Handout: Temporal Diversity

Z620: Quantitative Biodiversity, Indiana University

February 10, 2017

OVERVIEW

Biological systems are inherently variable through time. While some of this variation is stochastic, ecological systems can change though time because of abiotic (e.g., rising temperature, increasing CO_2, drought) or biotic conditions (e.g., invasive species, disease). While some biologists focus on processes that occur on contemporary scales (minutes, days, weeks, years), other scientists are interested in what drives patterns of biodiversity in the fossil record over geologic time scales. In this handout, we will first introduce you to a new ecological data structure, the time-by-site-species matrix. You will learn how to manage and manipulate this type of data structure using some new R packages. We will then introduce the basics of time series analysis, which will be followed by sections dealing with repeated-measures analysis of variance (RM-ANOVA), temporal beta diversity (i.e., turnover), and synchrony.

1) SETUP

Retrieve and Set Your Working Directory

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

Install Packages

We will be using several packages in this week's module. Let's load them now. 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')
for (package in package.list) {
  if (!require(package, character.only = T, quietly = T)) {
    install.packages(package)
    library(package, character.only = T)
  }
}
```

2) LOADING DATA

To learn about topics of temporal biodiversity, 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 biodiversity study was initiated in the late 1970s by Jim Brown and colleagues to look at species interactions, specifically competition between rodents and ants for plant seeds. Early experiments focused on the exclusion of rodents with fencing and trapping, which led to an increase in the ant population size as well as changes in plant species composition. These findings inspired Brown and colleagues to construct an improved experiment

that continues today. The Portal Project monitors 24 replicated 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)
```

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 merriami*) (plot_id: 3, 15, 19, 21)
3) Short-term Krat - short-term exclusion of Kangaroo rats (*Dipodomys merriami*) (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)

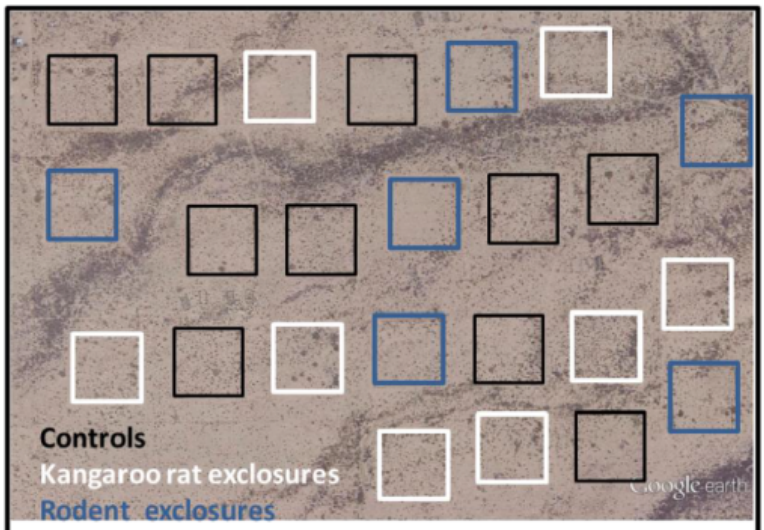


Fig 3. Experimental plots at the 20 ha Portal site. Black: Control plots where all rodents have access. White: Only kangaroo rats excluded. Blue: All rodents excluded.

3) DATA WRANGLING

The success of the Portal Project is based reflects insightful questions from a creative long-term experiment. In addition to generating a lot of quality information, it was critical for the Portal Project's data to be organized, accessible, and properly managed.

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, one needs to manipulate the long-format data for

visualization and statistical analysis. For example, many R packages and functions will want 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 fast and efficient, and thus are preferred ways for manipulating data in R compared to other control-flow statements (e.g., for loops and while 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.

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

# Make a taxon column 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 "pipes" (%>%). 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 is fed into the next 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 `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` on the split dataset to convert parts of a long format dataset to a wide format dataset.

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

Now we can create differerent site-by-species-by-time matrices

```
dplyr::filter(time.by.species, year == 1984) # return 1984 site-by-species
dplyr::filter(time.by.species, plot_id == 5) # return plot5 time-by-species
```

Last, let's convert the `tidyr` object to a dataframe

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

4) TIME SERIES ANALYSIS - A PRIMER

Because many questions in biodiversity science involve time, we are going to provide a brief introduction to some basic concepts and practices associated with time series analysis. Time series analysis involves a suite of statistical tools for detecting and decomposing trends in temporal data, and distinguishing this from other sources (e.g., seasonality, cycles, and error). We will demonstrate some of these tools using the abundance of rodents in a single site of the Portal Project. Because, the Chihuahuan Desert sits in a rain shadow created by the Sierra Madre Mountains, the Portal site is an arid ecosystems with rain falling primarily during the months of June through October. This variability in precipitation provides an opportunity for us to detect a signal of this *seasonality* in our dataset, while testing for long-term trends and forecasting rodent densities into the future.

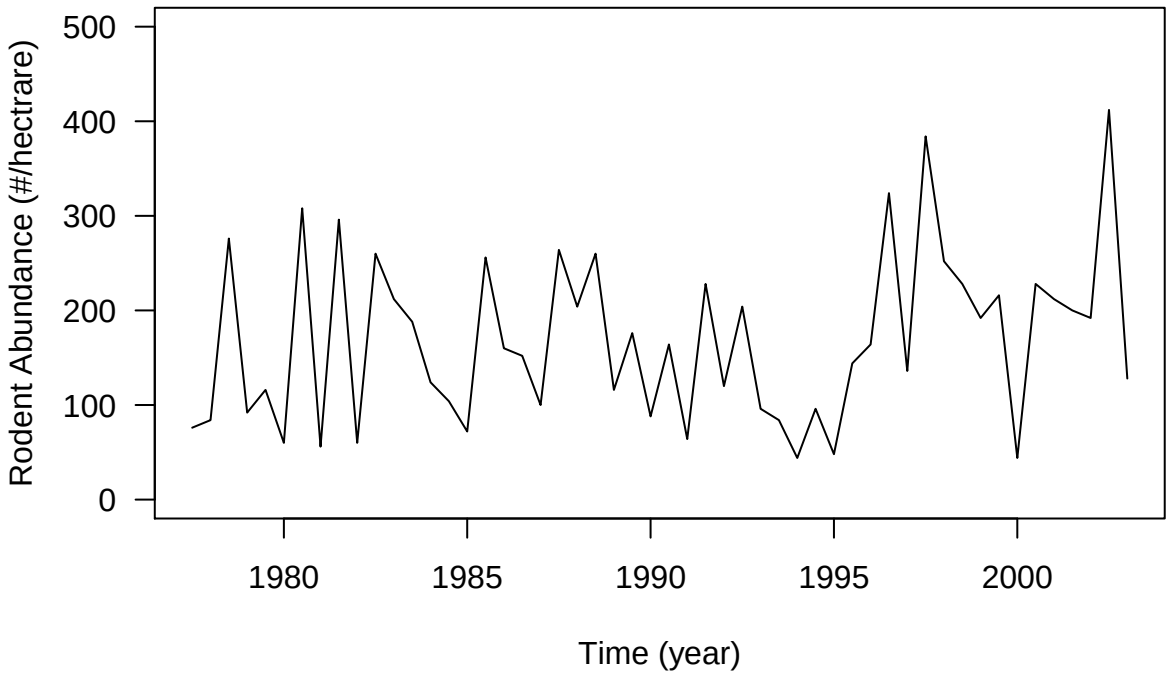
Data wrangling

In the following R chunk, we are going to manipulate `portal` using some of the tools that we 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 are retaining month so that 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)

# Create a seasonality variable using month number
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 and calculate rodent abundance for each season within a year. We will then express the abundance data on a per hectare basis (0.25 ha * 4 = 1 ha). Then, we are going to use a set of time series functions in the R base package along with some functions from the `tseries` package. The first thing we will do is to create a time series data set, which identifies the start time and seasonality of our sampling, which is determined by the frequency argument.

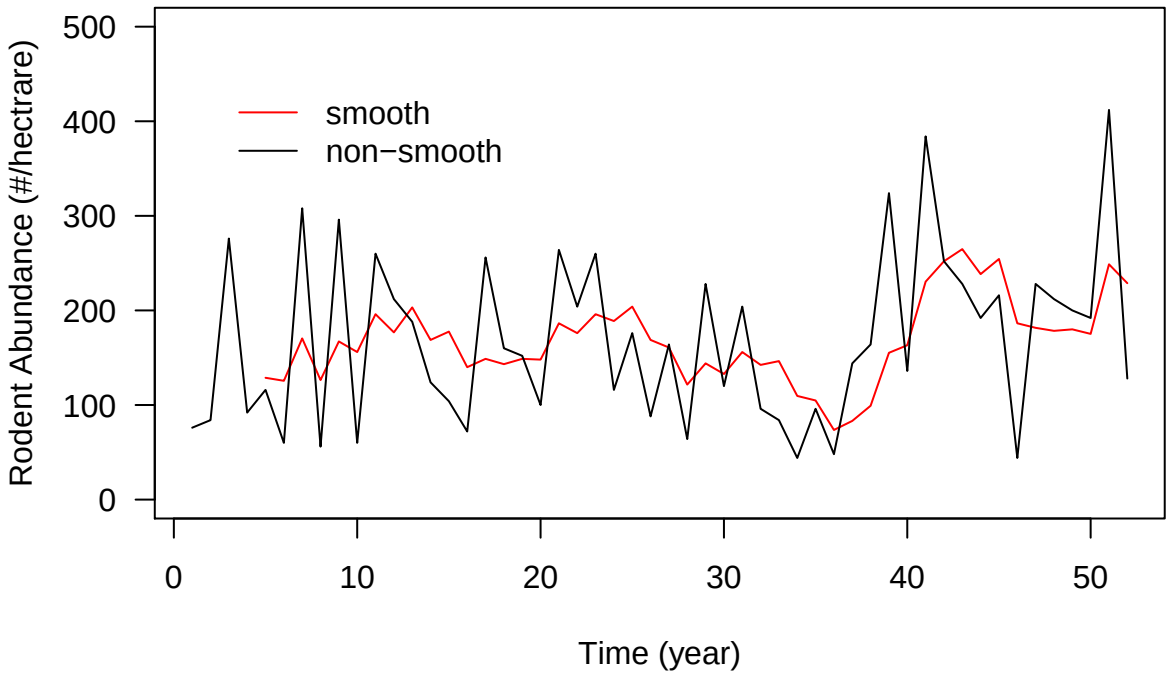
```
abund <- filter(time.by.spec.2, plot_id == 2) %>%
  group_by(year, season) %>%
  count(wt = n)

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 (#/hectrare)",
        xlab = "Time (year)", las = 1, ylim = c(0, 500))
```



Smoothing

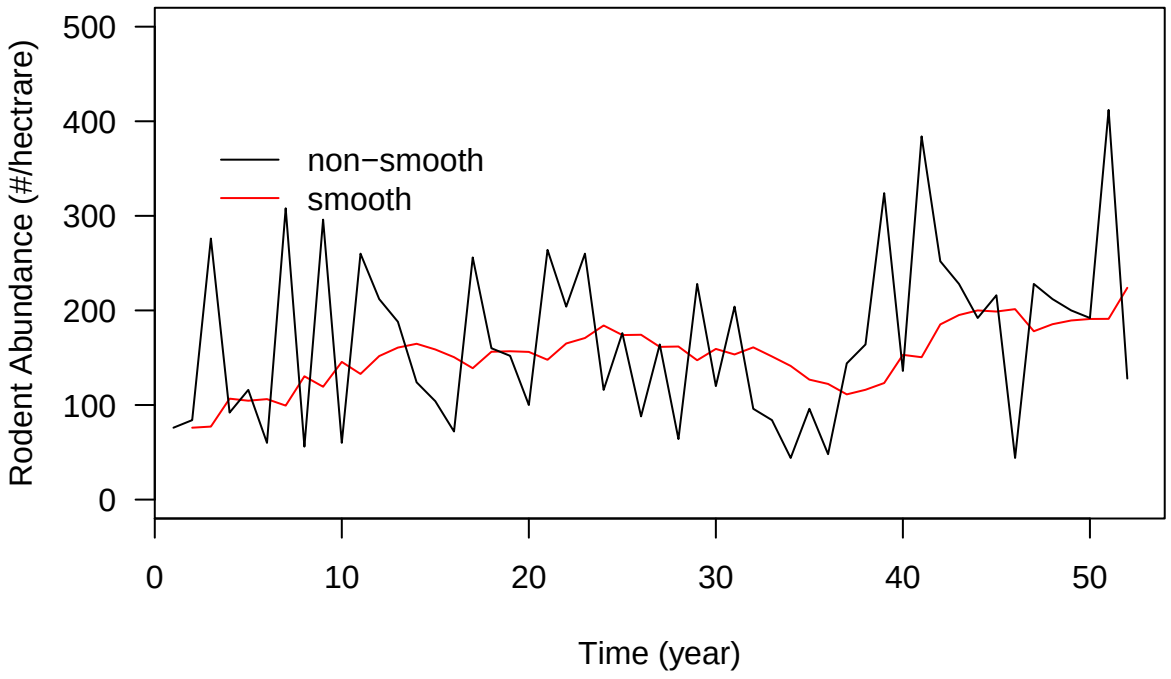
One technique that is commonly used, mostly for visualizing trends in a time series is *smoothing*. Smoothing techniques attempt to remove noise from a data set, and are used as an aid to detect signal in a dataset. The simplest form of smoothing is the *simple moving average*. Simple moving average 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 in the `TTR` package. Play around with n to visualize the effects of the smoothing window.

```
abund.sm <- SMA(abund$nn, n = 5)
plot(abund.sm, type = "l", col = "red", ylab = "Rodent Abundance (#/hectrare)",
     xlab = "Time (year)", las = 1, ylim = c(0, 500))
lines(abund$nn, col = "black")
legend(2, 450, col = c("red", "black"), lty = c(1,1), c("smooth", "non-smooth"), bty = "n", cex = 1)
```



One can also *exponentially smooth* a time series. Such procedures often use low-pass filters, which means that low frequency signal is retained. In addition, this smoothing process places exponentially less weight on observations over time. A common statistic for performing exponential smoothing is Holt-Winters filtering. 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.

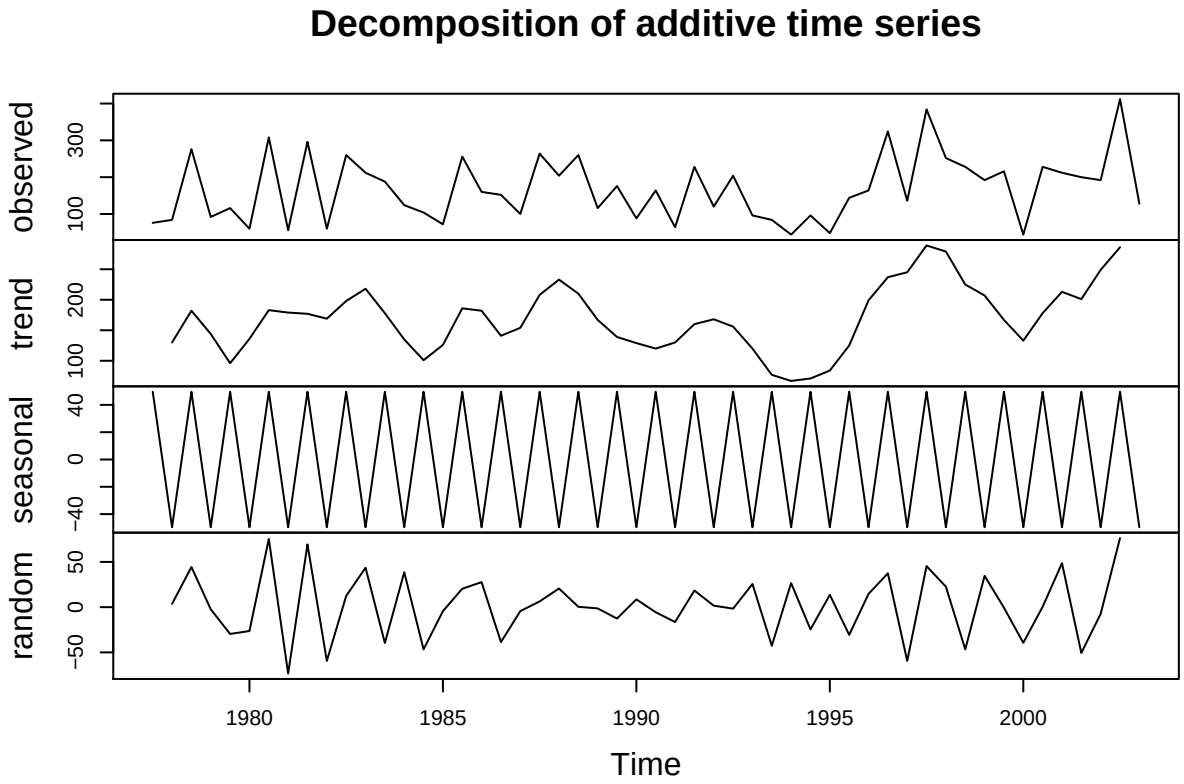
```
abund.hw <- HoltWinters(abund$nn, beta = FALSE, gamma = FALSE)
# abund.hw$fitted
plot(abund.hw, xlab = "Time (year)", ylim = c(0, 500),
     ylab = "Rodent Abundance (#/hectrare)", las = 1, main = NA)
legend(2, 400, col = c("black", "red"), lty = c(1,1),
       c("non-smooth", "smooth"), bty = "n", 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 that could be driven by patterns of 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

One of the most basic ways of analyzing a time series is with auto-regressive moving average (ARMA) models. ARMA approaches can identify trends in data and help to make forecasts about future observations. As such, ARMA is commonly 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 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 $\mu is the mean of the time series (Y$), 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 change as a function of 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 (t - 1) the data. Here we will use the `adf.test()` function in the `tseries` package, which implements the Augmented Dickey-Fuller Test.
This is a commonly applied statistic that tests stationarity in a time series. We will be testing the null hypothesis that the time series is non-stationary, which means we will conclude that a time series is stationary if the p-value > 0.05.

```
adf.test(abund.ts, alternative = "stationary") # small p = non-stationary # note data problem
```

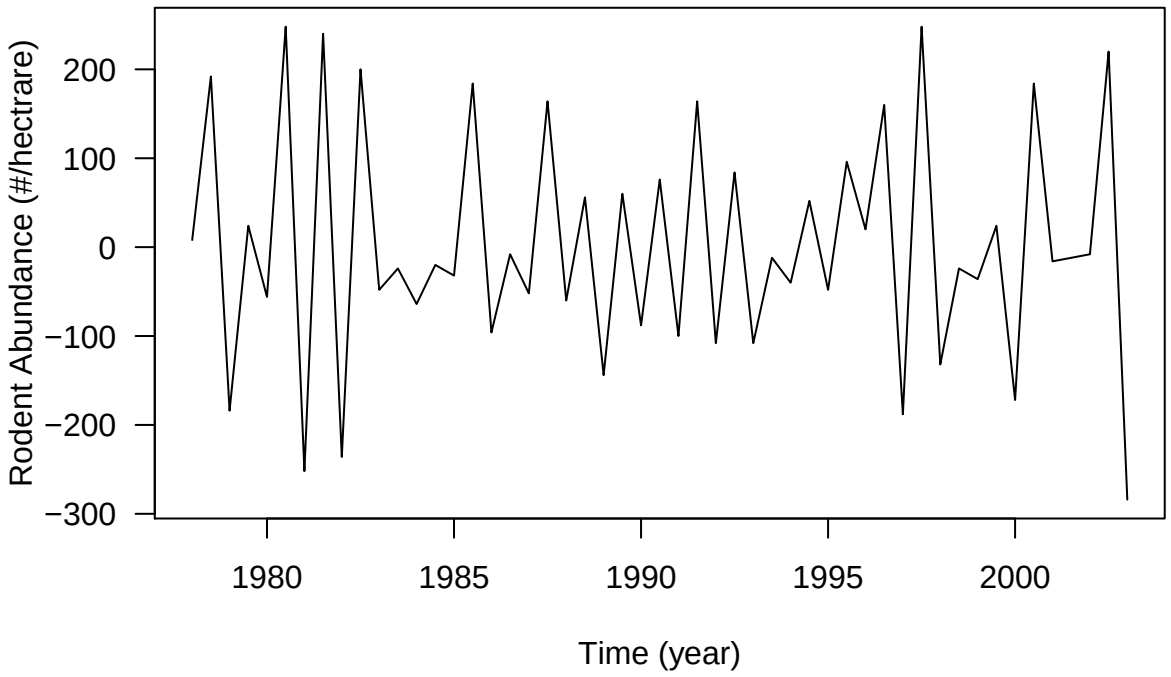
```
##
##  Augmented Dickey-Fuller Test
##
## data:  abund.ts
## Dickey-Fuller = -2.6196, Lag order = 3, p-value = 0.3252
## alternative hypothesis: stationary
```

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 take differences between observations t and t_1. We will then re-run the Dickey-Fuller test and take a look at a plot of the differences data.

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

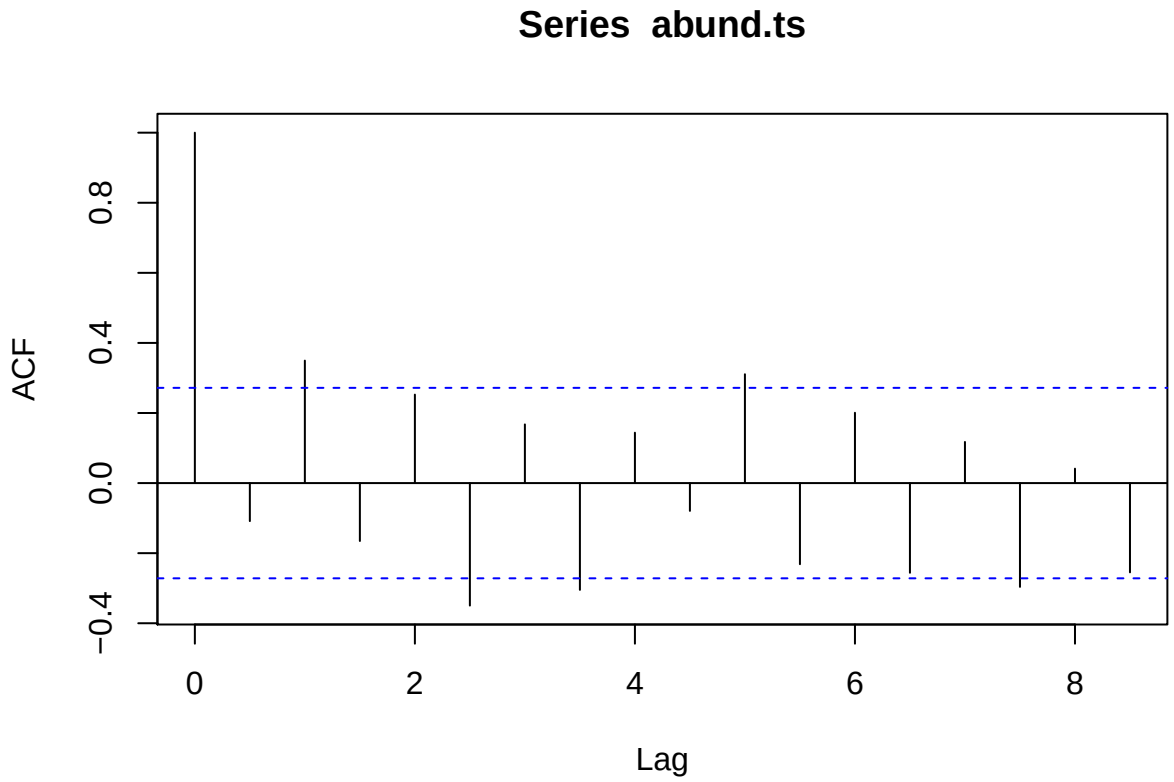
```
##
##  Augmented Dickey-Fuller Test
##
## data:  abund.ts.diff
## Dickey-Fuller = -3.9123, Lag order = 3, p-value = 0.02019
## alternative hypothesis: stationary
```

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



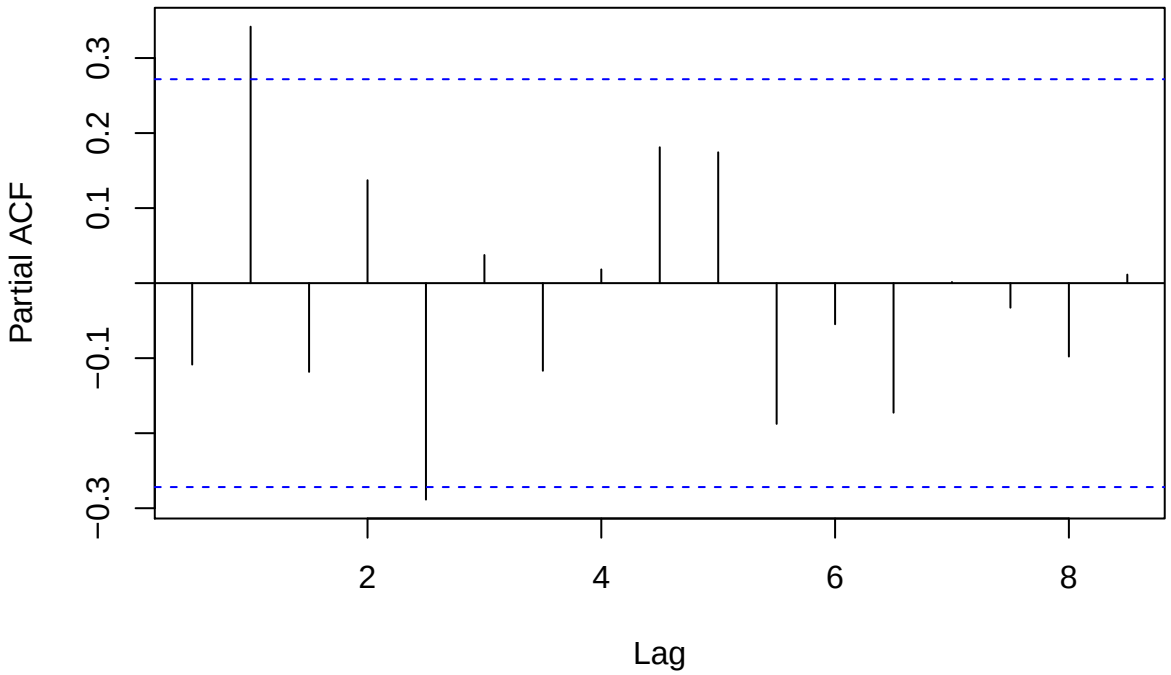
One of thing that we will do know is to look at the lags in our time series using the *autocorrelation function (ACF)*. These plots help us visualize structure in our data, but also help inform the parameterization of our ARMA models. Specifically, the ACF tells us about the lags of the forecast errors and thus tells us about the MA component of the model. The ACF simply looks at the correlation between lagged intervals in the time series. The ACF will always equal 1 with a zero lag. 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 use to inform 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 is reflecting autocorrelation on an annual cycle.

```
acf(abund.ts) # autocorrelation function; decays geometrically
```



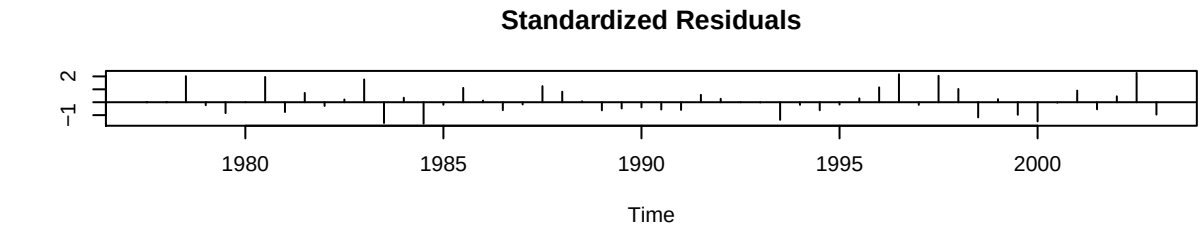
We will also look at the time series using the *partial autocorrelation function (PACF)*. PACF calculates correlations between two series after correcting or removing any correlation that might exist with another lagged series. In contrast to the ACF, significant partial correlation coefficients tell 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 = 1.

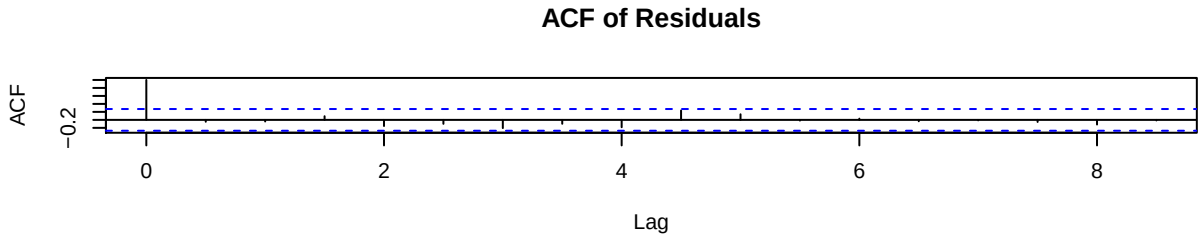
```
pacf(abund.ts) # partial autocorrelation function; decays geometrically
```

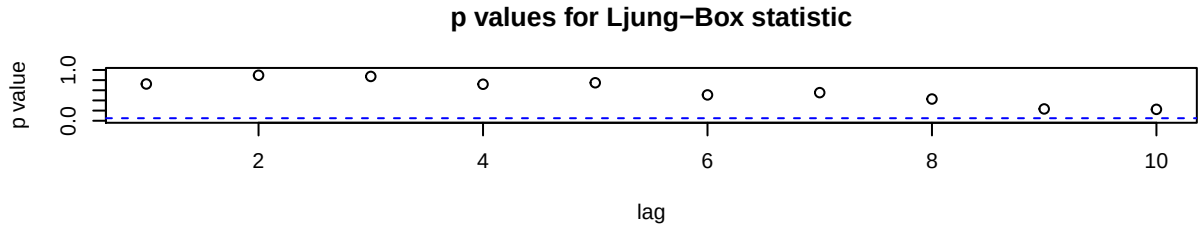


Let's use the information from the Augmented Dickey-Fuller Test, along with the ACF and PACF to construct an ARMA model. Actually, because we found that differencing helped us meet the assumption of stationarity, we are going to use an auto-regressive integrated moving average (ARIMA) model. ARIMA models 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 effects 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.

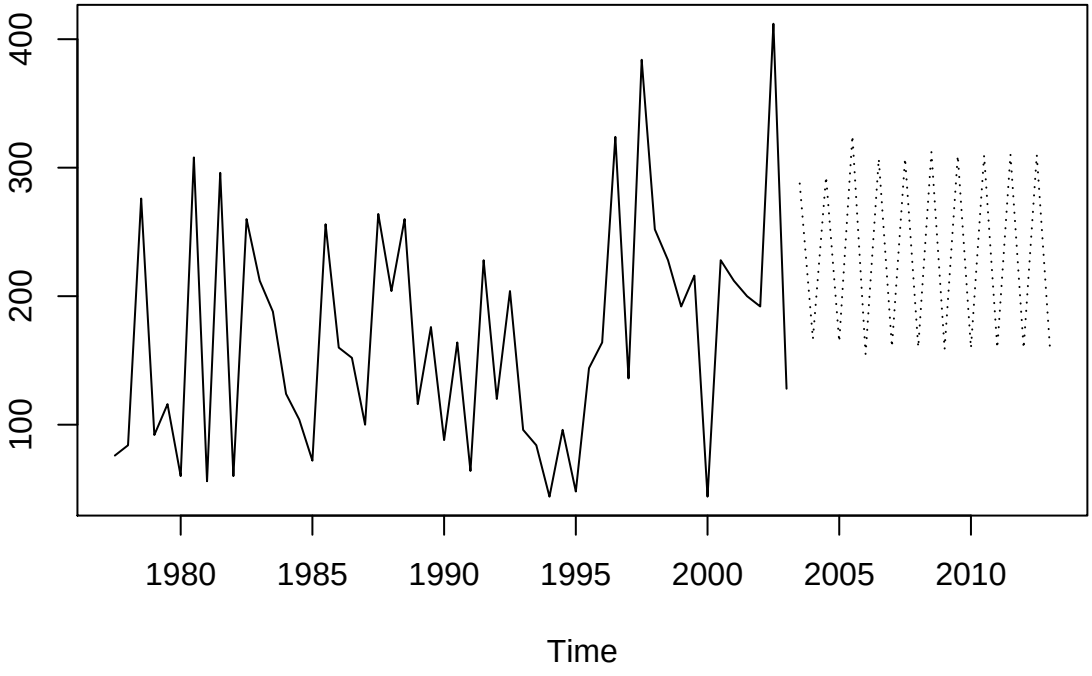
```
abund.arm <- auto.arima(abund.ts)
abund.arm <- arima((abund.ts), c(0, 0, 1), seasonal = list(order = c(2, 1, 0), period = 2), include.mea
tsdiag(abund.arm)
```








```
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 go to the trouble of setting up an experiment, whether it is in a laboratory, a hospital, or in the field, they tend to take more than one measurement on their experimental unit. Such studies are referred to as *longitudinal designs*. Historically, there has been confusion about how to deal with the fact that these measurements are non-independent and thus violate some of the major assumptions of most parametric statistics. Repeated measures Analysis of Variance (ANOVA) provides on way of analyzing factorically designed longitudinal studies. In the following section, we will show you how to perform and interpret a RM-ANOVA using data from the Portal Project.

Wrangle data for RM-ANOVA

Annual observed richness

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

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

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

# Pull out two of the five treatments
rich.treat <- rich.all[which(rich.all$plot_type == "Control" | rich.all$plot_type == "Rodent Exclosure"
names(rich.treat)[4] <- "richness"
#names(rich.treat)[names(rich.treat) == "rowSums.time.by.species....c.1..3..."] <- "richness"
```

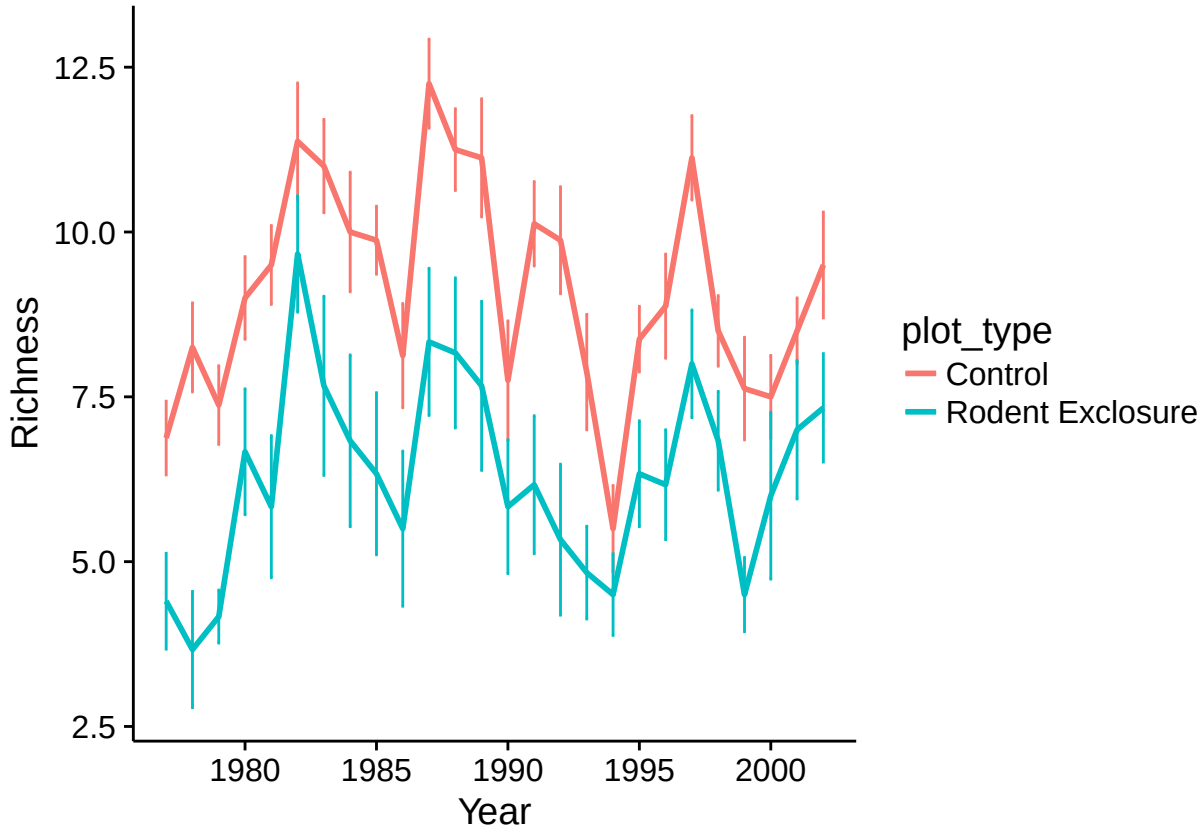
Plot data

In the following section, we'll return to the `group_by` function from `dplyr` along with pipes to retrieve the mean and standard deviation for the annual observed richness from the Control and Rodent Exclosure plots. After that we will introduce a new way of plotting data in R. While the base package in R can make beautiful plots and has a tremendous amount of flexibility, it can also require a lot of effort and code to generate figures. Another, perhaps easier, way to make figures in R is with the package `ggplot`. As you'll see below, with relatively few lines of code, we can make a nice-looking plot with `ggplot` showing the effects of rodent exlcosure on mammal richness.

After this, `dplyr` allows one to apply functions to the split dataset using functions like `summarise()`, which can return summary statistics (e.g., mean, min, max).

```
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.

### NW: I used your sem function below (as well as one from Week1) and it doesn't work on this data set

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)
```



Analyze the time series data with Repeated Measures Analysis of Variance (RM-ANOVA)

Background on RM-ANOVA. Why and when it's useful.

Expand to talk about covariance structure and how this ties in with ARIMA stuff above.

Have them do AIC tests with other structures.

Perhaps show them how to calculate LS-MEANS?

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

# We can use `summary()` to look at the output in detail
summary(rich.rm)
```

```
## Linear mixed-effects model fit by REML
##  Data: rich.treat
##        AIC      BIC    logLik
##   1590.462 1617.567 -788.2309
##
## Random effects:
```

```
##  Formula: ~1 | plot_id
##         (Intercept) Residual
## StdDev:    1.296979 2.246238
##
## Correlation Structure: AR(1)
##  Formula: ~1 | plot_id
##  Parameter estimate(s):
##     Phi
## 0.37097
## Fixed effects: richness ~ plot_type * year
##                                   Value Std.Error  DF    t-value p-value
## (Intercept)                    29.62737  57.07705 343  0.5190767  0.6040
## plot_typeRodent Exclosure     -62.03957  89.14103  12 -0.6959710  0.4997
## year                           -0.01033   0.02869 343 -0.3600129  0.7191
## plot_typeRodent Exclosure:year  0.02980   0.04480 343  0.6652702  0.5063
##  Correlation:
##                                (Intr) plt_RE year
## plot_typeRodent Exclosure      -0.64
## year                           -1.00   0.64
## plot_typeRodent Exclosure:year  0.64  -1.00  -0.64
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.98978687 -0.69549540 -0.07250359  0.68791058  2.73090608
##
## Number of Observations: 359
## Number of Groups: 14
```

```
# We can also use `pander()` to make a cleaner looking table of output
set.caption("RMANOVA for Portal")
pander(anova(rich.rm))
```

Table 1: RMANOVA for Portal

	numDF	denDF	F-value	p-value
(Intercept)	1	343	418.7	0
plot_type	1	12	12.28	0.00435
year	1	343	0.007381	0.9316
plot_type:year	1	343	0.4426	0.5063

Some practice chunks to be eventually deleted

```
# Some trials with calcuating abundance and richness
# time.by.species <- group_by(portal, year, plot_id) %>% count(taxon) %>% spread(key = taxon, value = n
# filter(time.by.species, plot_id==2)
# abundance<-rowSums(filter(time.by.species, plot_id==2)[,-c(1,2)])
# richness<-rowSums(filter(time.by.species, plot_id==2)[,-c(1,2)]>0)
#filter(time.by.species, plot_type == "Control", plot_type == "Rodent Exclosure")

richness<-rowSums(filter(time.by.species, plot_type == "Control", plot_type == "Rodent Exclosure")[,-c(
# biomass <- group_by(portal, year, plot_id) %>% summarize(rod.mass = sum(weight), na.rm = TRUE)
# p2<-filter(biomass, plot_id > 2)
```

```
# plot(rod.mass ~ year, p2, xaxt = "n", type = "l")
#
# port <- filter(portal, plot_id == 19)
# site2biomass <- group_by(portal, plot_id, year) %>%
#   summarise(
#     sum(na.omit(weight))
#   )
# plot(site2biomass, type = "l")
# plot(site2biomass[,1], site2biomass[,3])

# Richness plots by treatment

temp <- group_by(portal, plot_type, plot_id, year) %>% count(taxon) %>% spread(key = taxon, value = n, i
div <- vegan::diversity(temp[,-c(1:3)], metric = "richness")
temp$div <- div
tempdiv <- group_by(temp, plot_type, year) %>%
  summarise(
    mean = mean(div),
    sd = sd(div))

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

# Abundance plots by treatment
temp <- group_by(portal, plot_type, plot_id, year) %>% count(taxon) %>% spread(key = taxon, value = n, i
div <- rowSums(temp[,-c(1:3)])
temp$div <- div
tempdiv <- group_by(temp, plot_type, year) %>%
  summarise(
    mean = mean(div),
    sd = sd(div)/sqrt(n()))

plots <- ggplot(tempdiv, aes(x = year, y = mean, color = plot_type)) +
  geom_line(size = 1, show.legend = T) +
  geom_errorbar(aes(ymin = mean - sd, ymax = mean + sd), width = .1) +
  xlim(1977, 2002) +
  xlab("Year") +
  ylab("Div")
plot(plots)
```

6) TEMPORAL BETA DIVERSITY

The structure of an ecological community changes over time. We can conceptualize temporal turnover in much the same way as spatial turnover: how species change in abundance or presence over time.

A. Richness

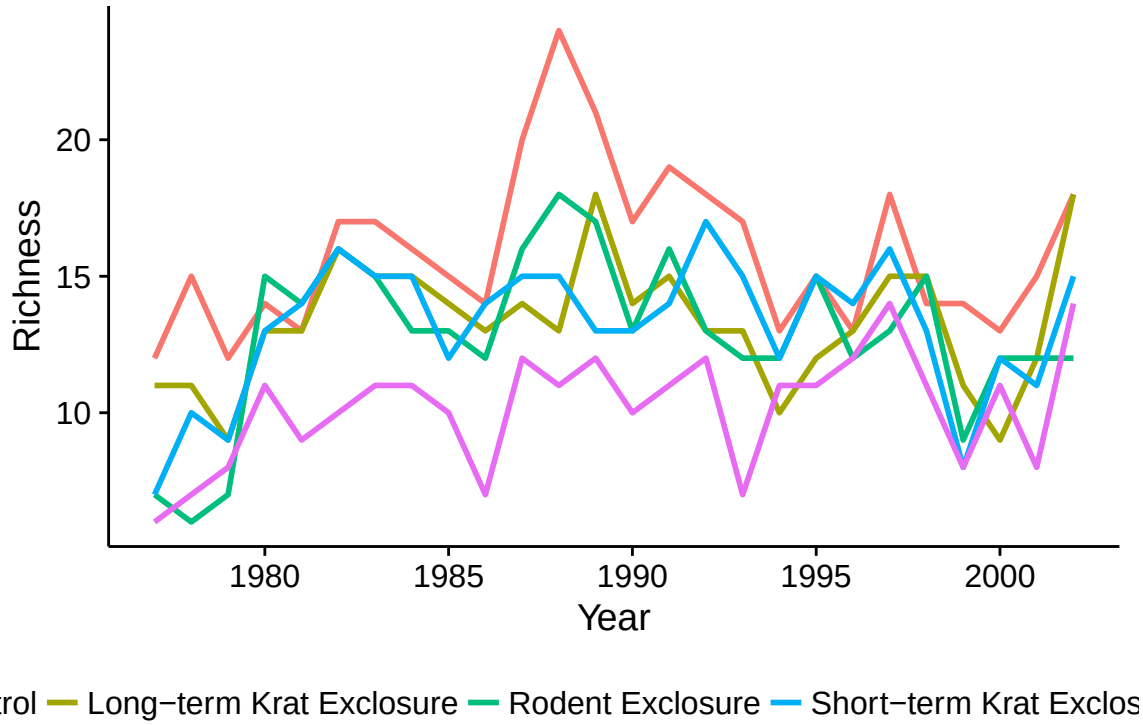
A simple measure of community change over time is how many species it gains or loses over the observed duration. In the Chihuahan Desert, disturbances such as drought may influence which species are found in the experimental plots over time. In addition, some species may simply be transients that happened to be censused one year, but not in others. The treatment design of the plots at the Portal site may influence which species are found in each group of sites. Before we proceed, we can generate a few hypotheses: * The control plots will have higher richness than the exclosure plots because many species are excluded from the exclosures. * Alternatively, the kangaroo rat exclosure treatments may have moderately high diversity if the kangaroo rat tends to competitively displace other small mammals. * Likewise, we can draw additional hypotheses about the exclosure of banner-tailed kangaroo rats (Spectab), and the exclosure of all rodents.

First, we can calculate the richness in each plot, in each year.

```
portal.richness.year <- group_by(portal, year, plot_type) %>%
  count(taxon) %>%
  summarise(richness = n())
```

Now, let's plot these on the same graph to visualize

```
rich.plot <- ggplot(
  portal.richness.year, aes(x = year, y = richness, color = plot_type)) +
  geom_line(size = 1, show.legend = T) +
  xlim(1977, 2002) +
  xlab("Year") +
  ylab("Richness") +
  theme(legend.position = "bottom")

plot(rich.plot)
```



It looks like the Control plots tend to have the highest richness, and had the highest peak richness. It is also worth noting that the Krat exclosures and rodent exclosures remained relatively similar, but that the Spectab exclosure consistently had low richness. There appears to have been an event that occurred around

1996 that led to a systematic crash in richness in all treatments, most severely in the exclosure treatments.

B. Turnover

Although we have just described general trends in richness, we have not yet learned anything about the change in species composition over time (i.e., species turnover). Turnover gives us an idea of how similar species composition is over time: low turnover means that the composition and its relative abundances remain relatively stable through time, while high turnover suggests a highly dynamic community structure. Just like turnover in space, turnover can be driven by the introduction of new species or the loss of resident species. Here we will calculate the overall turnover, but also assess how this metric is influenced by the introduction of new species versus the disappearance of resident species.

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

To perform these analyses, we will use the `codyn` package (Hallett et al. 2016), which calculates a number of metrics to analyze community dynamics.

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

# This data.table now contains a new column `n` that represents the species counts


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

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

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 now contains a column for the value of the turnover metric, the second year in the pairwise comparison, and the type of plot. If we examine the data structure, note that the only difference between each object the column containing the calculation of the metric. For easier plotting, we'll combine these columns into a single data table

```
# Let's join the columns by the 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")
```

```
# Here, let's turn this back into long-form using gather
portal.turnover <- gather(portal.turnover, key = metric, value = turnover,
       total, appearance, disappearance)
```

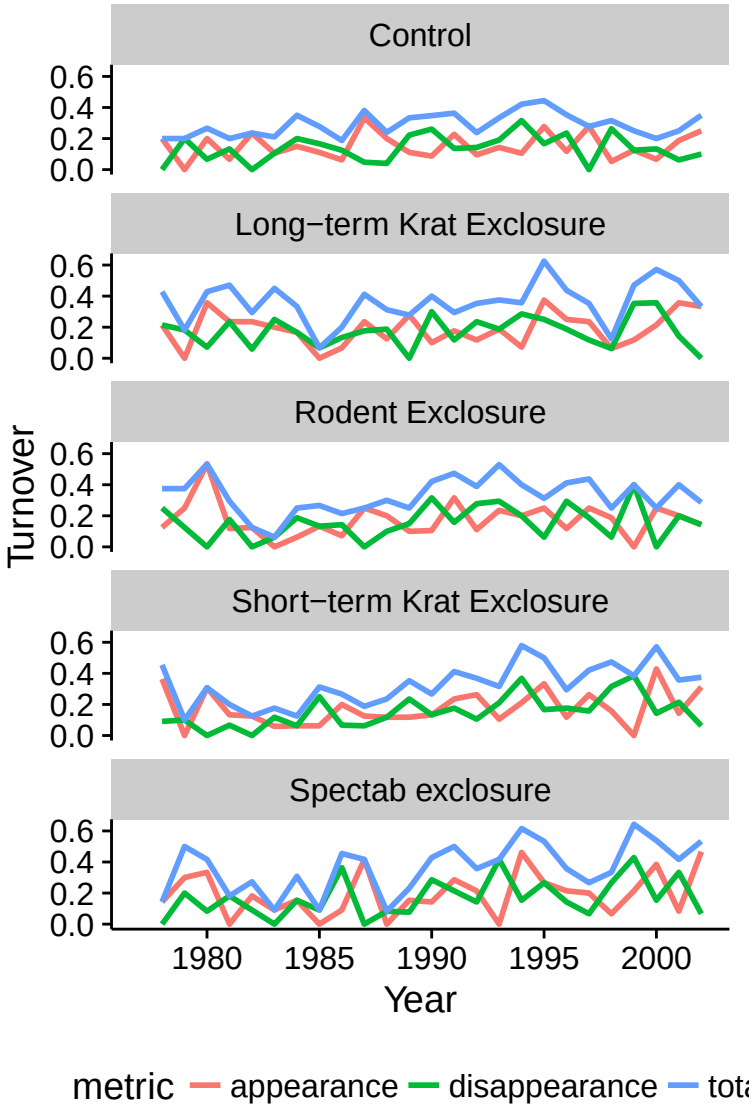
In the above code chunk, we did a lot of data wrangling using only a couple lines of code. Think about what we just did. We had three separate data frames containing different turnover metrics. But those metrics referred to the same plots and years. So, we joined them together using their shared year > plot_id index to create one data table. We used a pipe `%>%` to first join total and disappearance values, then piped this joined data table to be the first argument in another join function. Run each of these `full_join()` statements independently to see what it is doing at each step.

Next, to facilitate plotting, we can turn this wide form data table back into a long form. We did this using the `gather()` function. This function says: in `portal.turnover`, create a new column of keys and call it "metric". Create another new column of associated values and call it "turnover". The keys in the metric column will be the old columns we are gathering together (total, appearance, disappearance). The values that go in turnover will be 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)`).

Let's visualize this.

```
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

While turnover gives a measure of how dynamic a community is over time, it would also be interesting to learn how often dominance (i.e., the most abundant species) changes over time. If turnover is due mostly to gains and losses of rare species, with little change to the dominant members, this is not uncommon. However, if the community sees drastic rearrangements of community dominance (i.e., the most abundant species change ranks frequently) this could suggest something different is going on. Mean rank shifts measure the relative change in species rank abundances.

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

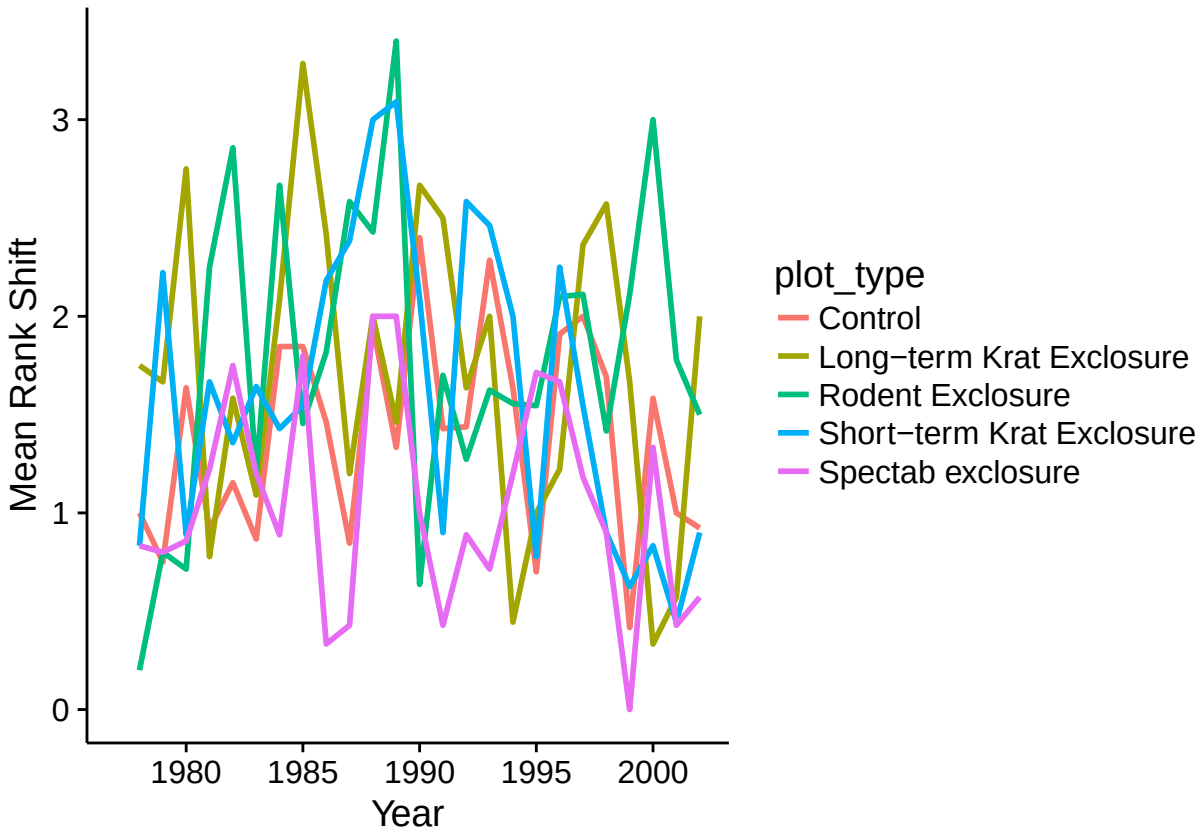
Here, we calculate MRS, plot it, and characterize its variability.

```
# above we calculated abundances of species in each site over time
portal.species.abunds
```

```
## Source: local data frame [1,702 x 4]
```

```
## Groups: year, plot_type [?]
##
##     year plot_type                    taxon     n
##    <int>    <fctr>                    <chr> <int>
## 1   1977   Control Chaetodipus_penicillatus     6
## 2   1977   Control       Dipodomys_merriami   108
## 3   1977   Control          Dipodomys_ordii     6
## 4   1977   Control    Dipodomys_spectabilis    47
## 5   1977   Control         Neotoma_albigula    22
## 6   1977   Control    Onychomys_leucogaster     5
## 7   1977   Control            Onychomys_sp.     2
## 8   1977   Control       Onychomys_torridus     9
## 9   1977   Control       Perognathus_flavus    13
## 10  1977   Control      Peromyscus_eremicus     3
## # ... with 1,692 more rows
```

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

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

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)
```




```
# How variable are each groups? Let's calculate the coefficient of variation
group_by(portal.rankshift, plot_type) %>% summarise(variability = sd(MRS) / mean(MRS))
```

```
## # A tibble: 5 × 2
##                   plot_type variability
##                       <chr>       <dbl>
## 1                   Control   0.3759483
## 2  Long-term Krat Exclosure   0.4482928
## 3          Rodent Exclosure   0.4361809
## 4 Short-term Krat Exclosure   0.4751513
## 5         Spectab exclosure   0.5234745
```

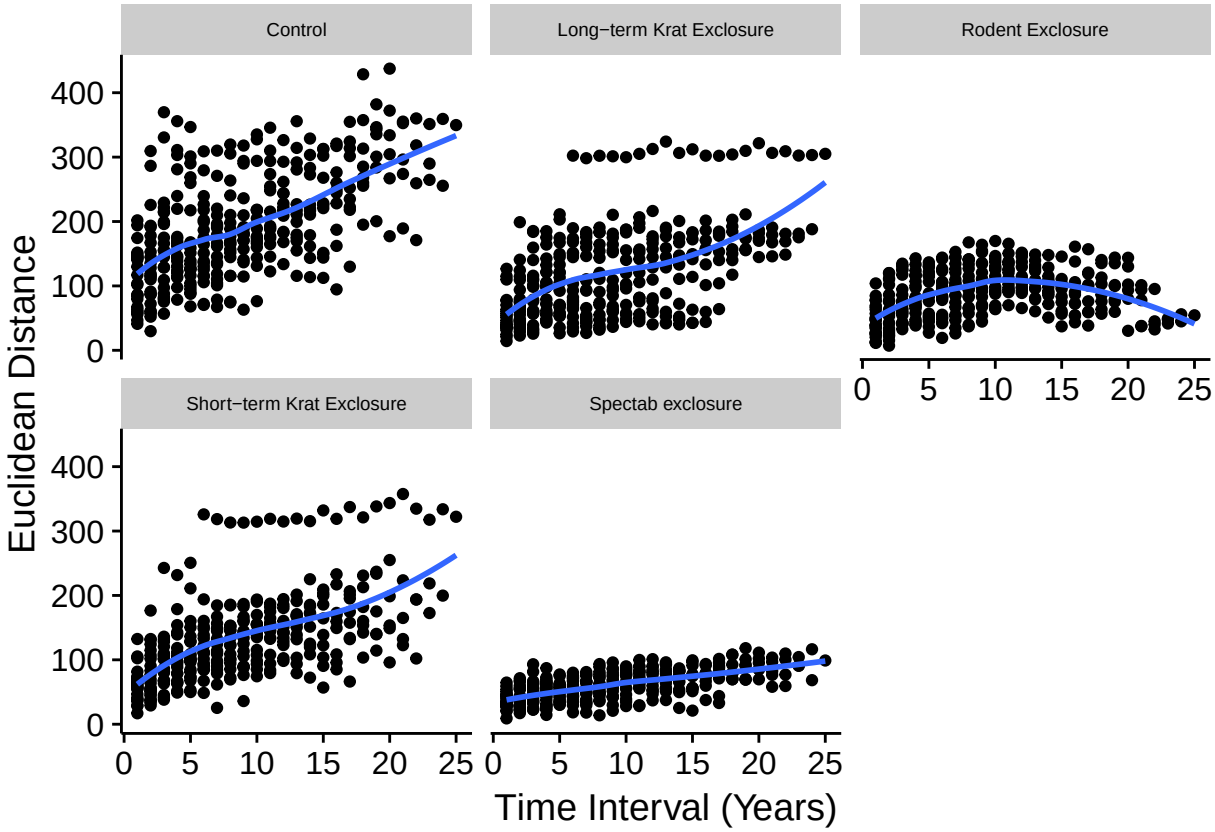
```
# The cyclic_shift function is not recognizing portal.species.abunds as the data frame. NW, are you fam

#cyclic_shift(df = as.data.frame(portal.species.abunds),
#  time.var = "year",
#  species.var = "taxon",
#  abundance.var = "n",
#  replicate.var = "plot_type",
#  FUN = rank_shift(),
#  bootnumber = 10)
```

D. Rate Change Interval

The temporal lag in community similarity suggests a time duration at which communities become sufficiently dissimilar to one another. For example, in highly dynamic systems with high species replacement and large mean rank shifts, temporal turnover may saturate after a relatively short time interval, say 5 years, after

which pairwise dissimilarity values remain sufficiently different. Alternatively, in more stable communities, comparably large pairwise dissimilarities may take tens if not hundreds of years.

```
portal.total.abunds <- portal.species.abunds %>% group_by(year, plot_type) %>% count(wt = n)
portal.change.int1 <- rate_change_interval(portal.species.abunds,
                      time.var = "year",
                      species.var = "taxon",
                      abundance.var = "n",
                      replicate.var = "plot_type")

rate.plot1 <- ggplot(portal.change.int1, 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("Euclidean Distance") +
  xlab("Time Interval (Years)")
rate.plot1
```



Each plot looks quite different from one another. The control plot tends to keep increasing, while the Krat exclosures resemble one another and seem to saturate. The Spectab exclosure shows relatively little dissimilarity even across the whole 25-year experiment. The rodent exclosures appear to have gone through a long-term fluctuation and seem to more closely resemble each other after 25 years apart than after 10 years apart.

But what is on the y-axis? Use `help(rate_change_interval)` to find out.

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, especially noting the hump-shaped pattern of the rodent exclosure treatment. 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.
```

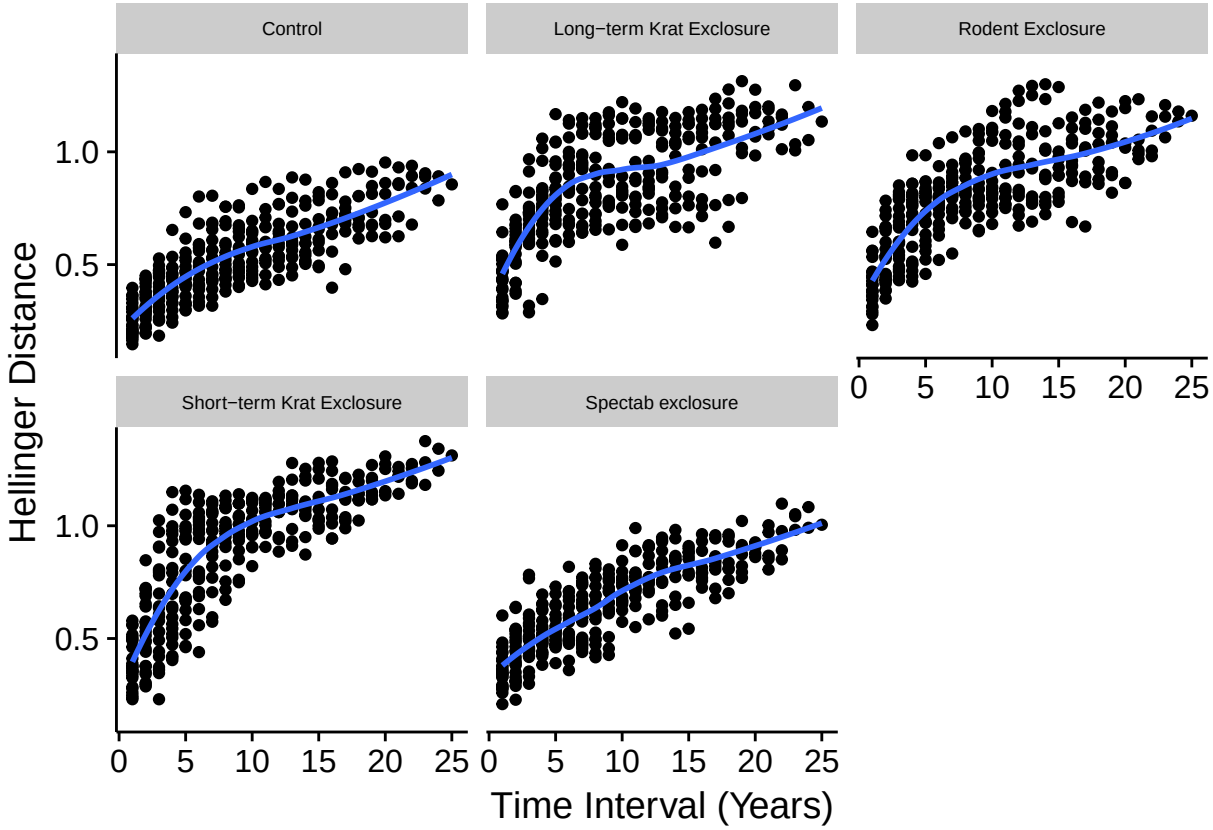
```
## Source: local data frame [130 x 3]
## Groups: year [?]
##
##     year                 plot_type    nn
##    <int>                    <fctr> <int>
## 1   1977                   Control   223
## 2   1977  Long-term Krat Exclosure    65
## 3   1977          Rodent Exclosure    58
## 4   1977 Short-term 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
```



These figures tell a much different story than the previous ones using just the Euclidean distances alone. Thus, it remains important to understand the distance metrics you are using and how they are influenced by species abundances.

7) SYNCHRONY, or covarying species; perhaps stability, compensatory dynamics, or variance ratio?

Species may also exhibit synchrony: their changes in abundance are not independent from one another. . . .

http://onlinelibrary.wiley.com/doi/10.1111/2041-210X.12188/epdf https://cran.r-project.org/web/packages/vrtest/vrtest.pdf

A. Species synchrony

```
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"
```

B. Variance Ratio

VR = 1 species do not covary, VR > 1 species positively covary VR < 1 species negatively covary Bootstrapping... CI returned.

```
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 = 100,
               average.replicates = T,
               level = 0.95)
```

C. Community Stability

Temporal mean over standard deviation of aggregate species abundances

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

# Let's take a look at stability metrics across exclosures
pander(full_join(portal.loreau, portal.gross) %>%
  full_join(portal.stab))
```

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

plot_type	loreau	gross	stability
Control	0.1869	0.1263	3.044

plot_type	loreau	gross	stability
Long-term Krat Exclosure	0.1578	0.07418	1.865
Rodent Exclosure	0.187	0.1423	1.864
Short-term Krat Exclosure	0.08773	0.001546	2.462
Spectab exclosure	0.1542	0.1393	2.911

8) Using environmental drivers to explain temporal biodiversity data (univariate, multivariate, both?)

loading and plotting precipitation data

perhaps take the sum rather than mean of precip below

```
precip1 <- read.table(file = "./data/Portal_precipitation_19801989.csv", header = T, sep = ",")
precip2 <- read.table(file = "./data/Portal_precipitation_19892002.csv", header = T, sep = ",")

precip.month <- unite(precip1, col = date, c(Year, Month), sep = ".", remove = F)
precip.month$date <- as.Date(precip.month$date, "%Y-%m")
precip.month$date <- as.numeric(precip.month$date)

precip2 <- read.table(file = "./data/Portal_precipitation_19892002.csv", header = T, sep = ",")
precip2 <- unite(precip2, col = date, c(Year, Month, Day, Hour), sep = "-", remove = F)
parse_date_time(precip2$date, "%y-%m-%d-%H%M")

monthly.precip <- group_by(precip2, Year, Month) %>%
  summarise(mean = mean(Precipitation))
monthly.precip$date <- unite(monthly.precip, col = date, c(Year, Month), sep = ".")
plot(monthly.precip$date, monthly.precip$mean)
```