Preparing our BLE data for analysis

Paul Galpern

Dear Emily ...

The purpose of this R Markdown document is to illustrate some basic steps that will be necessary to prepare a raw "query" from our BLE database for somebasic analyses. I will use the `tidyverse` idiom to do this. As mentioned this is a far easier way to do operations on data frames (that is once you have climbed the learning curve). I really recommend doing this as the benefit is the ability to quickly change analyses with *more* confidence that you are doing so in a manner that will not mangle your data.

A note on the data

Accompanying this R markdown document are three `CSV` files from which I will demonstrate the preparation of analysis-ready data. Note that they represent what is in the database as of September 4th, 2019. The arthropod data is exclusively from BLIDs numbered in the 31000 and 32000s (an indicator that they were part of Jess's 2018 experiment). But always worth checking the coordinates on these to make sure they are where you think they are. The number of animals represented here is likely to change in the next few weeks as we add in harvestmen, honey bees and beetles. However, the non-Apis bees should be authoritative.

Basic setup

In the following, I assume that you have files referenced below in the working directory (i.e. `getwd()` to check).

```
## install.packages("tidyverse")
## install.packages("kableExtra")
library(tidyverse)
## The next two are just to make the R Markdown document
library(knitr)
library(kableExtra)

site <- read_csv("site_BLE_Sep4_19.csv")
trap <- read_csv("trap_BLE_Sep4_19.csv")
arthropod <- read_csv("arthropod_BLID32000_BLE_Sep4_19.csv")
```

A first look at the data

In an R Markdown context we can get a glimpse of the data by using the `knitr::kable()` function. (That is, the `kable()` function in the `knitr` package.)

Site data

These data describe the BLID which is the 5 digit number indicating the "site." A site can have multiple trapping events (i.e. times at which traps were deployed) as well as multiple locations within the site (typically with an intertrap distance of < 200m).

```
## The following code first `restricts `filter`s rows to those in
## the relevant BLID range, then `slice`s out four rows for illustration,
## then `select`s a subset of the columns we care about, and finally
## does some `kable` magic to make it print nicely in this document.
site %>%
  filter(BLID >= 31000 & BLID < 33000) %>%
  slice(c(1, 2, 43, 44)) %>%
  select(BLID, lat, lon, locality, region, elevation, siteType) %>%
  kable(digits=2) %>%
  kable_styling(full_width=FALSE)
```

BLID	lat	lon	locality	region	elevation	siteType
32000	51.79	-114.21	Range Rd 23, Didsbury	Red Deer	1013	ditch
32007	52.12	-113.70	Range Rd 265, Penhold	Red Deer	966	ditch
31119	53.10	-112.83	8.2 km from Armena	Kinsella	739	inField_wetland
31105	53.00	-112.26	15.1 km from Daysland	Kinsella	707	inField_wetland

The crucial fields in the above table are `BLID`, `lat`, `lon` and `elevation` which should be obvious. Also note that the 31000 BLIDs are `inField_wetland` which are the restored sites we care about. The 32000 BLIDs are nearby `ditch` sites which Jess selected in part to get a background sense of the fauna that might be expected. You'll need to query her exactly on what drove this design. So for a strict community analysis of wetland restoration sites you would just use the `inField_wetland` BLIDs.

Also note that in using the `tidyverse` as below we use `%>%` which are called pipes. They shunt the output of the previous function in the next one. For example `site %>% filter(BLID >= 31000)` below could equally be written: `filter(site, BLID >= 31000)`. In other words, the missing variable to the `filter` function is assumed to be the output of the previous step.

Trap data

These data describe trapping events, and allow us to capture multiple times when samples were taken from a given BLID site. They also help resolve multiple traps that may be located at that site (either at exactly the same location, or within 200 m of one another).

There are a lot of relevant columns here, so we'll `select` them into two tables so we can talk about them clearly. We'll also only select two BLIDs from the relevant range and two `pass`es from those sites.

```
trap %>%
  filter(BLID %in% c(32000, 31028)) %>%
  filter(pass %in% c(0, 1)) %>%
  select(BLID, BTID, trapType, pass, replicate, endYear) %>%
  arrange(BLID) %>%
  kable(digits=2) %>%
  kable_styling(full_width=FALSE)
```

BLID	BTID	trapType	pass	replicate	endYear
31028	31028-0-WBV-2018	infield	0	WBV	2018
31028	31028-0-WPF-2018	infield	0	WPF	2018
31028	31028-0-WCC-2018	infield	0	WCC	2018
31028	31028-1-WBV-2018	infield	1	WBV	2018
31028	31028-1-WPF-2018	infield	1	WPF	2018
31028	31028-1-WCC-2018	infield	1	WCC	2018
32000	32000-0-DBV-2018	Blue Vane	0	DBV	2018
32000	32000-0-DPF-2018	Pit Fall	0	DPF	2018
32000	32000-1-DBV-2018	Blue Vane	1	DBV	2018
32000	32000-1-DPF-2018	Pit Fall	1	DPF	2018

First note that the `BTID` is a composite of the BLID, `pass`, `replicate` and `endYear` separated by dashes. `BTID`s are the key sample identifier and are the codes that appear on all pinned samples, all residue jars and are used at intermediate processing steps to keep track of things. It tells us everything we need to know about where and when the animal was sampled as well as how and for what purpose.

Where we sample is the `BLID`.

When we sampled is the `pass` which is numbered from 0 to 8 and refers to a sequential visit to collect things. Pass zero was the set-up pass and is in there mostly so we can estimate the total deployment time (i.e. no animals were collected, but information about vegetation may be recorded at pass 0 and are referenced using that code).

The details of how we sampled and any other experimental information are held within the `replicate`. For this particular set of sites we used `W` to indicate sampling near a wetland. `D` to indicate sampling in a field-side/road-side ditch, `BV` for a blue vane trap, `PF` for a pitfall trap, and `CC` for a coloured cup trap. The latter is basically an efficient way to mimic the yellow and blue plates people put out for short-term bee collection (here, just a supplementary bee trapping method).

Let's look at more columns.

```
trap %>%
  filter(BLID %in% c(32000, 31028)) %>%
  filter(pass %in% c(0, 1)) %>%
  arrange(BLID, pass) %>%
  select(BTID, startMonth:startMinute, endMonth:endMinute) %>%
  kable(digits=2) %>%
  kable_styling(latex_options="scale_down")
```

BTID	startMonth	startDay	startHour	startMinute	endMonth	endDay	endHour	endMinute
31028-0-WBV-2018	NA	NA	NA	NA	5	30	9	53
31028-0-WPF-2018	NA	NA	NA	NA	5	30	9	53
31028-0-WCC-2018	NA	NA	NA	NA	5	30	9	53
31028-1-WBV-2018	5	30	9	53	6	13	9	56
31028-1-WPF-2018	5	30	9	53	6	13	9	56
31028-1-WCC-2018	5	30	9	53	6	13	9	56
32000-0-DBV-2018	NA	NA	NA	NA	5	24	10	47
32000-0-DPF-2018	NA	NA	NA	NA	5	24	10	47
32000-1-DBV-2018	5	24	10	47	6	6	9	23
32000-1-DPF-2018	5	24	10	47	6	6	9	23

Notice here (for the same BTIDs) that we know the time very precisely at which the trap was deployed (`start`) and emptied (`end`). The zero pass BTIDs have no start times, only end times (i.e. because pass 1 started immediately after pass zero ended, and pass zero is a placeholder for the start). You'll note this in the table.

A last look and we can see two more highly essential columns.

```
trap %>%
  filter(BLID %in% c(32000, 31028)) %>%
  filter(pass %in% c(0, 1)) %>%
  arrange(BLID, pass) %>%
  select(BTID, deployedhours, midjulian) %>%
  kable(digits=2) %>%
  kable_styling(full_width=FALSE)
```

BTID	deployedhours	midjulian
31028-0-WBV-2018	NA	NA
31028-0-WPF-2018	NA	NA
31028-0-WCC-2018	NA	NA
31028-1-WBV-2018	336.05	157
31028-1-WPF-2018	336.05	157
31028-1-WCC-2018	336.05	157
32000-0-DBV-2018	NA	NA
32000-0-DPF-2018	NA	NA
32000-1-DBV-2018	310.60	151
32000-1-DPF-2018	310.60	151

`deployedhours` is the end time minus the start time. This variable will be **crucial** when doing abundance-based studies of individual species. Because every trapping period was not the same length we need to correct for trapping effort. For example, doing a Poisson or negative-binomial GLMM you might enter `deployedhours` as an offset term. This effectively makes the response variable a trapping rate of the number of animals per hour. In addition it is worth thinking about differences in effort when doing community-based analyses. We might need to consider rarefying community matrices that are input to various multivariate analyses to make sure they represent comparable estimates of biodiversity.

`midjulian` is the phenological variable we always use. It gives the day of year of the mid-point of the trapping event. As it turns out abundance and community structure is highly-dependent on the phenology. As such it may be necessary to do all abundance analyses as non-linear GAMs to capture seasonal phenology. When comparing sites across a large geographic area or among years we use another variable which is not yet available here. It is the growing degree days. This allows us to standardize climatically for bee emergence time across the province and between seasons. I don't think it is going to matter in this case (but you may want to consider it).

Arthropod data

```
## The query already filtered these data for BLIDs between 31000 and 33000
## This selects the 1st, 11th, 21st, etc. row as illustration
arthropod %>%
  slice(seq(1, 100, by=10)) %>%
  select(BBID, BTID, arthOrder, family, genus, species, caste, sex) %>%
  kable() %>%
  kable_styling(full_width=FALSE)
```

BBID	BTID	arthOrder	family	genus	species	caste	sex
1509601	31000-1-WBV-2018	Hymenoptera	Apidae	Bombus	terricola	queen	NA
1500412	31000-3-WBV-2018	Hymenoptera	Apidae	Bombus	rufocinctus	queen	F
1500422	31000-3-WBV-2018	Hymenoptera	Apidae	Bombus	rufocinctus	queen	F
1501321	31000-1-WBV-2018	Hymenoptera	Apidae	Bombus	rufocinctus	queen	F
1503017	31000-2-WBV-2018	Hymenoptera	Apidae	Bombus	rufocinctus	queen	F
1503027	31000-2-WBV-2018	Hymenoptera	Apidae	Bombus	rufocinctus	queen	F
1503037	31000-2-WBV-2018	Hymenoptera	Apidae	Bombus	rufocinctus	queen	F
1503436	31000-2-WBV-2018	Hymenoptera	Apidae	Bombus	rufocinctus	queen	F
1508638	31000-4-WCC-2018	Hymenoptera	Apidae	Bombus	rufocinctus	worker	F
1509323	31000-4-WBV-2018	Hymenoptera	Apidae	Bombus	rufocinctus	worker	F

This subset shows us two species of bumble bees including queens and workers. Note how the `sex` column has not been uniformly completed for bumble bees, as this can be determiend from the caste alone. `sex` should be authoritative for other bee species, however. You might need to clean this up if you do analyses by sex. Everything should be self explanatory, except the `BBID` which is the unique identifier of that particular specimen and appears on the pin label. The `BTID` also appears on the pin label in the collection.

Join the data

We have three data frames here `site`, `trap` and `arthropod`. These correspond to how the data is stored in our networked relational database and this is to make it efficient to enter data only once and let computers do the magic.

What we want before we can do analyses is to have a data frame where everything about each bee is on each row. This is trivial with the `tidyverse`. Without it, by the way, it's quite challenging.

```
## Produce an analysis-ready data frame
##
## Notice how this time we assign the result of the pipe chain to an
## object `allD`. We also remove an unneeded duplicated BLID column in
## arthropod before joining `select(-BLID)` as it gets renamed and
## causes trouble at the final join if we don't.
allD <- arthropod %>%
  select(-BLID) %>%
  left_join(trap, by="BTID") %>%
  left_join(site, by="BLID")
```

That's it! We've got a data frame where each row is a different bee and the spatial information as well as the temporal and experimental information is joined in additional columns. This was achieved using a thing called a `left_join` which takes the first data frame and matches up rows from the second data frame using the column `by` to make the match.

There are lot of other columns that don't really matter to you in `allD`. I won't show them here, but let's just prove to ourselves that it worked.

```
## Before the join
nrow(arthropod)
```

```
## [1] 22034
```

```
ncol(arthropod)
```

```
## [1] 20
```

```
ncol(site)
```

```
## [1] 20
```

```
ncol(trap)
```

```
## [1] 42
```

```
## After the join
nrow(allD)
```

```
## [1] 22034
```

```
ncol(allD)
```

```
## [1] 79
```

So we have the same number of rows before and after. That is we have data on 22,034 arthropods. We had `20+20+42=82` columns before, and after we had `79` which corresponds to a loss of three columns. Two of these were duplicated because they were used to make a match, and hence they were removed when the join occurred, and one was forcibly removed as the first step. So it worked!

Now check the data quality

It would be important to look for issues in these data. For example there will definitely be mispellings in the names of species that will make it look like you have two species when you only have one. There are ways these can be diagnosed. For example:

```
## First make a single variable that is the species binomial name
## put it in alphabetical order, then group by this binomial name
## then summarize by these groups to find the number of records of each name
allD %>%
  mutate(genusSpecies=paste(genus, species, sep="_")) %>%
  arrange(genusSpecies) %>%
  group_by(genusSpecies) %>%
  summarize(Nrecords = n()) %>%
  kable(longtable=TRUE) %>%
  kable_styling(latex_options=c("repeat_header"))
```

genusSpecies	Nrecords
Agapostemon_texanus	57
Alopecosa_aculeata	4
Andrena_amphibola	10
Andrena_lupinorum	8
Andrena_nivalis	12
Andrena_pertristis	3
Anthidium_tenuiflorae	1
Anthophora_bomboides	74
Anthophora_occidentalis	13
Anthophora_porterae	7
Anthophora_terminalis	529
Anthophora_ursina	1
Bombus_bifarius	20
Bombus_bohemicus	3
Bombus_borealis	2020
Bombus_centralis	222

(continued)

genusSpecies	Nrecords
Bombus_cryptarum	15
Bombus_fervidus	2
Bombus_flavidus	8
Bombus_flavifrons	58
Bombus_frigidus	59
Bombus_huntii	20
Bombus_insularis	119
Bombus_melanopygus	3
Bombus_mixtus	188
Bombus_nevadensis	299
Bombus_occidentalis	6
Bombus_perplexus	34
Bombus_rufocinctus	5099
Bombus_sandersoni	94
Bombus_sylvicola	117
Bombus_ternarius	6205
Bombus_terricola	230
Bombus_vagans	851
Coelioxys_funeraria	10
Coelioxys_rufitarsis	2
Coelioxys_sodalis	4
Diadasia_australis	8
Dufourea_maura	75
Habropoda_cineraria	2
Halictus_confusus	72
Halictus_ligatus	2
Halictus_rubicundus	133
Heriades_carinata	1
Heriades_variolosus	1
Holcopasites_pulchellus	5
Hoplitis_albifrons	1
Hoplitis_pilosifrons	30
Hoplitis_producta	9
Hoplitis_spoliata	5
Lasioglossum_colatum	1
Lasioglossum_leucozonium	459
Lasioglossum_paraforbesii	336
Lasioglossum_zonulum	1375
Lassioglossum_zonulum	60
Megachile_frigida	11
Megachile_inermis	43
Megachile_lapponica	7
Megachile_latimanus	17
Megachile_melanophaea	52
Megachile_montivaga	1
Megachile_perihirta	100
Megachile_pugnata	5
Megachile_rotundata	3
Melissodes_agilis	26
Melissodes_confusus	2383

(continued)

genusSpecies	Nrecords
Melissodes_rivalis	34
Osmia_bucephala	4
Pardosa_distincta	1
Pardosa_fuscula	1
Pardosa_groenlandia	1
Pardosa_modica	1
Pardosa_ontariensis	20
Pardosa_Pardosa_sp	11
Pardosa_xerampelina	1
Pseudopanurgus_parvus	329
Xysticus_ferox	1

It frankly doesn't look too bad, but you should probe other columns for things that don't make sense. *Lasioglossum zonulum* a very common introduced sweat bee seems to have two spellings and the single "s" version is the correct one. How would you correct this. Well, as follows:

```
## Notice how we start by including an old version of the data
## in the pipeline and assign it back to itself. We could easily
## assign it to another variable, however.
##
## The `ifelse` function checks every row in the genus column for the
## mispelled text and replaces it with the correct text if it finds it,
## otherwise it just inserts the text that was there `genus`
##
## Finally we remake the binomial species column to reflect this fix
allD <- allD %>%
  mutate(genus=ifelse(genus=="Lassioglossum", "Lasioglossum", genus)) %>%
  mutate(genusSpecies=paste(genus, species, sep="_"))
```

We should also probe the data in other ways to make sure there are no obvious typos or groups that make no sense. This could be done by grouping by various columns and summarizing as I have done. I'll show this for the BTID column, and we will count the number of distinct BTIDs, passes, and replicates by BLID.

```
allD %>%
  arrange(BLID) %>%
  group_by(BLID, siteType) %>%
  summarize(N_BTIDs = n_distinct(BTID),
            N_pass = n_distinct(pass),
            N_replicates = n_distinct(replicate),
            Nrecords=n()) %>%
  kable(longtable=TRUE) %>%
  kable_styling(latex_options=c("repeat_header"))
```

BLID	siteType	N_BTIDs	N_pass	N_replicates	Nrecords
31000	inField_wetland	9	4	3	335
31007	inField_wetland	4	2	2	395
31014	inField_wetland	4	3	2	131
31021	inField_wetland	6	3	2	460
31028	inField_wetland	10	4	3	288
31035	inField_wetland	3	2	2	55
31042	inField_wetland	10	4	3	112
31049	inField_wetland	5	2	3	210

(continued)

BLID	siteType	N_BTIDs	N_pass	N_replicates	Nrecords
31063	inField_wetland	8	4	3	459
31070	inField_wetland	8	4	3	234
31077	inField_wetland	8	4	2	226
31091	inField_wetland	7	3	3	108
31105	inField_wetland	9	4	3	594
31112	inField_wetland	6	3	2	122
31119	inField_wetland	6	4	3	463
31133	inField_wetland	7	3	3	78
31147	inField_wetland	7	4	3	164
31154	inField_wetland	10	4	3	99
31161	inField_wetland	10	4	4	113
31168	inField_wetland	11	4	3	666
31175	inField_wetland	8	3	3	136
31182	inField_control	8	4	2	284
31189	inField_wetland	10	4	3	318
31196	inField_wetland	11	4	3	357
31203	inField_wetland	8	4	3	145
31210	inField_wetland	5	2	3	444
31217	inField_control	6	3	3	153
32000	ditch	4	4	1	89
32007	ditch	5	4	2	157
32014	ditch	4	4	1	247
32021	ditch	7	4	2	292
32028	ditch	4	4	1	260
32035	ditch	4	4	1	383
32042	ditch	5	4	2	169
32049	ditch	7	4	2	294
32056	ditch	4	4	1	194
32063	ditch	3	3	1	57
32070	ditch	4	4	1	102
32077	ditch	6	4	2	88
32084	ditch	6	4	2	183
32091	ditch	4	4	1	177
32098	ditch	5	4	2	226
32105	ditch	4	4	1	295
32112	ditch	5	4	2	246
32119	ditch	6	4	2	704
32126	ditch	5	4	2	275
32133	ditch	5	4	2	143
32140	ditch	6	4	2	466
32147	ditch	4	4	1	221
32154	ditch	5	4	2	297
32161	ditch	4	4	1	318
32168	ditch	7	4	2	608
32175	ditch	8	4	2	346
32182	ditch	8	4	2	608
32189	ditch	6	4	2	524
32196	ditch	7	4	2	575
32203	ditch	7	4	2	390
32210	ditch	6	4	2	483

(continued)

BLID	siteType	N_BTIDs	N_pass	N_replicates	Nrecords
32217	ditch	6	4	2	673
32224	ditch	6	4	2	887
32231	ditch	7	4	2	610
32238	ditch	5	4	2	465
32245	ditch	5	3	2	571
32252	ditch	7	4	2	799
32259	ditch	5	4	2	338
32266	ditch	6	4	2	703
32273	ditch	4	4	1	129
NA	NA	160	1	1	293

Generally speaking the number of replicates multiplied by the number of passes should be greater than or equal to the number of BTIDs. Where there are fewer BTIDs it is because some traps may have had no animals, or those animals are of species groups that have not yet been entered (e.g. harvestmen, beetles). But anything that looks way out of line is worth inspecting using `filter`s of the data frame.

A final word ...

Finally don't forget to focus on only the animals you want to focus on! There's more than bees in there so you'll need to filter.

For abundance-related analyses of these data this data frame is ready to go. However to prepare the data into a community matrix of the sort that might be used for virtually any multivariate analysis in community ecology there are more steps to go. You will need to first summarize your data by the relevant grouping (e.g. are you analyzing by BLID or by BTID) and find the total number of each species (or species-sex) combination at each level of the group. (This is straightforward with the `tidyverse`, but bears thinking about how you want to aggregate information). Of course you might then want to filter out very rare or narrowly distributed species (maybe, maybe not) depending on the task.

Next you'll `spread()` your species data into columns. This is a magical thing that the `tidyverse` will do for you. And if (or when) you get to this step, I can help more if you need it. After that it's community analyses all the way. And rarefaction is likely going to need to be one of those steps.

Additional information that we need to integrate is of course land cover as well as information about time since restoration. That information we'll have to talk to Jess about. And indeed, anything else that you think might be important dependent on how you want to take this analysis, and the sorts of hypotheses you want to test.