Understanding MEs

Jeanette Lyerly

7/15/2023

Purpose

This is the example for “Understanding Mega-environments with real world breeding data” as part of the Wheat CAP Genomic Selection Workshop, July 2023, Raleigh, NC.

In this markdown we will import some data and create an example biplot.

Our goals in this example are to:
- Import some data
- Check the data
- Create a biplot
- Calculate correlations among environments
- Decide if those correlations make sense with our biplot output

These data sets are based on data provided by Jean-Luc Jannink (downloaded from T3). Data are from a wheat yield QTL study from the University of Nebraska.
More about T3: https://wheat.triticeaetoolbox.org (https://wheat.triticeaetoolbox.org)

Background

In this example we will begin with the output from Noah’s field analysis. In that analysis he imported the clean data, checked for row and column effects, fit the best model, and calculated BLUEs for each location.

Here we will take the BLUEs for those locations and create a biplot to visualize the relationship between the environments.

Load libraries

We are going to use the tidyverse, gge, and GGEBiplots packages.

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✔ ggplot2 3.4.2     ✔ purrr   1.0.1
## ✔ tibble  3.2.1     ✔ dplyr   1.1.2
## ✔ tidyr   1.3.0     ✔ stringr 1.5.0
## ✔ readr   2.1.0     ✔ forcats 0.5.1
```

```
## ── Conflicts ─────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(gge)
library(GGEBiplots)
library(here)
```

```
## here() starts at /Users/jeanette/Desktop/Genomic Selection 2022-2023/GS_workshop_J
uly2023/understanding_MEs
```

Import data

The input file with the BLUEs is in the data folder for our mega-environment analysis. We will import the data and check it.

```
mydata <- read_csv(file = here::here("data", "YLDQTL_allBLUEs.csv"), col_names = T)
```

```
## Rows: 2420 Columns: 4
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (3): TRAIT, LOC, ID
## dbl (1): BLUEs
##
## ℹ Use `spec()` to retrieve the full column specification for this data.
## ℹ Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
glimpse(mydata) #always a good idea to check the data after importing
```

```
## Rows: 2,420
## Columns: 4
## $ TRAIT <chr> "Yield", "Yield", "Yield", "Yield", "Yield", "Yield", "Yield", "…
## $ LOC   <chr> "Across Locations", "Across Locations", "Across Locations", "Acr…
## $ ID    <chr> "11_26", "11_73", "13_13", "13_59", "14_10", "14_2", "14_22", "1…
## $ BLUEs <dbl> 60.27991, 62.05436, 56.25072, 60.73065, 65.40685, 67.60363, 61.3…
```

In this file we have a variable for trait, a variable for location, the sample ID, and the phenotype data, or BLUEs.

Get the yield data

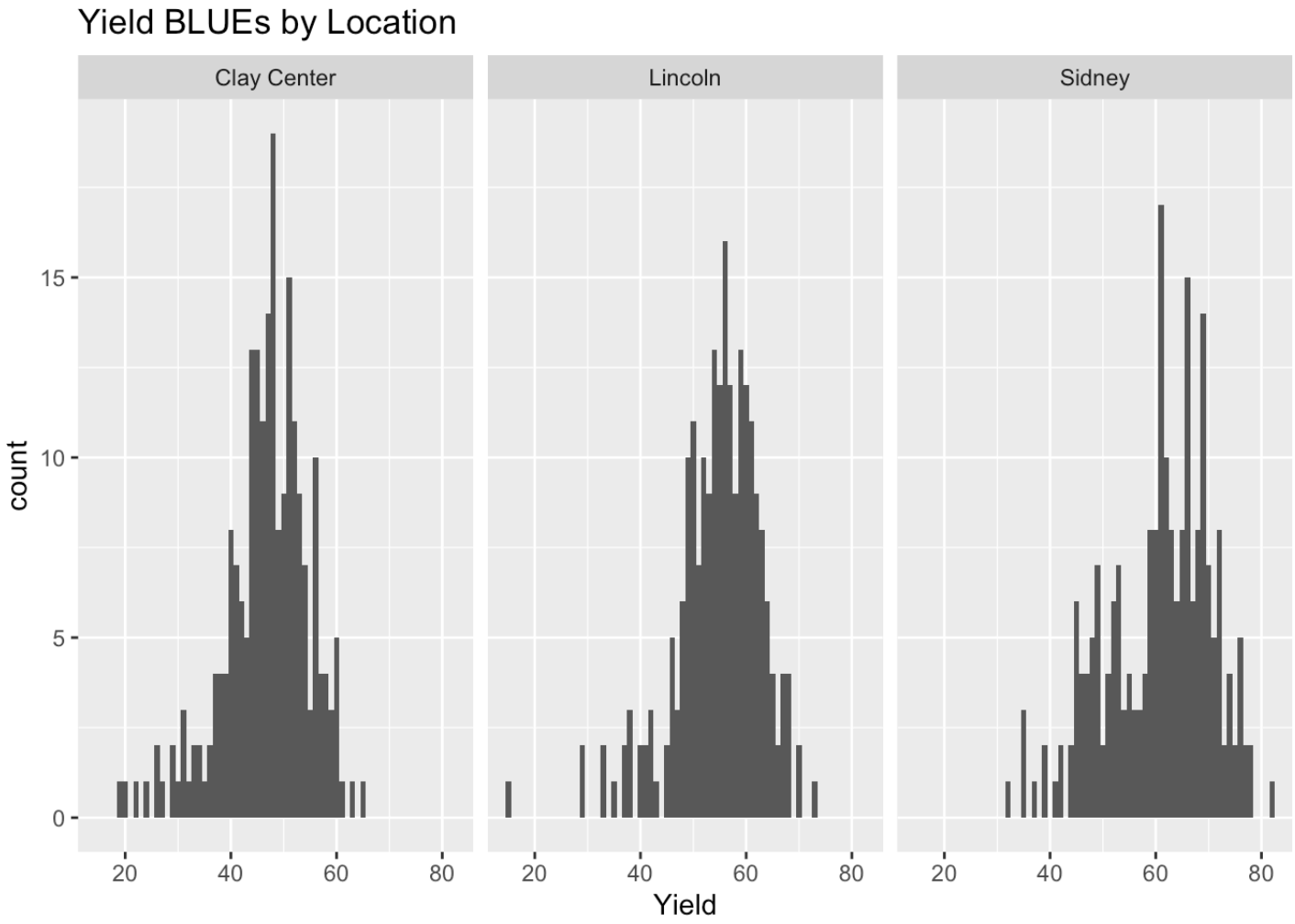
In this example we will look at the yield data. We will filter out the “Across Locations” values and focus on the individual environments for this year.

```
mydata <- mydata %>%
  filter(TRAIT %in% c("Yield"),  #filter for the yield trait
         !is.na(BLUEs), #filter to remove missing data
         !LOC %in% c("Across Locations")) #filter to remove Across Locations values
```

Distributions for traits

Let’s look at the distributions for the trait. A quick check of the data is rarely a bad idea.

```
ggplot(mydata, aes(x = BLUEs)) +
  geom_histogram(binwidth = 1) + #make a histogram of the yield data
  facet_wrap(vars(LOC)) + #wrap over locations
  labs(title = "Yield BLUEs by Location", x = "Yield") #label the plot
```

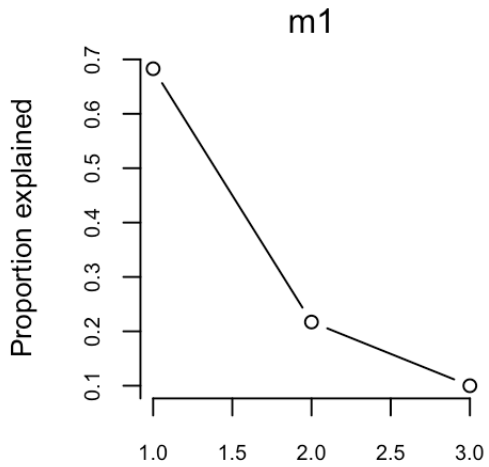


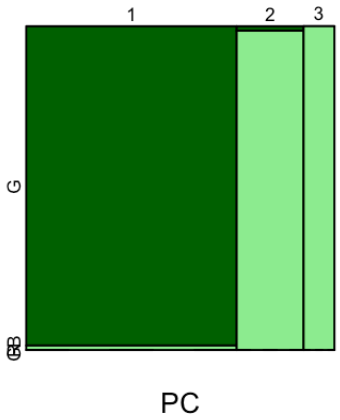
This distribution looks like we expected.

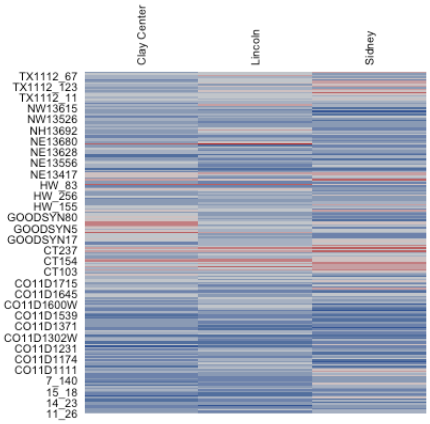
Analysis with gge package for yield

We will use the gge package to create our model for yield.
The GGEBiplots package will take the model from gge as input.

```
moddat1 <- mydata
m1 <- gge(moddat1, BLUEs ~ ID + ID * LOC, scale=F, center=T)
plot(m1)
```







In the scree plot the first couple of PCs should explain a large portion of the variation. Otherwise another technique is needed. Here the first PC accounts for a large portion of the variation.

Use package GGEBiplots

We can use this package to make plots using the gge model from above. The GGEPlot function generates the GGE biplot as an object of class 'ggplot' from a model produced by a call to either GGEModel or gge.

This package makes a number of different types of biplots. You can learn more about the package and the different types of plots here: https://cran.r-project.org/web/packages/GGEBiplots/GGEBiplots.pdf (https://cran.r-project.org/web/packages/GGEBiplots/GGEBiplots.pdf).

Type of biplot to produce (type argument in the function below)
1. Basic biplot.
2. Examine environment. See ExamineEnv
3. Examine genotype. See ExamineGen
4. Relationship among environments. See EnvRelationship
5. Compare two genotypes. See CompareGens
6. Which won where/what. See WhichWon

7. Discrimination vs. representativeness. See DiscRep
8. Ranking environments. See RankEnv
9. Mean vs. stability. See MeanStability
10. Ranking gentoypes See RankGen

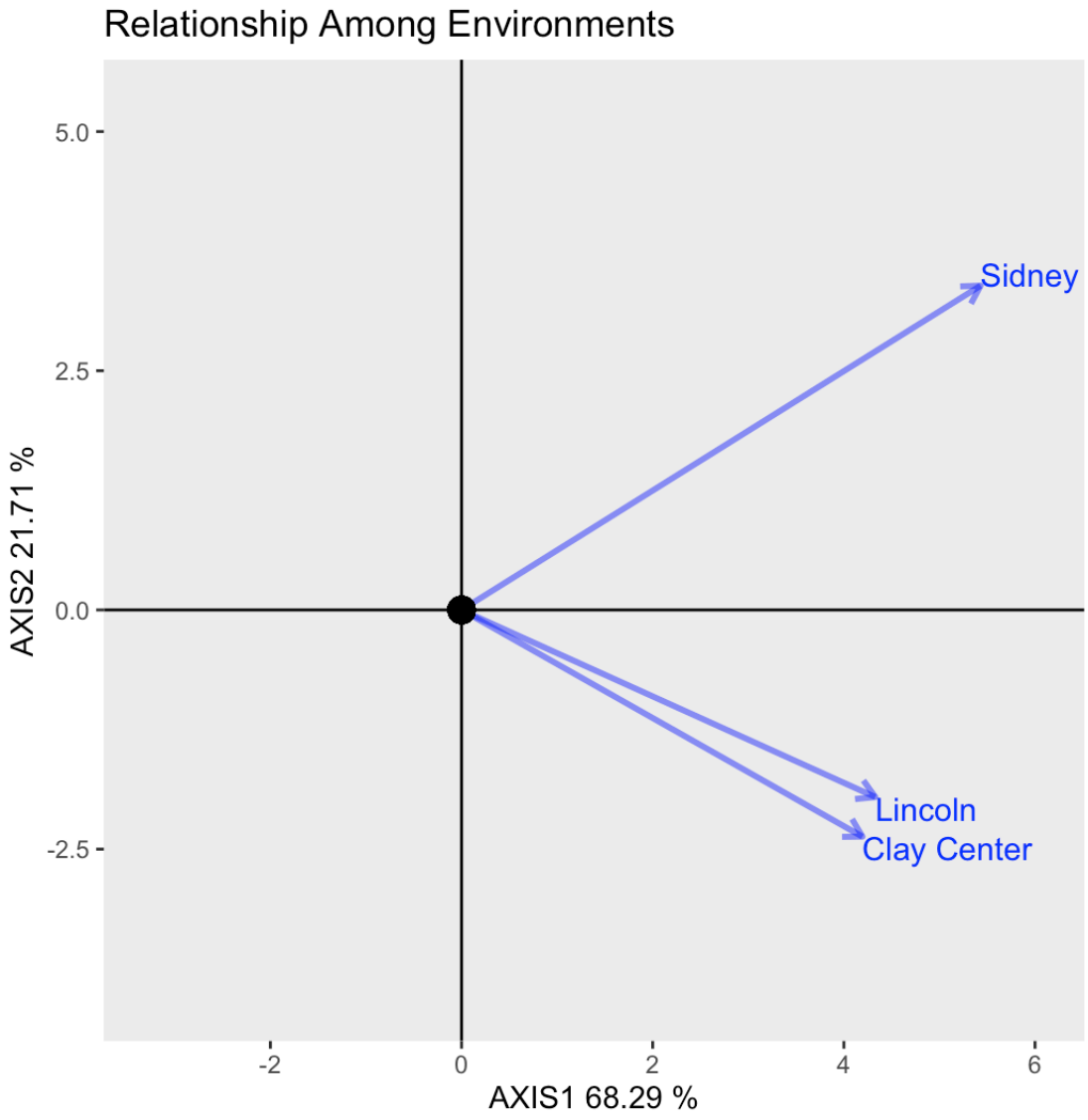
For this example we will focus on the Relationship Among Environments - type 4.

We can create this plot, and also save it to a file.

```
#relationship among environments - type 4
gge_type4 <- GGEPlot(m1, type = 4)
```

```
## Warning: Duplicated aesthetics after name standardisation: hjust and vjust
```

```
gge_type4
```



GGE Biplot showing components 1 and 2 explaining 90% of the total variation using Column Metric Preserving SVP and Tester-Centered G+GE with no scaling

```
#save this plot to a pdf file
#call the pdf command to start the plot
pdf(file = here::here("results", "YLDQTL_biplot_type4.pdf"),
    width = 6, # The width of the plot in inches
    height = 6) # The height of the plot in inches
gge_type4
#run dev.off() to create the file
dev.off()
```

```
## quartz_off_screen
##                 2
```

The biplot can show us large patterns in the data set.

The length of the arrow represents the standard deviation of means within an environment and tells us about the strength of the contribution for that environment.

The angle tells us about the correlation among the environments. If the vectors are close together in the same direction then we expect those environments to be positively correlated. If the angle is 90 degrees, then the environments are not likely to be correlated. If the vectors form a large angle, more than 90 degrees, then the environments are negatively correlated.

In this example, the vectors for Lincoln and Clay Center are close together. The angles between Lincoln and Sydney and Clay Center and Sydney are larger than the angle between Lincoln and Clay Center. The vectors are still going in the same direction, and the larger angles are less than 90 degrees.

Therefore, looking at this plot we expect a positive correlation for yield between Lincoln and Clay Center, and for each of those to have a lower positive correlation with Sydney. Lincoln should have a slightly higher correlation with Sydney than Clay Center.

If we calculate the correlations among the environments, we can see that this is true.

```
cordat <- mydata %>%
  pivot_wider(., id_cols = ID, names_from = LOC, values_from = BLUEs) #restructure th
e data for the correlation function
round(cor(cordat[,-1], use = "pairwise.complete.obs", method = "pearson"), digits = 2
) #calculate the correlation and round the result to 2 decimal places
```

```
##             Clay Center Lincoln Sidney
## Clay Center        1.00    0.65   0.45
## Lincoln            0.65    1.00   0.49
## Sidney             0.45    0.49   1.00
```

We have a positive correlation between Clay Center and Lincoln (0.65). Sydney has a lower positive correlation with the other locations, and it's slightly higher with Lincoln (0.49) than Clay Center (0.45).

Does this result make sense?

Lincoln is in the eastern part of NE. Clay Center is about 1.5 hours west of Lincoln. Lincoln and Clay Center both have a humid continental climate with cold winters and hot, humid summers. Sidney is toward the western edge of NE with a semi-arid climate with hot summers, cold winters, and low rainfall. Based on the geography and climate, we might expect Sidney to be very different from Lincoln and Clay Center, so our data and results make sense.

Session Info

```
sessionInfo()
```

```
## R version 4.1.2 (2021-11-01)
## Platform: aarch64-apple-darwin20 (64-bit)
## Running under: macOS 13.4.1
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRblas.
0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1-arm64/Resources/lib/libRlapac
k.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] here_1.0.1       GGEBiplots_0.1.3 gge_1.7          forcats_0.5.1
##  [5] stringr_1.5.0    dplyr_1.1.2      purrr_1.0.1      readr_2.1.0
##  [9] tidyr_1.3.0      tibble_3.2.1     ggplot2_3.4.2    tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] httr_1.4.2       sass_0.4.1       bit64_4.0.5      vroom_1.5.6
##  [5] jsonlite_1.7.2   modelr_0.1.8     bslib_0.3.1      assertthat_0.2.1
##  [9] nipals_0.8       highr_0.9        cellranger_1.1.0 yaml_2.2.1
## [13] pillar_1.9.0     backports_1.3.0  glue_1.6.2       digest_0.6.29
## [17] polyclip_1.10-0  rvest_1.0.2      colorspace_2.0-2 htmltools_0.5.3
## [21] plyr_1.8.6       pkgconfig_2.0.3  broom_1.0.4      haven_2.4.3
## [25] scales_1.2.1     tweenr_1.0.2     tzdb_0.3.0       ggforce_0.3.3
## [29] generics_0.1.3   farver_2.1.0     ellipsis_0.3.2   withr_2.5.0
## [33] cli_3.6.1        magrittr_2.0.3   crayon_1.4.2     readxl_1.3.1
## [37] evaluate_0.14    fs_1.6.1         fansi_1.0.4      MASS_7.3-54
## [41] xml2_1.3.2       tools_4.1.2      hms_1.1.1        lifecycle_1.0.3
## [45] munsell_0.5.0    reprex_2.0.1     compiler_4.1.2   jquerylib_0.1.4
## [49] rlang_1.1.1      grid_4.1.2       rstudioapi_0.14  labeling_0.4.2
## [53] rmarkdown_2.11   gtable_0.3.0     DBI_1.1.1        reshape2_1.4.4
## [57] R6_2.5.1         lubridate_1.8.0  knitr_1.36       fastmap_1.1.0
## [61] bit_4.0.4        utf8_1.2.3       rprojroot_2.0.2  stringi_1.7.5
## [65] parallel_4.1.2   Rcpp_1.0.7       vctrs_0.6.2      dbplyr_2.1.1
## [69] tidyselect_1.2.0 xfun_0.28
```