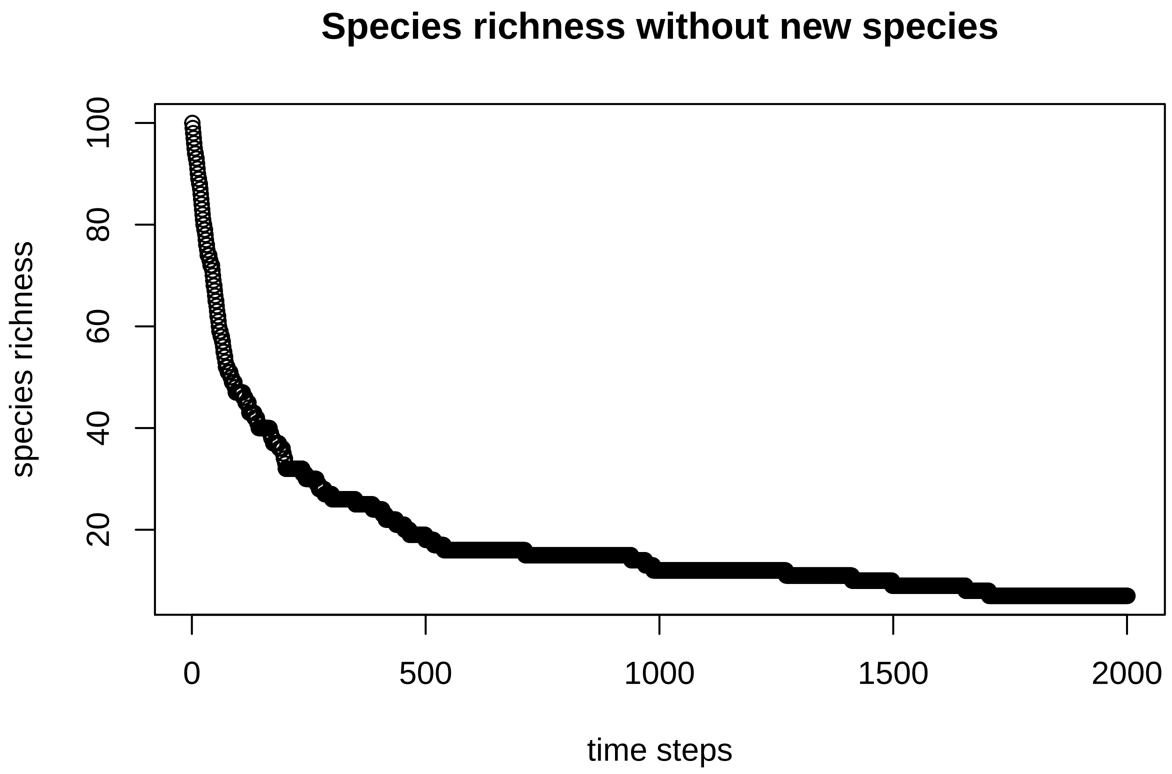
HPC Coursework

PetraGuy

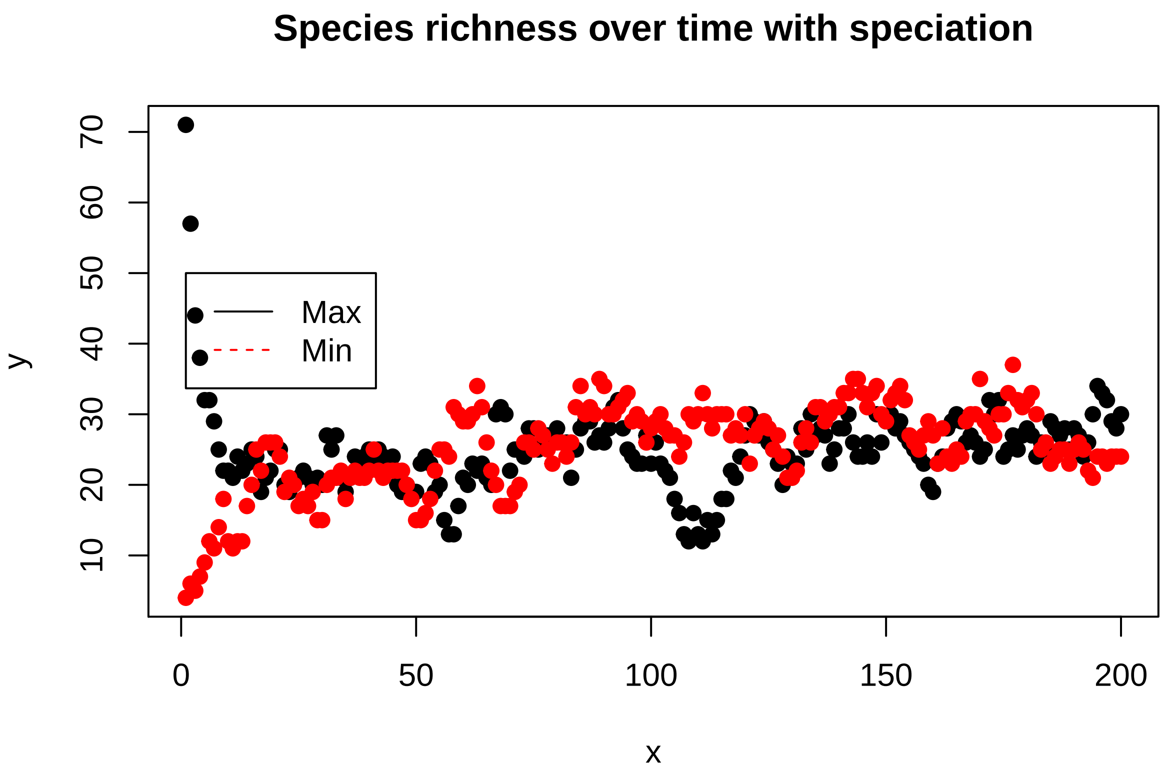
14th December 2017

```
question_8()
```



The graph plots species richness of a community starting with 100 different species over 2000 generations. In this model new species are not introduced, so each step either maintains the richness or reduces it. For example, if you begin with a community (1,2,3,4,5), one step could change this to (2,2,3,4,5) At the next step the community will either remain unchanged, or be reduced, for example (2,2,3,3,5). Since there is no mechanism to increase species richness, eventually, given enough steps, the richness will become 1.

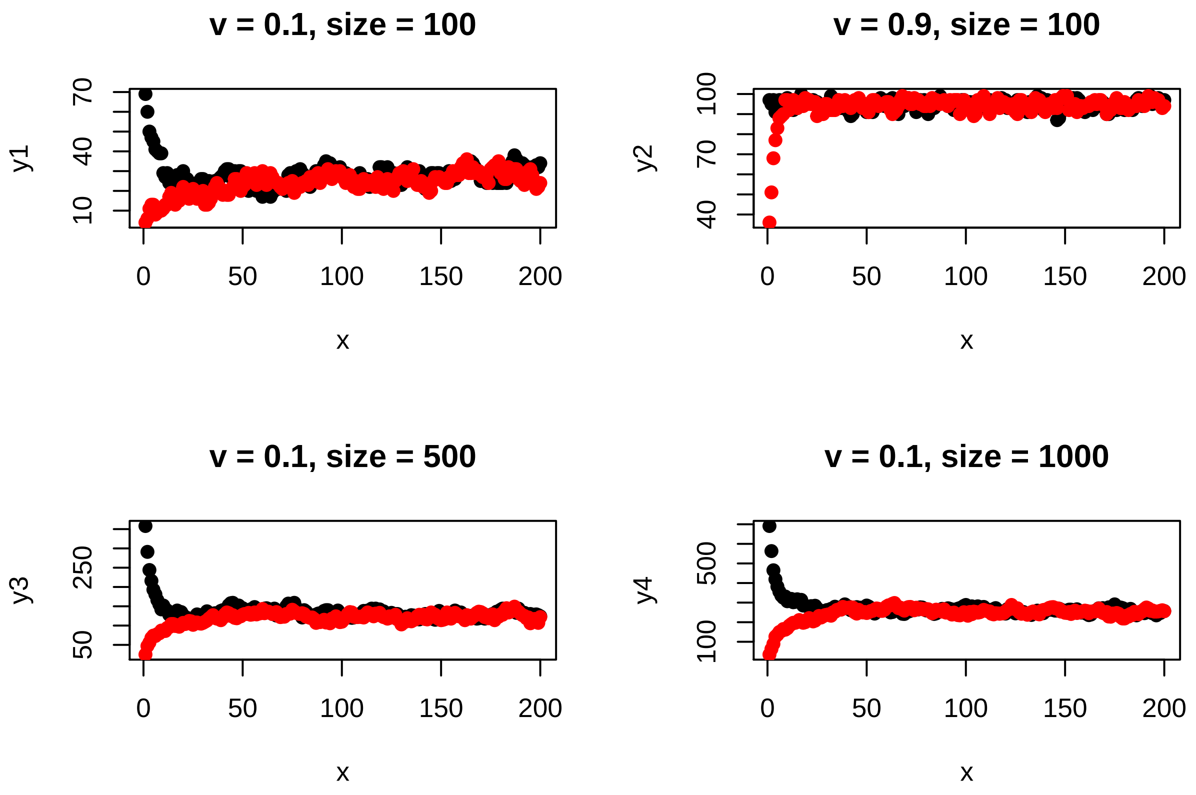
```
question_12()
```



The graph shows the species richness for two communities, one with maximum richness, one with minimum, both with 100 individuals and a speciation reate of 0.1. Because the speciation rate is the same both communities they tend to the same richness after sufficient generations. A richness of around 30 after 50 generations in this case. The higher the speciation rate, the greater the richness of the final community, because the algorithm will more refrequently follow the step of generating a new individual. The final community richness depends on the speciations rate and the size of the initial community - as the graphs below show

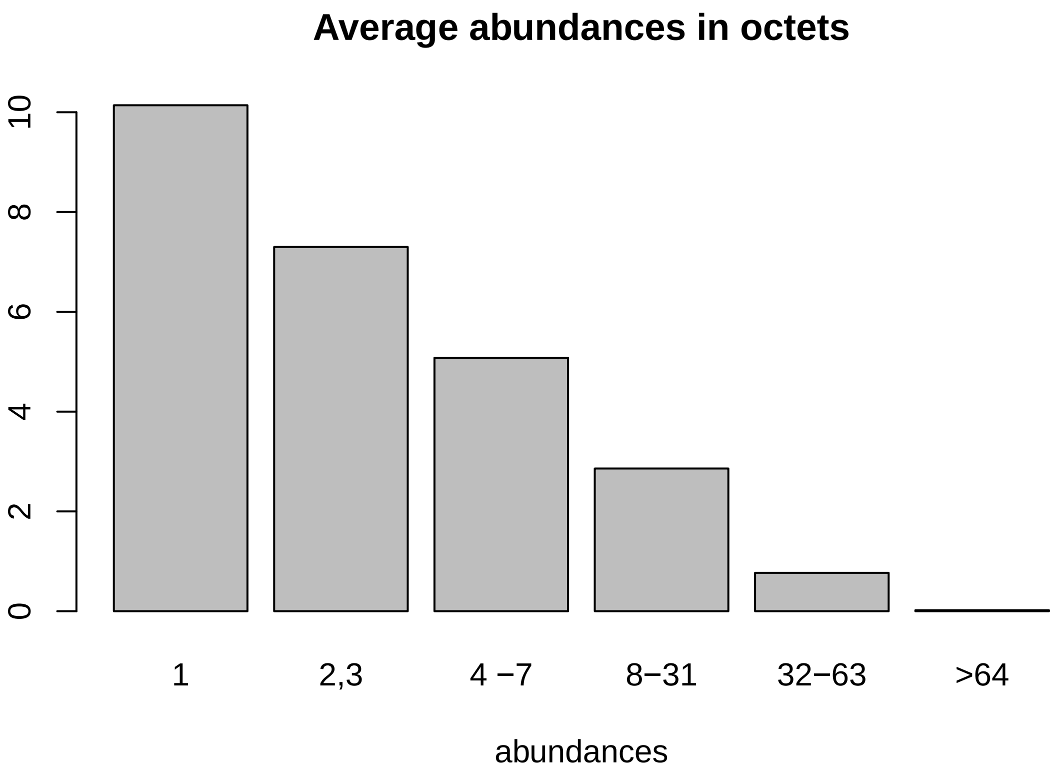
```
question_12b = function() {
  t = 200

  v = 0.1
  comm_max = initialise_max(100)
  comm_min = initialise_min(100)
  rich_max = neutral_time_series_speciation(comm_max, v, t)
  rich_min = neutral_time_series_speciation(comm_min, v, t)
  y1 =  cbind(rich_max, rich_min)


  v = 0.9
  comm_max = initialise_max(100)
  comm_min = initialise_min(100)
  rich_max = neutral_time_series_speciation(comm_max, v, t)
  rich_min = neutral_time_series_speciation(comm_min, v, t)
  y2 =  cbind(rich_max, rich_min)


  v = 0.1
  comm_max = initialise_max(500)
  comm_min = initialise_min(500)
  rich_max = neutral_time_series_speciation(comm_max, v, t)
  rich_min = neutral_time_series_speciation(comm_min, v, t)
  y3 =  cbind(rich_max, rich_min)


  v = 0.1
  comm_max = initialise_max(1000)
  comm_min = initialise_min(1000)
  rich_max = neutral_time_series_speciation(comm_max, v, t)
  rich_min = neutral_time_series_speciation(comm_min, v, t)
  y4 =  cbind(rich_max, rich_min)

  x = (1:t)
  par(mfrow=c(2,2))

  matplot (x, y1, pch = 19, col = 1:2, main = "v = 0.1, size = 100")

  matplot (x, y2, pch = 19, col = 1:2, main = "v = 0.9, size = 100")

  matplot (x, y3, pch = 19, col = 1:2, main = "v = 0.1, size = 500")

  matplot (x, y4, pch = 19, col = 1:2, main = "v = 0.1, size = 1000")


}
```

```
question_12b()
```



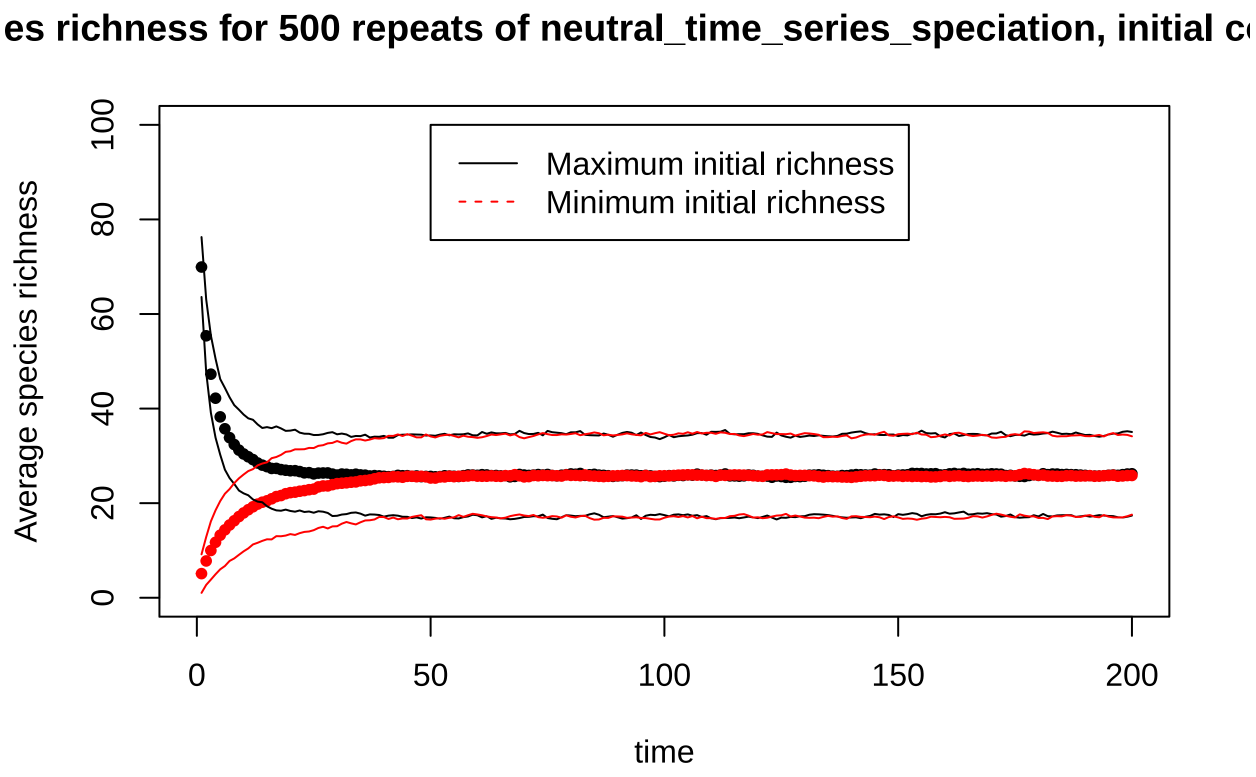
For community size of 100, the final species richness is around 30 for v = 0.1 and nearly 100 for v = 0.9. If the inital community size is increased to 500, the final richness increases to around 75, and for an inital community of 1000, the final richness is around 125.

```
question_16()
```



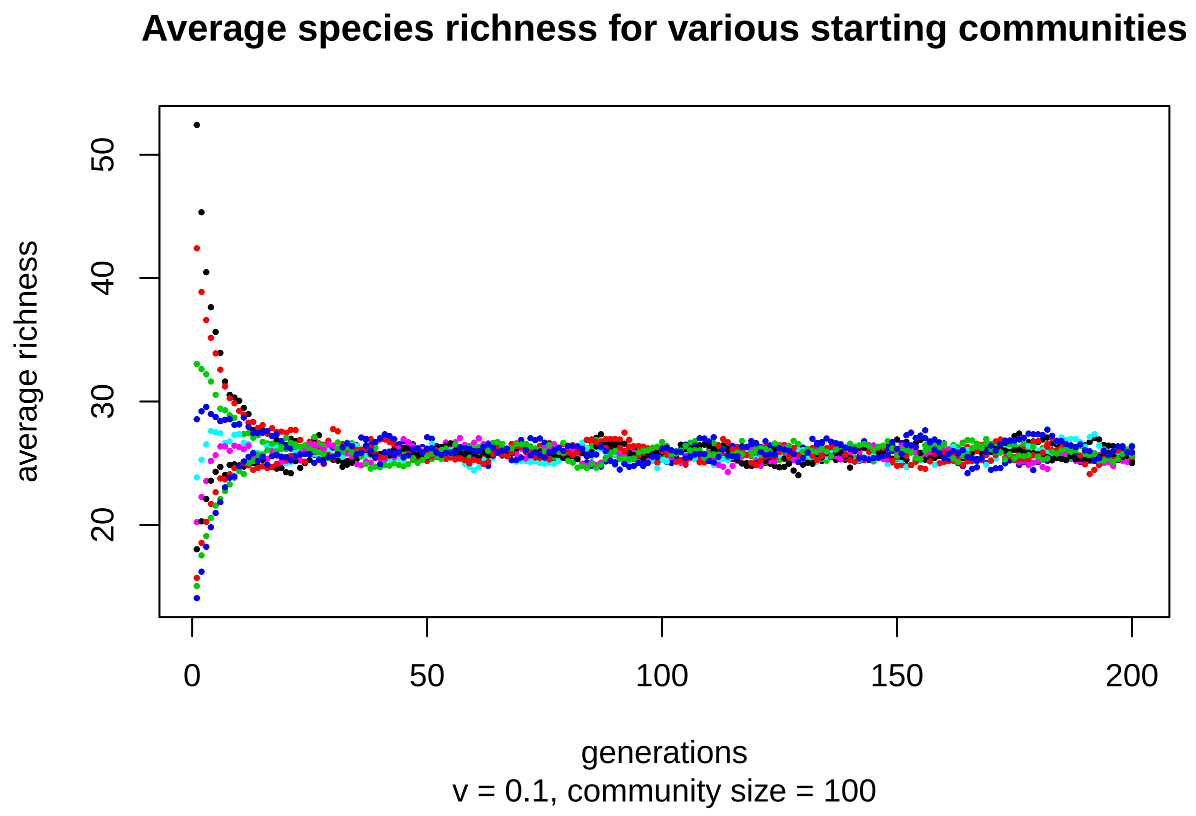
The graph shows the average abundances of the community after the burn-in of 200 generations. The distribution of abundances would change if the speciation rate changed. A higher speciation rate would result in a richer community and therefore the first octet would have a higher frequency. A low speciation rate results in a community of low richness and therefore the frequencies of the larger octets increase at the expense of the smaller. As we saw bove, a large initial community would also result in higher richness and larger first octet.

```
challenge_A()
```

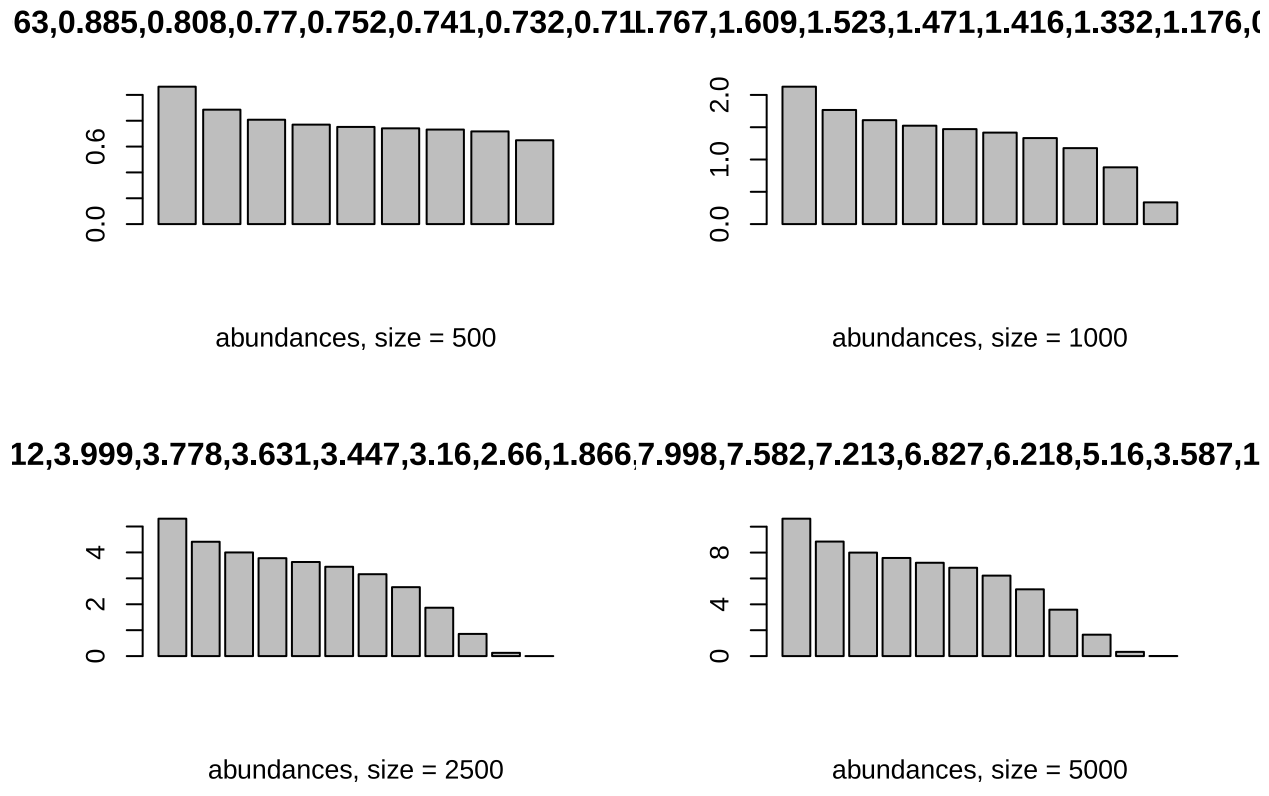


The graph is a plot of average richness values as the times increments. The 97.2% confidence inervals were calculated using z = 2.2 giving the 98.6th percentile - giving 1.4% in each tail. Dynamic equilibrium is reached at around 50 steps.

```
challenge_B()
```

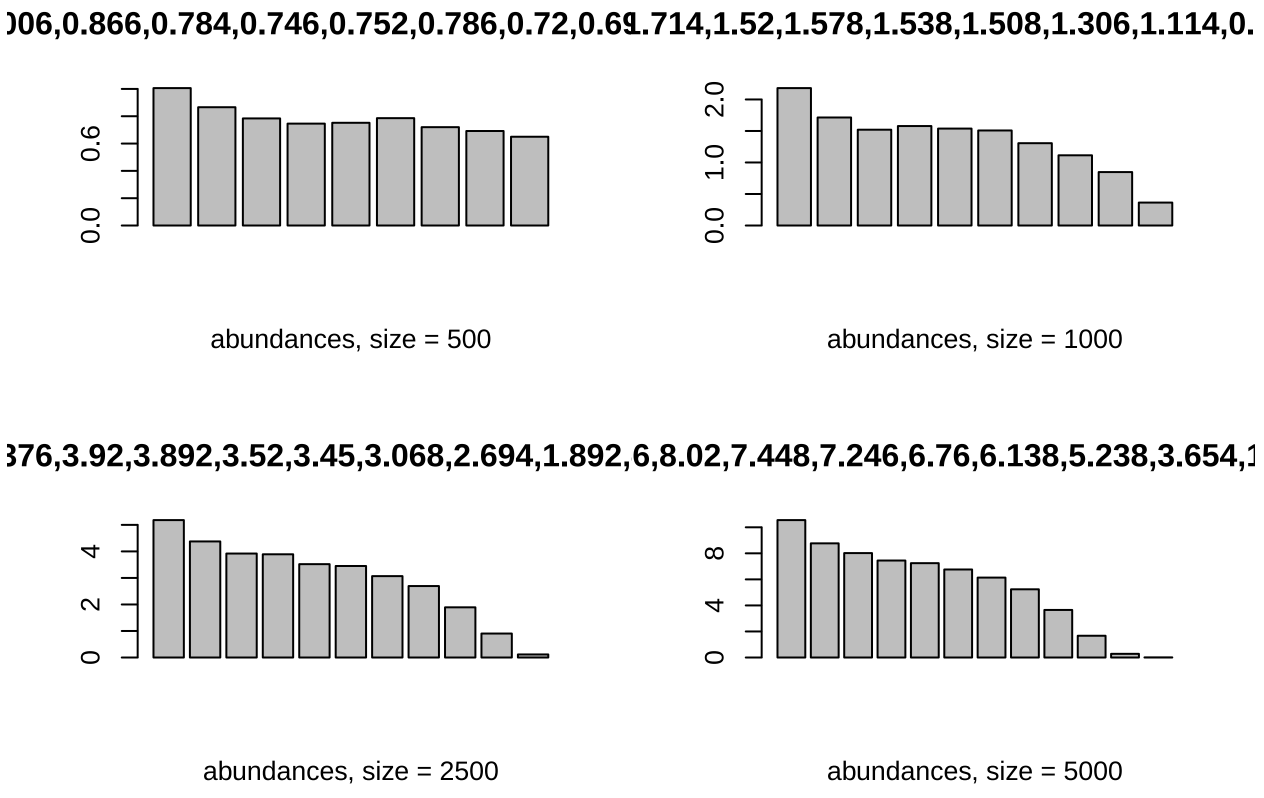


A range of initial community richness were used here. The average was calculated over 50 repeats for the time series of species richness. The graph shows that whatever the initial richness, the same richness will be reached eventually for all communities, given that they were all the same size and same speciation rate.



Question 20. These are the output average abundances, in octets, from the simulations run on the HPC.The values are shown again below for clarity. The speciation rate was very low for this simulation, therefore you would expect the final species richness to be low. The smaller communities show less richness than the larger ones, as shown in the graphs for question_12b above. With such a low speciation rate, even a community of 500 does not reach a very large final secies richness.

```
challenge_D()
```



```
## [1] 1.006 0.866 0.784 0.746 0.752 0.786 0.720 0.692 0.650
##  [1] 2.180 1.714 1.520 1.578 1.538 1.508 1.306 1.114 0.848 0.364
##  [1] 5.180 4.376 3.920 3.892 3.520 3.450 3.068 2.694 1.892 0.904 0.116
##  [1] 10.552  8.766  8.020  7.448  7.246  6.760  6.138  5.238  3.654  1.672
## [11]  0.282  0.010
```

These are the octets from the coalescence simulation, which look very similar to the results of the simulation from the HPC. Not totally sure why one is quicker - except that in the coalescence model you generate one community, then do the size of the community calculations, so for our sizes,that's a maximum of 10,000 cyles in the loop. For the neutral model you generate a new community of size N many times during the burn in, then you continue recreating new communities for the presest time in order to take an average over all the cyles. Therefore there are many more calculations.

Question 18

The fractal dimension,D is given by

$$D = \frac{\log(N)}{\log(r)}$$

N is the number of new shapes produced when you divide the original by r. So for the Menger gasket which starts as 1 solid square, cutting the sides into 3 parts results in 8 solid squares. The fractal dimension is therefore

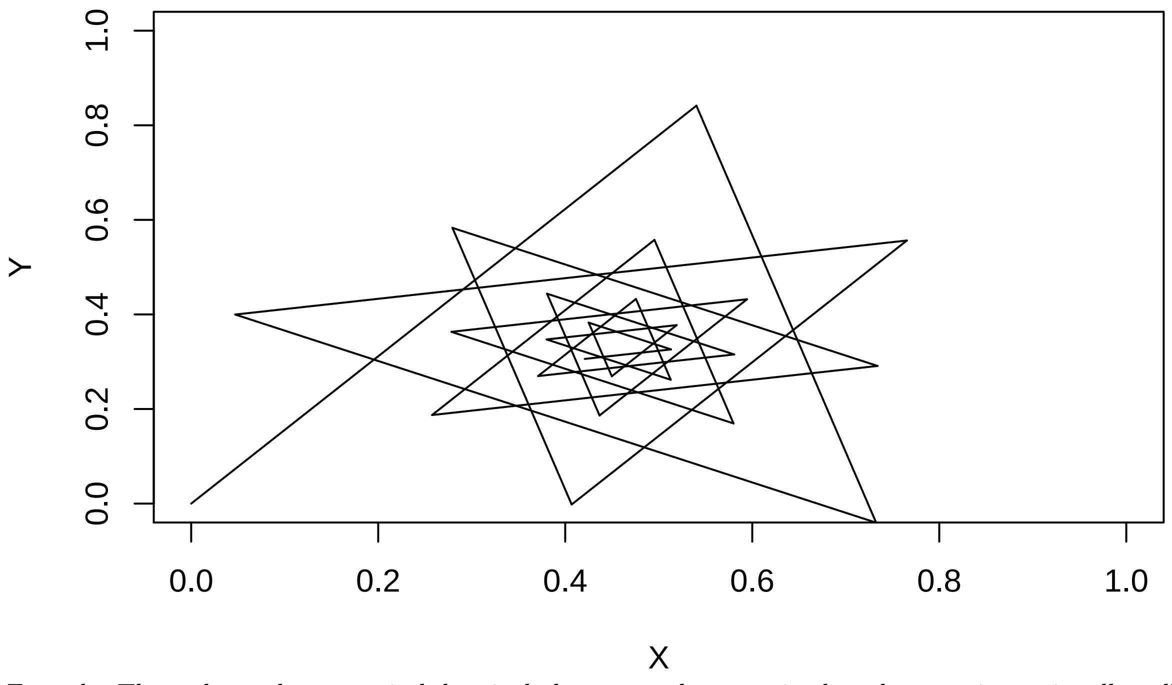
$$D = \frac{\log(8)}{\log(3)} = 1.893$$

In the Menger sponge, you start with a sold cube, cutting the sides ino 3 results in 20 smaller cubes, so the dimension is

$$D = \frac{\log(20)}{\log(3)} = 2.727$$

```
chaos_game()
```

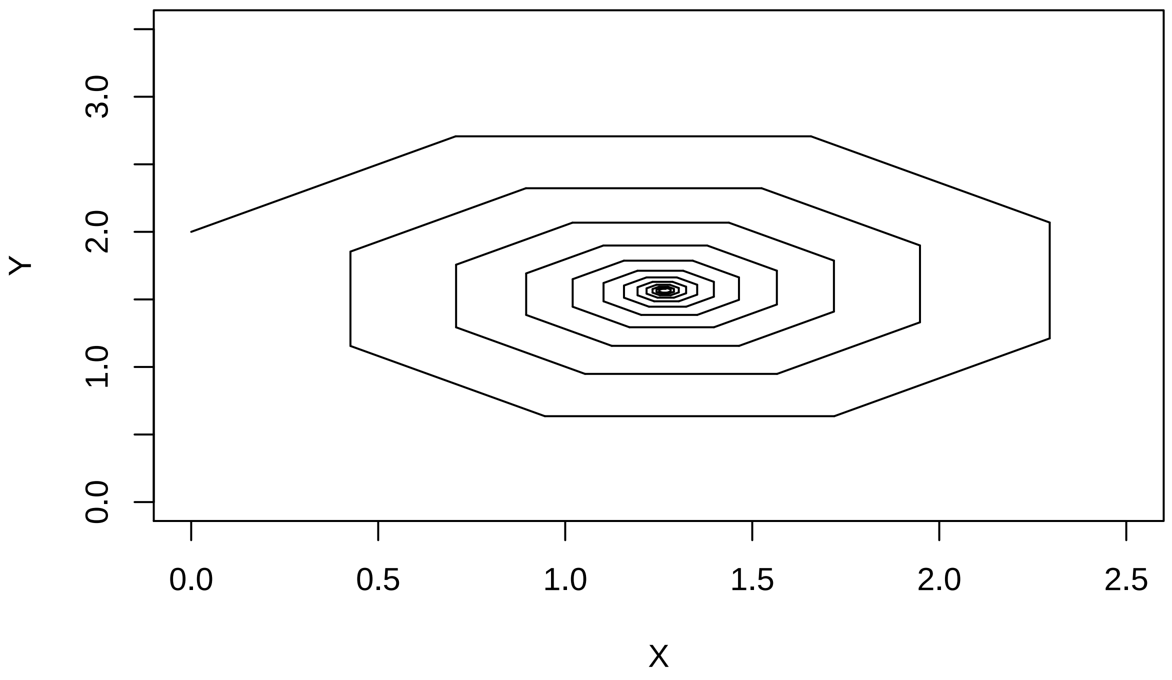
The code produces a Sierpinski Gasket type picture.

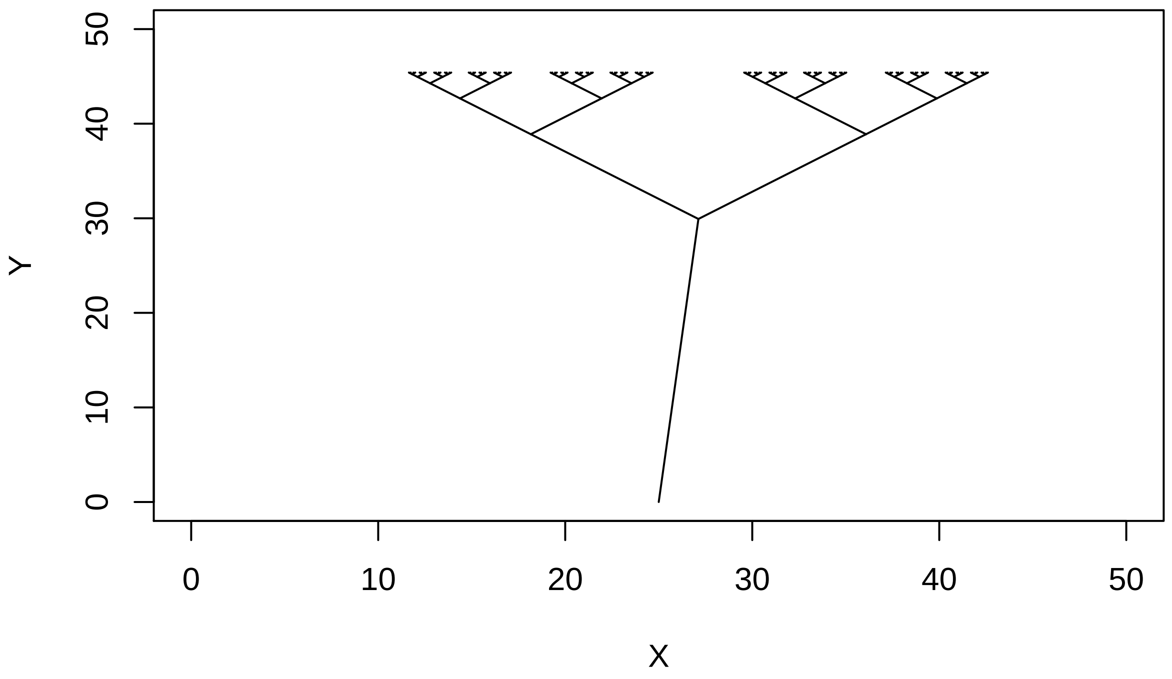


Q22

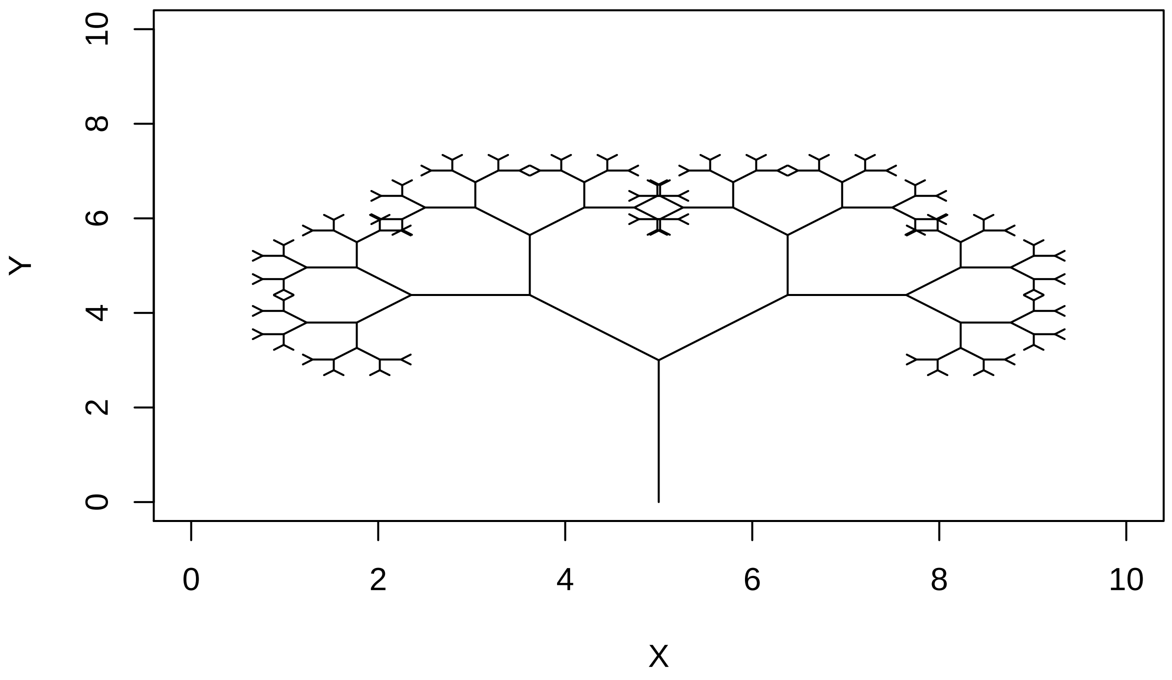
Fractals. The code produces a spiral, but includes a nested, recurssive loop because it continually calls itself. Putting an if (distance > 0.1) statement before calling spiral within spiral ensure that the programme will stop.

The angle was too large in the above spiral - so here's another

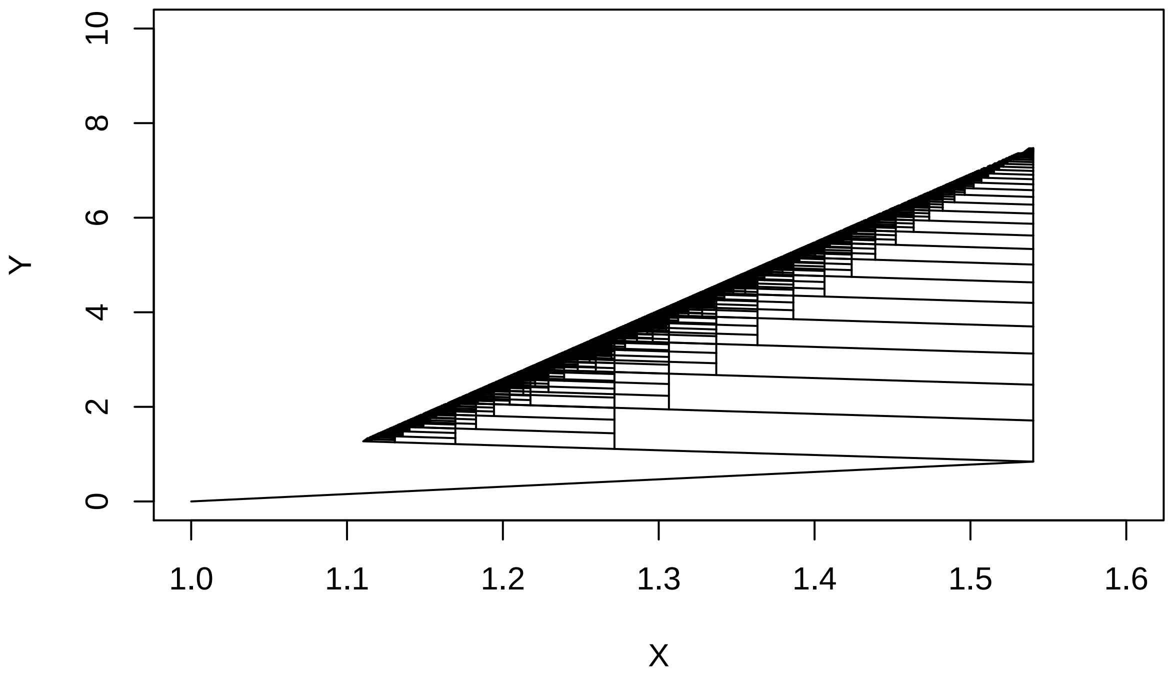




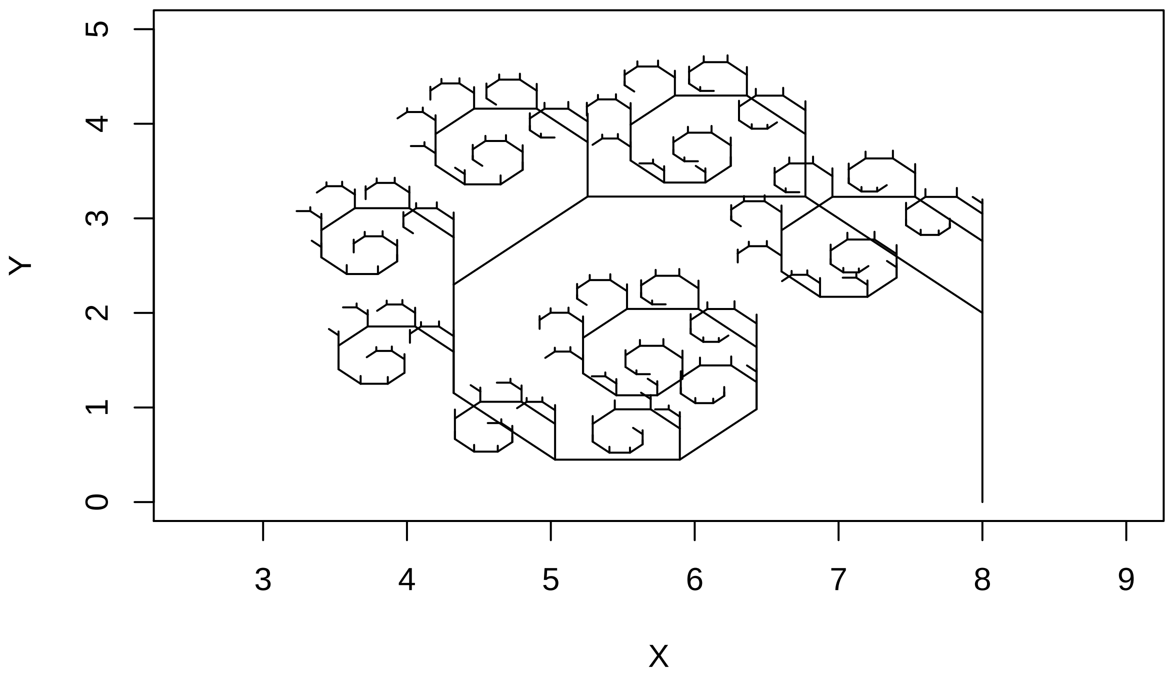
In this code the direction is always plus or minus pi/4



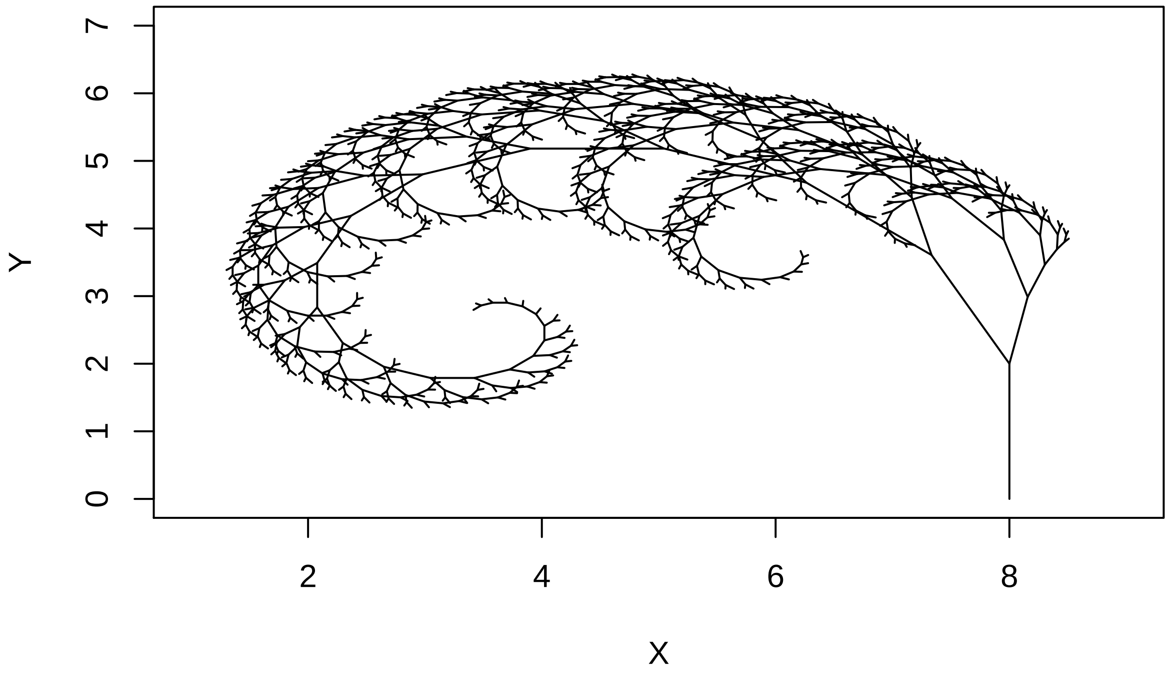
Here, the new direction is changed within each recursion so that the branches curl - i.e the new direction is added to the previous one.



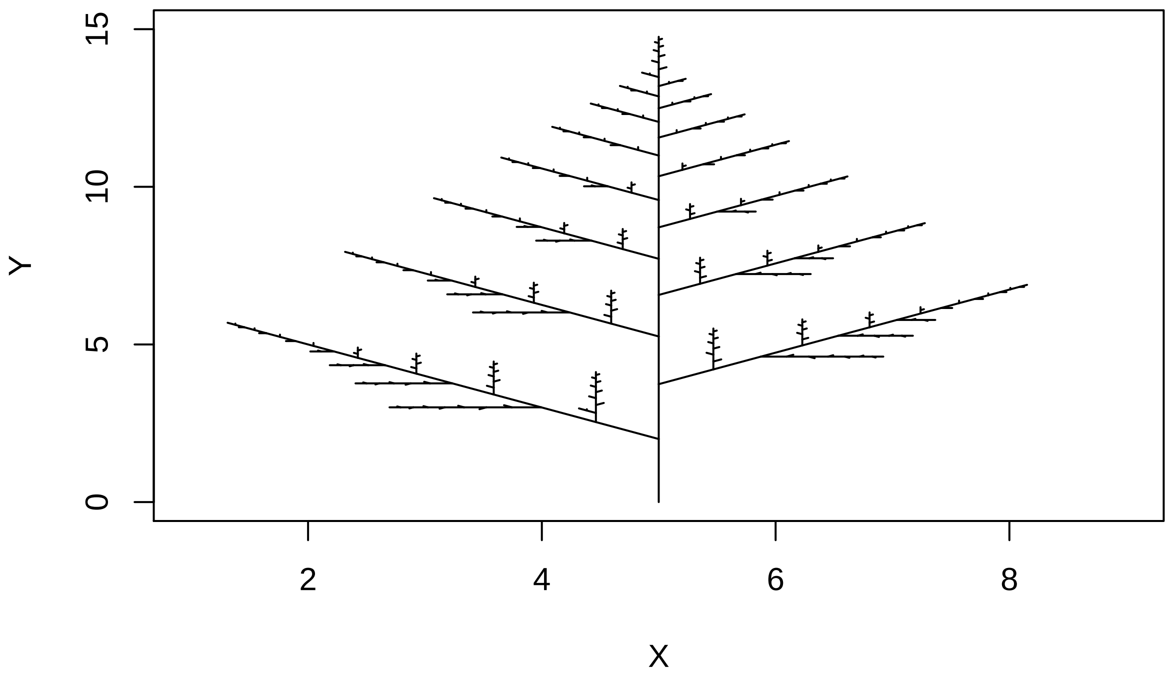
Like the first tree, the direction here is always 3pi/4 or pi/2 because they are hard coded into the function argument.



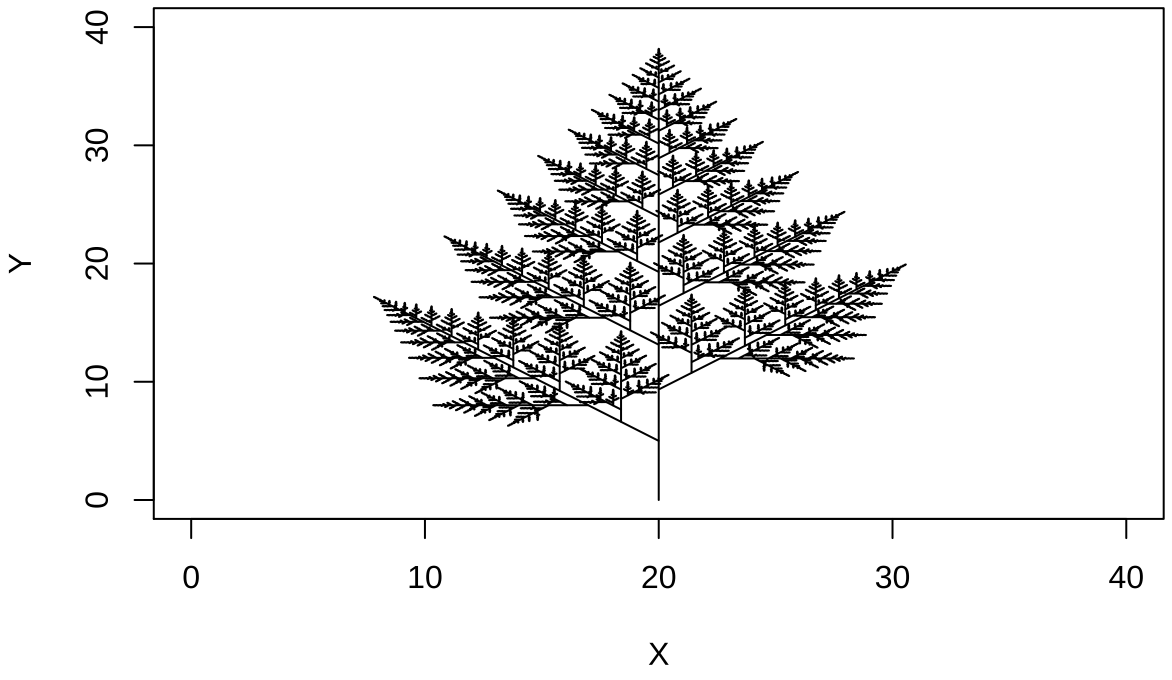
In this code the direction changes within each recursion giving the curled head of an opening dryopteris fern.



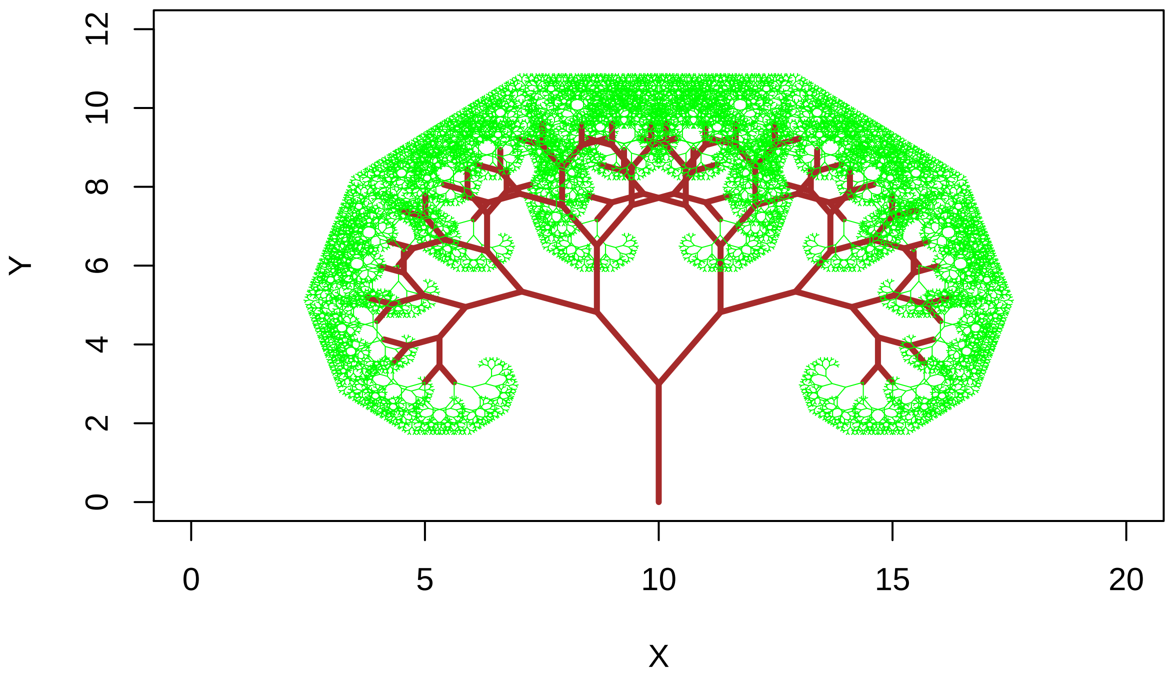
By slightly changing direction2 each time the fern gently curls, and by increasing the multiplier for the length we get more iterations for the limit length.



Here the dir*-1 command changes the direction of the plotting.



By adding a factor tht increases the size of the lateral branches, so that more occur befoer the limiting size, the fern can be made bushier.



Adding some lines to turtle to use different colours in segments depending on the length of the line