Preliminaries

If you haven't already downloaded and installed a copy of R and RStudio please do so now. Both programs are free to use. We will be using RStudio, but as this is simple a user-friendly 'wrap-around' for R, you should download both programs.

Download R for windows
https://cran.r-project.org/bin/windows/base/

Download R for mac
https://cran.r-project.org/bin/macosx/

Download RStudio (just select the free version)
- Then scroll down to select your O/S from the list at the bottom of the page
https://rstudio.com/products/rstudio/download/#download

Changing working directory

Immediately on opening R, the first thing you need to do is change your working directory to the folder where your data is currently kept. In the case of this PDF, you will need to unzip the Borneo veg data files and then change your working directory so that it is pointed at the file where the Borneo veg files have unzipped.

Changing working directory in RStudio

Changing the working directory in RStudio is the same for Windows and Mac.
Both: Select 'Session'   > Set working directory > Choose Directory...
> *navigate to folder where your data is stored (i.e. your csv files)*

Colour Coding

R code in this PDF is colour coded throughout. Functions (commands) are coloured **`orange`**. Names of objects or options (i.e. things that you will need to change or rename) are coloured **`blue`**. Basic syntactical signs (parenthesis, commas, plus or multiplication signs) are colour **`black`**. Package names are coloured **`purple`**. Note, which follow a hashtag (**#**) are not read by R, and serve as a place where you can make notes about your tests, code, and figures. These are coloured **`green`**.

In the following example, the orange words are functions that R will recognise as commands. Functions are typically placed to the immediate left of an opening parenthesis. The library '**`car`**' is denoted in purple, and is bracketed by black quote marks and parentheses. The object **`swallows.lm`** is an object that the biologist as created and named themselves.

```
install.pckages("car") # if not already installed
library(car)
crPlots(swallows.lm)
```

Species Accumulation Curves

The concept behind a species accumulation curve (SAC) is actually quite straightforward. Imagine you were hired to go into a forest and count all the species of frogs. Each day you go out, looking for more frog species. To start with you would get one, two or three new species each day, but over time you'd find fewer and fewer new species. Eventually you'd be looking for a few days, or a week or a month and not find any new species. At this point, we can probably assume you've found all or most of the frog species in the forest. This doesn't necessarily mean you've found all the species. Some might be cryptic and hard to see. Some might be inactive at the time of year you are looking. However, broadly speaking we can build a curve from the number of new species you find each day, and check where it levels out to get an estimate of how many species are probably present.

Assumptions & Data Set-up

Species accumulation curves do not have any testable assumptions per se, but they can provide meaningless or misleading results if the experimental design is faulty. The most common reason for faulty design is sampling across a clear, heterogeneous boundary. What might cause such a boundary? Imagine you are identifying frog species, and you are collecting data over several weeks. If it was dry for most of the time, and then it rains for three days, then returns to dry weather, you could easily find that there is a clear difference between frogs that you identified during the dry and wet weather. The same could happen if you cross a geographic boundary. If you are running transects through forest, then cross into a wetland, the species assortment would probably suddenly jump.

Mostly, avoiding problems of heterogeneity is a matter of good design and thinking carefully about field sites. When you come to plotting the data, a heterogeneous boundary will be obvious as a 'jump' in the data. If the curve is smooth, and then jumps or kinks upwards you may have a heterogeneity problem. How to cope with this is trickier. To some degree, methods like the random bootstrapping approach smooth out these kinks and assume the whole environment is homogeneous, and is some instances this may actually be acceptable. Otherwise, it may be necessary to split the data (i.e. wet and dry days, forest and wetland transects).

Gradual changes across a sampling range are less problematic than sudden jumps, although keep in mind that by applying a species accumulation curve you are assuming that the environment does represent a coherent whole--problems can still occur. It may be possible to run a transects 3km up a mountainside and see only gradual changes in species assortment, but it would still be questionable whether or not such a dataset is truly representing a 'homogenous' environment. The clearest indicator that you are running collection through several different and gradually changing environments is that you will not see a levelling out of the curve. It will just continue to increase. Imagine if you ran a sample all the way across a continent. It's quite possible the curve would never properly level out, because you are always encountering new species.

Measuring Effort

Species accumulation curves require some sort of index of effort. This could be quadrats along one or more transect lines, or days spent looking for frogs, or number of new species per individual counted (i.e. new frog species per frog), or area surveyed. In the example we'll look at, the data was collected in Borneo, examining morphotypes of understory plants. The data was collected in six transects through the rainforest with nine quadrats per transect. The biologists were interesting in identifying whether there might be a change in species richness near the tourist boardwalks in Mulu Forest Park. To this end, they positioned Quadrat 1 (1x1m) immediately next to the boardwalk, and then ran a transect at a right angle into the forest, so that Quadrat 9 was always the furthest quadrat and always the same distance into the forest.

We need to import quite a few datasets. The first dataset records presence and absence of a species in a given quadrat.

```
borneo <- read.table('borneo_binary.csv',header=T, sep = ',')
View(borneo)
```

As is, this is not easily usable for species accumulation curves, because the quadrat and transect columns will require extra coding to get around. We could use code to remove these columns, but instead I've just set up a separate csv file that is missing the first two columns.

```
borneo <- read.table('borneo_presabs.csv',header=T, sep = ',')
View(borneo)
```

We've also summed the occurrences together and grouped these by quadrat:

```
borneo.out <- read.table('borneo_away_fr_boardwalk.csv',header=T,
sep = ',')
View(borneo)
```

And we have also split the data by transect:

```
transect1 <- read.table('borneo_transect1.csv',header=T, sep = ',')
transect2 <- read.table('borneo_transect2.csv',header=T, sep = ',')
transect3 <- read.table('borneo_transect3.csv',header=T, sep = ',')
transect4 <- read.table('borneo_transect4.csv',header=T, sep = ',')
transect5 <- read.table('borneo_transect5.csv',header=T, sep = ',')
transect6 <- read.table('borneo_transect6.csv',header=T, sep = ',')
```

Finally, we are interested in whether there may be a different pattern if we reverse the data (i.e. flip the transects and run them from the forest towards the boardwalk). As such, we have an inverted dataset as well:

```
borneo.in <- read.table('borneo_towards_boardwalk.csv',header=T,
sep = ',')
```

Most of the code we will be using is from the vegan package, so you will need to load it now if you don't have it already installed and loaded.

```
install.packages("vegan")
library(vegan)
```

Let's start by applying some different methods for generating species accumulation curves. The method **`collector`** will add data in the order it was collected. The method **`random`** is a form of bootstrapping and adds data in a random order. The method **`exact`** finds the expected mean species richness and will only work if the data is in a presence/absence format (the other methods will accept accumulated species counts). The method **`coleman`** finds the expected species richness following Coleman *et al.* (1982) and **`rarefaction`** finds the mean when accumulating individuals instead of sites. If your data is already presented in the form of new species per individual plant or animal counted, then you have already set it up as a rarefaction analysis by default (i.e. no matter what you run it will come out as a form of rarefaction).

Let's work with the presence/absence data first.

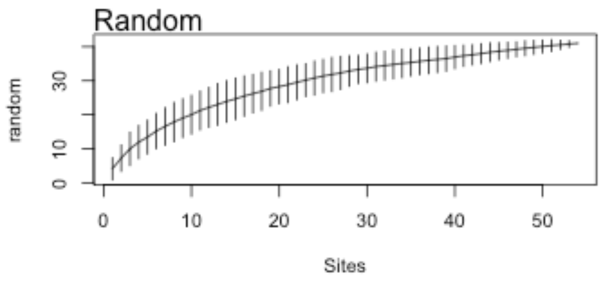
```
sp1 <- specaccum(borneo, "random")
sp2 <- specaccum(borneo, "collector")
sp3 <- specaccum(borneo, "exact")
sp4 <- specaccum(borneo, "coleman")
sp5 <- specaccum(borneo, "rarefaction")
```

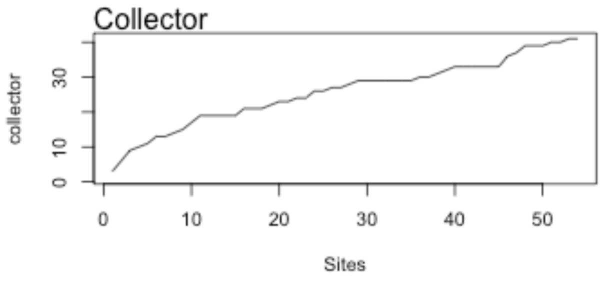
Graphing species accumulation curves

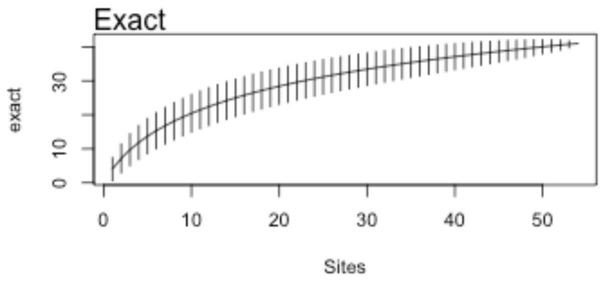
```
plot(sp1)
mtext("Random", side = 3, adj = 0, cex = 1, col = "black")

plot(sp2)
mtext("Collector", side = 3, adj = 0, cex = 1, col = "black")

plot(sp3)
mtext("Exact", side = 3, adj = 0, cex = 1, col = "black")

plot(sp4)
mtext("Coleman", side = 3, adj = 0, cex = 1, col = "black")

plot(sp5)
mtext("Rarefaction", side = 3, adj = 0, cex = 1, col = "black")
```

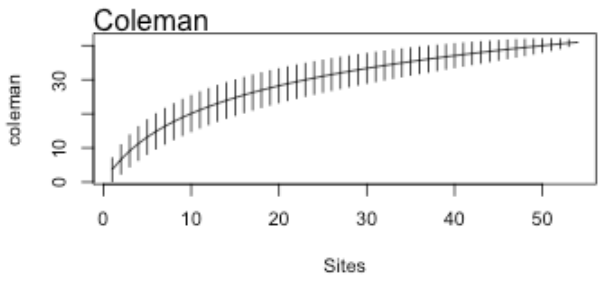
You should obtain the plots shown over-page. Note that ‘collector’ does not have standard error bars, as it is simply presenting the species in the order they were found. You can plot all the plots in one column by re-setting your par, but this will only work if you have a sufficiently large screen.

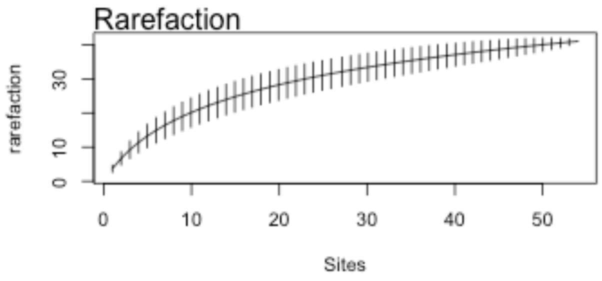
```
par(mfrow=c(5,1))
```


Random
random
0
10
30
0
10
20
30
40
50
Sites


Collector
collector
0
10
30
0
10
20
30
40
50
Sites


Exact
exact
0
10
30
0
10
20
30
40
50
Sites


Coleman
coleman
0
10
30
0
10
20
30
40
50
Sites


Rarefaction
rarefaction
0
10
30
0
10
20
30
40
50
Sites

Graphing species accumulation curves: adding colour

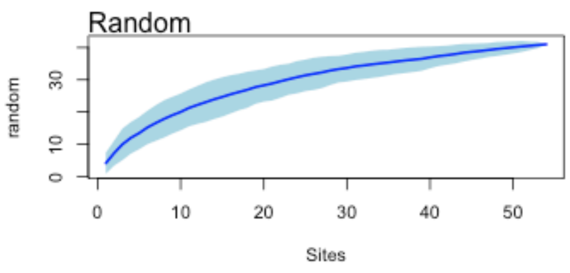
Let's try adding some colour and creating some nicer looking graphs. Just as an aside, scientific graphing conventions don't include titles (i.e. scientific graphs typically don't have titles). We are including titles just to keep track of what we are plotting because there are so many plots to look at.

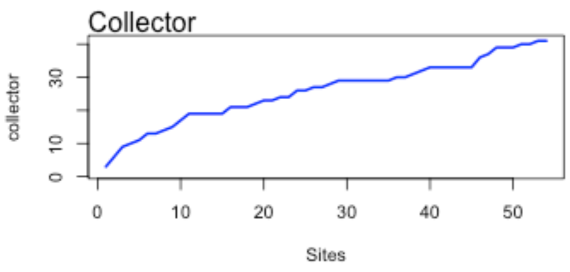
```
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Random", side = 3, adj = 0, cex = 1, col = "black")

plot(sp2, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Collector", side = 3, adj = 0, cex = 1, col = "black")

plot(sp3, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Exact", side = 3, adj = 0, cex = 1, col = "black")

plot(sp4, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Coleman", side = 3, adj = 0, cex = 1, col = "black")

plot(sp5, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue")
mtext("Rarefaction", side = 3, adj = 0, cex = 1, col =
"black")
```

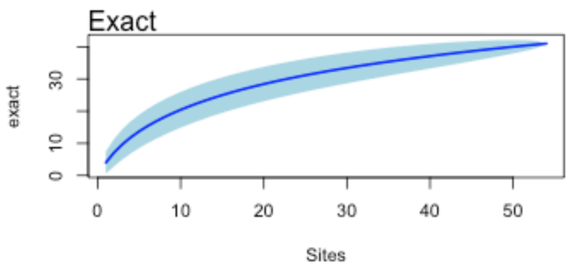
Note also that methods that use bootstrapping will generate a slightly different result each time because they are working from a randomly subsetting dataset. You can 'stablise' the result by setting a seed number. If you do this, you'll get the same result each time.

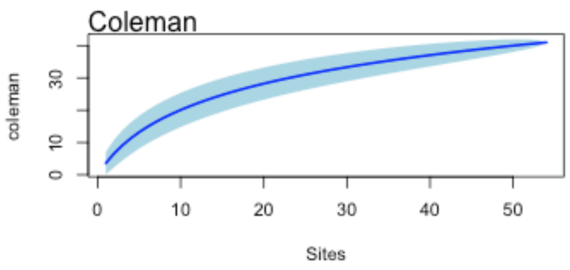
```
set.seed(42)
```

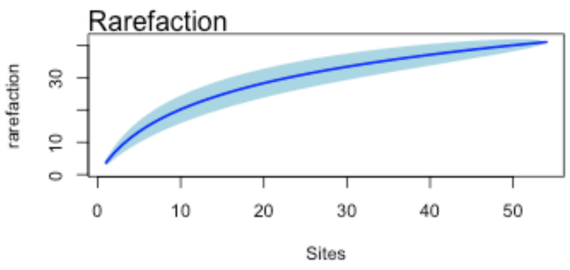
Now try re-running the random curve a couple times. You should find that you get the same result. You can set the seed to any number. I tend to use 42 just because of fond memories of reading Douglas Adams.


Random
random
0
10
30
0
10
20
30
40
50
Sites


Collector
collector
0
10
30
0
10
20
30
40
50
Sites


Exact
exact
0
10
30
0
10
20
30
40
50
Sites


Coleman
coleman
0
10
30
0
10
20
30
40
50
Sites


Rarefaction
rarefaction
0
10
30
0
10
20
30
40
50
Sites

The species accumulation curves are looking quite nice, but we have a problem, which is that R thinks the whole 6 transects by 9 quadrats is a single collecting event, giving over 50 samples. We can sum the species observations together, and organise them by transects, but if we do this the method **exact** will no longer work. That's okay, as we will focus on using the methods **random** and **collector** from here on.

However, let's start off by comparing the six transects before summing them together.

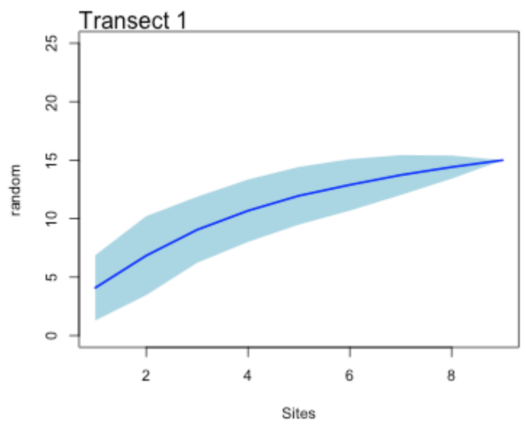
Using the random method (by transect)

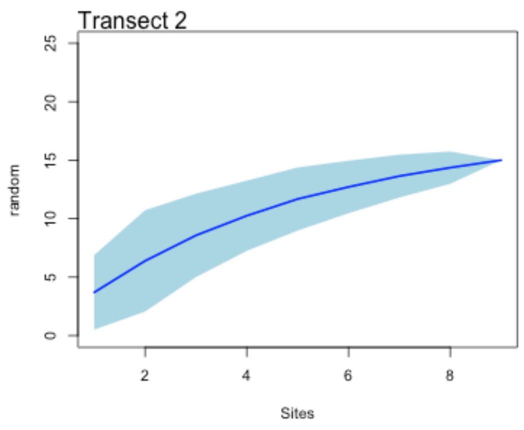
```
# RANDOM METHOD
# Set your graphics parameters to 3x2 only if you have sufficient
space on your screen. Otherwise just run these one at a time.

par(mfrow=c(3,2))

sp1 <- specaccum(transect1, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 1", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect2, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 2", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect3, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 3", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect4, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 4", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect5, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 5", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect6, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 6", side = 3, adj = 0, cex = 1, col = "black")
```

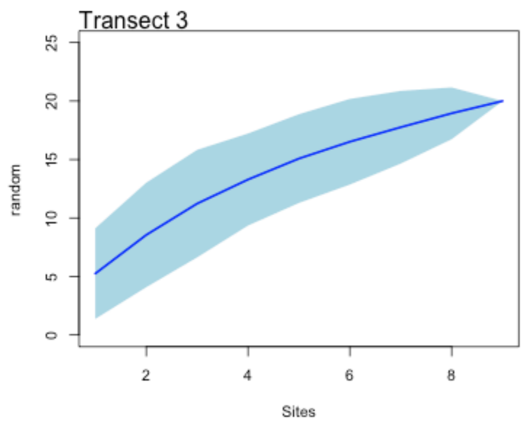
Using the collector method (by transect)

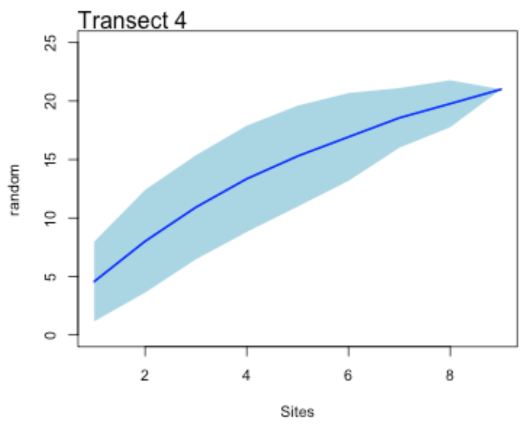
```
# COLLECTOR METHOD
# Set your graphics parameters to 3x2 only if you have sufficient
space on your screen. Otherwise just run these one at a time.


par(mfrow=c(3,2))

sp1 <- specaccum(transect1, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 1", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect2, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 2", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect3, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 3", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect4, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 4", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect5, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 5", side = 3, adj = 0, cex = 1, col = "black")

sp1 <- specaccum(transect6, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,25))
mtext("Transect 6", side = 3, adj = 0, cex = 1, col = "black")
```

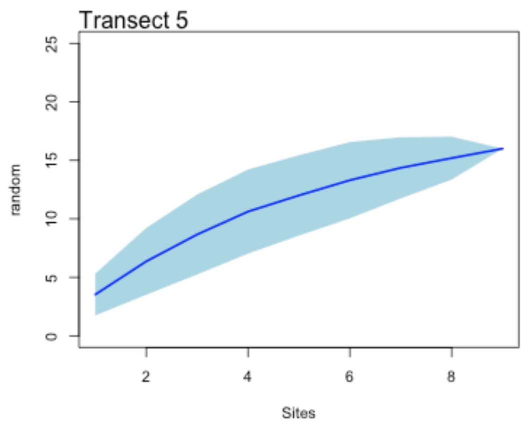
Using the random method:

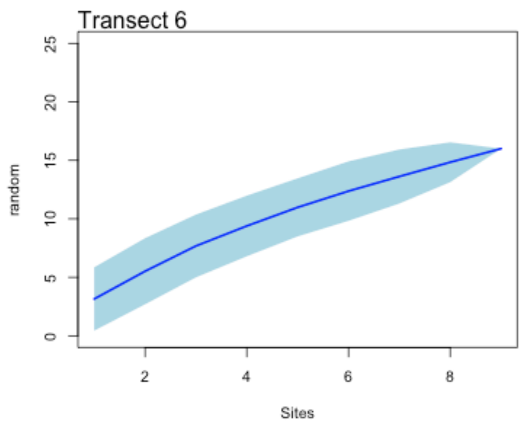




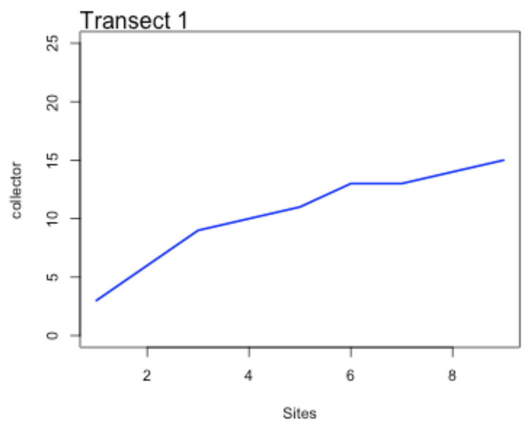


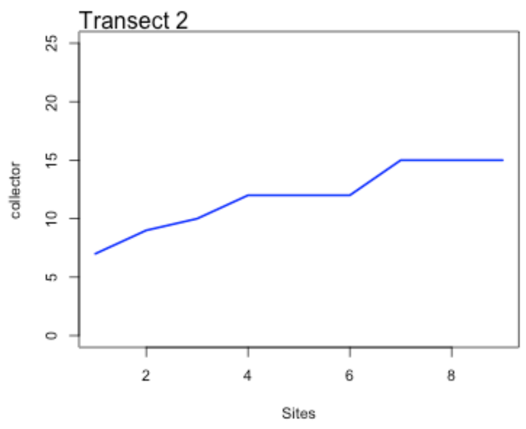


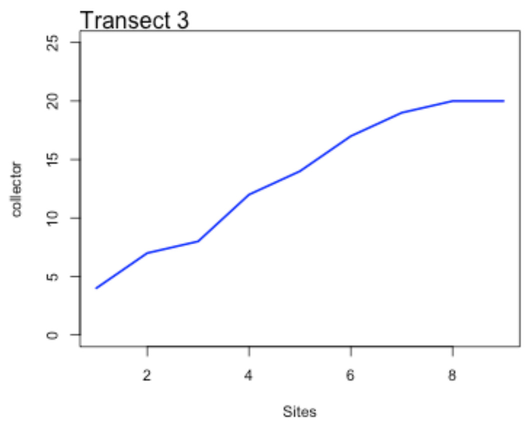


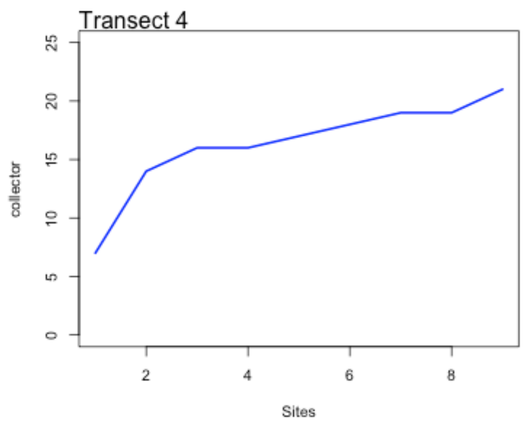


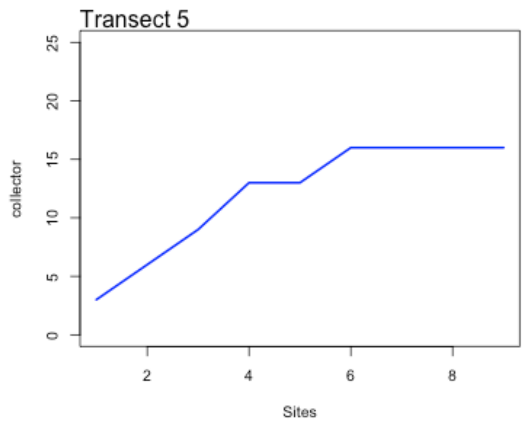
Using the collector method:

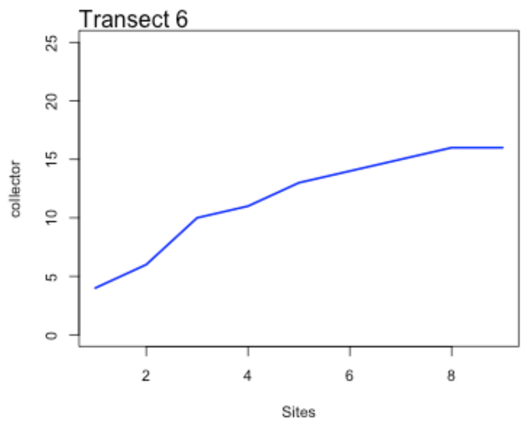












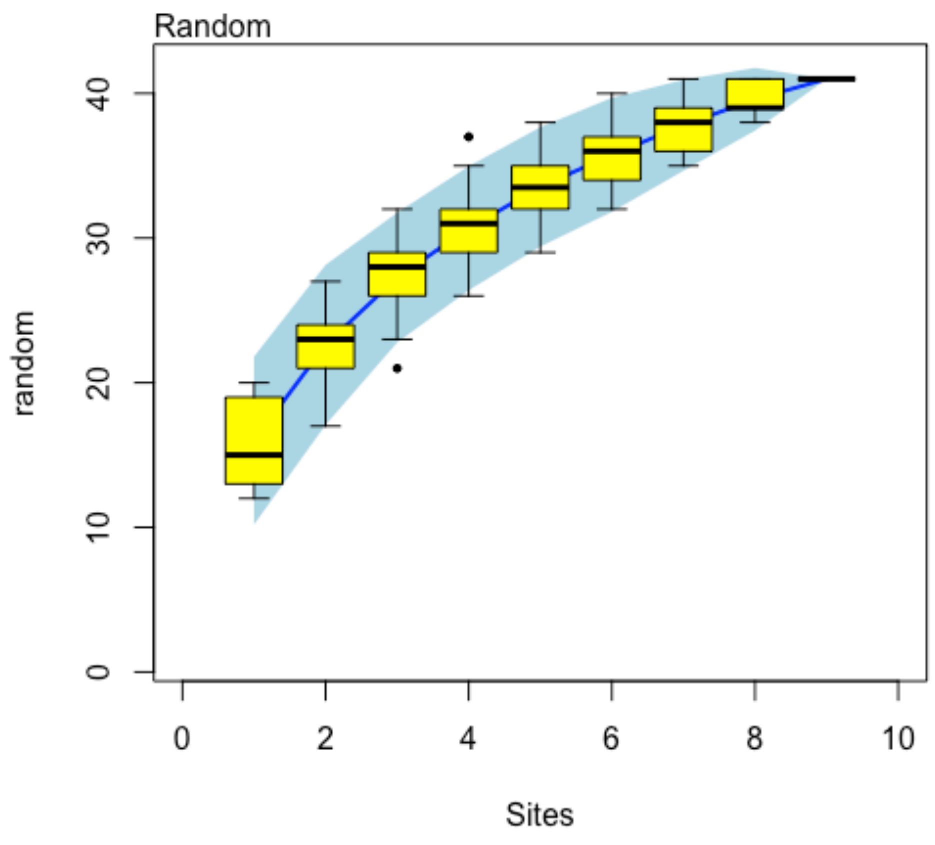
At this point we should check whether there are any large jumps or kinks in the data. These would indicate an ecotone change, which invalidates SACs (i.e. a SAC is invalid if you run a transect from a forest into wetland, for instance). There are a couple places where there seem to be minor jumps, but there is nothing consistent, and given the coarse granularity of the transect (just nine quadrats) some jumpiness is to be expected.

Let's now look at all transects combined. We'll add some additional estimates to the graphs, focusing on the data as it was collected. At this stage we are still working with the data arranged so that the collection starts at the boardwalk and then moves into the rainforest.

Plot Random resamples using boxplots

The following code plots the bootstrapped random resamples using boxplots. These are simply 'added' over the top of the standard error.

```
set.seed(42)

# Random bootstrapped model
# Boxplots show upper and low quartiles, median and range for each
transect point
sp1 <- specaccum(borneo.out, "random")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", xlim =c(0,10))
boxplot(sp1, col="yellow", add=TRUE, lty=1, pch=20, cex = 0.7)
mtext("Random", side = 3, adj = 0, cex = 1, col = "black")
```



Plot Random resamples with fitted curves

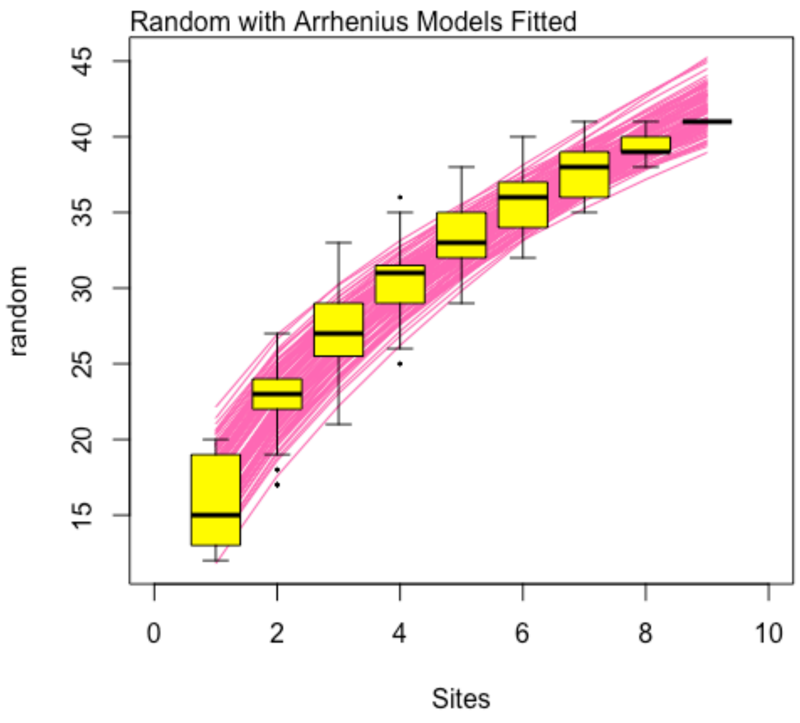
There are a number of different curve fitting options. Library 'vegan' uses self-starting non-linear regression curve fitting methods from 'nls'.

Arrhenius Model (Power Law)

The Arrhenius Model is among the oldest and simplest ways to model a species accumulation curve. It follows a straightforward power law. First, we will start by fitting Arrhenius models to all of our random resampled datasets and plot this. Here is the description from **`?arrhenius`**

`The Arrhenius model (SSarrhenius) is the expression k*area^z. This is the most classical model that can be found in any textbook of ecology (and also in Dengler 2009). Parameter z is the steepness of the species-area curve, and k is the expected number of species in a unit area.`

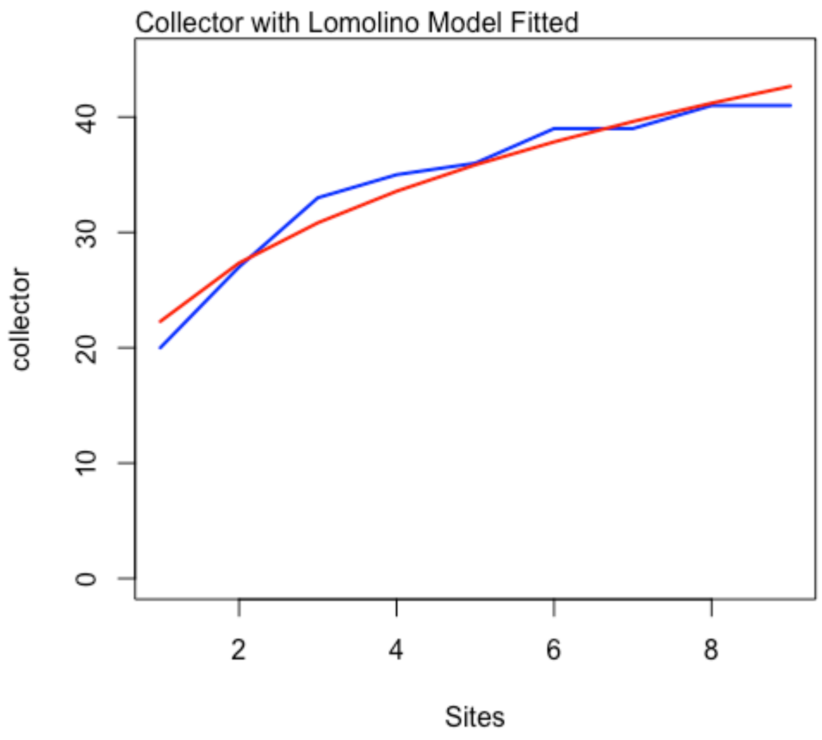
```
# Fit Arrhenius (power law) models to the random data
# Plot all curves together
sp1 <- specaccum(borneo.out, "random")
mods <- fitspecaccum(sp1, "arrhenius")
plot(mods, col="hotpink", xlim =c(0,10)))
boxplot(sp1, col="yellow", add=TRUE, lty=1, pch=20, cex=0.3)
mtext("Random with Arrhenius Models Fitted", side=3, adj=0, cex=1,
col="black")
```



Arrhenius Model (Power Law) applied to 'Collector'

The easiest way to obtain parameters for our dataset is to apply the power law (Arrhenius) model to the collector data. Here's code to generate a model and plot it:

```
# Fit Arrhenius (power law) model to collector curve only
# Collector = red
# Power law line of best fit = blue
sp1 <- specaccum(borneo.out, "collector")
plot(sp1, ci.type="poly", col="blue", lwd=2, ci.lty=0,
ci.col="lightblue", ylim=c(0,45))
mod1 <- fitspecaccum(sp1, "arrhenius") # collector
plot(mod1, add = TRUE, col=2, lwd=2)
mtext("Collector with Arrhenius Model Fitted", side=3, adj=0,
cex=1, col="black")
```



Obtaining parameters 'k' and 'z' for the curve

The parameter 'k' is sometimes also given as 'c'. We can obtain the parameters for the average of the random resampled datasets like so:

```
coef(mod1)
```

RESULT

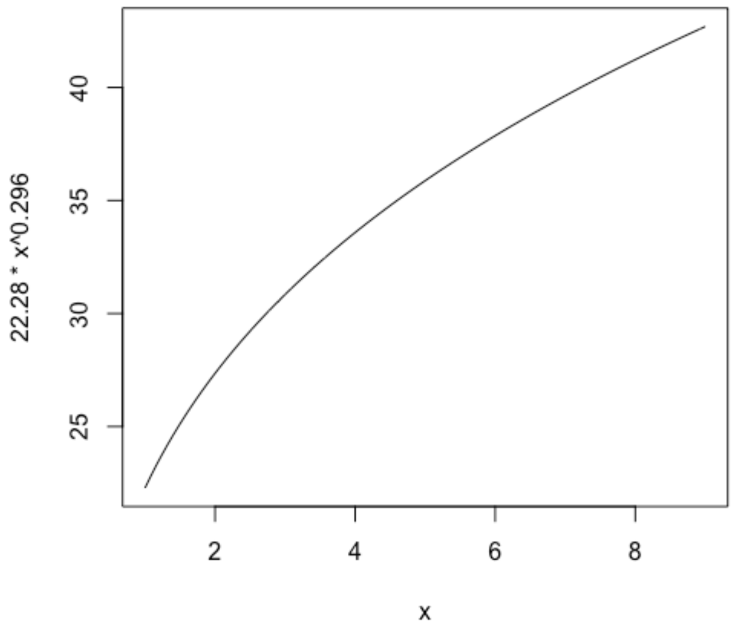
```
         k          z
22.2821985  0.2956144
```

These parameters can be slotted into a straightforward power law equation:

$$y = \mathrm{k} * x^{\mathrm{z}}$$

Where the x is the spatial measurement, often given as 'area' for species-area relationships, but it could be 'quadrat' for a species accumulation curve based on quadrats along a transect (as is the case here). Frequently z is simply assumed to be 0.25, although as we can see from our example it is preferable to model your data and establish is this is true for your environment of interest. In this case z is 0.296.

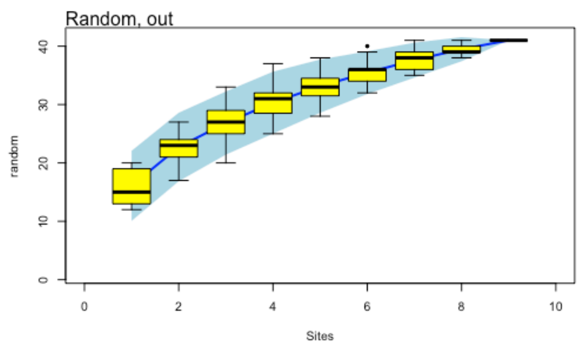
```
# Back to basics
# We can simply take the k and z values and plot the equation
curve(22.28 * x^0.296, from = 1, to = 9)
```

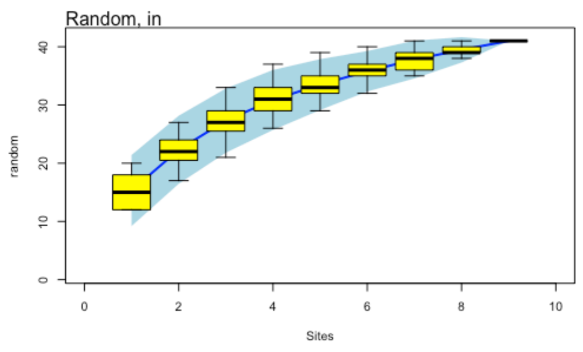


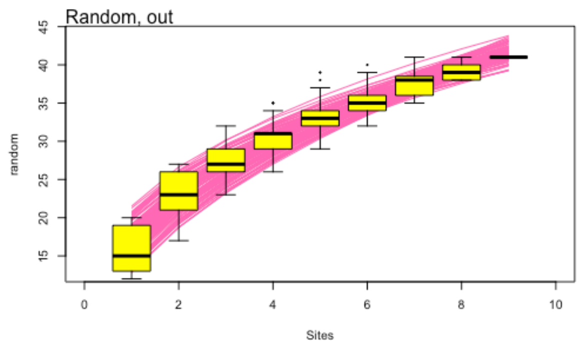
The original question was one of examining whether there is a difference in curves when the data is reversed (i.e. would moving towards the boardwalk produce a different curve to that generated by moving away from the boardwalk, even if the data were otherwise the same).

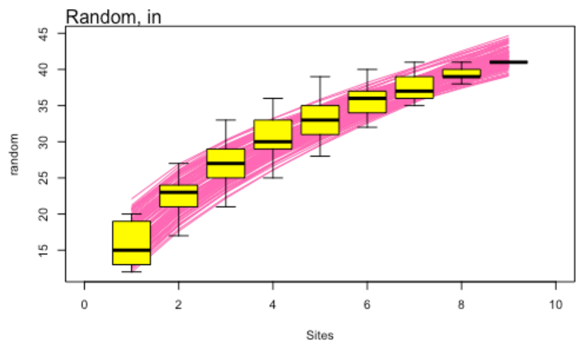
```
par(mfrow=c(3,2))
set.seed(42)

# Random bootstrapped model
# Boxplots show upper and low quartiles, median and range for each transect point

# Borneo out
sp1 <- specaccum(borneo.out, "random")
plot(sp1,  ci.type="poly",  col="blue",  lwd=2,  ci.lty=0,  ci.col="lightblue",  xlim
=c(0,10))
boxplot(sp1, col="yellow", add=TRUE, lty=1, pch=20, cex = 0.7)
mtext("Random, out", side = 3, adj = 0, cex = 1, col = "black")

# Borneo in
sp1 <- specaccum(borneo.in, "random")
plot(sp1,  ci.type="poly",  col="blue",  lwd=2,  ci.lty=0,  ci.col="lightblue",  xlim
=c(0,10))
boxplot(sp1, col="yellow", add=TRUE, lty=1, pch=20, cex = 0.7)
mtext("Random, in", side = 3, adj = 0, cex = 1, col = "black")

# Borneo out
sp1 <- specaccum(borneo.out, "random")
mods <- fitspecaccum(sp1, "arrhenius")
plot(mods, col="hotpink", xlim =c(0,10))
boxplot(sp1, col="yellow", add=TRUE, lty=1, pch=20, cex = 0.3)
mtext("Random, out", side=3, adj=0, cex=1, col = "black")

# Borneo in
sp1 <- specaccum(borneo.in, "random")
mods <- fitspecaccum(sp1, "arrhenius")
plot(mods, col="hotpink", xlim =c(0,10))
boxplot(sp1, col="yellow", add=TRUE, lty=1, pch=20, cex = 0.3)
mtext("Random, in", side=3, adj=0, cex=1, col = "black")

# Borneo out
sp1 <- specaccum(borneo.out, "collector")
plot(sp1,   ci.type="poly",   col="blue",   lwd=2,   ci.lty=0,   ci.col="lightblue",
ylim=c(0,45))
mod1 <- fitspecaccum(sp1, "arrhenius")
plot(mod1, add = TRUE, col=2, lwd=2)
mtext("Collector, out", side = 3, adj = 0, cex = 1, col = "black")

# Obtain 'k' and 'z' for the model
# k is sometimes also given as 'c'
coef(mod1)

# Borneo in
sp1 <- specaccum(borneo.in, "collector")
plot(sp1,   ci.type="poly",   col="blue",   lwd=2,   ci.lty=0,   ci.col="lightblue",
ylim=c(0,45))
mod1 <- fitspecaccum(sp1, "arrhenius")
plot(mod1, add = TRUE, col=2, lwd=2)
mtext("Collector, in", side = 3, adj = 0, cex = 1, col = "black")

# Obtain 'k' and 'z' for the model
# k is sometimes also given as 'c'
coef(mod1)
```

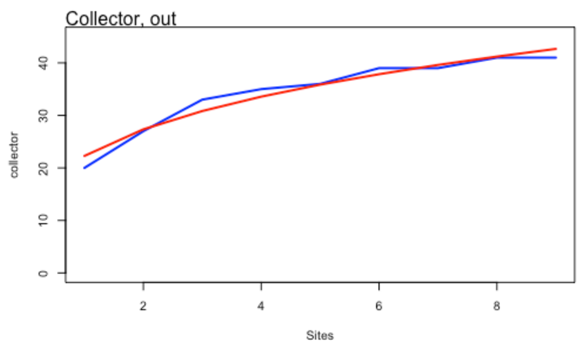
Here are the graphs produced using the code above, but for the data moving away from the boardwalk (out, left) and from the jungle towards the boardwalk (in, right).

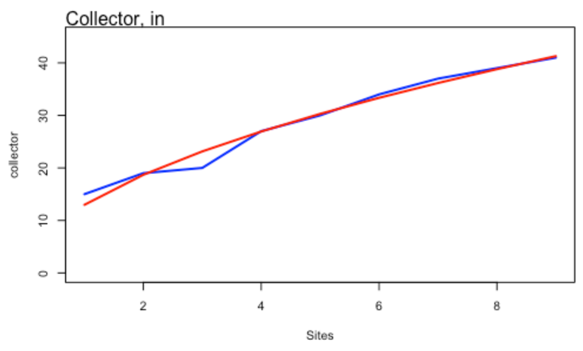












Out: k = 22.28, z = 0.296
In: k = 12.97, z = 0.527

This illustrates one problem with just relying on the data in the order it is obtained. This is exactly the same data reversed, but because there is a slight gradient in the environment (i.e. the boardward is an area of disturbance), the k and z values are completely different when the orientation is reversed.

To attempt to get around this, researchers sometimes find the 'best' (i.e. most parsimonious) of the random subsampled models and use this to calculate k and z instead. Let's see what happens if we do this.

First we will check the 'out' direction.

```
set.seed(42)

sp1 <- specaccum(borneo.out, "random")
mods <- fitspecaccum(sp1, "arrhenius")
best <- which((sapply(mods$models, AIC)) ==
min(sapply(mods$models, AIC)))

# summarizes the best model, including k and z values
# We can't use 'coef' here because of the model structure
mods$models[best]
```

RESULT

```
[[1]]
Nonlinear regression model
  model: y ~ SSarrhenius(x, k, z)
   data: parent.frame()
      k       z
19.6636  0.3417
 residual sum-of-squares: 3.71

Number of iterations to convergence: 2
Achieved convergence tolerance: 6.606e-06
```

Now, let's check the 'in' direction.

```
sp1 <- specaccum(borneo.in, "random")
mods <- fitspecaccum(sp1, "arrhenius")
best <- which((sapply(mods$models, AIC)) ==
min(sapply(mods$models, AIC)))

# summarizes the best model, including k and z values
# We can't use 'coef' here because of the model structure
mods$models[best]
```

RESULT

```
[[1]]
Nonlinear regression model
  model: y ~ SSarrhenius(x, k, z)
   data: parent.frame()
      k       z
19.1106  0.3578
 residual sum-of-squares: 3.456

Number of iterations to convergence: 3
Achieved convergence tolerance: 1.111e-07
```

This is certainly better. We now have k of 19.6 and 19.1 and z of 0.342 and 0.358. However, if you re-run this a number of times with no set.seed, you'll find that these numbers still wander around quite a bit. Increasing the number of iterations to 'stabilise' the output may still be needed.

The moral here is to be very careful with your data when building a species accumulation curve. It’s not only passing through a clear ecotone that can cause species accumulation curves to become sub-optimal. even just slight gradients in the environment (which may not always be obvious to a field researcher) can cause problems.

Finally, keep in mind that some more sophisticated models, such as **Lomolino**, may deliver better results. Use `?lomolino` to check information on that modelling method.