Convert xlsx to csv

All we're going to do here is convert our xlsx into a csv.

We're going to use `readxl` because thats how I know how to do it. First we'll load the data as a tibble.

```
require(readxl)
```

```
## Loading required package: readxl
```

```
setwd('~/projects/QB-2021-project/')
tib <- readxl::read_xlsx('data/species.xlsx')
str(tib)
```

```
## tibble[,13] [178 x 13] (S3: tbl_df/tbl/data.frame)
##  $ Genus  : chr [1:178] "Clarias" "Tilapia" "Cichlidae" "1enopus" ...
##  $ Species: chr [1:178] "sp." "sp." "Indet" "sp." ...
##  $ Bed_1L : num [1:178] 1 1 0 1 1 0 0 0 0 1 ...
##  $ Bed_1M : num [1:178] 1 1 1 1 1 0 1 1 1 1 ...
##  $ Bed_1U : num [1:178] 0 0 0 1 1 1 1 1 1 1 ...
##  $ Bed_2L : num [1:178] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_2M : num [1:178] 1 1 0 0 0 0 0 0 0 0 ...
##  $ Bed_2U : num [1:178] 1 1 0 0 0 0 0 0 0 0 ...
##  $ Bed_3  : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_4L : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_4M : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_4U : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_m  : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
```

Clean up future column names

This is just where sites (really layers for us) are rows while species are columns. We're going to want to adjust some stuff. Forexample we will need unique names for our columns. we'll name them using `genus_species` by manipulating the values in those columns we will do that using `tidyr`. We'll also clean up the text using `dplyr` functions with `stringr` functions.

```
require(tidyr)
```

```
## Loading required package: tidyr
```

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
```

```
##     intersect, setdiff, setequal, union
```

```
require(stringr)
```

```
## Loading required package: stringr
```

```
tib_combined <- tib %>%
  tidyr::unite(genus_species, c("Genus", "Species")) %>%
  dplyr::mutate(genus_species = tolower(genus_species)) %>%
  dplyr::mutate(genus_species = stringr::str_replace(genus_species, "[.]", "")) %>%
  dplyr::mutate(genus_species = stringr::str_replace(genus_species, " ", "_"))
str(tib_combined)
```

```
## tibble[,12] [178 x 12] (S3: tbl_df/tbl/data.frame)
##  $ genus_species: chr [1:178] "clarias_sp" "tilapia_sp" "cichlidae_indet" "1enopus_sp" ...
##  $ Bed_1L       : num [1:178] 1 1 0 1 1 0 0 0 0 1 ...
##  $ Bed_1M       : num [1:178] 1 1 1 1 1 0 1 1 1 1 ...
##  $ Bed_1U       : num [1:178] 0 0 0 1 1 1 1 1 1 1 ...
##  $ Bed_2L       : num [1:178] 0 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_2M       : num [1:178] 1 1 0 0 0 0 0 0 0 0 ...
##  $ Bed_2U       : num [1:178] 1 1 0 0 0 0 0 0 0 0 ...
##  $ Bed_3        : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_4L       : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_4M       : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_4U       : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
##  $ Bed_m        : num [1:178] 1 0 0 0 0 0 0 0 0 0 ...
```

Transpose the dataframe

This is going to look a little complicated but it will very "elegantly" get us what we want using `tidr` functions. It works like this:

1. Combine all of the columns other than `genus_species` into two columns, call them `layer` and `pres`. This gives us a mostly "long" format dataframe. Each row consists of a taxon, a layer, and whether or not the taxon is present in that layer.
2. Use the `genus_species` column to generate a bunch of new columns, one for each taxon. This returns us to a "wide" format.
3. Make all of the layer names lowercase letters only so we never have to think about capital letters again.

```
tib_t <- tib_combined %>%
  tidyr::gather(key = "layer", value = "pres", 2:ncol(tib_combined)) %>%
  tidyr::spread(key = "genus_species", value = "pres") %>%
  dplyr::mutate(layer = tolower(layer))
str(tib_t)
```

```
## tibble[,179] [11 x 179] (S3: tbl_df/tbl/data.frame)
##  $ layer                              : chr [1:11] "bed_1l" "bed_1m" "bed_1u" "bed_2l" ...
##  $ 1enocephalus_sp                    : num [1:11] 1 0 0 1 1 1 0 0 0 0 ...
##  $ 1enopus_sp                         : num [1:11] 1 1 1 0 0 0 0 0 0 0 ...
##  $ 1erus_sp                           : num [1:11] 0 0 1 0 0 0 0 0 0 0 ...
##  $ aepyceros_sp                       : num [1:11] 0 0 0 0 0 1 0 0 0 0 ...
##  $ aethomys_cf_lavocati               : num [1:11] 0 0 0 0 0 0 0 0 1 0 ...
##  $ afrochoerus_nicoli                 : num [1:11] 0 0 0 0 1 1 0 0 0 0 ...
##  $ agamidae_indet                     : num [1:11] 0 1 0 0 0 0 0 0 0 0 ...
##  $ alcelaphini_1                      : num [1:11] 0 0 0 0 0 0 1 0 0 1 ...
##  $ alcelaphini_2                      : num [1:11] 0 0 0 0 0 0 1 0 0 1 ...
##  $ alcelaphini_3                      : num [1:11] 0 0 0 0 0 0 0 0 0 0 ...
##  $ alcelaphini_indet                : num [1:11] 1 1 0 1 0 0 0 0 0 0 ...
##  $ ancylotherium_cf_hennigi         : num [1:11] 1 0 0 0 0 0 0 0 0 0 ...
##  $ antidorcus_sp                    : num [1:11] 0 0 0 0 0 0 1 0 1 0 ...
##  $ antilopini_indet                 : num [1:11] 1 0 0 1 1 1 0 0 0 1 ...
##  $ anura_indet                      : num [1:11] 0 0 0 1 1 1 1 0 1 1 ...
##  $ arvicanthus_niloticus            : num [1:11] 0 0 0 0 0 0 0 0 1 0 ...
##  $ aves_indet                       : num [1:11] 0 0 0 0 0 0 1 1 1 1 ...
##  $ beatragus_antiquus               : num [1:11] 0 0 1 0 1 0 0 0 0 0 ...
##  $ bitis_nasicornis                 : num [1:11] 0 1 0 0 0 0 0 0 0 0 ...
##  $ bitis_sp                         : num [1:11] 1 1 1 0 0 1 0 0 0 0 ...
##  $ boidae_indet                     : num [1:11] 1 1 0 0 0 0 0 0 0 0 ...
##  $ bovidae_indet                    : num [1:11] 0 0 1 0 0 0 0 0 0 0 ...
##  $ bovini_(small)_indet             : num [1:11] 1 1 1 1 1 1 0 0 0 0 ...
##  $ bufo_sp                          : num [1:11] 0 0 1 0 0 0 0 0 0 0 ...
##  $ bufonidae_indet                  : num [1:11] 0 1 1 0 0 0 0 0 0 0 ...
##  $ canidae_indet                    : num [1:11] 0 1 0 0 1 1 1 0 0 0 ...
##  $ cecopithecoidea_kimeui           : num [1:11] 0 0 0 0 0 0 0 0 1 1 ...
##  $ cephalopini_indet                : num [1:11] 0 0 0 0 0 0 0 0 0 0 ...
##  $ ceratherium_simum                : num [1:11] 1 1 1 1 1 1 0 1 1 1 ...
##  $ cf_acinony1_sp.                  : num [1:11] 0 0 0 1 0 0 1 0 1 0 ...
##  $ cf_cercocebus_sp.                : num [1:11] 0 0 0 0 0 1 0 0 0 0 ...
##  $ cf_grammomys_sp.                 : num [1:11] 0 1 0 0 0 0 0 0 0 0 ...
##  $ cf_myona1_sanguineus             : num [1:11] 0 1 0 0 0 0 0 0 0 0 ...
##  $ cf_steatomys_sp.                 : num [1:11] 0 1 1 0 0 0 0 0 1 0 ...
##  $ chamaeleontidae_indet            : num [1:11] 0 1 1 0 0 0 0 0 0 0 ...
##  $ chelonia_indet                   : num [1:11] 1 1 1 1 1 1 1 1 1 1 ...
##  $ cichlidae_indet                  : num [1:11] 0 1 0 0 0 0 0 0 0 0 ...
##  $ clarias_sp                       : num [1:11] 1 1 0 0 1 1 1 1 1 1 ...
##  $ colobinae_indet                  : num [1:11] 0 0 0 0 0 0 1 0 0 0 ...
##  $ colubridae_indet                 : num [1:11] 1 1 1 0 0 1 0 0 0 0 ...
##  $ crocidura_cf_hindei              : num [1:11] 0 1 1 0 0 0 0 0 0 0 ...
##  $ crocodylus_niloticus             : num [1:11] 1 1 1 1 1 1 0 0 0 0 ...
##  $ crocodylus_sp                    : num [1:11] 1 1 1 1 1 1 1 1 1 1 ...
##  $ crocodylus_spnov.                : num [1:11] 1 0 0 1 1 0 0 0 0 0 ...
##  $ crocuta_aff_ultra                : num [1:11] 1 1 1 0 0 1 0 0 0 0 ...
##  $ crocuta_crocuta                  : num [1:11] 0 0 1 0 0 0 0 0 0 0 ...
##  $ crocuta_sp                       : num [1:11] 1 1 0 0 0 0 0 0 0 0 ...
##  $ damaliscus_agelaius              : num [1:11] 0 0 0 0 0 0 1 0 0 0 ...
##  $ damaliscus_angusticornis         : num [1:11] 0 0 0 1 1 0 0 0 0 0 ...
##  $ damaliscus_niro                  : num [1:11] 0 0 0 0 1 1 1 0 0 0 ...
##  $ dasymus_incomptus                : num [1:11] 0 0 0 0 0 0 0 0 1 0 ...
##  $ deinotherium_cf_bozasi           : num [1:11] 1 0 1 1 0 0 0 0 0 0 ...
##  $ dendromys_sp                     : num [1:11] 0 1 1 0 0 0 0 0 0 0 ...
##  $ diceros_bicornis                 : num [1:11] 0 0 0 0 1 1 1 1 1 1 ...
##  $ ectopotamochoerus_dubius         : num [1:11] 1 1 1 0 0 0 0 0 0 0 ...
##  $ elapidae_indet                   : num [1:11] 1 1 1 0 0 0 0 0 0 0 ...
##  $ elephantulus_sp                  : num [1:11] 0 0 1 0 0 0 0 0 0 0 ...
##  $ elephas_recki_(early from)       : num [1:11] 1 1 1 1 0 0 0 0 0 0 ...
##  $ elephas_recki_(evolved form)     : num [1:11] 0 0 0 0 1 1 1 0 1 1 ...
##  $ equus_cf_oldowayensis            : num [1:11] 1 1 1 1 1 1 1 1 1 1 ...
##  $ erinaceus_cf_major               : num [1:11] 0 0 1 0 0 0 0 0 0 0 ...
##  $ felidae_indet                    : num [1:11] 1 0 1 1 0 0 0 0 0 0 ...
##  $ felis_cf_serval                  : num [1:11] 0 0 0 0 1 0 0 0 0 0 ...
##  $ galago_senegalensis              : num [1:11] 0 1 0 0 0 0 0 0 0 0 ...
##  $ galago_sp                      : num [1:11] 1 0 0 0 0 0 0 0 0 0 ...
##  $ gazella_sp                     : num [1:11] 0 0 0 0 1 1 0 0 0 0 ...
##  $ gazella_wellsi                 : num [1:11] 1 1 1 1 0 0 0 0 0 0 ...
##  $ gazellini_sp                   : num [1:11] 0 0 0 0 0 0 1 1 0 1 ...
##  $ genetta_sp                     : num [1:11] 0 1 1 0 0 0 0 0 0 0 ...
##  $ gerbillus_sp                   : num [1:11] 0 0 1 0 0 0 0 0 0 0 ...
##  $ giraffa_gracilis               : num [1:11] 0 0 0 1 0 0 0 0 0 0 ...
##  $ giraffa_jumae                  : num [1:11] 0 1 0 0 1 0 0 1 1 1 ...
##  $ giraffa_sp                     : num [1:11] 1 0 1 1 1 1 0 0 0 0 ...
##  $ giraffa_stillei                : num [1:11] 0 0 0 0 0 0 0 0 0 1 ...
##  $ gorgon_olduvaiensis            : num [1:11] 0 0 1 0 1 1 0 0 0 0 ...
##  $ herpestes_sp                   : num [1:11] 1 0 1 0 0 0 0 0 0 0 ...
##  $ herpestinae_indet              : num [1:11] 0 0 1 0 0 0 0 0 0 0 ...
##  $ heterocephalus_sp              : num [1:11] 0 0 1 1 1 0 0 0 1 0 ...
##  $ hipparion_cf_ethiopicum        : num [1:11] 0 0 0 0 0 0 1 0 1 1 ...
##  $ hippopotamus_gorgops           : num [1:11] 1 1 1 1 1 1 1 1 1 1 ...
##  $ hippopotamus_sp                : num [1:11] 0 0 0 0 0 1 0 0 0 0 ...
##  $ hippotragini_indet             : num [1:11] 0 1 1 1 1 0 0 0 0 0 ...
##  $ hippotragus_gigas              : num [1:11] 1 1 0 0 1 1 1 0 0 0 ...
##  $ hyaenidae_indet                : num [1:11] 0 0 0 0 0 0 1 0 0 0 ...
##  $ hystri1_sp                     : num [1:11] 1 1 1 0 0 0 0 0 0 1 ...
##  $ insectivora_indet              : num [1:11] 0 1 0 1 0 1 0 0 0 0 ...
##  $ jaculus_sp                     : num [1:11] 0 0 0 0 0 0 0 0 0 0 ...
##  $ kobus_ellipsiprymnus           : num [1:11] 0 0 0 0 0 0 1 0 0 0 ...
##  $ kobus_kob                      : num [1:11] 0 0 0 0 0 0 1 0 0 0 ...
##  $ kobus_sigmoidalis              : num [1:11] 0 1 0 0 0 0 0 0 0 0 ...
##  $ kobus_sp                       : num [1:11] 1 1 1 1 1 1 0 0 0 0 ...
##  $ kolpochoerus_limnetes          : num [1:11] 0 0 0 0 0 0 1 1 1 1 ...
##  $ kolpochoerus_major             : num [1:11] 0 0 0 0 0 0 0 0 0 1 ...
##  $ lagomorpha_indet               : num [1:11] 0 0 0 0 0 0 1 0 0 1 ...
##  $ libytherium_olduvaiensis       : num [1:11] 1 1 1 1 1 1 0 0 0 0 ...
##  $ lutra_sp                       : num [1:11] 1 0 0 0 0 0 0 0 0 0 ...
##  $ lutrinae_indet                 : num [1:11] 0 0 0 0 0 0 1 0 0 0 ...
##  $ machairodontinae_indet         : num [1:11] 1 1 0 1 1 0 0 0 0 0 ...
##   [list output truncated]
```

Save the dataframe as a csv

Now we have *essentially* a site-by-species matrix for our data that is formatted to play very nicely with `tidyverse` stuff. We just need to save it which we can do with the built in function `write.csv`.

```
# uncomment to write file, its curently giving me an error because of working dir or something
# write.csv(tib_t, 'data/layer_by_species.csv')
```