**A Primer Using a Well Pad Plant Community Dataset to Learn How to Run Multivariate Ecological Analysis for Unconstrained Non-metric Multidimensional Scaling (NMS) Ordination, perMANOVA, and Indicator Species Analysis using R (4.0.2) and RStudio Desktop**

Michelle Rude, Chantal Ricard, Randi Lupardus, Anne CS McIntosh

Table of Contents

[1. Introduction: 3](#_Toc73973722)

[1.1 About This Primer 3](#_Toc73973723)

[1.2 Background Information 3](#_Toc73973724)

[1.3 Getting Started with RStudio (Installation) 5](#_Toc73973725)

[2. Creating a Project in RStudio: 5](#_Toc73973726)

[2.1. Opening RStudio – and a Brief Overview of Windows 5](#_Toc73973727)

[2.2. Saving Your Rstudio Files 7](#_Toc73973728)

[2.3. Entering Text and Code Into Your R script File 8](#_Toc73973729)

[2.4. Downloading and Installing Packages 9](#_Toc73973730)

[3. Importing and Modifying Datasets 11](#_Toc73973731)

[3.1 Importing Datasets 12](#_Toc73973732)

[4. NMS Ordination: 13](#_Toc73973733)

[4.1. Stress (Scree) Plots – Selecting Number of Axes/Dimensions for Ordination 14](#_Toc73973734)

[4.2 Running the Ordination 16](#_Toc73973735)

[4.3 Ordination Graph Plotting Options 18](#_Toc73973736)

[4.4 Vectors (Environmental Variables) 23](#_Toc73973737)

[4.5 Preparing to Present 24](#_Toc73973738)

[5. Permutational Multivariate Analysis of Variance (perMANOVA): 27](#_Toc73973739)

[5.1 Running perMANOVAs 27](#_Toc73973740)

[6. Indicator Species Analysis (ISA): 28](#_Toc73973741)

[6.1 Identifying Species by Sites 28](#_Toc73973742)

[7. Summary Statistics: 28](#_Toc73973743)

[7.1 Reporting Cover Values 28](#_Toc73973744)

[8. Reporting Methods and Results: 32](#_Toc73973745)

[8.1 Reporting Methods – Including Demonstration 32](#_Toc73973746)

[8.2 Reporting Results 33](#_Toc73973747)

[9. References: 33](#_Toc73973748)

[9.1 Additional Online Resources You May Find Useful 33](#_Toc73973749)

[10. Appendix 35](#_Toc73973750)

[10.1 Species codes 35](#_Toc73973751)

# 1. Introduction:

## 1.1 About This Primer

This primer is structured to assist you step-by-step with importing and modifying datasets, and running community-level ecological statistical analyses using RStudio. RStudio, which uses the R programming language, is beneficial to learn because it is versatile and free and a very powerful statistical platform! However, learning how to use RStudio and R can be tricky for individuals without a computer science background because they require knowledge of coding and are very different than using traditional point-and-click statistical software. We have developed this primer to help walk you through the steps for conducting multivariate ecological analyses in RStudio and then reporting them in your research paper/assignment. You will learn how to conduct three main types of multivariate visualization and analyses: i) visualize your data using a non-metric multidimensional scaling (NMS) ordination, ii) test for differences among groups of samples in your dataset (using perMANOVA), iii) identify species associated with groups (using Indicator Species Analysis (ISA)), and iv) calculate descriptive/summary statistics for plant species of interest. The primer uses a published sample community ecology dataset on well pad recovery that you will download and use as you work through the primer. Helpful hints will be provided throughout the primer for how to effectively use RStudio.

## 1.2 Background Information

R is a programming language used to develop an open-source integrated suite of statistical software tools. R can be used to conduct a diverse array of statistical analyses and both numeric and graphic summaries of data (see <https://www.r-project.org/about.html> for detailed information). R, unlike other software programs that you may have used (e.g., IBM SPSS, JASP), is based on writing code rather than clicking on menu-driven commands to conduct your data organization and analyses. While this can be intimidating for introductory users of the program, once you gain some experience working with it you will start to appreciate the benefits it has. For example, R is beneficial because you can quickly replicate what you did by saving your code and then readily re-running it in the future. RStudio is a separate open-source ‘front end’ desktop application that integrates the R programming language running in its ‘back end’ (see https://RStudio.com/products/RStudio/). RStudio was developed as a more user-friendly easy-to-learn interface compared with R. In this primer, you will use RStudio to write and run all of your analyses.

Statistical software programs can aid users in conducting many different types of statistical analyses. Univariate statistical analyses focus on a single response variable, whereas multivariate analyses focus on simultaneously studying multiple response variables at the same time. In this primer we will be exploring how to import and modify a dataset, and then focus on conducting three different types of multivariate visualization and analyses of a sample community ecology dataset:

* Visualize patterns in the dataset using non-metric multidimensional scaling (NMS) ordination. Ordination is a way of visualizing complex multivariate datasets in a reduced number of dimensions (often 2-dimensions) that enables the viewer to look for patterns in the data points. While ordination is challenging to understand – the important thing to note is that an ordination visualizes sample units based on the relative similarity or dissimilarity among the aspects of the data that you have captured in your multivariate dataset. Study units that have more similar composition will be positioned more closely together on the ordination plot, whereas study points that are more dissimilar in their composition will be further away in the visualized ordination plot. Non-Metric Multidimensional Scaling (**NMS**, MDS, NMDS, or NMMDS) is an ordination method that is well suited to data that are non-normal or are on arbitrary, discontinuous, or otherwise questionable scales. NMS is generally the best ordination method for community-level ecological data (McCune and Grace 2002
* Testing for differences among groups (permutational multivariate analysis of variance (perMANOVA). A perMANOVA is used to test for significant differences in the multivariate data within your study among two or more levels of a grouping (categorical) variable that are of interest to you. It functions like a univariate t-test (two levels only) or ANOVA (more than than two levels).
* Identifying species associated with different groups (Indicator Species Analysis). Inidicator species analysis is used to identify species (or whatever other aspect of your multivariate dataset you are studying) that are associated with each level of your grouping factor. This is only done when your perMANOVA (or another type of test that is used to test for multivariate differences) has indicated that there are significant differences among the levels of your grouping factor. It gives you the opportunity to explore which species, etc. that is contributing to the separation of your groups.

For this primer you will be using an ecological dataset collected in 2014 that quantified recovery of forest understory plant communities on well pads that were subsequently reclaimed. Data were collected on both reclaimed oil and gas well pads and adjacent undisturbed reference sites (forested land). The study area was located in both the Central Mixedwood (n=15) and Lower Foothills (n=15) Natural Subregions in Alberta, Canada. The area is characterized by deciduous, coniferous and mixedwood forested landscapes, with varying topographical, hydrological and ecological regimes (Natural Regions Committee 2006). For the purposes of the analyses in this primer, we are interested in comparing the patterns in the plant community on well pads recovering from this anthropogenic disturbance agent, with undisturbed reference forest sites to evaluate whether or not they have recovered in the time since they were reclaimed.

Be sure to have downloaded and saved the two sample dataset .csv files that you will be using in this analysis (Primarydataset.csv and Seconddataset.csv)

Throughout this primer where you see Example: **with bold text that follows** – you should type in the bolded information to the right of the example into the R script window in the upper left box of RStudio.

Example: **maindata <- read.csv("Primarydataset.csv")**

You would paste into the R script:

**maindata <- read.csv("Primarydataset.csv")**

Note: RStudio doesn’t like angled quotes “” so be sure you are using **"")**

## 1.3 Getting Started with RStudio (Installation)

Start by downloading and installing the two open-source free statistical programs that you will need to use this primer: R and RStudio Desktop (follow instructions provided on websites depending on which type of operating system you have):

* R URL: [https://cran.RStudio.com/](https://cran.rstudio.com/) (the back-end programming language that RStudio uses)
* RStudio URL: <https://www.RStudio.com/products/RStudio/download/> (the user-friendly interface for working with R that you will be using).

Note: For this primer we have downloaded R v 4.0.2 for Windows (32/64 bit). The version of R that you download may be a newer version or on a different operating system. We do not expect issues with functionality as R is updated and used on different operating systems, so expect that this primer should still operate and not affect your results, but if there are issues please alert us.

# 2. Creating a Project in RStudio:

## 2.1. Opening RStudio – and a Brief Overview of Windows

1. Open **RStudio** (you will not be opening R – it will be working behind the scenes with RStudio). When you first open RStudio you will see three (the first time you open Rstudio – Fig. 1a) or once you have a R script window open you will see four different windows (also called panes) open (Fig. 1b). These panels are described below (Note that this background information is provided for your information – do not worry if all of this information doesn’t make sense to you when you start to use R – it will make more sense once you have used R):

* **The script editor panel:** This panel appears in the upper left after you have opened or created a new script (see steps below). You can open the script editor by either creating a new empty script or by opening an existing script. You can open a new empty script by clicking the New File icon in the upper left of the main RStudio toolbar. This icon looks like a white square with a white plus sign in a green circle. It is the script panel where you will be inputting all of your R code that you will want Rstudio to run for you.
* **The command console/panel:** The console is in the left of the Rstudio interface and after initial opening when you have a script file open, this console is in the lower left of Rstudio. This panel executes R commands and responds to those commands (except for graphs that will display in the plots panel) that you have input and run in the script editor panel (we will come back to this below).
* **The environment panel**: The environment panel is in the upper right of the Rstudio interface and displays current active R objects. Objects are ‘containers’ in R – they will provide you with information on the data that you have imported into R and other information about the data as it is modified based on your commands in R.
* **The history panel:** This panel also in the upper right, tabbed with the environment panel, shows all of the code that has been executed in the current session.
* **The files panel:** This panel in the lower right provides you with the contents of the current working folder. The file names in this panel can be helpful when trying to remember the name of the data file you want to load for your current session.
* **The plots panel:** This panel is also tabbed in the lower right. Any plots that you execute in Rstudio will show up in this panel.
* **The packages panel:** This panel is also tabbed in the lower right. Packages are collections of R code that are functions and datasets that have been previously developed by R users. These pieces of code will help the program know what to do with your data and are much more efficient at conducting analyses than starting from scratch to analyze your data. In this primer you will be using multiple packages (e.g., VEGAN). This panel shows all of the packages that are installed on your computer and those that are being used in the current session will have a check-mark next to them.
* **The help panel:** This panel (tabbed in the lower right) contains help information.
* **The viewer panel:** This panel is also tabbed in the lower right. It can be used to view local web content within Rstudio.

a)

Graphical user interface, text, application

Description automatically generated

b)

Graphical user interface, text, application

Description automatically generated

Fig. 1. a) the R Script windows/panes/panels the first time you open RStudio, and b) the R Script windows/panes/panels. Red box outlines the script window to type code into in the upper left. Here we have added the setwd to the beginning. The lower left window shows the console where your code output will appear. Upper right window contains the imported and created datasets under the tab Environment. The lower right window contains files, plot output, and common packages. The Run icon featured in small red box, runs requests entered in the R script window when the cursor is placed at the end of the line (Note: Run keyboard shortcut = CTRL+Enter).

## 2.2. Saving Your Rstudio Files

1. To begin, create a new RStudio project which will keep your project associated files organized in a single location that is named after your project: select **File -> New Project** a pop-up will prompt you with several options. Select **New Directory -> New Project**. Save the new project as a meaningful Project name e.g. **Well\_vs\_Ref** (Fig. 2) within the directory where your files that you are going to open are found (find this folder using **Browse…**). Files throughout the steps that follow need to be saved in the same location that was set under **Directory Name** when saving the **New Project** . To locate your project directory Select **Browse** and find the location where your .csv files (Primarydataset.csv and Seconddataset.csv) are stored in the drop-down menu (Fig. 1). Then click on **Create Project**.
2. Next you will create a new script file within this project (select **File - New File - R script)**. A fourth window, R script, will then open (Fig. 2). You should then save your R script file (select **File -> Save as)** with a meaningful name e.g., WellvsRef.R . The .R extension is used for R script files. This file is saved within your project folder that you just created above. (hint: you should remember to save your R script file regularly as it won’t autosave for you as you enter new code and text in it)

Graphical user interface, text, application, email

Description automatically generated

Fig. 2. Naming a new project and setting directory for file imports. Directory needs to be set to the location of the files you are importing. Leave “Use renv with this project” and “Open in new session” unchecked.

## 2.3. Entering Text and Code Into Your R script File

1. Now that you have a R script file, try adding an introduction line of text to your file that provides a description of what this file is going to contain. This way when you open it back later (or someone else does) your RScript file will make more sense. Any notes or comments within your R script file that you want to make should be started with # (e.g., Fig. 3a). (Hint: if you put a pound **#** symbol on a line of code – RStudio will know it is text and will ignore it rather than try to analyze or “Run” it). It is very helpful to get in the habit of providing yourself notes throughout the R script file so that next time when you come back you can help remind yourself (or others who read your code) what you did. (Hint: try to get in the habit of doing this as you go through this primer!)

a)

Text

Description automatically generated

b)

Text

Description automatically generated

c)

Graphical user interface, application

Description automatically generated

**Fig. 3.** a) Demonstration of the RScript code with introductory information added using a # to start the code so that R knows not to try to read that information as code. You can highlight the code that you want R to run or put the cursor at the end of a single line and click on Run and it will run your code for you. (Hint: CTRL+Enter = shortcut to run a single line of code where your cursor is). b) Demonstration of how RStudio demonstrates an error in a line of code – in this case there is a missing quote before vegan that has caused the error. c) Demonstration of how if you hover over the red x you will get a clue about what the error in the code is.

1. Next up you need to set the working directory for your project in your R script file (e.g., Fig. 3a). This working directory is where you have saved your project and R script file and all of the datasets that you are going to later import into R (instructions below). All of your files that you are working with should be located in this directory.

***\* Every time you close the R program you will have to reset the working directory.\****

You will add this to the start of your R script window using the set working directory code: “**setwd**” and putting in the appropriate directory where your files are saved, for example type a second line into the R script window/pane (Fig. 3a):

Example: **setwd("G:/My Drive/WellRefDataset")**

(hints: RStudio can only read upright quotes not smart quotes “” and directory locations must be using forward slash “/” rather than backwards slash “\”. When setting your working directory, be sure to type in the location where YOUR data live – not just copy and paste the G: location where we saved our datasets..

## 2.4. Downloading and Installing Packages

For the ecological multivariate analyses that we will be conducting, you will need to download packages (reminder - these are pieces of code in R that will help the RStudio program know what to do with your data).

In this primer we will be using the following packages: **vegan, MASS, goeveg, indicspecies, psych, permute, lattice.**

You can use your R script to open the packages for you – just like you did for setting the working directory above. Input the following code into your R script file (upper left window/pane from Fig. 1b):

**install.packages(c("vegan", "MASS", "goeveg", "indicspecies", "psych", "permute", "lattice"))**

(Reminder: You have to put your cursor at the end of the line and click on the “**RUN**” button in the upper right of the script panel or hit short-cut keys of CTL+Enter ). (Fig. 3)

You will only need to download packages once during a session (R will put them in a temporary directory). Later during your analysis after you have packages downloaded you will need to call them (‘load’ them) – which you will learn about in the NMS section below. The console window (Fig. 4) tells you whether the actions you completed – in this case the installation of the packages was successful or not (red text indicates issues – although you may also encounter some red text in your console panel that is then resolved as your Rscript code continues to run). (Note: any time you close the R program you will have to re-open these packages to continue running analysis).

Text

Description automatically generated

**Fig. 4.** Example output from successfully downloading the packages: **vegan, MASS, goeveg,** **indicspecies**, **psych, permute, lattice**. (There may be temporary error messages above this).

Note: You may get error/warning code in R – take a look and see if the warnings are resolved as you work through your analyses (e.g., you may get a warning at this stage “WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:… and then it installs the RTools for you and proceeds”. So in this case we do not have to worry about this being an unresolved error.

Note: RStudio will add a red circled x to any line where there is an error in the code that you have provided in your RScript file (e.g., Fig. 3b). If you hover your mouse cursor over the red x, it generally tells you what the issue is, which is super helpful (e.g., Fig. 3c).

# 3. Importing and Modifying Datasets

Now that you have RStudio open and have set up your project and script it is time to bring in the datasets that you want to analyze. They need to be in the proper format for RStudio to analyze them – take a look at the two comma separated value (.csv) files to familiarize yourself with them: Primarydataset.csv and Seconddataset.csv (Figs. 5,6). Keep in mind, these data have already been organized for you from the raw data that the scientists collected and input. Your own raw data that you would collect in a study would need to be organized in the same fashion to perform the multivariate analyses we are doing in this primer. Note: Data for individual sample units are in rows and each variable/property is in a column. The ecological data are separated into two separate files: the primarydataset.csv is where the plant community dataset that you are interested in studying are housed – in our study, the individual columns are percent cover for individual plant species recorded within each of the well pads and reference sites. The seconddataset.csv is where the remaining information that are associated with each plot in our dataset live – including all of the categorical/grouping variables (in our plant community dataset this includes a grouping variable to distinguish between wellsite and reference sites) and any additional environmental variables that were collected. The two datasets are linked based on the ID field that uniquely identifies each study site.

A picture containing text, indoor

Description automatically generated

**Fig. 5.** The main dataset for the understory vegetation community response variables saved in **csv** format. The first ID column is the primary key/unique identifier for each study unit and corresponds to the ID column in the second dataset. The remaining columns are the percent cover values for each species within the understory vegetation community using seven letter species codes (first four letters are the start of the genus and last three letters are for the species).

A screenshot of a computer

Description automatically generated

**Fig. 6.** The second dataset for the environmental variables or design variables saved in csv format. In this example the unique identifier ID is in column A, categorical variables are columns B-D. The quantitative variables are columns E-U.

## 3.1 Importing Datasets

Now you will import Primarydataset.csv and Seconddataset.csv into RStudio. Remember that they need to be within your main project folder working directory (in our case Well\_vs\_Ref).

1. When importing data for analysis or re-loading packages to run analysis you will need to run these requests in your R script window (upper left screen window) (Fig. 1b).
2. You must ensure that the datasets you are trying to import are saved as comma separated value files (csv). To do this in MS Excel, open up your spreadsheet file and select **File > Save as.** Select the location on your computer you have set for your working directory. Before you hit **save,** select the drop-down menu next to **save as type,** and scroll until you find **csv (comma delimited)**.
3. When importing datasets the names must be written identically to their file names within quotations (**"** **"**) and ending in **.csv**. Once your dataset is imported, you can call your datasets whatever names you want in R and they don’t have to match up with what your .csv file names are – here we will use maindata and seconddata as the names for our datasets in R.

First, name the file in R by typing an arrow and operation: **""** **<- read.csv ("").** (reminder: R doesn’t like angled quotes “”). You should also tell R that your first row of data are the headers/labels for each column in your dataset, so it doesn’t try to treat them like data values. That is why you include header=TRUE in the code below. In addition row.names=1 indicates that each row has a unique identifier/name associated with it.

For our study enter the following:

**maindata <- read.csv("Primarydataset.csv", row.names=1, header = TRUE)**

Place your cursor after each line and select **Run.**

You have now imported the plant community dataset, or primary dataset naming the file **maindata**. Follow the same steps to import the environmental variables dataset (second dataset).

For our study:

**seconddata <- read.csv ("Seconddataset.csv", row.names=1, header = TRUE)**

Place your cursor after each line and select **Run.**

You have now imported the grouping and environmental variables from your second csv file.

Note: You will see these datasets appear in the Environment screen window in the upper right (**Fig. 7**) after they are successfully uploaded. When datasets are imported they can be opened for viewing by double clicking on their name which appears in the environment screen (top right) (Fig. 2). Open datasets to ensure that they are correct before proceeding with analyses.

a)

Text

Description automatically generated

b)

Table

Description automatically generated

**Fig. 7.** a) The environment panel displaying the two datasets that have been imported into RStudio – maindata and seconddata. b) If you click on one of the datasets it will show in your Left script window as a separate file so you can review and ensure it imported correctly.

# 4. NMS Ordination:

Once your two datasets have been properly imported and organized, you can proceed with running an ordination. While ordination is challenging to understand – the important thing to note is that an ordination visualizes your study units based on the relative similarity or dissimilarity among their plant communities. Study units that have more similar plant community composition will be positioned more closely together on the ordination plot, whereas study points that are more dissimilar in their plant community composition will be further away in the visualized ordination plot. Once you create your plot this should make more sense to you. Non-Metric Multidimensional Scaling (**NMS**, MDS, NMDS, or NMMDS) is an ordination method that is well suited to data that are non-normal or are on arbitrary, discontinuous, or otherwise questionable scales. NMS is generally the best ordination method for community-level data (McCune and Grace 2002).

## 4.1. Stress (Scree) Plots – Selecting Number of Axes/Dimensions for Ordination

1. Stress (“scree”) plots are a way to evaluate how many dimensions (axes) your final ordination plot should have for optimal simplified representation of your data. This information is needed prior to running your ordination. To visually represent the ordination output stress data using a scree plot, load the goeveg package:

**library (goeveg)**

Followed by the script below:

**dimcheckMDS (maindata, distance = "bray", k = 6, trymax = 20, autotransform = TRUE)**

1. The dimcheckMDS function produces a stress plot (scree plot) in your **Plots** pane/window (Fig. 8), visually representing the preferred dimensions for plotting your ordination based on stress values. You can copy and paste this into a Word doc (Rt click on the imagine in the Plots pane and see options) and once in Word adjust the shape of it for optimal viewing. For the stress plot we are testing the community dataset (maindata). K = max number of dimensions/axes (default =6). In this example we are specifying a reduction in stress with step-down in dimensionality from six all the way to one, testing for the preferred dimensionality to represent the data. Essentially you are taking the X # of dimensions that your original dataset has (based on how many species you have in the study) and trying to look at them collectively in a much reduced dimensionality that you can graph and interpret. There are no fixed statistical criteria that will provide you with the correct number of dimensions to select. However, one thing to consider is the interpretability and ease of use – and the less dimensions your solution has, the easier you will be to display it on a graph and describe the visual patterns that you observe (Kruskal and Wish 1978). However, there are trade-offs depending on the number of dimensions you select. If you reduce dimensionality too much (e.g., to one dimension solution), then your solution will have over simplified the patterns in your data and not provide you with a meaningful solution. In our experience researchers tend to select either a two or three-dimensional solution – as long as the stress meets the criteria of Table 1 below.

In this example we have also specified the distance measure “bray”, set the maximum number of random configurations to 20 iterations, and left the transformation argument to the default (TRUE). Refer to Table 1 for interpretation of stress values. You can also see McCune and Grace (2002) pgs. 129-133. We are looking for the ‘elbow’ where there is the most decrease in stress that is around stress of 20 or below.

A screenshot of a cell phone

Description automatically generated

**Fig. 8.** Plot maindata stressplot (screeplot) for visual representation. The graph suggests a 2-D solution based on the placement of the first ‘elbow’ shown in red. This value is just above the cut-off of 20 for being fair in terms of interpretive value (see Table 1). Note: there are trade-offs between number of axes and how useful they are to interpret! Users often stick with a 2-dimensional solution for simplicity even when borderline, and we will do that here.

**Table 1**. Example “Rules of thumb” for interpreting stress values from unconstrained NMS ordination. Table adapted from McCune and Grace (2002) Kruskal’s and Clarke’s rule of thumb recommendations. Ultimately it is up to the user to decide what stress level they are comfortable with and be able to justify it. The more dimensions a final solution has the more difficult it is to interpret. 3 dimensions tends to be the maximum we have observed in publications.

|  |  |
| --- | --- |
| **Stress (%)** | **Recommendation** |
| <5 | Excellent: Can be completely confident that this is a meaningful solution |
| 5 - 10 | Good: there is minimal chance of false inferences from this solution |
| 10 - ~20 | Fair: At the lower level of stress you can be fairly confident in inferences from this solution, but the closer you get to 20 the more potential there is for a solution with false inferences |
| > 20 | Poor: The further you get past 20% stress, the more the plot from this solution should not be relied upon and is likely to provide false inferences |

## 4.2 Running the Ordination

1. Remember that you will continue to input your code in your R script file. The ordination will be run by R using the **metaMDS** function within the **vegan** package. For vegan package to run, you first need to load **permute** and **lattice**. Then you will need to load the vegan package (which you previously installed above – remember that loading and installing are not the same thing). To load a package you type **library ( )** and the package name between the parentheses. Remember that you will have to load each package separately that you use.

**library (permute)**

**library (lattice)**

**library (MASS)**

**library (vegan)**

Note: Remember to always Select “Run” from the upper left box to run each line of code. Highlighting multiple lines at a time will run all of them at once.

1. Before running or plotting the ordination, it is a good idea to set the seed and scale so the NMS dimensions of the plot will not change. Ordination results are based on random iterations that minimize stress, therefore, if you do not set your seed (starting point), your graph may not be the exact same in appearance every time.

**set.seed ( )** function is a pseudo random number generator. By selecting a place to start, you will get the same results with the same seed number.

Example: **set.seed (100)**

You will also want to create a scaling factor that controls the stretching/shrinking of the plot. Name the variable, followed by an arrow **<-** and a numeric value to keep the axes equal. 3 is commonly used for scaling.

Example: **scl <- 3**

1. Running the ordination requires you to select a grouping variable from your environmental dataset (**seconddata**) that will be helpful for you to use to distinguish sample points when visualizing your ordination plot. In this case we are interested in comparing well pad and reference forest plant communities so we will use that as our grouping factor. So, create a grouping factor, **WellorRef** and use that as the grouping variable throughout. To call the variable, remember that you must place a **$** before the variable so RStudio knows you are selecting a column of a particular name from the second dataset file. The variable name must match exactly with the name in the data file (Note: remember that R is case sensitive). You will enter what you want to label this grouping factor (e.g., **WellorRef)** followed by an arrow and the argument **as.factor.** Next enter the dataset name and variable of interest in parentheses separated by **$**. In this example we are using the variable created from the second dataset, “**WellorRef**”, to visualize (and later test for) differences between the well and adjacent forested reference sites.

Example: **WellorRef <- as.factor (seconddata$WellorRef)**

In the Environment window you will now see a “Values” section below your two datasets: WellorRef: Factor with 2 levels “Ref” and “Well”

1. Now it is time to run the NMS ordination. Name the ordination output file (e.g. **ordmain)** followed by an arrow and the **metaMDS** function (this is the code that will run the ordination in RStudio). Enter the name of the main dataset (**maindata**), followed by the distance measure (for ecological data, the Bray-Curtis (R calls this **bray**) distance measure is most commonly used), trace (select = False), trymax (The trymax value represents the maximum number of random starts R will perform before coming to a final solution (we recommend 100)), and k (the number of axes/dimensions for your solution based on the number of dimensions from your scree plot – for our example dataset this is k=2). If you do not specify k, R will provide the default of a 2-dimensional (2 axes) solution**.** The entire line of code should look like this:

Example:**ordmain <- metaMDS (maindata, distance = "bray", trace= FALSE, trymax= 100, k=2)**

Reminder that R does not recognize “Smart” quotes from MS Word (“text”) – instead the quotes need to be straight (").

You could also try a larger dimensional solution by changing the k value. For example, you could run a 3-dimensional solution (e.g., ordmain3 with k=3) or a 4-dimensional solution (e.g., ordmain4 with k=4). For our example here we will stick with the two dimensional solution (**ordmain**).

1. Now that you have run the NMS ordination you want to see the result output from code you have ran. Add a line to your code with the name of the object (e.g., **ordmain**) and select “Run” the line. Values will appear in the output console (Fig. 9). You will want to copy and paste these important values into a word document, as RStudio will clear the console when you close the program. This output will give you the dimensions and stress values you need to report in your results section. Note that R does some transformations to your data – as demonstrated by the scaling line below: centring, PC rotation, halfchange scaling (it is beyond the scope of this assignment to understand the process of transforming/standardizing the data).

Text, letter

Description automatically generated

**Fig. 9.** Viewing NMS result output from maindata. This ordination was run using the Bray-Curtis distance measure on data that were first Wisconsin transformed (which you don’t have to understand the details of for this assignment). Results in red box suggest a 2-D graph with stress value of 20.9% after 20 tries. Note that this is right around the rule of thumb (Table 1) for poor results.

## 4.3 Ordination Graph Plotting Options

In NMS ordination analysis, the closer the points are together on the graph, the more similar they are, while the farther apart they are, the more dissimilar they are from one another. Your ordination plot will help you to evaluate these differences/similarities among your study sites.

1. The ordination plot output will appear in the bottom right window under the plot tab. You can adjust the size of the plot by selecting the window border and making it larger or taller (initially when the window is not expanded your plot will look squished! When displaying the data, we want axes to be equal, therefore specify the scaling parameters **“scl”** in the plot that was set earlier.Every time you want a clear graph, enter:

Example**: plot (ordmain, type = "n", scaling = scl)** [see Fig. 10].

For better control of ordination graphics you can first draw an empty plot (type = "n") and then add species and sites separately using point or text functions. (hint: anytime you want to clear your graph and get back to a blank graph use the code **type="n")** Based on your version of R, you may get red output saying **“scaling is not a graphical parameter.”** This will not affect your results, ignore and continue with analyses - you will continue to get this error message when graphing ordination.

1. You can plot the sites Well or Ref using the study site (row) labels to see where the individual sites lie on the plot. This code is telling R to use the individual well pad site labels to identify points on the graph by their specific Site ID (e.g., Bor4\_Well) in place of individual symbols in the ordination plot (Hint: this plot is very busy and would not be a recommended type of format to submit in a presentation or assignment).

Example: **text (ordmain, display = "sites", cex=0.8, col= (1:2)[WellorRef])**

Specifying cex allows you to adjust the size of graph points or text and you can see that in this example we are assigning the colors using the grouping factor (Well vs Ref) that we created above (Fig. 11). Note that in this code we have opted to use colors 1 and 2 (you can google to find out other colors or try out some different values; each number represents a color – e.g., col=1 is black, col=2 is red and col=3 is green.) In this example we have used a “:” to delineate our two colors – you could alternatively use the following code with commas instead **col=c(1,2)** and it would do the same thing. When you have features that are not directly adjacent – e.g, if you wanted to pick colors black and green then you would use the following **col=c(1,3)** approach in place, as if you selected col=(1:3) then it would be using the colors 1, 2, and 3 and because there are only two levels of the grouping factor in this example (well or ref) it would only use the black and red.

A screenshot of a cell phone

Description automatically generated

**Fig. 10.** Blank ordination plot with equal axes**.**

A screenshot of a cell phone

Description automatically generated

**Fig. 11.** Text labels identifying points on the graph by their specific Site ID. Reference sites shown in black and wellsites shown in red (as indicated by the col 1:2 whereby col=color and color 1 is black, color 2 is red.

Reminder: To help keep your code organized, you can make notes for yourself in the script box by putting a hash tag **#** in front of script (e.g. #site labels). Lines of code with **#** in front will not be run by RStudio (Fig. 12).

A screenshot of a cell phone

Description automatically generated

**Fig. 12.** Example line of code taken from the Rscript window. The red box outlines the line of code which will not run because it starts with a hash tag. The green box outlines text within quotation marks which needs to be highlighted in green for the program to recognize. The blue boxes outline number values that need to be highlighted in blue for cex (size of the symbols) and col (colors of the symbols) to be recognized.

1. To identify sites by colour-coding and distinguishing using shapes of datapoints to coincide with WellorRef levels, first clear the graph.

Example: **plot (ordmain, type = "n", scaling = scl)**

Next, plot the desired variable within square brackets, in this case WellorRef, with the shape (pch which is short for point character) and color (col) outlines specified within parentheses.

Example: **with(maindata, points(ordmain, display = "sites", scaling = scl, pch= c(3,4)[WellorRef], col = (1:2) [WellorRef]))** [Fig. 13]

[in this case the well pads are red x’s and the reference sites are black + signs.

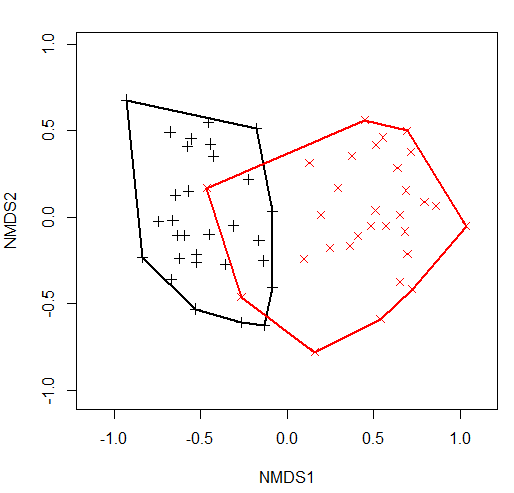
For the point character pch we have delineated the types of characters we are using (3= plus symbol, 4= cross) using **pch=c(3,4)**. We could also have substituted with the following code to get the same results: **pch=(3:4)**. Try using different symbols (e.g., pch 1 and 2).

1. You can then add polygon borders to distinguish and enclose all the sites that belong to each group within a grouping factor.

Example: **ordihull(ordmain, WellorRef, col=1:2, scaling = "symmetric", lwd = 2, show.groups = WellorRef)**

Altering **lwd** changes the thickness of the polygon lines (Fig. 13).

1. In this example we can see a distinction between reference sites and wellsites but also some overlap among them in the middle of the ordination plot (Fig. 12). (Recall: In an ordination analysis, the closer the points are together on the graph, the more similar they are to one another; the farther apart they are, the more dissimilar they are). Note that in this case we have not added a legend that tells the reader what the two types of colored points represent. You will learn how to do that below.



**Fig. 13.** Non-metric Multidimensional Scaling (NMS) ordination plot using polygon hulls around each of the two grouping factor levels. In this example we can see separation between the majority of references sites and wellsites indicated by the difference in color and shape of points on the graph; +=reference, x=wellsite).

1. The **ordiellipse** command shows confidence intervals displayed as an ellipse. If groups overlap, they are likely not significantly different from one another. However, groups still need to be tested for significant differences in their plant community composition using a perMANOVA to confirm this (you will do this below). You can overlay ordiellipse (confidence intervals) with ordihull (outline of all sites within a group). You can leave ordihull and ordiellipse as outlines or you can specify to add fill (draw =”polygon”). To alter color and change shape, enter alternative values for **col=** and **pch=** arguments respectively. First you should re-clear the plot.

Example: **plot (ordmain, type = "n", scaling = scl)**

**#in this example below we have changed the colors and drawn the confidence interval ellipse**

**with(maindata, points(ordmain, display = "sites", scaling = scl, pch=c(1,2) [WellorRef], col = (4:5) [WellorRef]))**

**ordiellipse(ordmain, WellorRef, col=4:5, scaling = "symmetric", show.groups = WellorRef, draw="polygon")**

**ordihull(ordmain, WellorRef, col=4:5, scaling = "symmetric", show.groups = WellorRef)** [Fig. 14].

1. To add a legend, use the **with ( )** function and specify the location of the legend, the grouping factor, the legend type (bty), colors (col), shapes (pch), and the background color for legend points (pt.bg).

Example: **with (ordmain, legend ("topright", legend=levels (WellorRef), bty="n", col = (4:5), pch = c(1,2), pt.bg = (4:5) [WellorRef]))** [Fig. 14].

A picture containing map, text

Description automatically generated

**Fig. 14.** Ordiellipse (filled) command shows confidence intervals of the groups based on the variable **WellorRef**. The ordihull (outline) command encases all sites within each group. The confidence intervals do not overlap, suggesting significant differences between the reference sites and wellsites. A perMANOVA test for differences is required.

## 4.4 Vectors (Environmental Variables)

If you want to overlay the quantitative environmental variables from your second dataset on your ordination plot you can then visualize which variables are significant and in which direction they are correlated with the ordination plot as indicated with an arrow.

1. Start bycreating a new environmental factors dataset (this will contain the variables that you want to overlay onto your plot) followed by an arrow (e.g. **ef <-** ). Next specify your environmental variable dataset (e.g., seconddata) and the quantitative (numeric) variables you are running the analysis on. You can open the dataset and hover your cursor over the headings to determine information about the parameters (e.g., type of data and range of values for numeric data). In our example the quantitative/numeric variables are found from columns 4 (Age\_postcert) through 20 (non\_native\_cover).

Example: **ef<-seconddata[4:20]**

1. Next, use the **score** function to specify you are interested in accessing the site scores from the ordination.

Example: **score=ordmain$points**

1. Use the **vec** function to stack vectors and matrices. The **envfit** function fits the environmental factors onto the ordination. You can specify the arguments within the function to set the projection of points to have maximum p-values (**p.max**)

Example: **vec=envfit(score, ef, nperm=999, na.rm = TRUE)**

You can highlight **vec** to view the environmental vector output and their significance (Fig. 15)

A screenshot of a cell phone

Description automatically generated

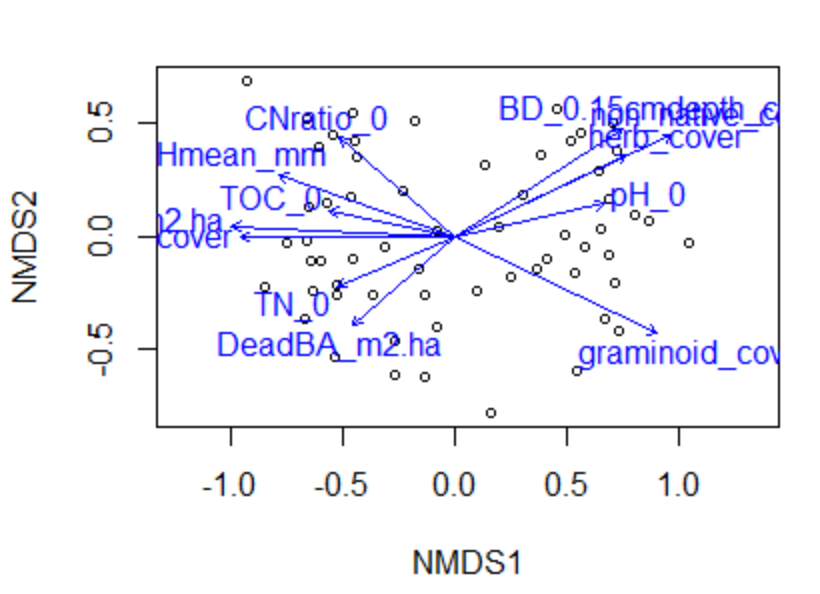
**Fig. 15.** Vector output. Red box outlines the p-values for all quantitative variables.

1. Next plot the vectors on the ordination graph. The output will include significant codes at the threshold you specify. You can adjust the threshold value of variables included in your graph by adjusting the **p.max** value. First set up the plot.

Example: **plot(ordmain, dis="site", scaling=scl)**

Then overlay the environmental variables with choice of threshold value on top of this graph.

Example: **plot(vec, p.max=0.001)** [Fig. 16].



**Fig. 16.** Vectors represented in blue represent the strength and direction with respect to arrows. Threshold for environmental variables on ordination plot = 0.001.

## 4.5 Preparing to Present

Now that you have learned how to make a graph with vectors – you may find that the vector names can be rather messy/busy. Here is some background information on how to update column names to make the visuals more friendly to readers.

1. You can change the names of the columns of environmental variables to be more intuitive for the reader. You can change the names in the excel csv file (hint: you could do this before running any analyses) and re-download the dataset or use code. To show output of all the column names within a dataset, enter:

**colnames (seconddata)**

Alternatively, working from within RStudio, you can double click on the dataset to open and view column names. To rename significant vectors with untidy column names, enter the current name and updated name as follows – here we are only updating Bulk Density column name:

**names (seconddata) [names (seconddata) == "BD\_0.15cmdepth\_cm****"] <- "Bulk\_Density"**

In both sets of parentheses enter the environmental dataset you are adjusting (e.g. seconddata). In the first set of quotations, inside the square brackets, enter the name of the column you are changing exactly as the name is displayed in the second dataset. After the square brackets enter an arrow followed by the new, more meaningful name within quotations. To determine column abbreviations, see the metadata that accompanies your dataset. To ensure column name changes were successful, re-enter:

**colnames (seconddata)**

1. After column name changes you will have to re-enter the parameters and the code for environmental variables:

**ef <- seconddata [4:20]**

**vec = envfit (score, ef, nperm=999, na.rm=TRUE)**

Then, re-plot the ordination:

**plot(ordmain, type= "n", scaling = scl)**

**points(ordmain, disp="sites", col = (1:2) [WellorRef], pch = c(1,2)[WellorRef])**

**ordihull(ordmain, WellorRef, col=1:2, scaling = "symmetric", lwd= 2, show.groups = WellorRef)**

**with (ordmain, legend ("topright", legend=levels (WellorRef), bty="n", col = (1:2), pch = c(1,2), pt.bg = (1:2) [WellorRef]))**

Next overlay the environmental variable threshold:

**plot (vec, p.max = 0.001)** (Fig. 17).

A close up of a map

Description automatically generatedA close up of a map

Description automatically generated

**Fig. 17.** Example on left shows ordination distinguishing sites with significant vectors in blue at threshold 0.001. Example on right shows ordination with ordihull command and the column name change of the environmental variable in the dataset, **seconddata**. Name change for column “BD\_0.15cmdepth\_cm” to “Bulk\_Density.” Vectors shown in blue with a threshold of 0.001. There are still lots of ways you could improve the quality and readability of this graph!

1. You may want to clean up your legend. It is good practice to keep your point colors in grey scale, since most publishers require this format. But if you are going to be presenting a poster, color is preferred to help draw in your reader! To make your ordination in grayscale, create a factor name, followed by the c function and specified color parameters.

Example: **cols <- c("gray10", "gray70")**

You can now re-plot your ordination and legend using the variable you created (**cols**) as the input for **col =**

If including vectors you can also specify a color (e.g. col =”black”, or col = “gray45”, etc).

**plot (ordmain, type = "n", scaling = scl)**

**with(maindata, points(ordmain, display = "sites",scaling = scl, pch=c(1,2)[WellorRef], col = cols[WellorRef]))**

**ordihull(ordmain, WellorRef, col=cols, scaling = "symmetric", lwd = 2, show.groups = WellorRef)**

**with (ordmain, legend ("topright", legend=levels (WellorRef), bty="n", col = cols [WellorRef], pch = c(1,2), pt.bg = cols [WellorRef]))**

**plot(vec, p.max=0.001, col = " black")** (Fig. 18).

A close up of a map

Description automatically generated

**Fig. 18.** Example ordination in grayscale using the code above (exported to MS Word).

1. When you are satisfied with your plot and ready to save and want a high quality image, rather than just right clicking on the image in the Plots pane, select **Export** in the plot window (bottom right screen). If you want to further edit the plot, for instance, to move vector names that are partially visible on the plot, make sure to select **Save as PDF** and you will be able to make changes in a PDF editor (e.g. Adobe). *This is a limitation in the functionality of graphing in R*.

# 5. Permutational Multivariate Analysis of Variance (perMANOVA):

Proceed with a permutation-based multivariate analysis of variance (perMANOVA). The **adonis2** function in R is analogous to perMANOVA. This analysis will be run using your second dataset. In our example, for the sake of illustrating how to run a perMANOVA test on one variable, we will assume that the variable **WellorRef** is the only factor of interest. However, as you will see in seconddataset.csv there is an additional grouping variable (Bor1Foot2) – you could also try comparing differences among these two natural regions (Boreal and Foothills Natural Regions) after running the perMANOVA below to get more practice running perMANOVAs.

## 5.1 Running perMANOVAs

A perMANOVA will test for significant differences in the plant community composition between the well pad plant communities and the reference forest plant communities (it is the same idea as a t-test when comparing two univariate variables). The perMANOVA requires you to select a grouping variable to compare community composition. In this example we use “WellorRef” to test for differences in plant community composition between wellsites and reference sites.

1. Use the **adonis** function to perform multivariate analysis of variance. To call a variable place **$** before the variable. The variable name must match exactly with the name in the data file because R is case sensitive.

Example: **adonis2(maindata ~ seconddata$WellorRef, distance="bray")** [Fig. 19].

Text

Description automatically generated

**Fig. 19.** Sample RStudio Results from perMANOVA using “**WellorRef**” as a variable. Note that individual responses may vary slightly. So your results may not be identical to those in this example, but the P-value should be very similar. Output suggests there was a significant difference with **p = 0.001 (In this case R is presenting the P-value as Pr(>F))**

*If you are analyzing more than two levels you will need to then perform post-hoc pairwise comparisons because the perMANOVA test only tells you there is a significant difference among at least two of the sites but it does not tell you which group(s) are different. With ecological data, it is preferred to use the Bonferroni correction for p-values (or another correction factor of your choice (e.g., Holm). However, this dataset only has two levels so proceed without a pairwise comparison because you now know that there is a difference between the wellsite and reference sites. To see a more complex example of a perMANOVA with two factors and post-hoc tests, we recommend visiting Andreas Hamann’s class dataset here: https://sites.ualberta.ca/~ahamann/teaching/renr690/index.html and completing the Unit 6 section on inferential statistics.*

# 6. Indicator Species Analysis (ISA):

## 6.1 Identifying Species by Sites

Indicator species analysis is used to identify species that are associated with each level of your grouping factor. This is only done when your perMANOVA has indicated that there are significant differences among the levels of your grouping factor. In our case we did see significant differences in plant community composition between well pads and reference forest. Before you can run ISA, you will have to load the program **indicspecies** (e.g. library (indicspecies). You will need to create a new dataset name followed by an arrow and then the function **multiplatt** (fyi this is short for multi-level pattern analysis).

**library (indicspecies)**

When using **multipatt** you can avoid site group combinations by using duleg = TRUE. In this example we have specified the number of permutations to be 999.

**Indval=multipatt(maindata, seconddata$WellorRef, duleg = TRUE, control = how(nperm=999))**

To view the output, enter summary and specify the parameters. Using Indvalcomp = TRUE is equivalent to alpha = 0.05.

Example: **summary(Indval, indvalcomp=TRUE)** [Fig. 20].

The multiplatt function is used to determine lists of species associated to groups (e.g. WellorRef).Alternatively, you can view all the species, both significant and not, by running the code:

**Indval$sign**

Or you can manually enter a threshold alpha value.

Example: **summary (Indval, alpha = 0.001)**

Output will appear for significant indicator species with specified threshold and the group which they coincide with (Fig. 21).

# 7. Summary Statistics:

## 7.1 Reporting Cover Values

When you identify indicator species, it is of interest to know more about these particular species. For example, it is likely of interest to the reader to report the mean cover values of each species for their respective groups and standard deviations.

1. To obtain descriptive statistics, first load the package **psych**

**library (psych)**

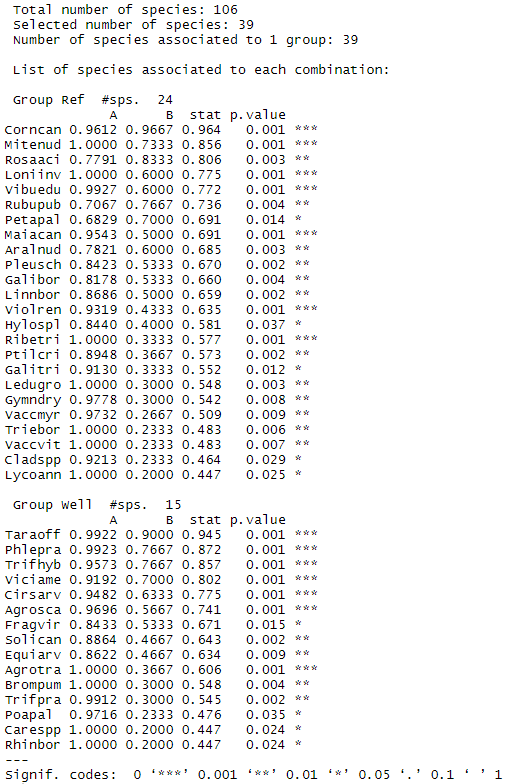
1. Next, because there is a long list of species you will need to override the max output setting.

**options (max.print = 10000)**

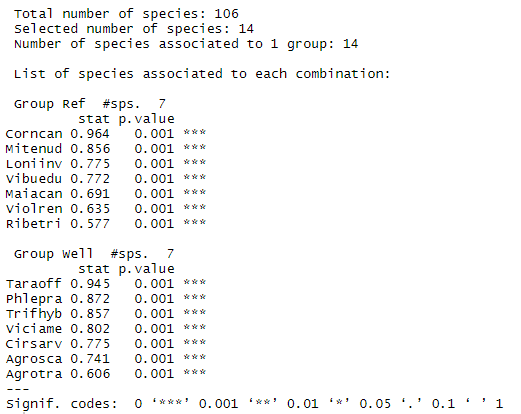
1. Now you can specify the grouping variable and community dataset from which you want the summary output. Example:

**describeBy(maindata, group= WellorRef)**

Specifying WellorRef for the group will show output for each species for both sites (Fig. 22).



**Fig. 20.** Significant indicator species partitioned by site “Well” and “Ref.” Group coincides with the species code see Appendix 10.1). Output suggests that out of 106 identified species, only 39 were indicator species (24 for reference sites and 15 for wellsites). Based on the A and B values, the species Corncan is a strong indicator species for reference sites because it is almost solely found on these sites A = 96% (The A value of 1 = 100%) and it is found across most of reference sites B = 96% (The B value of 1 = 100%).



**Fig. 21.** Indicator species with significance at threshold 0.001.

A screenshot of a cell phone

Description automatically generated

**Fig. 22.** Example summary output for reference sites. Summary output of importance outlined in red box.

# 8. Reporting Methods and Results:

## 8.1 Reporting Methods – Including Demonstration

Here we are demonstrating how you could record your statistical analyses that were conducted in R in a demonstration Methods section.

Refer back to your saved results files for this information (the values you copied and pasted from the R output window). This section should include the following:

* Software used
* Distance measure used
* Number of runs with random start (trymax)
* Number of dimensions of the final test selected by R (e.g. 2D)
* What additional statistical tests were completed (e.g., perMANOVA, and/or ISA methods)

NMS Ordination Example:

“To visualize the understory plant community composition of the well pad and reference forests, a nonmetric multidimensional scaling (NMS) unconstrained ordination was carried out using the ‘metaMDS’ function in R ‘vegan’ package (R-4.0.2) with Bray-Curtis as the distance measure. The number of dimensions for the final solution was determined by evaluating the stress plot and ordination output starting with a six-dimensional solution and stepping down to a one-dimensional solution (using the ‘dimcheckMDS’ function in R ‘goeveg’ package (R-4.0.2)).”

perMANOVA Example:

“A permutation-based perMANOVA using the ‘adonis’ function in R ‘vegan’ package (R-4.0.2) was used to determine if there were statistically significant (alpha=0.05) differences among the well pads and the reference sites understory plant communities. For the perMANOVA, 999 randomizations (permutations) were used, and significance was based on the proportion of randomized trials with a response value greater than or equal to the observed response value.”

ISA methods example:

“The Indicator Species Analysis (ISA; Dufrêne and Legendre 1997) was performed using the ‘multipatt’ function in R ‘indicspecies’ package to identify which understory plant species were contributing to the separation of the plant communities among sites. We used a quantitative or binary response with a randomization test (number of permutations =999). ISA was conducted using R (R-4.0.2).”

## 8.2 Reporting Results

Make sure it is clear what alpha you are using for threshold for determining significance of any of your analytical results that have a p-value associated with them.

NMS Ordination section should include the following:

* Number of iterations for the final solution (final stress)
* Ordination plot and a description of the patterns observed in the ordination (make sure to include legend and labels for your individual sample points)
* Standard interpretive aids visualized on the ordination plot (vectors) – if there are a lot of vectors then an option is to include two ordination plots – one with vectors and one without

perMANOVA section should include:

* no. of permutations
* Report the p-value and state whether there is a significant difference.

Indicator Species Analysis section should include:

* Number of indicator species for each group (e.g., Well or Ref.)
* Compiled table with only the relevant species (including species code, name, mean cover for each group, and your A and B values) Note: A values give insight into how exclusive a species is within a particular treatment, with an A value of 1 meaning a species is solely found in one treatment type. B values indicate the consistency that a species is found across all sample sites of a particular treatment, with a B value of 1 meaning the species was found in all sites within a treatment. We recommend that you organize your species in descending order of their A-value within each group.

# 9. References:

Anderson MJ. 2017. Permutational Multivariate Analysis of Variance (PERMANOVA). Wiley StatsRef Stat Ref Online.:1–15. doi:10.1002/9781118445112.stat07841.

Dufrene, M. and Legendre, P. (1997) Species Assemblages and Indicator Species: The Need for a Flexible Asymmetrical Approach. Ecological Monographs, 67, 345-366.  
http://dx.doi.org/10.2307/2963459

McCune B, Grace JB.  2002.  Analysis of Ecological Communities.  MjM Software, Gleneden Beach, Oregon.  304 pages.

## 9.1 Additional Online Resources You May Find Useful

This list is not at all exhaustive. Note that there are many useful R discussion boards and resources you can find online.

RStudio (use the Resources drop-down for many options):

<https://www.rstudio.com/>

Additional info on running NMS:

<https://jonlefcheck.net/2012/10/24/nmds-tutorial-in-r/>

Information on customizing ordination plots in vegan:

<https://fromthebottomoftheheap.net/2012/04/11/customising-vegans-ordination-plots/>

Information on different ways to color points on graphs:

<https://bookdown.org/rdpeng/exdata/plotting-and-color-in-r.html>

Calculating summary statistics in R:

<https://www.statmethods.net/stats/descriptives.html>

Conducting perMANOVA post-hoc tests and other multivariate tools:

<https://sites.ualberta.ca/~ahamann/teaching/renr690/index.html>

Additional resources that Anne McIntosh has used when teaching this in her class:

* [https://iqss.github.io/dss-workshops/R/Rintro/base-r-cheat-sheet.pdf](https://www.dataquest.io/blog/rstudio-tips-tricks-shortcuts/)
* [https://github.com/StevisonLab/R-Mini-Course](https://www.dataquest.io/blog/rstudio-tips-tricks-shortcuts/)
* [https://support.rstudio.com/hc/en-us/articles/201141096-Getting-Started-with-R](https://www.dataquest.io/blog/rstudio-tips-tricks-shortcuts/)
* [http://adv-r.had.co.nz/Style.html](https://www.dataquest.io/blog/rstudio-tips-tricks-shortcuts/)
* <https://www.dataquest.io/blog/rstudio-tips-tricks-shortcuts/>
* <https://bookdown.org/ndphillips/YaRrr/>

# 10. Appendix

## 10.1 Species codes

**Table 1.** Species list of scientific names and common names for the main dataset.

| Species Code | Genus | Species | Common Name |
| --- | --- | --- | --- |
| Abiebal | *Abies* | *balsamea* | Balsam Fir |
| Achimill | *Achillea* | *millefolium* | Common Yarrow |
| Agrosca | *Agropyron* | *scabra* | Tickle Grass |
| Agrotra | *Agropyron* | *trachycaulum* | Slender Wheatgrass |
| Alnucri | *Alnus* | *crispa* | Green Alder |
| Alnurug | *Alnus* | *rugosa* |  |
| Amelaln | *Amelanchier* | *alnifolia* | Saskatoon |
| Apocand | *Apocynum* | *androsaemifolium* | Spreading Dogbane |
| Aquican | *Aquilegia* | *canadensis* | Canada Columbine |
| Aralnud | *Aralia* | *nudicaulis* | Wild Sarsparilla |
| Arnicor | *Arnica* | *cordifolia* | Heart-Leafed Arnica |
| Astecil | *Aster* | *ciliolatus* | Lindley's Aster |
| Astecon | *Aster* | *conspicuus* | Showy Aster |
| Betupap | *Betula* | *papyrifera* | Paper Birch |
| Botrvir | *Botrychium* | *virginianum* | Virginia Grape Fern |
| Bracsal | *Brachythecium* | *salebrosum* | Golden Ragged Moss |
| Bromcil | *Bromus* | *ciliatus* | Fringed Brome |
| Bromine | *Bromus* | *inermis* | Smooth Brome |
| Brompum | *Bromus* | *pumpellianus* | Pumpelly brome |
| Calacan | *Calamagrostis* | *canadensis* | Bluejoint |
| Carespp | *Carex* | *Spp* | Upland Carex Spp |
| Castmin | *Castilleja* | *miniata* | Red Indian Paintbrush |
| Chamang | *Chamerion* | *angustifolium* | Fireweed |
| Circalp | *Circaea* | *alpina* | Small Enchanter's-Nightshade |
| Cirsarv | *Cirsium* | *arvense* | Canada Thistle |
| Cladspp | *Cladonia* | *spp* |  |
| Corncan | *Cornus* | *canadensis* | Bunchberry |
| Cornsto | *Cornus* | *stolonifera* | Dogwood |
| Desccae | *Deschampsia* | *caespitosa* | Tufted Hairgrass |
| Dicrsco | *Dicranum* | *scoparium* | Broom Moss |
| Dryoaus | *Dryopteris* | *austriaca* | Spinulose Shield Fern |
| Elymspp | *Elymus* |  |  |
| Equiarv | *Equisetum* | *arvense* | Common Horsetail |
| Equipra | *Equisetum* | *pratense* | Meadow Horsetail |
| Equisyl | *Equisetum* | *sylvaticum* | Woodland Horsetail |
| Eurhpul | *Eurhynchium* | *pulchellum* | Common Beaked Moss |
| Evermes | *Evernia* | *mesomorpha* | Spruce Moss |
| Fragves | *Fragaria* | *vesca* | Woodland Strawberry |
| Fragvir | *Fragaria* | *virginiana* | Wild Strawberry |
| Galetet | *Galeopsis* | *tetrahit* | Hemp-Nettle |
| Galibor | *Galium* | *boreale* | Northern Bedstraw |
| Galitri | *Galium* | *triflorum* | Sweet-scented Bedstraw |
| Geumale | *Geum* | *aleppicum* | Yellow Avens |
| Gymndry | *Gymnocarpium* | *dryopteris* | Common Oak Fern |
| Habehyp | *Habenaria* | *hyperborea* | Northern Green Orchid |
| Haledef | *Halenia* | *deflexa* | Spurred Gentian |
| Heralan | *Heracleum* | *lanatum* | Cow-Parsnip |
| Hierumb | *Hieracium* | *umbellatum* | Narrow-leaved Hawkweed |
| Hylospl | *Hylocomium* | *splendens* | Stairstep Moss |
| Impacap | *Impatiens* | *capensis* | Spotted Touch-Me-Not |
| Ledugro | *Ledum* | *groenlandicum* | Labrador Tea |
| Leyminn | *Leymus* | *innovatus* | Hairy Wild Rye Grass |
| Linnbor | *Linnaeus* | *borealis* | Twinflower |
| Loniinv | *Lonicera* | *involucrata* | Bracted Honeysuckle |
| Lycoann | *Lycopodium* | *annotinum* | Stiff Club-moss |
| Maiacan | *Maianthemum* | *canadense* | Wild Lily-of-the-valley |
| Melioff | *Melilotus* | *officinalis* | Yellow Sweet Clover |
| Mertpan | *Mertensia* | *paniculata* | Tall Lungwort |
| Mitenud | *Mitella* | *nuda* | Bishop's Cap |
| Orthsec | *Orthilia* | *secunda* | One-sided Wintergreen |
| Oryzasp | *Oryzopsis* | *asperifolia* | Rough-Leaved Ricegrass |
| Peltcan | *Peltigera* | *canina* | Dog Lichen |
| Petapal | *Petasites* | *palmatus* | Palmate-Leaved Coltsfoot |
| Petasag | *Petasites* | *sagitatus* | Arrow Leaved Coltsfoot |
| Phalaru | *Phalaris* | *arundinacea* | Reed Canary Grass |
| Phlepra | *Phleum* | *pratense* | Timothy |
| Picegla | *Picea* | *glauca* | White Spruce |
| Picemar | *Picea* | *mariana* | Black Spruce |
| Plagcus | *Plagiomnium* | *cuspidatum* | Woodsy Leafy Moss |
| Platyrep | *Platygyrium* | *repens* | common flat-brocade moss |
| Pleusch | *Pleurozium* | *schreberei* | Big Red Stem |
| Poapal | *Poa* | *palustris* | Fowl Bluegrass |
| Pohlnut | *Pohlia* | *nutans* | Copper Wire Moss |
| Polyjun | *Polytrichum* | *juniperinum* | Juniper Hair-Cap |
| Popubal | *Populus* | *balsamifera* | Balsam Poplar |
| Poputre | *Populus* | *tremuloides* | Trembling Aspen |
| Ptilcri | *Ptilium* | *crista-castrensis* | Knight's Plume |
| Pyroasa | *Pyrola* | *asarifolia* | Common Pink wintergreen |
| Rhinbor | *Rhinanthus* | *borealis* | Yellow Rattle |
| Ribelac | *Ribes* | *lacustre* | Black Gooseberry |
| Ribeoxy | *Ribes* | *oxycanthoides* | Canadian gooseberry |
| Ribetri | *Ribes* | *triste* | Wild Red Currant |
| Rosaaci | *Rosa* | *acicularis* | Prickly Rose |
| Rubucha | *Rubus* | *chamaemorus* | Cloudberry |
| Rubuida | *Rubus* | *idaeus* | Raspberry |
| Rubupub | *Rubus* | *pubescens* | Dewberry |
| salix | *Salix* | *Spp* |  |
| Scirmic | *Scirpus* | *microcarpus* | Small-Fruited Bulrush |
| Shepcan | *Shepherdia* | *canadensis* | Canada buffaloberry |
| Smileste | *Smilacina* | *stellata* | false Solomon's seal |
| Solican | *Solidago* | *canadensis* | Canada Goldenrod |
| Soncarv | *Sonchus* | *arvensis* | Perennial Sow-thistle |
| Soncasp | *Sonchus* | *asper* | Spiny Annual Sow-thistle |
| Stelspp | *Stellaria* | *Spp* | Chickweed Spp |
| Sympalb | *Symphoricarpos* | *albus* | Common Snowberry |
| Sympcil | *Symphyotrichum* | *ciliolatum* |  |
| Sympocc | *Symphoricarpos* | *occidentalis* | Buckbrush |
| Taraoff | *Taraxacum* | *officinale* | Common Dandelion |
| Thuiabi | *Thuidium* | *abietinum* | Wiry Fern Moss |
| Triebor | *Trientalis* | *borealis* | Northern Starflower |
| Trifhyb | *Trifolium* | *hybridum* | Alsike Clover |
| Trifpra | *Trifolium* | *pratense* | Red Clover |
| Urtidio | *Urtica* | *dioica* | Stinging Nettle |
| Vacccae | *Vaccinium* | *caespitosum* | Dwarf Bilberry |
| Vaccmyr | *Vaccinium* | *myrtilloides* | Common Blueberry |
| Vaccvit | *Vaccinium* | *vitis-idaea* | Lingonberry |
| Vibuedu | *Viburnum* | *edule* | Mooseberry |
| Viciame | *Vicia* | *americana* | American Vetch |
| Violcan | *Viola* | *canadensis* | Canadian white violet |
| Violren | *Viola* | *renifolia* | Kidneyleaf Violet |