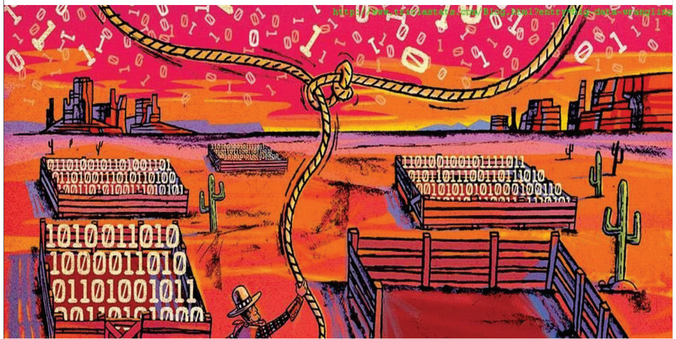
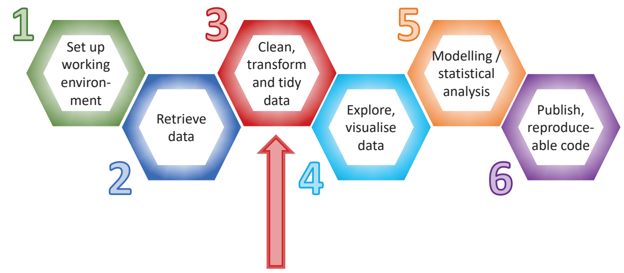
Palaeodata wrangling with the Tidyverse



Steve Juggins

Newcastle University

Palaeodata Workflow



Not sexy, not clever, not always straightforward, and can be very time consuming.
Having a good set of tools in R can:
- Increase efficiency
- Reduce errors
- Self-documenting & reproducible

Based on Fig 3, Slater et al. 2019. Using R in hydrology: a review of recent developments and future directions. Hydrology and Earth System Sciences **23:2939-2963.**

Guiding principles

- **Everything** is possible in **Base R** but solutions may be ad-hoc and need a mix functions & packages with different syntax: the Wild West that is Base R!
- **Tidyverse** offers a consistent approach to data wrangling, fast, concise & logical syntax, and very flexible.
- Some things may be **faster** in Base R but speed is rarely a limiting factor.
- **Focus** on code that is simple, readable and easy to understand (promote learning, easier debugging, and reproducibility).
- **Today** - don't get bogged down in the details of individual functions and operations. Focus on the big picture - the details will come with practice.

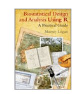
Resources

Whickham & Grolemund, R for Data Science,
free online https://r4ds.had.co.nz/

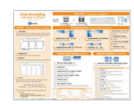
Locke, R Fundamentals 2: Data Manipulation (£11)
https://itsalocke.com/company/books/

Bruno Rodrigues, Modern R with the tidyverse
free online https://b-rodrigues.github.io/modern_R/ or buy pdf for < £10

Murray Logan's website, excellent tutorials & book
https://www.flutterbys.com.au/stats/



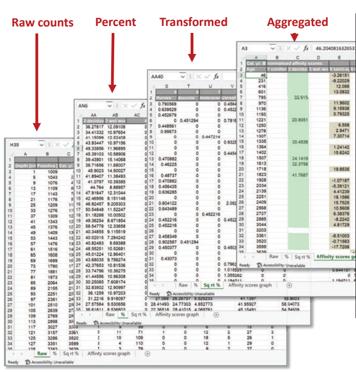
Rstudio cheatsheets
https://www.rstudio.com/resources/cheatsheets/



stackoverflow + 1001 other websites…

Do you need the tidyverse?

Do you have many separate files / sheets of the same data? (counts, percentages, subsets, aggregated)



If yes, then you should aim to have only one file and do all processing in R.

- More **efficient** (fewer files to keep track of)
- Reduces **errors** (and easier to correct errors and update data – only 1 file to change)
- **Reproducible** (record of what you did and why)

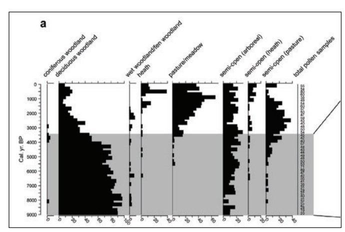
Example: Impact of the Neolithic Agricultural Transition

Jessie Woodbridge, Ralph Fyfe & colleagues
Woodbridge et al. (2014) & Fyfe et al. (2010)

Fyfe, et al. 2010. A pollen-based pseudobiomisation approach to anthropogenic land-cover change. The Holocene **20:1165-1171.**

Woodbridge, et al. 2014. The impact of the Neolithic agricultural transition in Britain: a comparison of pollen-based land-cover and archaeological 14C date-inferpoll population change. Journal of Archaeological Science **51:216-224.**

Aim: Reconstruct anthropogenic land-cover change in Britain over the last 9000 years



Land-cover reconstructions for Britain
Figure 3 from Woodbridge et al. (2014)

Thanks to Jessie Woodbridge and Ralph Fyfe for sharing their data!

Day 1 aims

Morning

Aims: To reproduce Woodbridge et al. Figure 3 using Tidyverse methods & explore changes in palynological richness.

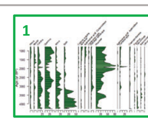
1. Work through analysis of a single site using base R
2. Repeat using Tidyverse
3. Apply to all (41) sites & create synthesis figure
4. Modelling species richness with the Tidyverse

Afternoon

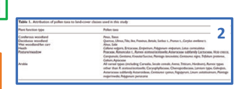
- Demonstration of my new package `riojaPlot` for plotting stratigraphic diagrams
- Work with your own data, ask questions, get help

Example: Impact of the Neolithic Agricultural Transition

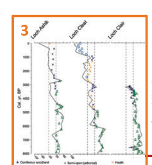
1. Extract dated pollen profiles from European pollen database (+ other sources), 41 sites in total.



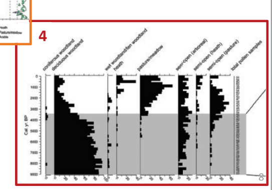
2. Classify each pollen type to a land cover class (LCC)



3. For each site, transform species data to major vegetation types (land cover class: LCC) and calculate the dominant LCC for each sample.

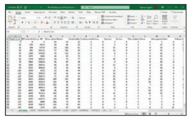


4. Sum dominant LCCs for all sites in 200 year time slices, convert to % of each LCC & plot.



The data

1. **Woodbridge_et_al_2014_Data.xlsx**: Excel file with pollen counts for 42 sites, each site on a separate tab. Columns for pollen types, Depth and Cal. Age BP + alternative chronologies, sample IDs & and pollen sum.



2. **LCC_Info.xlsx**: Excel file with the following worksheets:

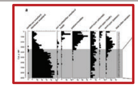
LCC_lookup: Lookup table of taxon names and corresponding land cover classes.

	A	B	C	D	E
1	VarName	LCC_ID	LCC_name	LCC_group	P_Group
2	Alnus	3	Wet/fen woodland	A	TRSH
3	Alnus glutinosa	3	Wet/fen woodland	A	TRSH
4	Alnus glutinosa-type	3	Wet/fen woodland	A	TRSH
5	Apiaceae	6	Pasture/meadow	C	HERB
6	Apiaceae u	6	Pasture/meadow	C	HERB
7	Apiaceae undiff.	6	Pasture/meadow	C	TRSH
8	Asteraceae	7	Arable indicators	C	HERB
9	Asteraceae subf. Asteroideae	7	Arable indicators	C	HERB
10	Asteraceae subf. Asteroideae cf. Tussilago	7	Arable indicators	C	HERB
11	Asteraceae subf. Asteroideae undiff.	7	Arable indicators	C	HERB
12	Asteraceae subf. Cichorioideae	6	Pasture/meadow	C	HERB
13	Asteraceae subf. Cichorioideae undiff.	6	Pasture/meadow	C	HERB
14	Asteraceae undiff.	6	Pasture/meadow	C	HERB
15	Asteroidea	7	Arable indicators	C	HERB
16	Avena/Triticum	7	Arable indicators	C	HERB
17	Avena/Triticum-type	7	Arable indicators	C	HERB
18	Avena/Triticum t.	7	Arable indicators	C	HERB

Site_info: Information about each site, including region, grid references etc.

Site_no	Site_code	Site_name	Region	Region2	Longit	Latitud	Site_t	Reference
1	ANFSHEIL	Allt na Feithe	East Scotland	Scotland	-3.7	56.23	Bog	Birks, H.H. 19
21	ASHIK	Loch Ashik	North Scotla	Scotland	-5.8	57.25	Small	Birks, H.J.B.,
2	BREDCOUN	Brede Bridge	Southeast En	England	-0.6	50.93	Mediu	Waller, M.P.
5	CHROISGP	Loch a'Chrois	North Scotla	Scotland	-5.3	57.57	Mediu	Pennington,
26	CLAIR	Loch Clair	North Scotla	Scotland	-5.3	57.56	Large	Waller, M.P.

Land Cover Classification (LCC) method



Step1: Calculate LCC for each site

1. Import data
2. Clean data - remove unwanted columns
3. Rename columns
4. Convert counts to percentages
5. Allocate each pollen type to an LCC class and sum sqrt-percentages across each class
6. Normalise the sqrt-percent LCC data to 100%
7. Identify the dominant LCC class at each level

Step2: Aggregate multiple sites

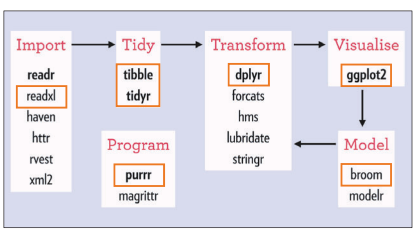
8. Aggregate data from all sites and count the number of levels in each LCC class in 200 year time slices (N)
9. Convert N to percentage of each LCC class in each time slice
10. Plot the aggregated data

Part 1: Code Step 1 in Base R

Palaeodata wrangling with the tidyverse

What is the Tidyverse?

- Collection of R packages designed for data science that share common interface standards, grammar and data structures
- You spend more time on concepts and less sorting out syntax
- Promotes code readability



- Both **general purpose** packages: tibble for data frames, tidyr for tidying, dplyr for transforming and ggplot for visualisation, and **specialised** packages for data import (readxl), dates & time (lubridate, hms), strings (stringr), factors (forcats) plus others.

Core features

1. Uses tibbles rather than data frames
   refined print method
   strict about $ subsetting, doesn't like rownames
   easier to created nested data frames

2. Encourages use of the pipe `%>%`
   Instead of `summary(lm(y~x, data=df))`
   Use `df %>% lm(y~x, data=.) %>% summary()`

3. Promotes use of tidy data
   Every column is a variable
   Every row is an observation
   Every cell is a single value

Tidy data

Not tidy !

(because the variable is pollen type and the attributes are taxon and count)

Depth (cm)	Cal. yr. BP	Pinus sylve	Taxus bacc	Betula	Betula/Co
132	1514	3	1	40	2
164	1618.5	3	3	33	1
196	1723	3	2	34	2
228	1827.5	4	4	33	1
260	1932	5	3	34	3
268	2143.75	4	3	43	0
276	2355.5	14	2	40	0
284	2567.25	11	1	60	0
292	2779	10	3	54	2

Wide format

Convert from wide to long with
`pivot_longer()`

```
df %>% pivot_longer(
   cols=-c(`Depth (cm)`, `Cal. yr. BP`),
   names_to="Taxon", values_to="Count")
```

Tidy !

(Each column is a variable, pollen counts are described by key/value pairs)

Depth (cm)	Cal. yr. BP	Taxon	Count
132	1514	Pinus sylvestris	3
132	1514	Taxus baccata	1
132	1514	Betula	40
132	1514	Betula/Corylus/Myrica	2
164	1618.5	Pinus sylvestris	3
164	1618.5	Taxus baccata	3
164	1618.5	Betula	33
164	1618.5	Betula/Corylus/Myrica	1
196	1723	Pinus sylvestris	3

Long format

Convert from long to wide with
`pivot_wider()`

```
df %>% pivot_wider(
   id_cols=c(`Depth (cm)`, `Cal. yr. BP`),
   names_from=Taxon, values_from=Count)
```

In package tidyr

dplyr functions for data transformation

dplyr function
Base R equivalent

select: select columns (subset)

filter: subsetting rows based on condition (subset)

arrange: sorting (sort, order)

distinct: find unique values (unique)

slice: selecting rows based on position ([])

mutate: create new columns (transform)

group_by: defining groups of rows to process subsets

summarise: summarise data (optionally by group) (aggregate)

*_join: merging data sets (merge)

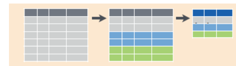
bind_rows, bind_cols: combine multiple dfs by row or column (rbind, cbind)

Creating new variables using mutate

Depth (cm)	Cal. yr. BP	Taxon	Count
132	1514	Pinus sylvestris	3
132	1514	Taxus baccata	1
132	1514	Betula	40
132	1514	Betula/Corylus/Myrica	2
164	1618.5	Pinus sylvestris	3
164	1618.5	Taxus baccata	3
164	1618.5	Betula	33
164	1618.5	Betula/Corylus/Myrica	1
196	1723	Pinus sylvestris	3

```
df %>%
   mutate(SQRT_Count=sqrt(Count))
A tibble: 9 x 5
  `Depth (cm)` `Cal. yr. BP` Taxon                 Count SQRT_Count
         <dbl>         <dbl> <chr>                 <dbl>      <dbl>
1          132          1514  Pinus sylvestris         3       1.73
2          132          1514  Taxus baccata            1       1
3          132          1514  Betula                  40       6.32
4          132          1514  Betula/Corylus/Myrica    2       1.41
5          164          1618. Pinus sylvestris         3       1.73
6          164          1618. Taxus baccata            3       1.73
7          164          1618. Betula                  33       5.74
8          164          1618. Betula/Corylus/Myrica    1       1
9          196          1723  Pinus sylvestris         3       1.73
```


Summarising by group



Split -> Apply -> Combine

Depth (cm)	Cal. yr. BP	Taxon	Count
132	1514	Pinus sylvestris	3
132	1514	Taxus baccata	1
132	1514	Betula	40
132	1514	Betula/Corylus/Myrica	2
164	1618.5	Pinus sylvestris	3
164	1618.5	Taxus baccata	3
164	1618.5	Betula	33
164	1618.5	Betula/Corylus/Myrica	1
196	1723	Pinus sylvestris	3

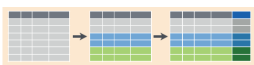
```
df %>% summarise(Total=sum(Count))

     Total
     <dbl>
1       89
```

```
df %>%
  group_by(`Depth (cm)`) %>%
  summarise(Total=sum(Count))

   `Depth (cm)` Total
          <dbl> <dbl>
1           132    46
2           164    40
3           196     3
```


Compute new variables by group



Depth (cm)	Cal. yr. BP	Taxon	Count
132	1514	Pinus sylvestris	3
132	1514	Taxus baccata	1
132	1514	Betula	40
132	1514	Betula/Corylus/Myrica	2
164	1618.5	Pinus sylvestris	3
164	1618.5	Taxus baccata	3
164	1618.5	Betula	33
164	1618.5	Betula/Corylus/Myrica	1
196	1723	Pinus sylvestris	3

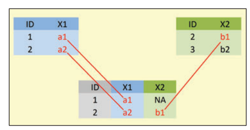
```
df %>%
   group_by(`Depth (cm)`) %>%
   mutate(Percent=Count/sum(Count)*100)

`Depth (cm)` `Cal. yr. BP` Taxon                 Count Percent
        <dbl>         <dbl> <chr>                 <dbl>   <dbl>
1         132          1514  Pinus sylvestris         3    6.52
2         132          1514  Taxus baccata            1    2.17
3         132          1514  Betula                  40   87.0
4         132          1514  Betula/Corylus/Myrica    2    4.35
5         164          1618. Pinus sylvestris         3    7.5
6         164          1618. Taxus baccata            3    7.5
7         164          1618. Betula                  33   82.5
8         164          1618. Betula/Corylus/Myrica    1    2.5
9         196          1723  Pinus sylvestris         3  100
```


Merging tables

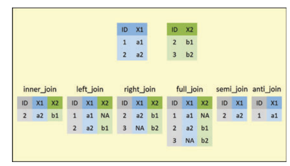
Join or merge tables using column(s) common to both tables

Left join – all rows from left df, matching rows from right




```
Df1 %>% left_join(df2, by='ID')
```

Other join types



Mutating joins: combine variables from the 2 sources

Filtering joins: use right hand df to filter rows in lh df

Figures from https://statisticsglobe.com/r-dplyr-join-inner-left-right-full-semi-anti

Working across rows using across()

Depth (cm)	Cal. yr. BP	Pinus sylve	Taxus bac	Betula	Betula/Co
132	1514	3	1	40	2
164	1618.5	3	3	33	1
196	1723	3	2	34	2
228	1827.5	4	4	33	1
260	1932	5	3	34	3
268	2143.75	4	3	43	
276	2355.5	14	2	40	
284	2567.25	11	1	60	
292		10	3	54	2

```
# Base R replace in situ
df[is.na(df)] <- 0

# Base R create new df
replace(df, is.na(df), 0)

# Tidyverse, replace NAs in a single column
df %>% mutate(Betula = replace_na(Betula, 0)))
```

Use `across()` to select, rename, or apply a transformation to multiple columns

```
across(.cols = everything(), .fns = NULL, ..., .names = NULL)

.cols = columns to transform
.fns = function to apply to each column
.names = how the new columns will be names (default is to use original names)
```

```
# Tidyverse, replace NAs except in cols 1 & 2
df %>% mutate(across(-(1:2), ~replace_na(.x, 0)))
```

placeholder for passed column

Part 2: Code it using the Tidyverse

Code comparison: Part 1

Base R

```
polldata <- read_excel("Woodbridge_et_al_2014_Data.xlsx",
sheet="REDMERE")
lcc_lookup <- read_excel("LCC_info.xlsx", sheet="LCC_Lookup")
lcc_lookup
non_pollen <- c("Sample","Radiocarbon years B.P.", "EPD default
[yrs.BP.]", "EPD [yrs.BP.]", "Fossilva [yrs.BP.]", "Sum")
del <- colnames(polldata) %in% non_pollen
polldata <- polldata[, !del]
depth_age <- subset(polldata, select=c(`Depth (cm)`, `Cal. yr. BP`))
poll_count <- subset(polldata, select=-c(`Depth (cm)`, `Cal. yr. BP`))
colnames(depth_age) <- c("Depth", "Age_BP")
poll_pc <- poll_count / rowSums(poll_count) * 100
poll_sqrt <- sqrt(poll_pc)
sel <- match(colnames(poll_sqrt), lcc_lookup$VarName)
taxa_lcc <- lcc_lookup[sel, ]
poll_lcc <- t (rowsum(t(poll_sqrt), group=taxa_lcc$LCC_name))
poll_lcc <- data.frame(poll_lcc, check.names=FALSE)
poll_norm <- poll_lcc / rowSums(poll_lcc) * 100
dominant_class <- apply(poll_norm, 1, which.max)
poll_norm$lcc_class <- factor(colnames(poll_norm)[dominant_class])
sum_arboreal <- rowSums(poll_norm[, c("Coniferous woodland",
                                      "Deciduous woodland",
                                      "Wet/fen woodland")])  # arboreal
sum_open <- rowSums(poll_norm[, c("Heath",
                                      "Pasture/meadow",
                                      "Arable indicators")])  # open
poll_norm$Affinity <- sum_arboreal - sum_open

plot(depth_age$Age_BP, poll_norm$Affinity, type="l",
     xlab="Age (years BP)",
     ylab="Affinity")
points(depth_age$Age_BP, poll_norm$Affinity, pch=19, cex=1,
col=as.integer(poll_norm$lcc_class))
legend("topleft", pch=19, col=1:7, legend=levels(poll_norm$lcc_class))
```

Tidyverse

```
polltidy <- read_excel("Woodbridge_et_al_2014_Data.xlsx", sheet="REDMERE")
lcc_lookup <- read_excel("LCC_info.xlsx", sheet="LCC_Lookup")
non_pollen <- c("Sample","Radiocarbon years B.P.", "EPD default [yrs.BP.]",
"EPD [yrs.BP.]", "Fossilva [yrs.BP.]", "Sum")
polltidy_long <- polldata %>%
  rename("Depth"=`Depth (cm)`, "Age_BP"=`Cal. yr. BP`) %>%
  pivot_longer(cols=-c("Depth", "Age_BP"),
               names_to="VarName", values_to="Count") %>%
  filter(!(VarName %in% non_pollen))
polltidy_sqrt <- polldata_long %>%
  group_by(Depth) %>%
  mutate(Percent = Count / sum(Count) * 100,
         SQRT_PC = sqrt(Percent)) %>%
  ungroup()
polltidy_norm <- polltidy_sqrt %>%
  left_join(lcc_lookup, by="VarName") %>%
  group_by(Depth, Age_BP, LCC_name, LCC_group) %>%
  summarise(SQRT_PC=sum(SQRT_PC), .groups="drop") %>%
  group_by(Depth, Age_BP) %>%
  mutate(Norm_SQRT_PC=SQRT_PC/sum(SQRT_PC)*100) %>%
  ungroup() %>%
  select(-SQRT_PC)
polltidy_lcc <- polltidy_norm %>%
  group_by(Depth, Age_BP) %>%
  mutate(A=sum(Norm_SQRT_PC[LCC_group=="A"]),
         C=sum(Norm_SQRT_PC[LCC_group=="C"]),
         Affinity=A-C) %>%
  select(-c(A, C)) %>%
  slice_max(Norm_SQRT_PC, n=1, with_ties=FALSE) %>%
  select(-c(Norm_SQRT_PC)) %>%
  ungroup()
ggplot(polltidy_lcc, aes(x=Age_BP, y=Affinity, col=LCC_name)) +
  geom_line(col="grey") +
  geom_point(size=2) +
  scale_x_continuous(breaks=seq(0, 10000, by=500)) +
  labs(x="Years BP") +
  scale_colour_discrete(name="LCC") +
  theme(legend.position="top") +
  guides(col=guide_legend(nrow=2))
```


Working with multiple datasets – base R

How to apply our LCC code for 1 site to many?

Use a for loop!

```
for (val in sequence) {
   statement
}
```

`sequence` is a vector and `val` takes on each of its values during the loop. In each iteration statement is `evaluated`.

```
x <- 1:5
for (i in x) {
    print(i)
}
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
```

Rewriting code as a function

- To apply our LCC function to multiple sites we can use a loop and collate results at each iteration.
- We could include all the LCC code within the loop but this is not good – makes the code more complex, and we should aim to reuse code, not repeat it.
- Encapsulate the code in a function, and call the function from within the loop.

Defining a function:

```
my_fun <- function(arg1, arg2, ...) {
   statements
   return(object)
}
```

Example:

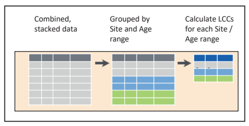
```
my_fun <- function(a, b) {
  c <- a + b
  return(c)
}
my_fun(2, 3)
[1] 5
```


Part 3: Code to aggregate multiple sites Base R

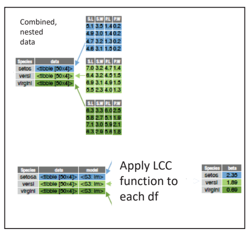
Working with multiple datasets – Tidyverse

Two approaches with Tidy functions that avoid loops and produce more readable, elegant code:

1. Convert each df to long format, and stack, and perform summaries / transformations by grouping of Site and Age/Depth



2. Convert to nested data frame that stores data of each site within the cell of a larger organising table. Apply a function to each nested table and collate results.



We use second approach to clean data and convert to long format and stack the data, then use first approach to calculate LCC from the stacked dataset.

Figures from tidyverse cheat sheets

Working with nested data using **purrr**

Useful functions:

tidyr::nest(df, cols): creates data frame with cols nested within grouped defined by the non-nesting columns.

tidyr:: unnest(df, cols): unnests a nested column col of a nested data frame.

purrr::map applies a function to each element of a list or vector (ie. each data frame of a nested data frame)

Working with nested data frames

Original data frame

```
iris
 A tibble: 150 x 5
   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
          <dbl>       <dbl>        <dbl>       <dbl> <fct>
 1          5.1         3.5          1.4         0.2 setosa
 2          4.9         3            1.4         0.2 setosa
 3          4.7         3.2          1.3         0.2 setosa
 4          4.6         3.1          1.5         0.2 setosa
 5          5           3.6          1.4         0.2 setosa
 6          5.4         3.9          1.7         0.4 setosa
 7          4.6         3.4          1.4         0.3 setosa
 8          5           3.4          1.5         0.2 setosa
 9          4.4         2.9          1.4         0.2 setosa
10          4.9         3.1          1.5         0.1 setosa
```

Nested data frame

```
n_iris <- iris %>% nest(data = -Species)
n_iris
 A tibble: 3 x 2
  Species    data
  <fct>      <list>
1 setosa     <tibble [50 x 4]>
2 versicolor <tibble [50 x 4]>
3 virginica  <tibble [50 x 4]>
```

List-column named "data"

Use `map` to apply function `summary` to each nested table

```
n_iris$data %>% map(summary)
[[1]]
  Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
 Min.   :4.300   Min.   :2.300   Min.   :1.000   Min.   :0.100
 1st Qu.:4.800   1st Qu.:3.200   1st Qu.:1.400   1st Qu.:0.200
 Median :5.000   Median :3.400   Median :1.500   Median :0.200
 Mean   :5.006   Mean   :3.428   Mean   :1.462   Mean   :0.246
 3rd Qu.:5.200   3rd Qu.:3.675   3rd Qu.:1.575   3rd Qu.:0.300
 Max.   :5.800   Max.   :4.400   Max.   :1.900   Max.   :0.600
[[2]]
 Sepal.Length    Sepal.Width     Petal.Length   Petal.Width
 Min.   :4.900   Min.   :2.000   Min.   :3.00   Min.   :1.000
```


Working with nested data frames

Apply base R function **summary** to each nested table and add results to original nested df

```
n_iris %>% mutate(summary = map(data, summary))
 A tibble: 3 x 3
  Species    data              summary
  <fct>      <list>            <list>
1 setosa     <tibble [50 x 4]> <table [6 x 4]>
2 versicolor <tibble [50 x 4]> <table [6 x 4]>
3 virginica  <tibble [50 x 4]> <table [6 x 4]>
```

Use formula version to supply arguments to function (in this case **lm**)

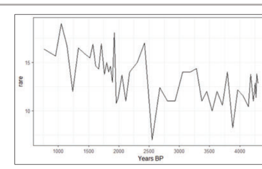
```
n_iris %>% mutate(model = map(data, ~lm(Sepal.Width~Sepal.Length, data=.x)))
 A tibble: 3 x 3
  Species    data              model
  <fct>      <list>            <list>
1 setosa     <tibble [50 x 4]> <lm>
2 versicolor <tibble [50 x 4]> <lm>
3 virginica  <tibble [50 x 4]> <lm>
```

Use ".x" to pass the df in the first argument to map to the mapped function

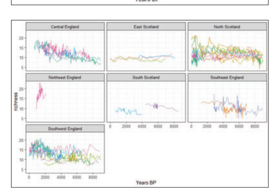
Part 4: Code to aggregate multiple sites using Tidyverse

Part 5: Modelling with the Tidyverse

- Calculate palynological richness using rarefaction



- Apply model to all sites



Very simple questions:
Which sites show significant trend in richness?
How does richness vary with landscape "openness"

broom for tidying model output

The problem

Tidy(ish) data ⇒ Model ⇒ Not tidy! No common format

The broom solution

Model ⇒ Convert to tidy with consistent output format for multiple model types (currently >130 included, easy to add your own).

tidy:	tibble that summarises model findings (e.g. coefficients, p-values)
glance:	concise one-row summary (e.g. r-squared, degf)
augment:	columns original data was modelled on (e.g. predictions, residuals)

Afternoon (13:30-15:30): Bring your own data