, disappearance), the second year in the pairwise comparison, and the type of plot (i.e., treatment). For easier plotting, let's combine these turnover metrics into a single data table:

```
# Use `join()` from `dplyr` to join the columns by shared year & plot type columns
portal.turnover <- full_join(portal.total, portal.disappearance) %>%
  full_join(portal.appearance)
```

```
## Joining, by = c("year", "plot_type")
## Joining, by = c("year", "plot_type")
```

```
# Use `gather()` from `tidyr` to convert back to long-form
portal.turnover <- gather(portal.turnover, key = metric, value = turnover,
       total, appearance, disappearance)
```

In the R code chunk above, we did a lot of data wrangling using only a couple lines of code. Take a moment to reflect on what we just did. We had three separate data frames containing different turnover metrics. But those metrics referred to the same plots and years. So, we joined them together using their shared year and plot_type index to create a single data-table. We first joined the total and disappearance values, then piped (`%>%`) this joined data-table to be the first argument in another join function. Try running each of these `full_join()` statements separately to see what is happening at each step.

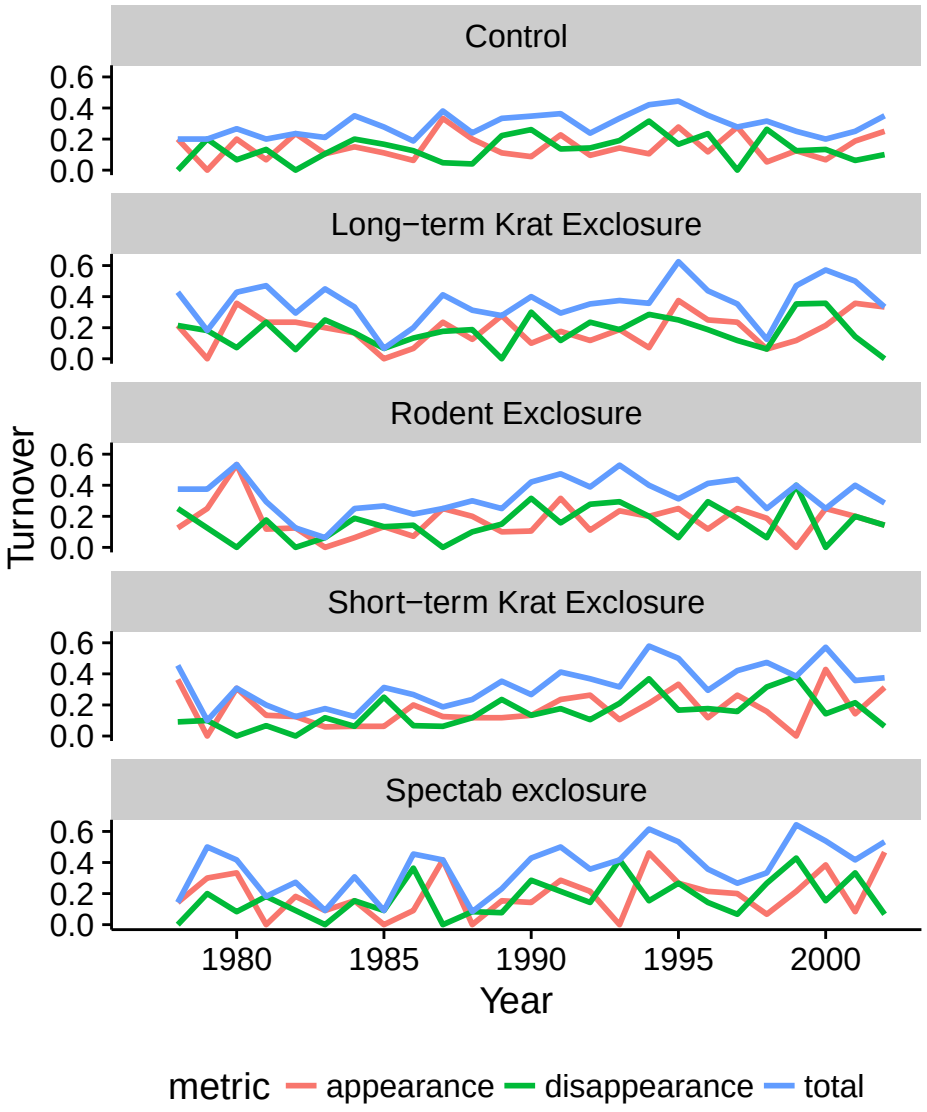
Next, we turned the wide-form data table back into a long-form table to facilitate plotting. We did this using the `gather()` function. Essentialy, this function takes `portal.turnover` and creates a new column of keys and calls it "metric". It then creates another new column of associated values and calls it "turnover". The keys in the metric column are the old columns we gathered together (i.e., total, appearance, disappearance). The values that go in turnover are the values that were previously in the total, appearance, disappearance columns.

Take a look at our new data table with the three different turnover metrics calculated for each year and treatment (try `View(portal.turnover)`).

Now, let's use `ggplot` to visualize the turnover metrics in all five of the Portal treatments over the entire annual time series.

```
turn.plot <- ggplot(
  portal.turnover, aes(x = year, y = turnover, color = metric)) +
  geom_line(size = 1, show.legend = T) +
  facet_wrap(~plot_type, ncol = 1) +
  xlim(1977, 2002) +
  xlab("Year") +
  ylab("Turnover") +
  theme(legend.position = "bottom")
```

```
plot(turn.plot)
```



C. Rank Shift

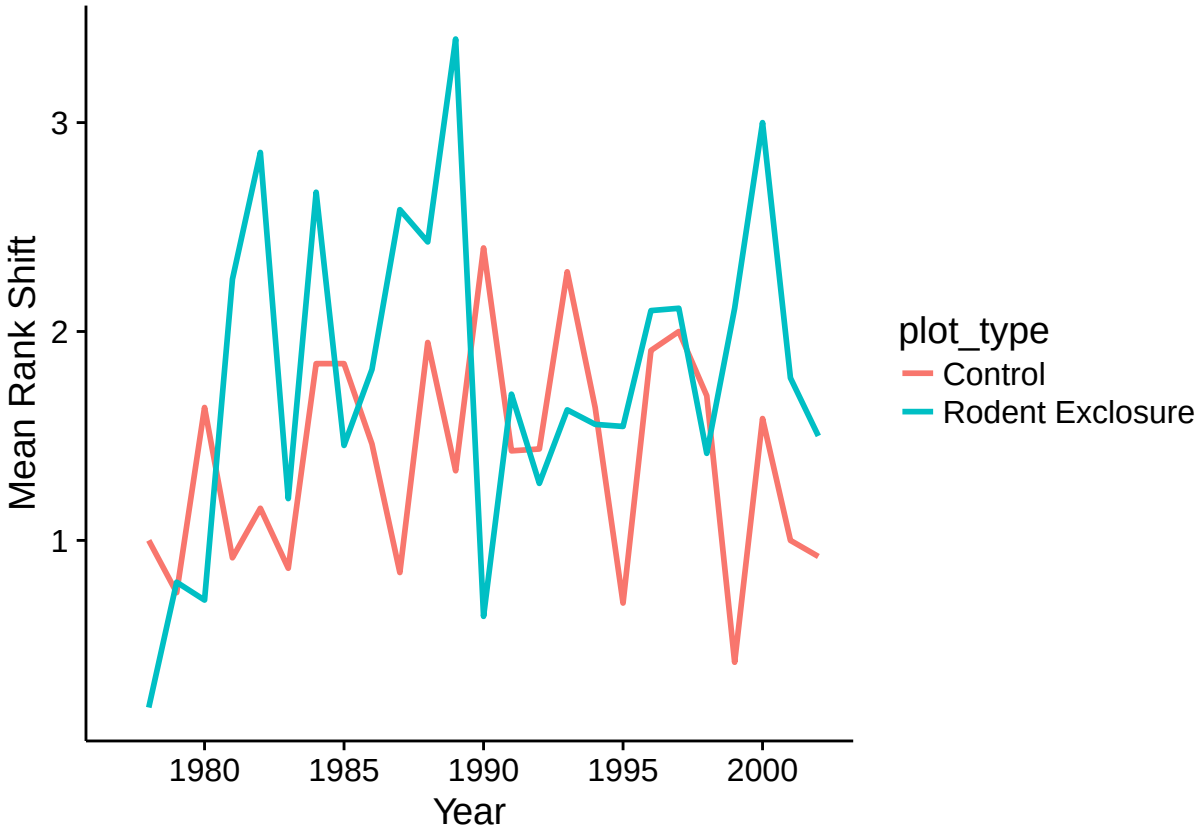
Turnover estimates place equal weight on the gain and loss of a species irrespective of its abundance. However, the gain or loss of a common vs. rare species may have important implications for species interactions and ecosystem functioning. For example, the "mass ratio theory" (Grime 1998) states that species should contribute to community processes in proportion to their abundance. However, more recent studies suggest that rare species contribute disproportinately to community processes (Leitao et al. 2016). In this section, we introduce a β-diversity metric called **mean rank shift (MRS)** (Collins et al. 2008) that allows us to calculate the shift in ranks using the following equation:

$$MRS = \sum_{i=1}^{n} \left(|R_{i,t+1} - R_{i,t}| \right) / n$$

where R_i is the rank of a species found in samples at time t and time $t + 1$. MRS then sums the absolute value of the rank changes for n species that are shared between samples. Thus, the higher the MRS index,

the more change there is in the commonness and rarity of taxa. In the following R chunk, we calculate MRS from `portal.rankshift` which is the abundance of species in each site over time (see above). We then plot MRS and characterize its variability.

```
# Pull out the two treatments we analyzed earlier
portal.abunds.cont.rodent <- filter(portal.species.abunds,
                        plot_type == "Control" | plot_type == "Rodent Exclosure")

# Calculate MRS
portal.rankshift <- rank_shift(
  df = as.data.frame(portal.abunds.cont.rodent),
  time.var = "year",
  species.var = "taxon",
  abundance.var = "n",
  replicate.var = "plot_type")

# Replace the year range with a single value to plot
portal.rankshift$year <- as.numeric(substr(portal.rankshift$year_pair, 6, 9))

# Create ggplot
rankshift.plot <- ggplot(portal.rankshift, aes(x = year, y = MRS, color = plot_type)) +
  geom_line(size = 1) +
  xlim(1977, 2002) +
  xlab("Year") +
  ylab("Mean Rank Shift")

plot(rankshift.plot)
```



```
# Does one plot type show higher or lower MRS, on average?
group_by(portal.rankshift, plot_type) %>%
  summarise(
    mean = mean(MRS),
    cv = sd(MRS)/mean)
```

```
## # A tibble: 2 × 3
##          plot_type     mean        cv
##              <chr>    <dbl>     <dbl>
## 1          Control 1.400675 0.3759483
## 2 Rodent Exclosure 1.788980 0.4361809
```

D. Rate Change Interval

How different can communities become over time? And how fast does this happen? These are important questions for basic science, but are also important for topics in restoration ecology where legacy effects often impede management practices. For example, in highly dynamic systems, turnover may quickly saturate at a high level, while in more stable communities, it may take decades or centuries for communities to reach a similar level of dissimilarity, if they ever do. In the following section, we use the `codyn` package to calcualte the **rate change interval** for species abundances in the Portal site and plot this information using `ggplot`.

If we take a look at `help(rate_change_interval)`, we see that this function calculates the Euclidean distance between two communities. Remember from our discussion of beta-diversity, that Euclidean distances are not appropriate for calculating community dissimilarity because the distance between sites that share no species in common may be shorter than between sites that share species but in different abundances! So, we need to transform these data with an appropriate transformation before making strong inferences about what is going on in our dataset. Recall that one of the appropriate transformations is the Hellinger transformation, advocated by Legendre and Gallagher (2001). The Euclidean distance calculated on the Hellinger-transformed species abundances generates the Hellinger distance.

The Hellinger transformation is the square root of the relative abundances: $y'ij = \sqrt{\frac{yij}{y_{i+}}}$

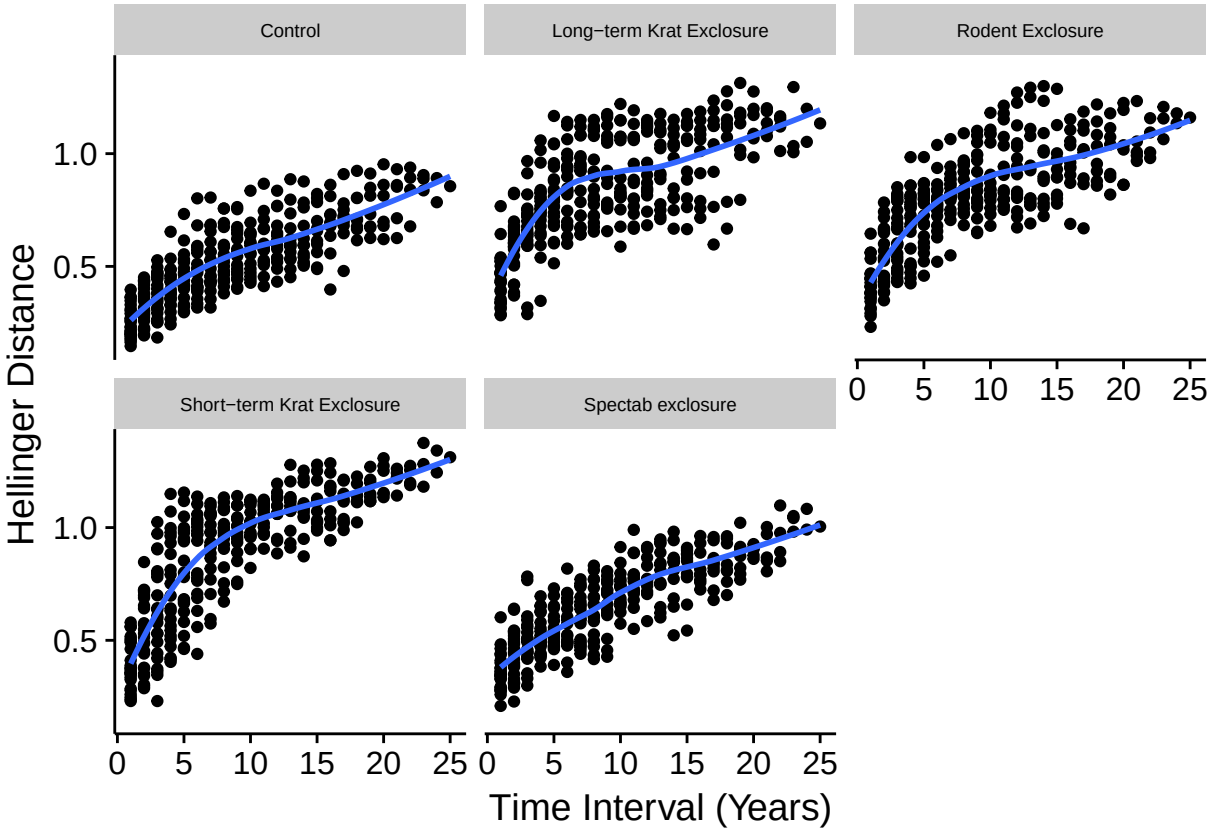
```
# In order to calculate relative abundances, we need total abundances
# First, let's count the total abundances
total.abunds <- portal.species.abunds %>%
  group_by(year, plot_type) %>%
  count(wt = n)  # The wt sums the values of n within a year
total.abunds # We now have a column nn that is the site total abund.
```

```
## Source: local data frame [130 x 3]
## Groups: year [?]
##
##     year                 plot_type    nn
##    <int>                    <fctr> <int>
## 1   1977                   Control   223
## 2   1977  Long-term Krat Exclosure    65
## 3   1977           Rodent Exclosure    58
## 4   1977 Short-term Krat Exclosure    92
## 5   1977         Spectab exclosure    49
## 6   1978                   Control   502
## 7   1978  Long-term Krat Exclosure    78
## 8   1978           Rodent Exclosure    74
## 9   1978 Short-term Krat Exclosure   205
## 10  1978         Spectab exclosure   133
## # ... with 120 more rows
```

```
# Now, let's join these total counts with the counts for each species
portal.hellinger.transf <- inner_join(
  portal.species.abunds, total.abunds, by = c("year", "plot_type")) %>%
  mutate(hellinger.transf = sqrt(n / nn))

# The mutate function creates a new column "hellinger.transf"
# by taking the square root of species relative abundance
portal.hellinger.transf
```

```
## Source: local data frame [1,702 x 6]
## Groups: year, plot_type [130]
##
##     year plot_type                   taxon     n    nn hellinger.transf
##    <int>    <fctr>                   <chr> <int> <int>            <dbl>
## 1   1977   Control Chaetodipus_penicillatus     6   223       0.16402997
## 2   1977   Control       Dipodomys_merriami   108   223       0.69592021
## 3   1977   Control          Dipodomys_ordii     6   223       0.16402997
## 4   1977   Control    Dipodomys_spectabilis    47   223       0.45908859
## 5   1977   Control         Neotoma_albigula    22   223       0.31409347
## 6   1977   Control    Onychomys_leucogaster     5   223       0.14973819
## 7   1977   Control            Onychomys_sp.     2   223       0.09470274
## 8   1977   Control       Onychomys_torridus     9   223       0.20089486
## 9   1977   Control       Perognathus_flavus    13   223       0.24144557
## 10  1977   Control      Peromyscus_eremicus     3   223       0.11598670
## # ... with 1,692 more rows
```

```
# We can use this new column as our "abundance" vector
portal.change.int2 <- rate_change_interval(portal.hellinger.transf,
                      time.var = "year",
                      species.var = "taxon",
                      abundance.var = "hellinger.transf",
                      replicate.var = "plot_type")

rate.plot2 <- ggplot(portal.change.int2, aes(interval, distance)) +
  geom_point() +
  facet_wrap(~plot_type) +
  theme(strip.text.x = element_text(size = 7)) +
  stat_smooth(method = "loess", se = F, size = 1) +
  ylab("Hellinger Distance") +
  xlab("Time Interval (Years)")
rate.plot2
```



7) STABILITY

The stability of ecological systems has been at the forefront of ecological research for decades (Ives and Carpenter 2007). There are many definitions of stability, and these are beyond the scope of this course (but see Pimm 1984, Grimm et al. 1997). In mathematical models, notions of stability often revolve around responses to perturbations and whether these perturbations grow away from or return to a stable equilibrium, but applying these ideas to real datasets is often less straightforward. Another way to think about stability is the lack of variability in some value (e.g., community biomass, population densities, etc.). Stability can be analyzed with respect to the individual species that make up a community (i.e., compositional variability), or some measure that integrates across species (i.e., aggregate variability), such as total biomass, total abundance, or richness (Micheli et al. 1999). Understanding how species diversity relates to measures of stability is still a highly active area of research (McCann 2000), but some general trends seem to be that increasing diversity decreases the stability of populations but increases aggregate measures of stability (e.g., biomass) (Tilman 1999). Below, we explore some metrics that have been used to analyze ecological stability.

A. Community Stability

The stability of an aggregate measure of an ecological system can be assessed by measuring its variability. One way to characterize variability is the Coefficient of Variation (CV). The CV relativizes the standard deviation of a variable to its mean value because variance scales with the mean. By using the CV, we can more easily compare the variability of systems with different mean values. We can calculate the CV as follows:

$$CV = \frac{\sigma}{\mu}$$

where σ is the standard deviation and μ is the mean value.

Higher CV indicates more variability, and lower CV indicates less variability. Therefore, we can measure stability as:

$$\text{Stability} = \frac{1}{CV}$$

Let's calculate stability within each plot type now.

```
portal.stab <- community_stability(df = as.data.frame(portal.species.abunds),
                     time.var = "year",
                     abundance.var = "n",
                     replicate.var = "plot_type")
pander(portal.stab)
```

plot_type	stability
Control	3.044
Long-term Krat Exclosure	1.865
Rodent Exclosure	1.864
Short-term Krat Exclosure	2.462
Spectab exclosure	2.911

B. Species Synchrony

One potential mechanism underlying stability is synchrony. Species synchrony is a measure of whether population densities fluctuate independently or not, and if not, how strongly they positively or negatively covary. When species exhibit strong synchrony, species densities positively covary. This suggests that species respond similarly to changes in the environment, and has been hypothesized to be a key driver of instability of aggregate community metrics (e.g., biomass)—if all species decline together, a single disturbance can be severely destabilizing. When species dispaly strong asynchrony, species densities negatively covary. This suggests that species respond differently to changes in the environment, which has been hypothesized to stabilize aggregate community metrics—some species compensate in density for losses in other species, a phenomenon called compensatory dynamics (Gonzalez and Loreau 2009).

Here, we will calculate two measures of community-wide synchrony that range from -1 for perfect asynchrony to $+1$ for perfect synchrony. Loreau and de Mazancourt (2008) describe a synchrony metric that compares aggregate community variance to population-level variance. Gross et al. (2014) present a different metric, which compares average species-level correlations with the aggregate community. See the derivations in the original papers for more details.

```
portal.loreau <- synchrony(df = as.data.frame(portal.species.abunds),
          time.var = "year",
          species.var = "taxon",
          abundance.var = "n",
          replicate.var = "plot_type",
          metric = "Loreau")
names(portal.loreau)[2] <- "loreau"
portal.gross <- synchrony(df = as.data.frame(portal.species.abunds),
          time.var = "year",
          species.var = "taxon",
          abundance.var = "n",
          replicate.var = "plot_type",
          metric = "Gross")
names(portal.gross)[2] <- "gross"
pander(full_join(portal.loreau, portal.gross))
```

```
## Joining, by = "plot_type"
```

plot_type	loreau	gross
Control	0.1869	0.1263
Long-term Krat Exclosure	0.1578	0.07418
Rodent Exclosure	0.187	0.1423
Short-term Krat Exclosure	0.08773	0.001546
Spectab exclosure	0.1542	0.1393

The synchrony values are all positive, but pretty far from 1, suggesting there is probably not strong positive synchrony at these temporal and spatial scales.

The field of biodiversity–stability research often test the hypothesis that greater richness relates to greater stability of some aggregate measure or process. Let's test this idea here.

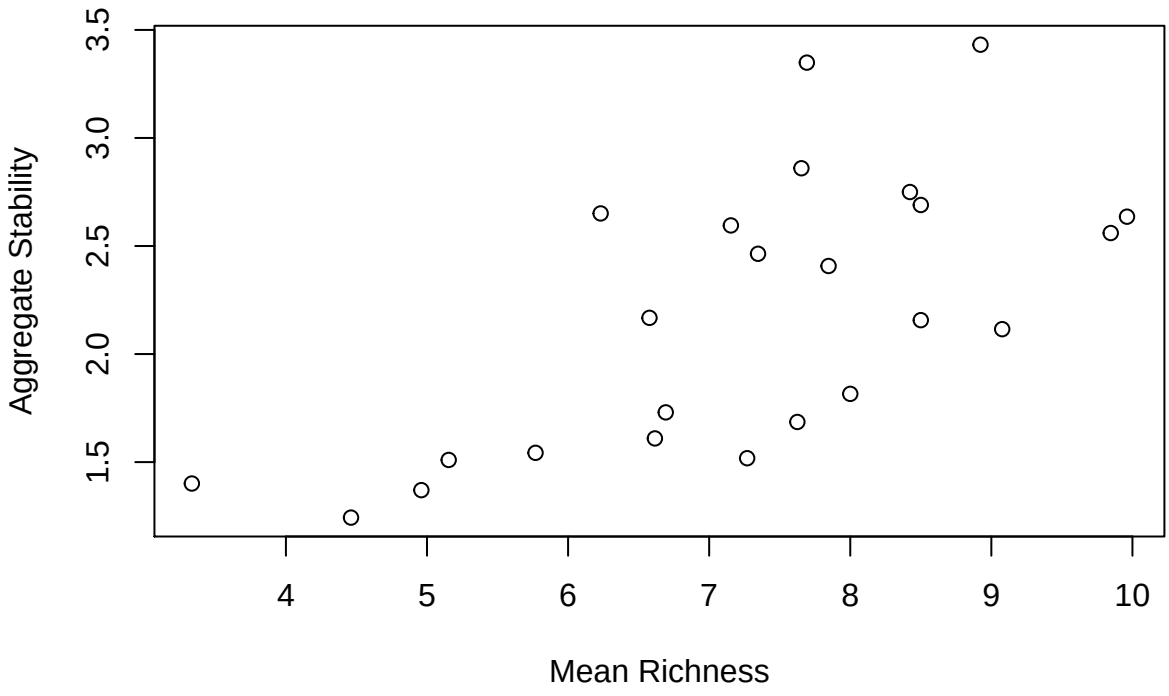
```
# Recall, earlier we calculated richness in each plot type in each year
head(rich.all)
```

```
##   year plot_id                 plot_type richness
## 1 1977       1         Spectab exclosure        4
## 2 1977       2                   Control        9
## 3 1977       3  Long-term Krat Exclosure        8
## 4 1977       4                   Control        5
## 5 1977       5          Rodent Exclosure        6
## 6 1977       6 Short-term Krat Exclosure        4
```

```
# First, let's ungroup this and regroup only by plot id
# Then, we will summarise average yearly richness in each plot type
portal.mean.rich.plot <- rich.all %>% ungroup() %>%
  group_by(plot_id) %>%
  summarise(mean.rich = mean(richness))

# Let's take a look at how stability metrics relate with mean richness
portal.plot.abunds <- as.data.frame(group_by(portal, year, plot_id) %>% count(taxon))
portal.stab.plot <- community_stability(df = portal.plot.abunds,
                    time.var = "year",
                    abundance.var = "n",
                    replicate.var = "plot_id")

# Join richness and stability
portal.div.stab <- rich.all %>% group_by(plot_id) %>%
  summarise(avg.rich = mean(richness)) %>%
  inner_join(portal.stab.plot)
```

```
## Joining, by = "plot_id"
```

```
plot(portal.div.stab$stability ~ portal.div.stab$avg.rich,
     xlab = "Mean Richness", ylab = "Aggregate Stability")
```



C. Variance Ratio

Similar to the above tests of community-wide synchrony, here we present another metric that tests for positive or negative species covariance. The ratio of aggregate abundances compared to the sum of variances of individual species can generate three qualitatively different outcomes: *VR = 1 species do not covary,* VR $>$ 1 species positively covary *VR $<$ 1 species negatively covary In order to assess the significance of this relationship, we will compare the observed variance ratio to a distribution of ratios calculated on a number of randomized null communities.

```
portal.vr <- variance_ratio(df = as.data.frame(portal.species.abunds),
                time.var = "year",
                species.var = "taxon",
                abundance.var = "n",
                replicate.var = "plot_type",
                bootnumber = 1000,
                average.replicates = T,
                level = 0.95)
pander(portal.vr)
```

lowerCI	upperCI	nullmean	VR
0.6847	1.372	1.003	1.325

Here, we notice a variance ratio of ~ 1.3. This value is greater than 1, suggesting positive covariation among species. However, we also note that the confidence intervals (based on null matrix randomization) bracket 1. Based on this observation, it is probably best to remain conservative based on this analysis alone and claim that we have insufficient evidence for significant covariation among species. In general, this agrees with the above synchrony calculations, suggesting we see weak, positive covariation among species that likely respond to environmental variability (e.g., drought, fire) in similar ways.

REFERENCES

Grime, J. P. (1998). Benefits of plant diversity to ecosystems: immediate, filter and founder effects. J. Ecol. 86, 902–910.

Collins https://www.ncbi.nlm.nih.gov/pubmed/19137958

Leitao et al. 2016 http://rspb.royalsocietypublishing.org/content/283/1828/20160084