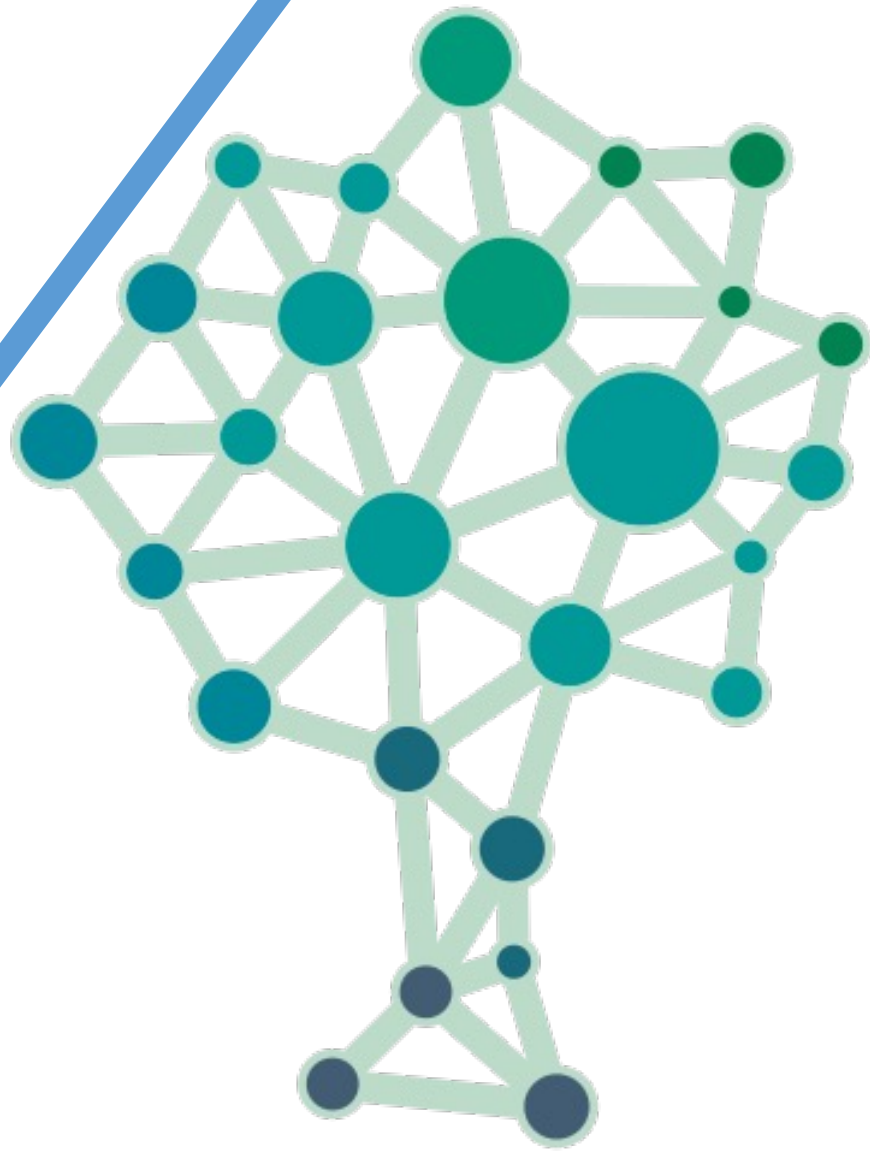
Introduction to Biological Computing

Computational Ecology; Schedule; the R Environment



Dr Josh Hodge

jhodge@ic.ac.uk

About Me

First generation student
Working class

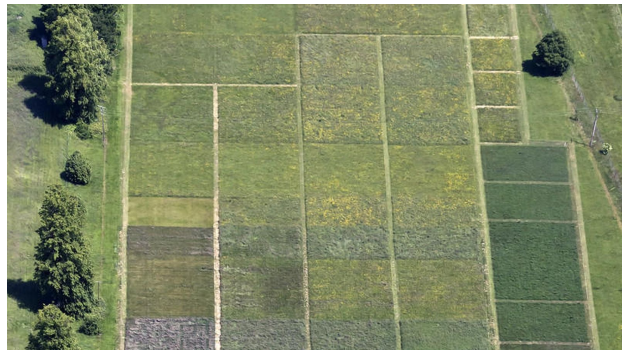
BSc Biological Sciences
Project: Fisheries in the South China Sea

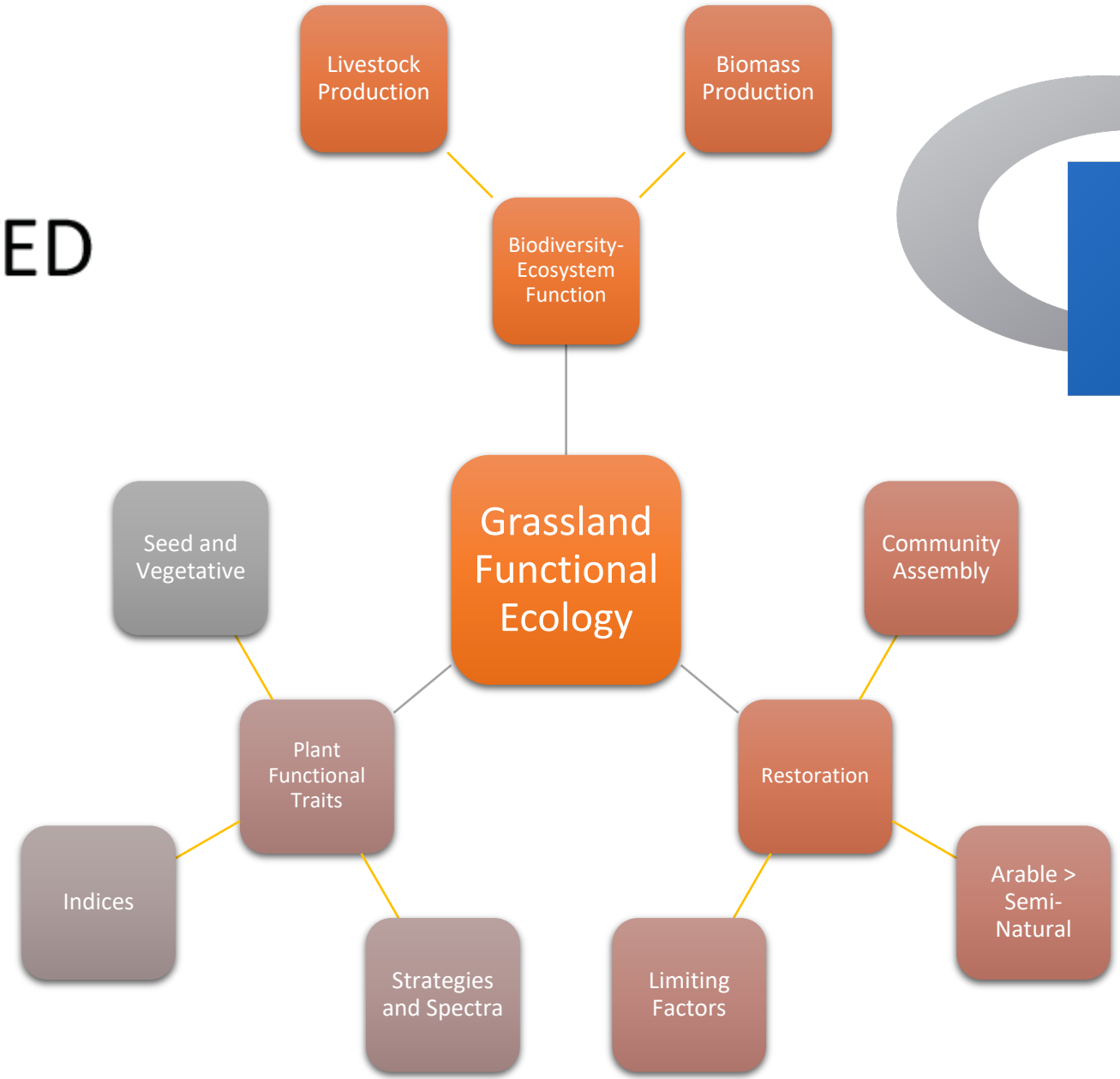
MSc Environmental Bioscience
Project: Biodiversity Offsetting

PhD Plant and Environmental Science

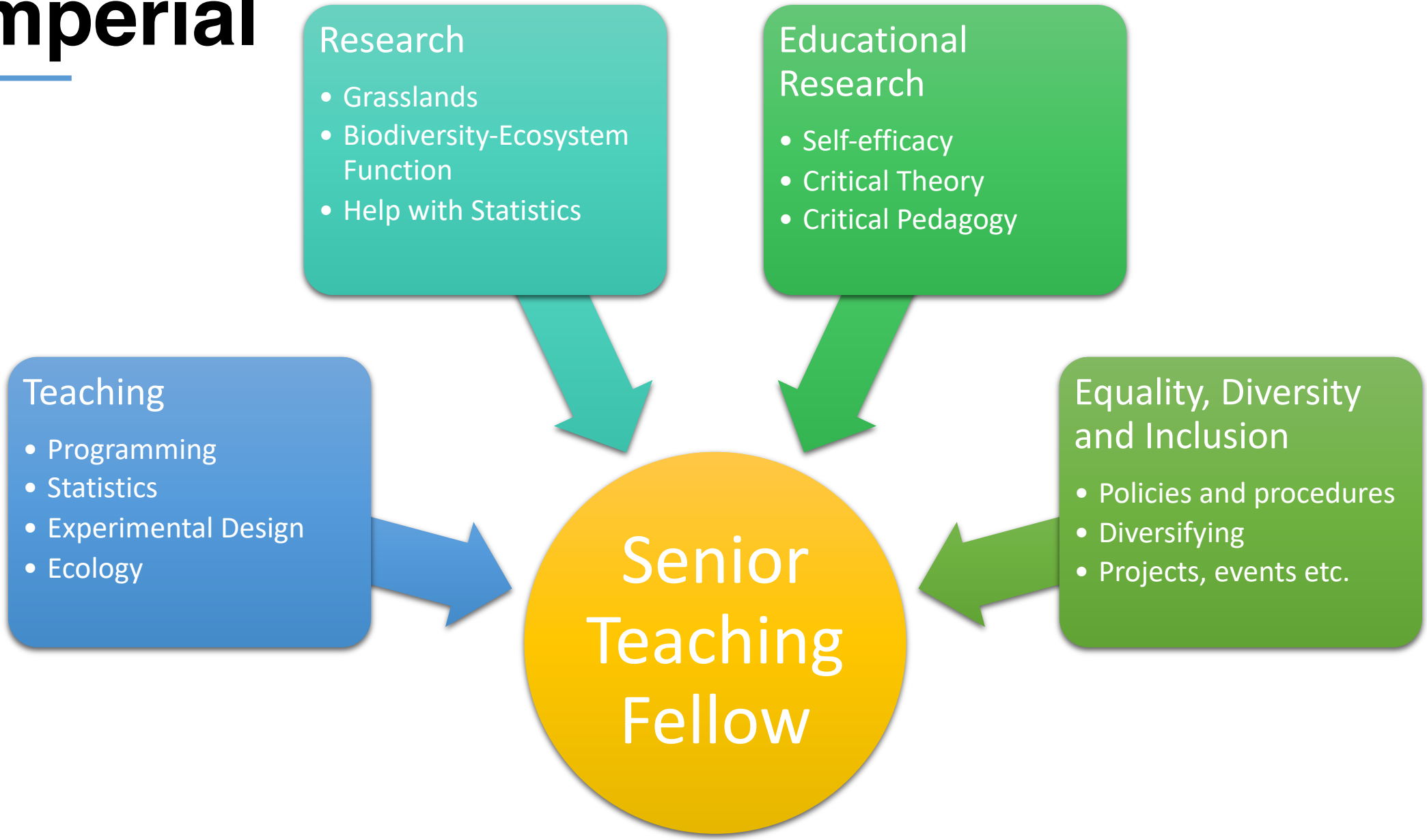
PhD Life

ROTHAMSTED RESEARCH





Imperial


Research
• Grasslands
• Biodiversity-Ecosystem Function
• Help with Statistics
Educational Research
• Self-efficacy
• Critical Theory
• Critical Pedagogy
Teaching
• Programming
• Statistics
• Experimental Design
• Ecology
Equality, Diversity and Inclusion
• Policies and procedures
• Diversifying
• Projects, events etc.
Senior Teaching Fellow

The Three Ecologists

- The Theoretician/Modeler:

"Using observations and computations to d
laws by which ecological interactions occu

- The Empiricist:

"Using the natural conditions in the field to
modeler's laws and make observations"

- The Conservationist:

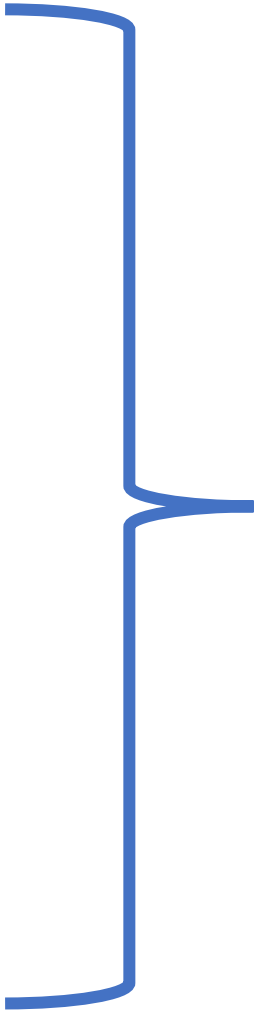
"Using the theories together with the field experiments to manage the biological resources and diversity"

Mentimeter

menti.com
and enter the code
2647 4438

Module Structure/Topics

- Biological Computing in R
  - 17th Oct – 21st Oct
  - Me
- Statistics in R
  - 24th Oct – 28th Oct
  - Me and Dr Julia Schroeder
- Spatial Analyses and GIS in R
  - 31st Oct - 4th Nov
  - Dr David Orme
- Genomics and Bioinformatics
  - 7th Nov – 11th Nov
  - Dr Mike Tristem



Coursework assessment (if MSc)

Intended Learning Outcomes

By end this module, you will be able to:

- Plan and justify the analysis or analyses to answer your research question(s).
- Write scripts to perform statistical, spatial and/or genomic analyses.
- Generate and interpret the results of statistical, spatial and/or genomic analyses.
- Write the results of your analyses to in a publication format.


Biological Computing in R

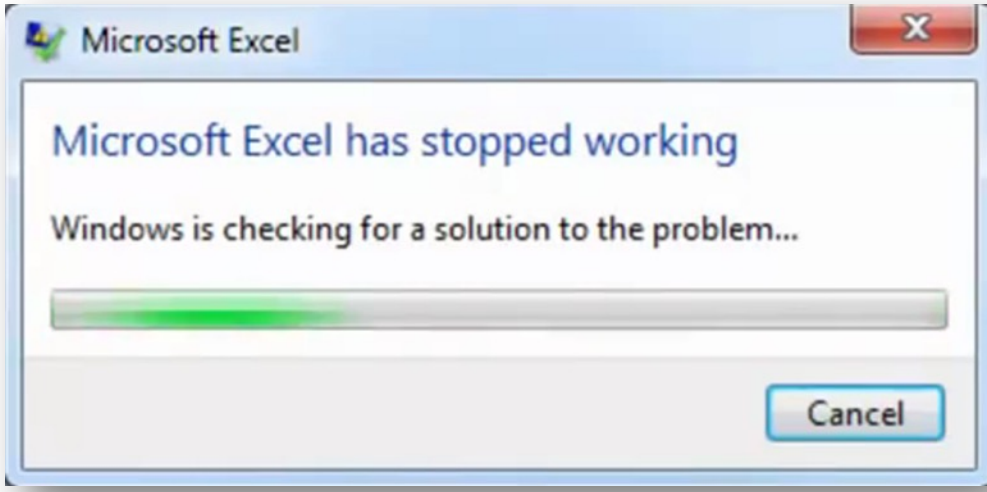
Intended Learning Outcomes

By end this week/topic, you will be able to:

- Navigate the R environment
- Perform basic commands to import, process and export data
- Write reproducible scripts
- Load and execute functions from various packages

Why do we program?

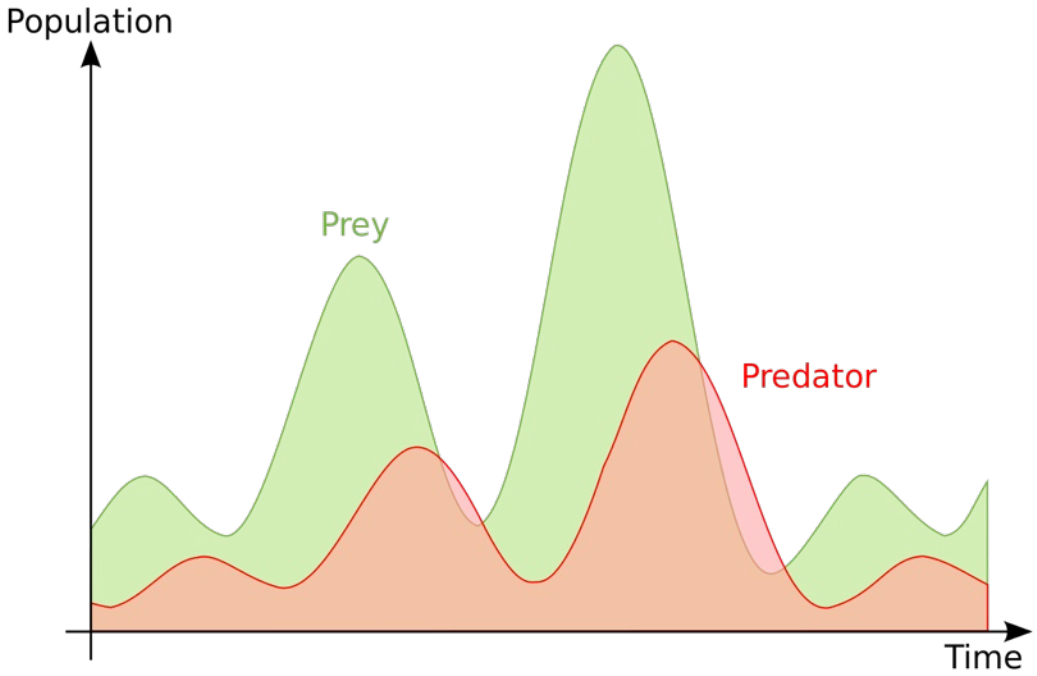
- Save time
- Generate data
- Necessity for large datasets



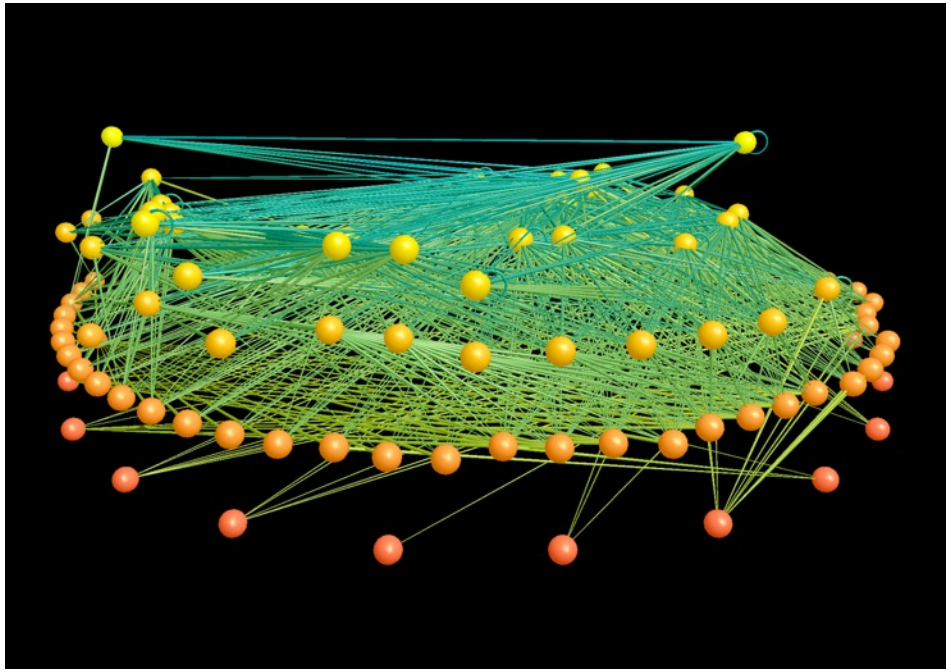
- Important for computational biology and ecology

Computational Ecology

- Lotka and Volterra (predator-prey interactions).



- Consumer-resource interactions embedded in larger ecological network.



Network of Tropic Interactions for Little Rock Lake, Wisconsin.

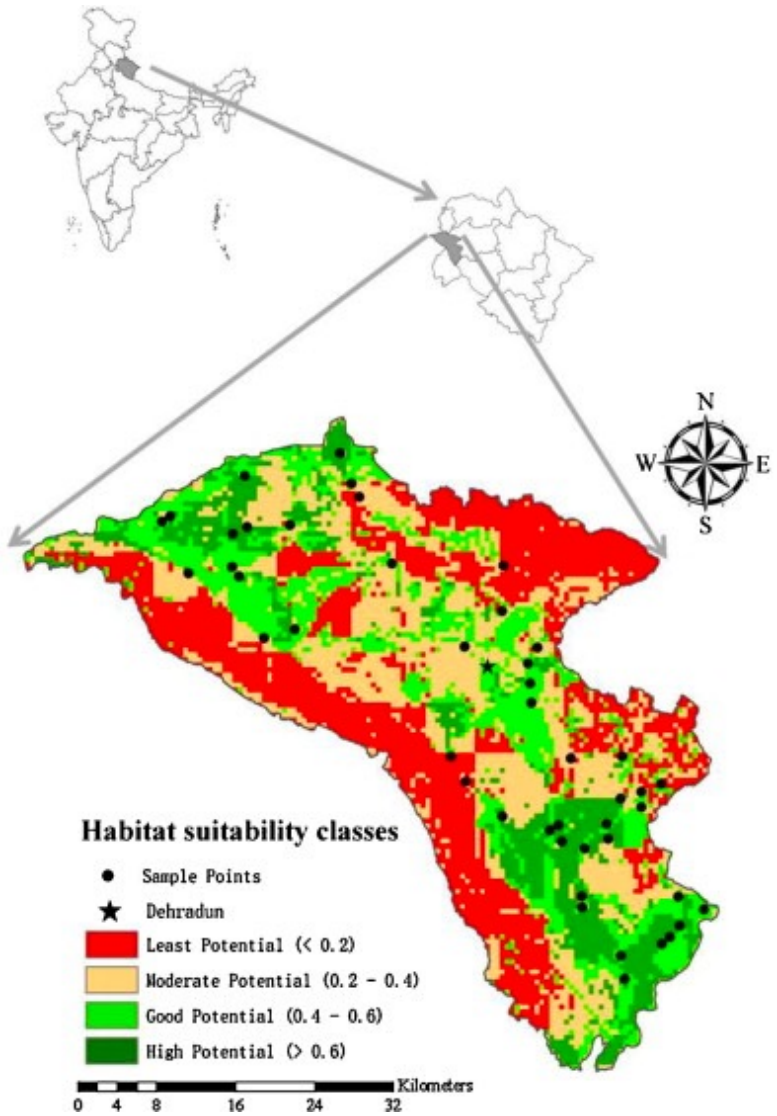
Pascual, M. 2005. Computational Ecology: From the Complex to the Simple and Back.

Distribution Models

Species Occurrences

+

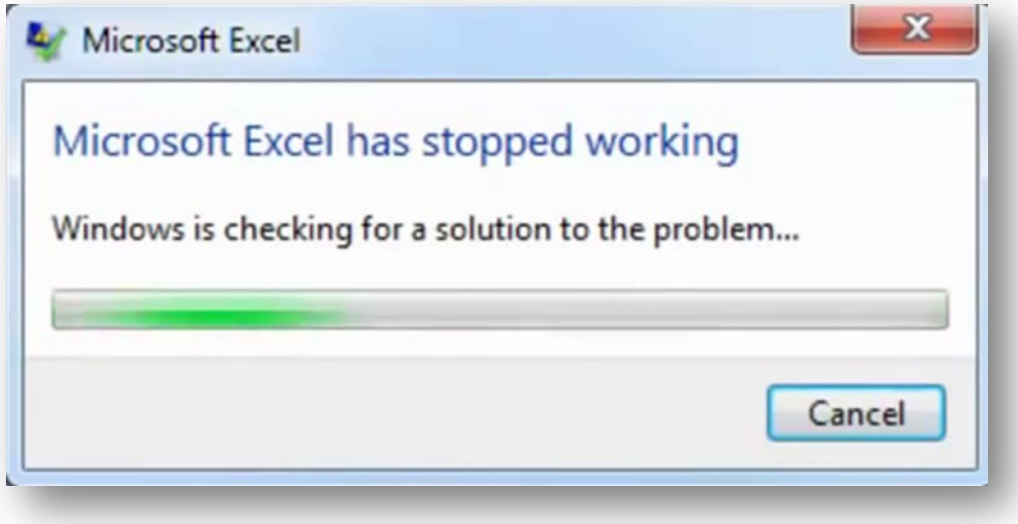
Environment



Yang, X-Q., et al. 2013. Maxent modelling for predicting the potential distribution of medicinal plant, Justicia adhatoda L. in Lesser Himalayan foothills.

Why do we learn to program?

- Save time
  - Generate data
- Necessity for large datasets
- Grounding in computational biology
- **EMPLOYABILITY**



Coding Languages

Employers

The R Environment

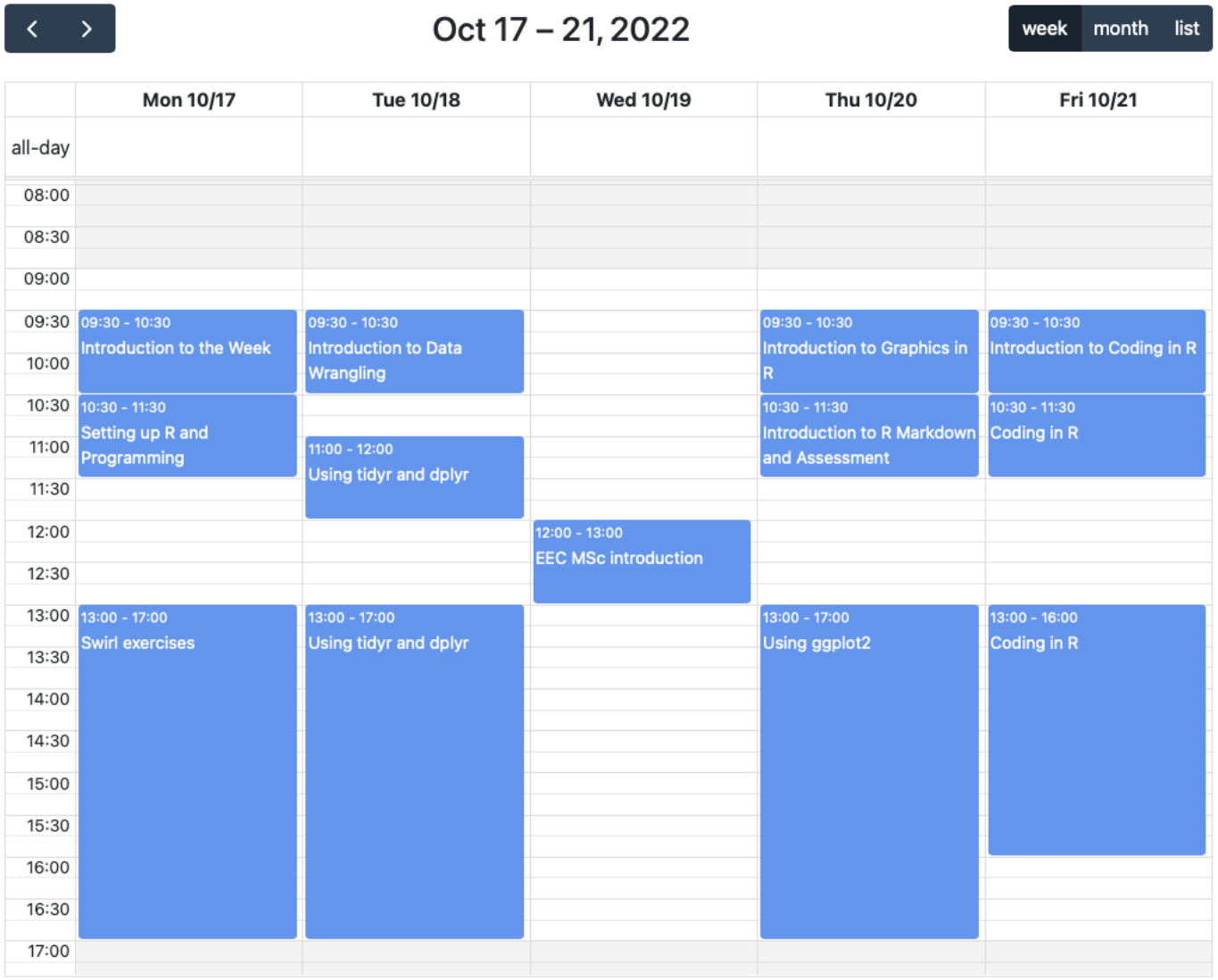
- Freely available statistical software
- Many packages for:
  - Data-handling
  - Processing
  - Analyses
  - Graphing and Visualization
- Scriptable


at first I was like...
...but now it's like...
Horst '18

Schedule

- Lectures in the morning and practicals in the afternoon
- Monday: swirl exercises to get comfortable with the R programming language
- Tuesday: Data Wrangling in R using the tidyverse
- Wednesday: Commemoration Day
- Thursday: Plotting in R using ggplot2
- Friday: Coding in R

Schedule



[Timetable]

Base R
Cheat Sheet

Getting Help

Accessing the help files

?mean
Get help of a particular function.
help.search('weighted mean')
Search the help files for a word or phrase.
help(package = 'dplyr')
Find help for a package.

More about an object

str(iris)
Get a summary of an object's structure.
class(iris)
Find the class an object belongs to.

Using Packages

install.packages('dplyr')
Download and install a package from CRAN.

library(dplyr)
Load the package into the session, making all its functions available to use.

dplyr::select
Use a particular function from a package.

data(iris)
Load a built-in dataset into the environment.

Working Directory

getwd()
Find the current working directory (where inputs are found and outputs are sent).

setwd('C://file/path')
Change the current working directory.

Use projects in RStudio to set the working directory to the folder you are working in.

Vectors

Creating Vectors

c(2, 4, 6)	2 4 6	Join elements into a vector
2:6	2 3 4 5 6	An integer sequence
seq(2, 3, by=0.5)	2.0 2.5 3.0	A complex sequence
rep(1:2, times=3)	1 2 1 2 1 2	Repeat a vector
rep(1:2, each=3)	1 1 1 2 2 2	Repeat elements of a vector

Vector Functions

sort(x)
Return x sorted.
rev(x)
Return x reversed.
table(x)
See counts of values.
unique(x)
See unique values.

Selecting Vector Elements

By Position

x[4]	The fourth element.
x[-4]	All but the fourth.
x[2:4]	Elements two to four.
x[-(2:4)]	All elements except two to four.
x[c(1, 5)]	Elements one and five.

By Value

x[x == 10]	Elements which are equal to 10.
x[x < 0]	All elements less than zero.
x[x %in% c(1, 2, 5)]	Elements in the set 1, 2, 5.

Named Vectors

x['apple']	Element with name 'apple'.

Programming

For Loop

```
for (variable in sequence){
   Do something
}
```

Example

```
for (i in 1:4){
   j <- i + 10
   print(j)
}
```

While Loop

```
while (condition){
   Do something
}
```

Example

```
while (i < 5){
   print(i)
   i <- i + 1
}
```

If Statements

```
if (condition){
   Do something
} else {
   Do something different
}
```

Example

```
if (i > 3){
   print('Yes')
} else {
   print('No')
}
```

Functions

```
function_name <- function(var){
   Do something
   return(new_variable)
}
```

Example

```
square <- function(x){
   squared <- x*x
   return(squared)
}
```

Reading and Writing Data

Also see the **readr** package.

Input	Ouput	Description
df <- read.table('file.txt')	write.table(df, 'file.txt')	Read and write a delimited text file.
df <- read.csv('file.csv')	write.csv(df, 'file.csv')	Read and write a comma separated value file. This is a special case of read.table/ write.table.
load('file.RData')	save(df, file = 'file.Rdata')	Read and write an R data file, a file type special for R.

Conditions

a == b	Are equal	a > b	Greater than	a >= b	Greater than or equal to	is.na(a)	Is missing
a != b	Not equal	a < b	Less than	a <= b	Less than or equal to	is.null(a)	Is null

Doctors: Googling stuff online does not make you a doctor.

Programmers: