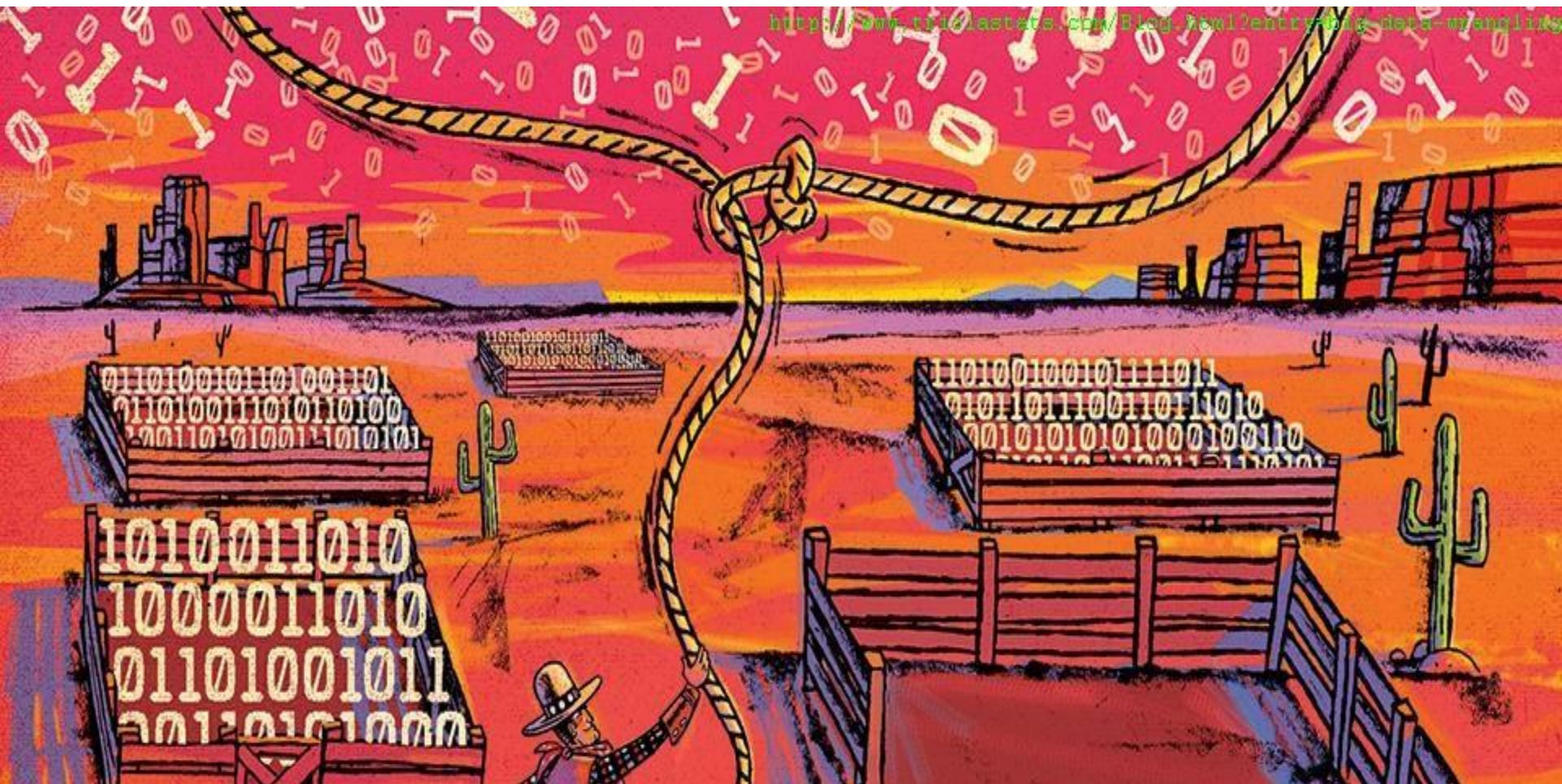
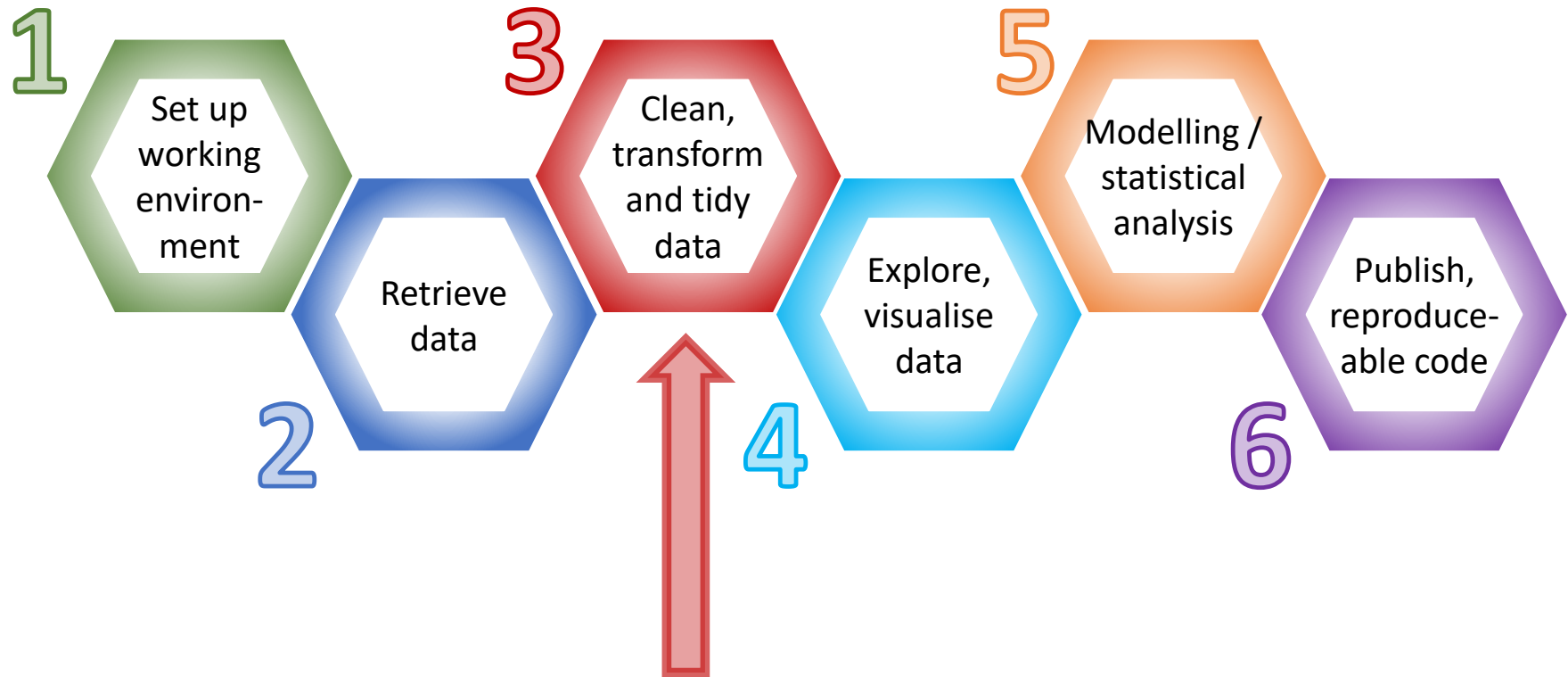


Palaeodata wrangling with the Tidyverse



Steve Juggins

Palaeodata Workflow



Not sexy, not clever, not always straightforward, and can be very time consuming.

Having a good set of tools in R can:

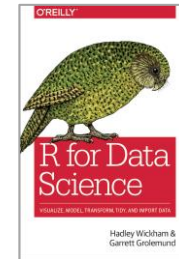
- Increase efficiency
- Reduce errors
- Self-documenting & reproducible

Guiding principles

- Everything is possible in **Base R** but solutions may not be obvious, use a mix functions with different syntax, and require ad-hoc packages: 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 (things that promote learning, easier debugging, reproducibility and code sharing).
- 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

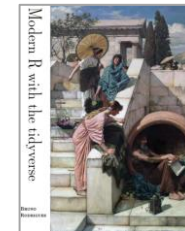
Whickham & Grolemund, R for Data Science,
free online <https://r4ds.had.co.nz/>



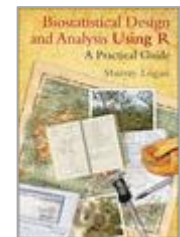
Locke, R Fundamentals 2: Data Manipulation (£11)
<https://itsalocke.com/company/books/>



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



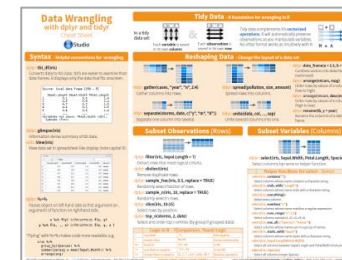
Murray Logan's website, excellent tutorials & book
<https://www.flutterbys.com.au/stats/>



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



+ 1001 other websites...



Example: Impact of the Neolithic Agricultural Transition

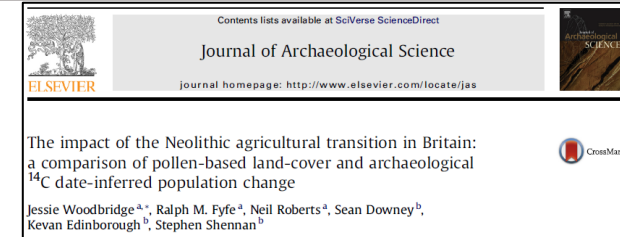
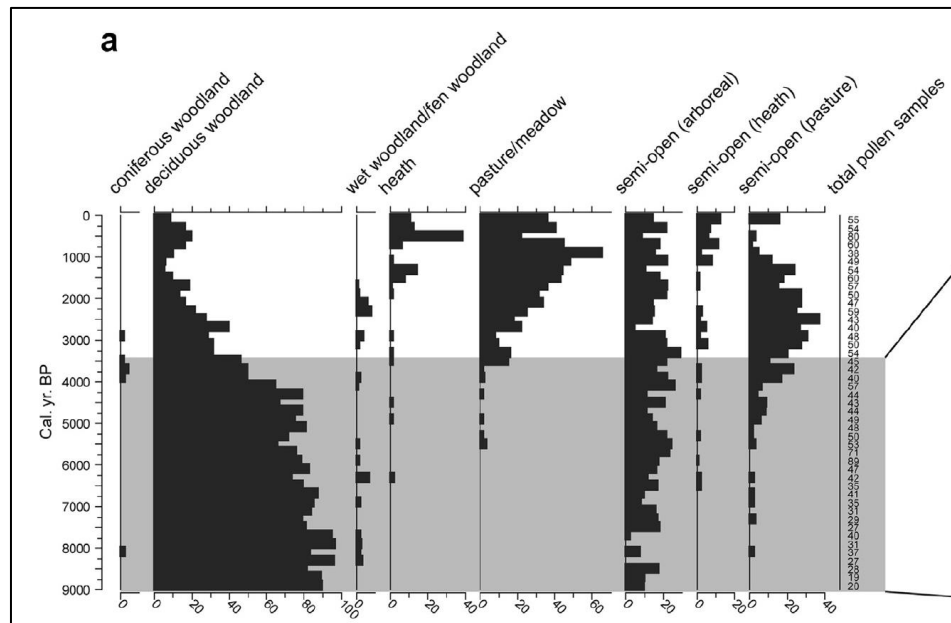
Jessie Woodbridge, Ralph Fyfe & colleagues

Woodbridge et al. (2014) & Fyfe et al. (2010)

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)



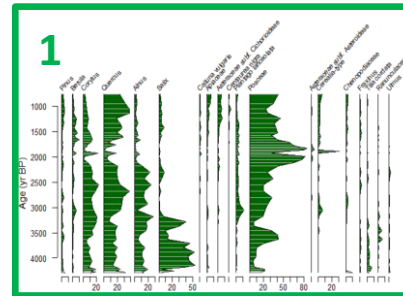
Fyfe, et al. 2010. A pollen-based pseudobiomisation approach to anthropogenic land-cover change. The Holocene **20**:1165-1171.

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

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

Example: Impact of the Neolithic Agricultural Transition

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

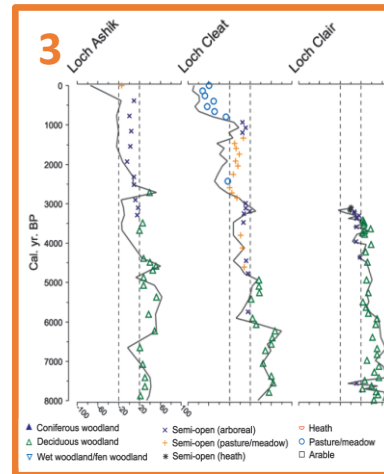


2. Classify each pollen type to a land cover class (LCC)

Table 1. Attribution of pollen taxa to land-cover classes used in this study

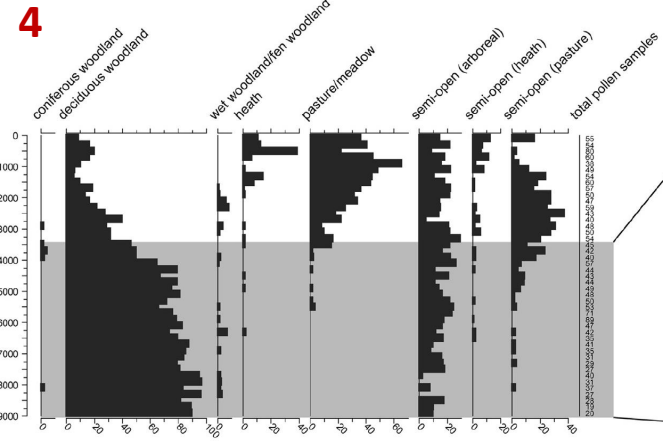
Plant function type	Pollen taxa
Coniferous woodland	<i>Pinus</i> , <i>Taxus</i>
Deciduous woodland	<i>Quercus</i> , <i>Ulmus</i> , <i>Tilia</i> , <i>Ilex</i> , <i>Fraxinus</i> , <i>Betula</i> , <i>Sorbus</i> s., <i>Prunus</i> s., <i>Corylus avellana</i> s.
Wet woodland/fen carr	<i>Alnus</i> , <i>Salix</i>
Heath	<i>Calluna vulgaris</i> , <i>Ericaceae</i> , <i>Empetrum</i> , <i>Polygonum viviparum</i> , <i>Lotus corniculatus</i>
Pasture/meadow	<i>Poa</i> spp., <i>Rumex acetosella</i> , <i>Asteraceae</i> subfamily <i>Lactucaceae</i> , <i>Vicia cracca</i> , <i>Campylosiphia</i> , <i>Geranium</i> , <i>Knautia</i> , <i>Succisa</i> , <i>Plantago lanceolata</i> , <i>Centaurea nigra</i> , <i>Trifolium pratense</i> , <i>Gallium</i> , <i>Apocynaceae</i>
Arable	All cereal types (including <i>Cerealia</i> , <i>Secale cereale</i> , <i>Avena</i> , <i>Triticum</i> , <i>Hordeum</i>), <i>Rumex</i> types other than <i>R. acetosella</i> , <i>Caryophyllaceae</i> , <i>Chenopodiaceae</i> , <i>Lamiaceae</i> type, <i>Galeopsis</i> , <i>Asteraceae</i> subfamily <i>Asteraceae</i> , <i>Centaurea cyanea</i> , <i>Fagopyrum</i> , <i>Linum catharticum</i> , <i>Plantago major/minor</i> , <i>Polygonum persicaria</i>

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



Fyfe et al. (2010)

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



Day 1 aims

Morning

Aims: To reproduce Woodbridge et al. Figure 3 using Tidyverse methods & do some modelling (richness, rate-of-change).

1. Work through analysis of a single site using base R
2. Repeat using Tidyverse
3. Apply to all (42) sites & create synthesis figure
4. Modelling with the Tidyverse

Afternoon

- Work with your own data, ask questions, get help

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_Taxon_List: Lookup table of taxon names and corresponding land cover classification (numeric code, 1-7).

VarName	LCC_ID	P_Group
Alnus	3	TRSH
Alnus glutinosa	3	TRSH
Alnus glutinosa-type	3	TRSH
Apiaceae	6	HERB
Apiaceae u	6	HERB
Apiaceae undiff.	6	TRSH
Asteraceae	7	HERB
Asteraceae subf. Asteroideae	7	HERB

LCC_Taxon_Classes: Lookup table of LCC taxon class and names (3 woodland, 3 open classes).

LCC_ID	LCC_name	LCC_group
1	Coniferous woodland	A
2	Deciduous woodland	A
3	Wet/fen woodland	A
4	Unused	B
5	Heath	C
6	Pasture/meadow	C
7	Arable indicators	C

LCC2_Assemblage_Classes: Lookup table of LCC2 assemblage classes and names (3 woodland, 3 open, 4 semi-open classes).

LCC2_ID	LCC2_name
1	Coniferous woodland
2	Deciduous woodland
3	Wet/fen woodland
4	Unused
5	Heath
6	Pasture/meadow
7	Arable
8	Semi-open (arboreal)
9	Semi-open (heath)
10	Semi-open (pasture)
11	Semi-open (arable)

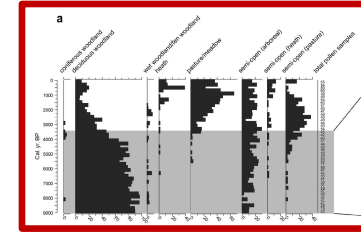
Site_info: Information about each site, incl. Region.

Site_no	Site_code	Site_name	Region	Region2	Longitude	Latitude	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 Scotland	Scotland	-5.8	57.25	Small Birks	H.J.B.,
2	BREDCOUN	Brede Bridge	Southeast England	England	-0.6	50.93	Medi	Waller, M.P.
5	CHROISGP	Loch a'Chrois	North Scotland	Scotland	-5.3	57.57	Medi	Pennington,
26	CLAIR	Loch Clair	North Scotland	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. Convert counts to percentages
4. Allocate each pollen type to an LCC class and sum sqrt-percentages across each class
5. Normalise the sqrt-percent LCC data to 100%
6. Calculate the sum of woodland (A) and open (C) LCC classes
7. Calculate Affinity score as A-C
8. Calculate the LCC class with highest %



Affinity	# LCC class with max Affinity		# LCC2_name
Affinity > 20	1 Coniferous woodland	→	1 Coniferous woodland
	2 Deciduous woodland	→	2 Deciduous woodland
	3 Wet/fen woodland	→	3 Wet/fen woodland
	4 Unused		4 Unused
Affinity < 20	5 Heath	→	5 Heath
	6 Pasture/meadow	→	6 Pasture/meadow
	7 Arable indicators	→	7 Arable
Affinity -20 to 20	1 Coniferous woodland	→	8 Semi-open (arboreal)
	2 Deciduous woodland	→	8 Semi-open (arboreal)
	3 Wet/fen woodland	→	8 Semi-open (arboreal)
	5 Heath	→	9 Semi-open (heath)
	6 Pasture/meadow	→	10 Semi-open (pasture)
	7 Arable indicators	→	11 Semi-open (arable)

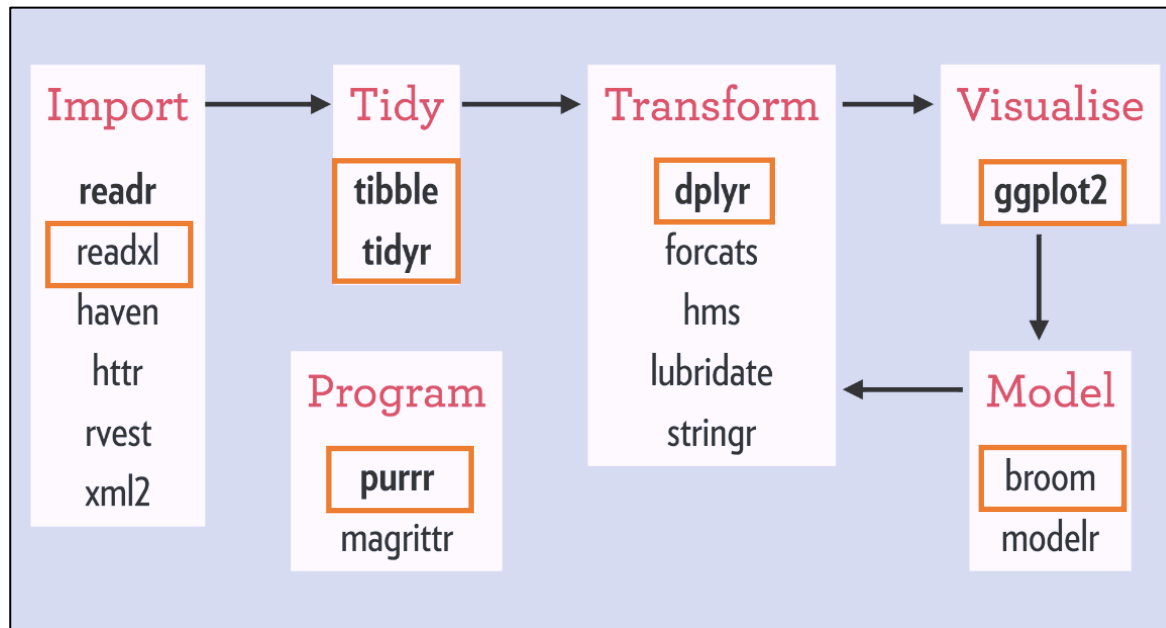
Step2: Aggregate multiple sites

1. Aggregate data from all sites and count the number of levels in each LCC2 class in 200 year time slices (N)
2. Convert N to percentage of each LCC2 class in each time slice

Part 1: Code it in Base R

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 create **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 sylv	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	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**

Tidy data?

Site_Depth	Cal. yr. BP	Pinus sylv	Taxus bac	Betula	Betula/Co
HOCKHAM_132	1514	3	1	40	2
HOCKHAM_164	1618.5	3	3	33	1
HOCKHAM_196	1723	3	2	34	2
HOCKHAM_228	1827.5	4	4	33	1
HOCKHAM_260	1932	5	3	34	3
HOCKHAM_268	2143.75	4	3	43	0
HOCKHAM_276	2355.5	14	2	40	0
HOCKHAM_284	2567.25	11	1	60	0
HOCKHAM_292	2779	10	3	54	2

```
df %>% tidyr::separate(Site_Depth,  
  into=c("Site_code", "Depth" ))
```

Depth (cm)	Cal. yr. BP	Pinus sylv	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	2779	10	3	54	2

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

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

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

dplyr functions for data transformation

select: select columns (**subset**)

dplyr function
Base R equivalent

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)` <dbl>	`Cal. yr. BP` <dbl>	Taxon <chr>	Count <dbl>	SQRT_Count <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

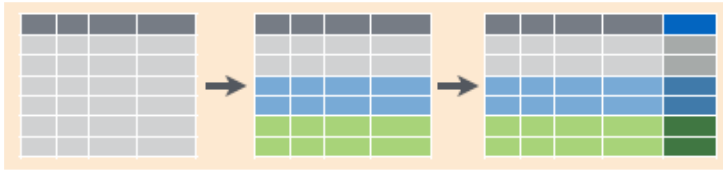
```
df %>% summarise(Total=sum(Count))
```

```
      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)` <dbl>	`Cal. yr. BP` <dbl>	Taxon <chr>	Count <dbl>	Percent <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

Summarising across rows

SiteCode	MaxDepth	Area	Alt.	Ca	Mg	Na	K
CW001	13.0	14	370	100.5	33.5	109	4
CW002	36.0	85	150	88.8	46	175	6.5
CW003	2.8	26	10	1987.5	634	842.5	69.8
CW004	1.8	9	4	1515.8	567	1340.8	64.8
CW005	3.0	19	4				
CW007	7.2	10	305	173	127.3	288.7	15.3
CW008	8.7	5	310	199.3	129.7	246	13.7

```
df %>% mutate(sum_cat = sum(Na+K+Mg+Ca))
```

```
A tibble: 124 x 9
```

```
  SiteCode MaxDepth Area Alt. Ca Mg Na K sum_cat
  <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 CW001      13     14   370  100.  33.5  109    4 436277.
2 CW002      36     85   150   88.8   46   175   6.5 436277.
3 CW003       2.8    26    10 1988.  634   842.  69.8 436277.
```

```
df %>%
```

```
  rowwise %>%
```

```
  mutate(sum_cat = sum(Na+K+Mg+Ca))
```

```
A tibble: 124 x 9
```

```
Rowwise:
```

```
  SiteCode MaxDepth Area Alt. Ca Mg Na K sum_cat
  <chr>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 CW001      13     14   370  100.  33.5  109    4   247
2 CW002      36     85   150   88.8   46   175   6.5  316.
3 CW003       2.8    26    10 1988.  634   842.  69.8 3534.
4 CW004       1.8     9     4 1516.  567  1341.  64.8 3488.
```

Merging tables

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

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

ID	X1	ID	X2
1	a1	2	b1
2	a2	3	b2

ID	X1	X2
1	a1	NA
2	a2	b1

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

Other join types

ID	X1	ID	X2
1	a1	2	b1
2	a2	3	b2

ID	X1	X2
2	a2	b1

ID	X1	X2
1	a1	NA
2	a2	b1

ID	X1	X2
2	a2	b1
3	NA	b2

ID	X1	X2
1	a1	NA
2	a2	b1
3	NA	b2

ID	X1
2	a2

ID	X1
1	a1

Mutating joins: combine variables from the 2 sources

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

R object	Description
Single site data	
poll	original pollen data
poll_long	pollen counts in long format
poll_long_trans	pollen percentages & sqrt in long format
poll_long_LCC	pollen with totals for LCC classes
poll_long_LCC2	pollen with LCC classification for each sample
Combined data (41 sites)	
allpoll_nested	nested list of original pollen data
allpoll_LCC2	pollen with LCC classification for each sample in long format
allpoll_agg	LCC2 classification, aggregated by 200-year time slices
allpoll_rich	palynological richness for all sites / levels, in long format

Part 2: Code it using the Tidyverse

Code comparison: Part 1

Base R

```
del <- colnames(poll) %in% non_pollen
poll <- poll[, !del]
depth_age <- subset(poll, select=c(`Depth (cm)`, `Cal. yr. BP`))
spec <- subset(poll, select=c(`Depth (cm)`, `Cal. yr. BP`))
colnames(depth_age) <- c("Depth", "Age_BP")
spec_pc <- spec / rowSums(spec) * 100
spec_sqrt <- sqrt(spec_pc)
sel <- match(colnames(spec_sqrt), LCC_taxon_list$VarName)
LCC_code <- LCC_taxon_list$LCC_ID[sel]
spec_LCC <- t(rowsum(t(spec_sqrt), group=LCC_code))
spec_LCC <- data.frame(spec_LCC, check.names=FALSE)
spec_LCC <- spec_LCC / rowSums(spec_LCC) * 100
spec_LCC$A <- rowSums(spec_LCC[, c("1", "2", "3")])
spec_LCC$C <- rowSums(spec_LCC[, c("5", "6", "7")])
spec_LCC$Affinity <- spec_LCC$A - spec_LCC$C
A_nms <- c("1", "2", "3")
C_nms <- c("5", "6", "7")
AC_nms <- c(A_nms, C_nms)
LCC2_ID <- kit::nif(
  spec_LCC$Affinity > 20, as.integer(A_nms[apply(spec_LCC[, A_nms], 1,
    which.max)]),
  spec_LCC$Affinity < -20, as.integer(C_nms[apply(spec_LCC[, C_nms],
    1, which.max)]),
  default = ifelse(as.integer(AC_nms[apply(spec_LCC[, AC_nms],
    1, which.max)]) < 4, 8L,
    as.integer(AC_nms[apply(spec_LCC[, AC_nms], 1,
    which.max)]) + 4L)
)
spec_LCC$LCC2_ID <- LCC2_ID
LCC2 <- cbind(depth_age, spec_LCC)
LCC2 <- merge(LCC2, LCC2_assem_classes, by="LCC2_ID",
  all.x=TRUE, sort=FALSE)
LCC2 <- LCC2[order(LCC2$Depth), ]
LCC_names <- LCC_taxon_classes$LCC_name[as.integer(AC_nms)]
```

Tidyverse

```
poll_long <- poll %>%
  rename("Depth"=`Depth (cm)`, "Age_BP"=`Cal. yr. BP`) %>%
  pivot_longer(cols=c("Depth", "Age_BP"),
    names_to="VarName", values_to="Count") %>%
  filter(Count > 0 & !(VarName %in% non_pollen))

poll_long_trans <- poll_long %>%
  group_by(Depth) %>%
  mutate(Percent = Count / sum(Count) * 100,
    Sqrt_PC = sqrt(Percent)) %>%
  ungroup()

poll_long_LCC <- poll_long_trans %>%
  left_join(LCC_taxon_list, by="VarName") %>%
  group_by(Depth, Age_BP, LCC_ID) %>%
  summarise(Percent=sum(Percent),
    Sqrt_PC=sum(Sqrt_PC),
    .groups="drop_last") %>%
  mutate(Norm_Sqrt_PC=Sqrt_PC/sum(Sqrt_PC)*100) %>%
  ungroup() %>%
  left_join(LCC2_assem_classes, by="LCC_ID")

poll_long_LCC2 <- poll_long_LCC %>%
  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) %>%
  slice_max(Norm_Sqrt_PC, n=1, with_ties=FALSE) %>%
  ungroup() %>%
  mutate(LCC2_ID=case_when(
    Affinity >= -20 & Affinity <= 20 & LCC_ID %in% 1:3 ~ 8,
    Affinity >= -20 & Affinity <= 20 & LCC_ID %in% 5:7 ~ LCC_ID + 4,
    TRUE ~ LCC_ID)) %>%
  left_join(LCC2_assem_classes, by="LCC2_ID")
```


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
```

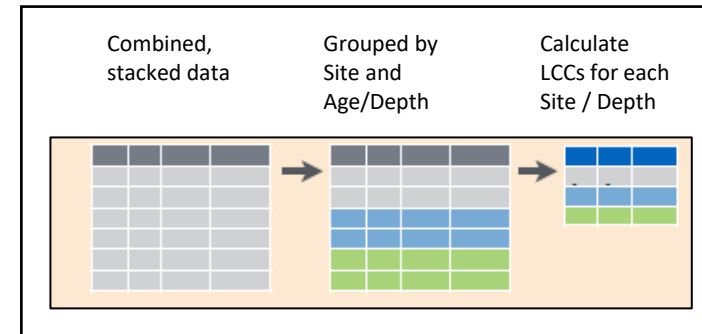
Part 3A: Code to aggregate multiple sites

Base R

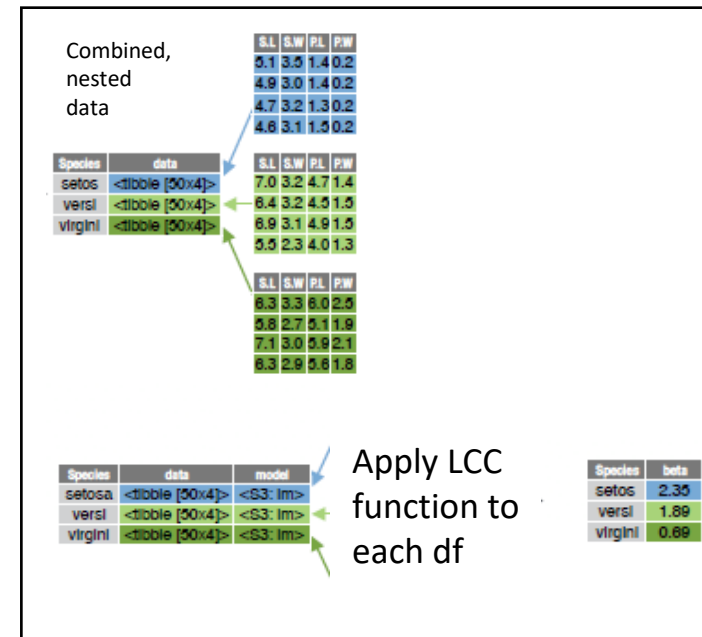
Working with multiple datasets – Tidyverse

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.



Second approach is more flexible as it allows us to apply more complex modelling functions to each nested df.

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

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=.)))
```

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 "." to pass the df to the mapped function

Part 3B: Code to aggregate multiple sites using Tidyverse

Code comparison: Part 2

Base R

```
LCC2 <- NULL
for (i in sites) {
  print(i)
  d <- read_excel("Woodbridge_et_al_2014_Data.xlsx", sheet=i)
  tmp <- fun_LCC2_base(d, LCC_taxon_list=LCC_taxon_list,
                      LCC_taxon_classes=LCC_taxon_classes)
  tmp <- data.frame(Site_code=i, tmp)
  LCC2 <- rbind(LCC2, tmp)
}

cuts <- seq(0, 20000, by=200)
Age2 <- cut(LCC2$Age_BP, breaks=cuts, labels=FALSE,
            include.lowest=TRUE) * 200 - 200
LCC2_count <- with(LCC2, table(Age2, LCC2_ID))
LCC2_percent <- LCC2_count[, -1] /
  rowSums(LCC2_count[, -1]) * 100
LCC2_percent <- as.data.frame.matrix(LCC2_percent)
LCC2_percent <- data.frame(Age2=as.integer(
  rownames(LCC2_count)),
  LCC2_percent, check.names=FALSE)
LCC2_percent <- subset(LCC2_percent, Age2 <= 9000)
LCC2_names <- LCC2_assem_classes$LCC2_name[
  as.integer(colnames(LCC2_percent)[-1])]
```

Tidyverse

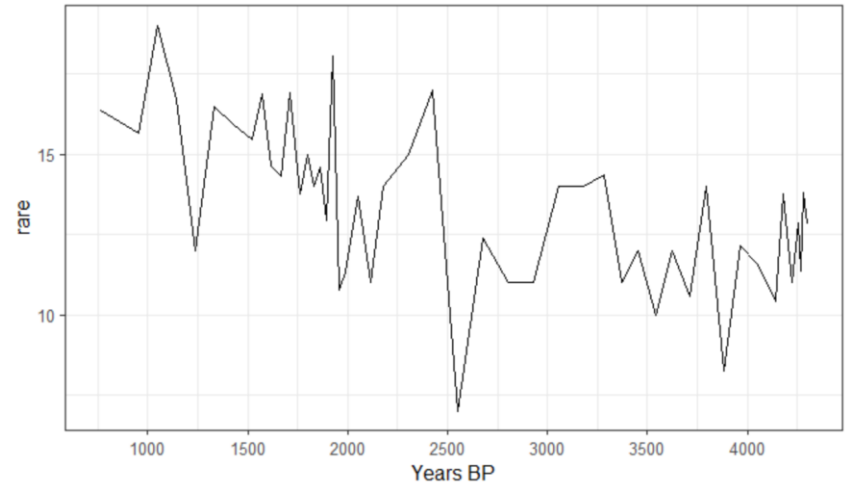
```
allpoll_nested <- tibble(Site_code=names(allpoll_list),
                        data=allpoll_list)

allpoll_LCC2 <- allpoll_nested %>%
  mutate(data = map(data, ~fun_long_tidy(.x,
    non_pollen=non_pollen))) %>%
  mutate(data = map(data, ~fun_LCC2_tidy(.x,
    LCC_taxon_list=LCC_taxon_list,
    LCC_taxon_classes=LCC_taxon_classes,
    LCC2_assem_classes=LCC2_assem_classes))) %>%
  unnest(data)

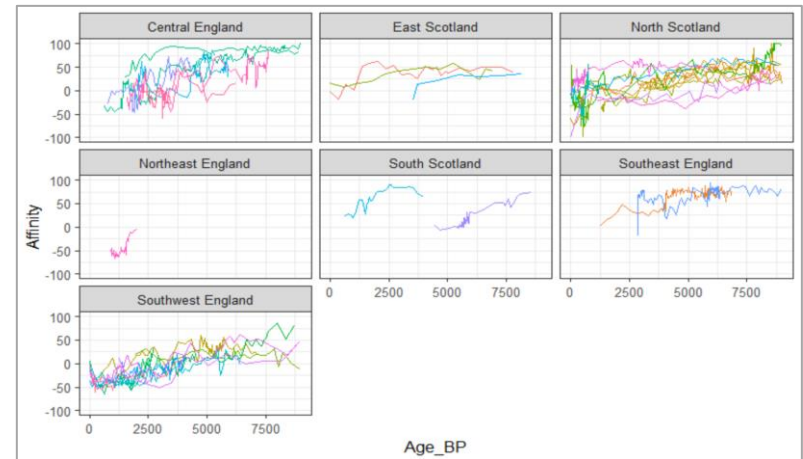
allpoll_agg <- allpoll_LCC2 %>%
  filter(Age_BP <= 9000) %>%
  mutate(Age2=cut_width(Age_BP, boundary=0, width=200,
    labels=FALSE) * 200 - 200) %>%
  group_by(Age2, LCC2_ID) %>%
  summarise(N=n(), .groups="drop_last") %>%
  mutate(PC=N/sum(N)*100) %>%
  pivot_wider(id_cols=c(Age2), names_from=LCC2_ID, values_from=PC,
    values_fill=0) %>%
  pivot_longer(cols=-c(Age2), names_to="LCC2_ID", values_to="PC") %>%
  mutate(LCC2_ID=as.integer(LCC2_ID)) %>%
  left_join(LCC2_assem_classes, by="LCC2_ID")
```

Modelling with the Tidyverse

Part 1: Calculated palynological richness using rarefaction



Part 2: Calculated affinity scores for each site using `purrr::map`
(100 = dominated by arboreal taxa
-100 = dominated by open taxa)

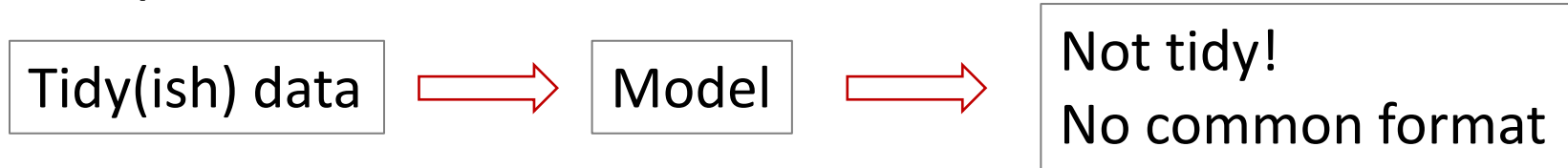


Very simple question:
How do richness and affinity
covary?

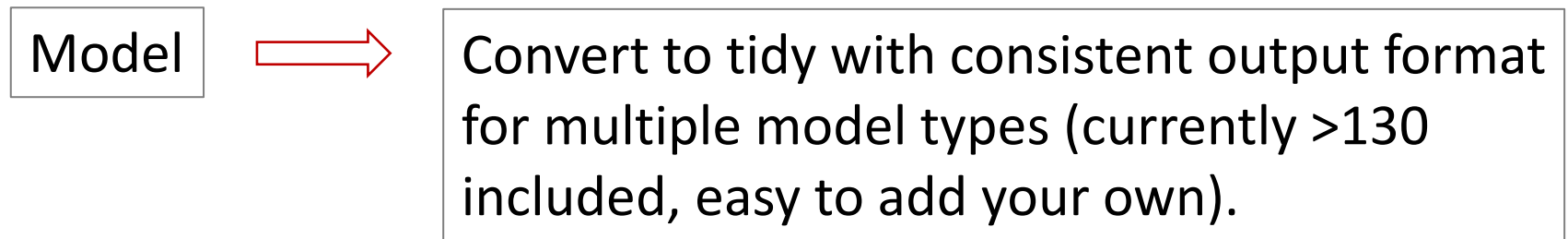
Part 4A: Modelling with the Tidyverse

broom for tidying model output

The problem



The **broom** solution



tidy: tibble that summarises model findings (e.g. coefficients, p-values)

glance: concise one-row summary (e.g. r-squared, degf)

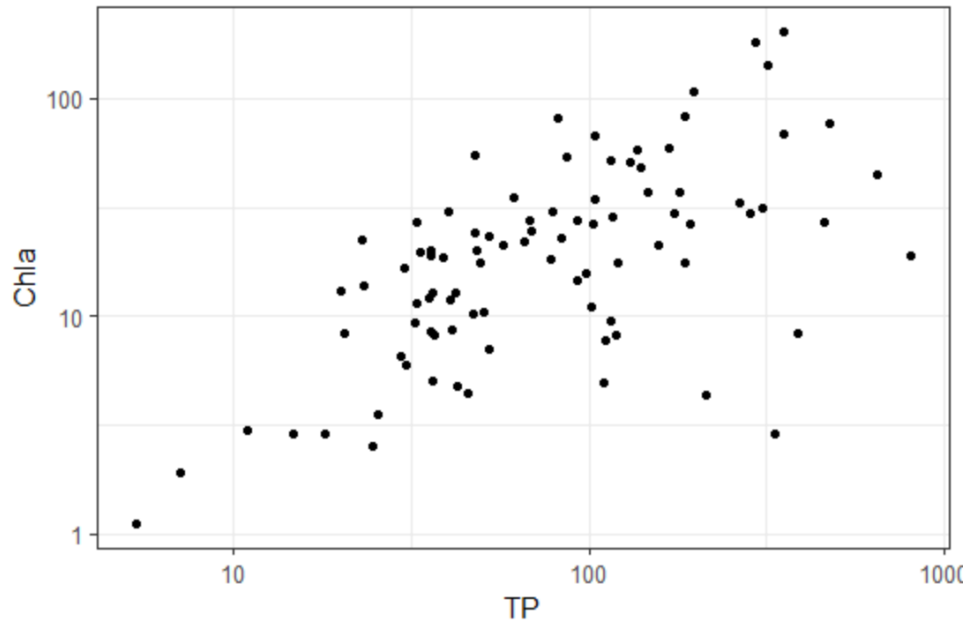
augment: columns original data was modelled on (e.g. predictions, residuals)

Part 4B: Modelling with the Tidyverse

UK Lakes water chemistry

Total phosphorus (TP), Chlorophyll-*a* (Chl*a*) and maximum depth (MaxDepth) measurements for 90 UK lakes.

What is the relationship between Chl*a* & TP?



Does this relationship vary with lake depth?

Part 4: Modelling with the Tidyverse

Your turn!