Data cleaning with R

Jeanette Lyerly

7/11/2023

Purpose

In this markdown we will import and clean some example data with R as part of the Wheat CAP Genomic Selection Workshop, July 2023, Raleigh, NC.

Our goals in this example are to:
- Import some data
- Look at the data and calculate some summaries
- Identify issues with the data and do any cleaning
- Export a new data file for downstream analysis

These data sets are based on data provided by Jean-Luc Jannink (downloaded from T3).

Data cleaning

What is data cleaning? When we talk about data cleaning we want to be sure our data is 1) correct, 2) consistent, and 3) useable. Don't underestimate the importance of data cleaning, or the time it will take to do this.

garbage in = garbage out

Common issues that we run into in our data cleaning:
Special characters
Inaccuracies/data entry errors
White spaces
Zeros instead of null values

Keep tidy data principles in mind:
- Each variable must have its own column
- Each observation must have its own row
- Each value must have its own cell

Background

Our data scenario begins with some data from breeder(s). It's common (for us) to receive data in a spreadsheet from each person/location.
In our example we have data coming in from four locations: Ithaca, Sydney, Lincoln, and Clay Center. Each

location has a data sheet.
At the end we want to have one combined data set that we can use for downstream analysis.

Load libraries

We are going to use the tidyverse and a couple of functions from the janitor package to work with this data.

```
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(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(here)
```

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

Import data

As data files have been sent from the breeder(s), we have stored them in a folder called “data_raw”. At this point we haven’t really looked at these, just put them in a folder as they arrived. We did use descriptive names so we know what the files are.

Let's import some data.
We will create a list of the files we have in the folder, then import them into individual data frames.

```
#make a list of the files
file_list <- list.files(path = here::here("data_raw"), pattern='*.csv')
file_list
```

```
## [1] "YLDQTLVAL_2014_NECC.csv" "YLDQTLVAL_2014_NEI.csv"
## [3] "YLDQTLVAL_2014_NEL.csv"  "YLDQTLVAL_2014_NES.csv"
```

```
#since we only have a few files, and we're assuming we haven't really looked at them, let's read them in separately
df1 <- read_csv(file = here::here("data_raw", file_list[1]), col_names = T)
```

```
## Rows: 440 Columns: 16
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (7): PROGRAM, STUDY, DESCRIPTION, DESIGN, LOC, ID, TYPE
## dbl (8): YR, REP, BLOCK, PLOT, ROW, COL, YLD, HT
## lgl (1): NOTES
##
## 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.
```

```
df2 <- read_csv(file = here::here("data_raw", file_list[2]), col_names = T)
```

```
## Rows: 418 Columns: 17
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (7): PROGRAM, STUDY, DESCRIPTION, DESIGN, LOC, ID, TYPE
## dbl (9): YR, REP, BLOCK, PLOT, ROW, COL, TW, YLD, HT
## lgl (1): NOTES
##
## 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.
```

```
df3 <- read_csv(file = here::here("data_raw", file_list[3]), col_names = T)
```

```
## Rows: 440 Columns: 17
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (7): PROGRAM, STUDY, DESCRIPTION, DESIGN, LOC, VAR, TYPE
## dbl (9): YEAR, REP, BLOCK, PLOT, ROW, COL, TWT, YIELD, PLTHT
## lgl (1): NOTES
##
## 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.
```

```
df4 <- read_csv(file = here::here("data_raw", file_list[4]), col_names = T)
```

```
## Rows: 440 Columns: 17
## ── Column specification ────────────────────────────────────────────────────────
## Delimiter: ","
## chr (7): PROGRAM, STUDY, DESCRIPTION, DESIGN, LOC, ID, TYPE
## dbl (9): YR, REP, BLOCK, PLOT, ROW, COL, TW, YLD, HT
## lgl (1): NOTES
##
## 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.
```

Looking at the file list we see four files, one per location.

Look over our imported data

Now we have four data frames, each with data from a location. Let's take a look at them.

```
glimpse(df1)
```

```
## Rows: 440
## Columns: 16
## $ YR          <dbl> 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014…
## $ PROGRAM     <chr> "University of Nebraska", "University of Nebraska", "Unive…
## $ STUDY       <chr> "YldQtl-Val_2014_ClayCenter", "YldQtl-Val_2014_ClayCenter"…
## $ DESCRIPTION <chr> "Yield QTL Validation, Yield QTL Validation,", "Yield QTL …
## $ DESIGN      <chr> "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "C…
## $ LOC         <chr> "Clay Center, NE", "Clay Center, NE", "Clay Center, NE", "…
## $ ID          <chr> "11_26", "11_73", "13_13", "13_59", "14_10", "14_2", "14_2…
## $ TYPE        <chr> "plot", "plot", "plot", "plot", "plot", "plot", "plot", "p…
## $ REP         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ BLOCK       <dbl> 10, 10, 20, 20, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 9, 9, 16, 16…
## $ PLOT        <dbl> 2097, 2098, 2191, 2192, 2035, 2032, 2039, 2038, 2037, 2033…
## $ ROW         <dbl> 10, 10, 20, 20, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 9, 9, 16, 16…
## $ COL         <dbl> 40, 41, 34, 35, 38, 35, 42, 41, 40, 36, 34, 39, 37, 43, 36…
## $ YLD         <dbl> 3773, 3679, 3295, 3423, 4654, 4459, 4277, 4096, 5071, 4708…
## $ HT          <dbl> 84, 84, 74, 78, 82, 82, 82, 80, 82, 92, 80, 80, 88, 82, 82…
## $ NOTES       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
```

```
glimpse(df2)
```

```
## Rows: 418
## Columns: 17
## $ YR          <dbl> 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014…
## $ PROGRAM     <chr> "University of Nebraska", "University of Nebraska", "Unive…
## $ STUDY       <chr> "YldQtl-Val_2014_Mead", "YldQtl-Val_2014_Mead", "YldQtl-Va…
## $ DESCRIPTION <chr> "Yield QTL Validation, Yield QTL Validation, Planting erro…
## $ DESIGN      <chr> "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "C…
## $ LOC         <chr> "Ithaca, NE", "Ithaca, NE", "Ithaca, NE", "Ithaca, NE", "I…
## $ ID          <chr> "11_26", "11_73", "13_59", "14_10", "14_2", "14_23", "14_3…
## $ TYPE        <chr> "plot", "plot", "plot", "plot", "plot", "plot", "plot", "p…
## $ REP         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ BLOCK       <dbl> 17, 17, 19, 8, 8, 8, 8, 8, 8, 8, 8, 8, 1, 1, 17, 17, 5, 5,…
## $ PLOT        <dbl> 2170, 2169, 2182, 2078, 2073, 2074, 2076, 2080, 2075, 2072…
## $ ROW         <dbl> 17, 17, 19, 8, 8, 8, 8, 8, 8, 8, 8, 8, 1, 1, 17, 17, 5, 5,…
## $ COL         <dbl> 10, 9, 2, 8, 3, 4, 6, 10, 5, 2, 9, 7, 3, 4, 5, 6, 9, 6, 7,…
## $ TW          <dbl> 39, 26, 51, 51, 39, 39, 39, 64, 39, 51, 51, 39, 64, 77, 26…
## $ YLD         <dbl> 11903, 12038, 12105, 12307, 11971, 12307, 12172, 11836, 12…
## $ HT          <dbl> 80, 82, 80, 90, 85, 83, 89, 84, 84, 83, 90, 89, 85, 85, 78…
## $ NOTES       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
```

```
glimpse(df3)
```

```
## Rows: 440
## Columns: 17
## $ YEAR        <dbl> 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014…
## $ PROGRAM     <chr> "University of Nebraska", "University of Nebraska", "Unive…
## $ STUDY       <chr> "YldQtl-Val_2014_Lincoln", "YldQtl-Val_2014_Lincoln", "Yld…
## $ DESCRIPTION <chr> "Yield QTL Validation, Yield QTL Validation,", "Yield QTL …
## $ DESIGN      <chr> "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "C…
## $ LOC         <chr> "Lincoln, NE", "Lincoln, NE", "Lincoln, NE", "Lincoln, NE"…
## $ VAR         <chr> "11_26", "11_73", "13_13", "13_59", "14_10", "14_2", "14_2…
## $ TYPE        <chr> "plot", "plot", "plot", "plot", "plot", "plot", "plot", "p…
## $ REP         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ BLOCK       <dbl> 1, 1, 19, 19, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 4, 4…
## $ PLOT        <dbl> 2009, 2010, 2187, 2188, 2097, 2091, 2092, 2093, 2099, 2100…
## $ ROW         <dbl> 1, 1, 19, 19, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 4, 4…
## $ COL         <dbl> 39, 40, 37, 38, 37, 31, 32, 33, 39, 40, 38, 35, 36, 34, 35…
## $ TWT         <dbl> 811, 803, 774, 763, 803, 817, 780, 786, 790, 792, 798, 786…
## $ YIELD       <dbl> 4398, 4452, 4398, 5138, 5219, 4748, 5037, 4687, 5239, 4842…
## $ PLTHT       <dbl> 33.07, 33.86, 37.80, 37.01, 33.86, 35.43, 33.86, 29.92, 33…
## $ NOTES       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
```

```
glimpse(df4)
```

```
## Rows: 440
## Columns: 17
## $ YR          <dbl> 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014…
## $ PROGRAM     <chr> "University of Nebraska", "University of Nebraska", "Unive…
## $ STUDY       <chr> "YldQtl-Val_2014_Sidney", "YldQtl-Val_2014_Sidney", "YldQt…
## $ DESCRIPTION <chr> "Yield QTL Validation, Yield QTL Validation,", "Yield QTL …
## $ DESIGN      <chr> "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "C…
## $ LOC         <chr> "Sidney, NE", "Sidney, NE", "Sidney, NE", "Sidney, NE", "S…
## $ ID          <chr> "11_26", "11_73", "13_13", "13_59", "14_10", "14_2", "14_2…
## $ TYPE        <chr> "plot", "plot", "plot", "plot", "plot", "plot", "plot", "p…
## $ REP         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ BLOCK       <dbl> 16, 16, 11, 11, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 19, 19, 16, …
## $ PLOT        <dbl> 2158, 2157, 2101, 2102, 2058, 2059, 2051, 2053, 2056, 2054…
## $ ROW         <dbl> 16, 16, 11, 11, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 19, 19, 16, …
## $ COL         <dbl> 8, 7, 1, 2, 8, 9, 1, 3, 6, 4, 2, 10, 7, 5, 6, 5, 9, 10, 10…
## $ TW          <dbl> 842, 842, 839, 830, 773, 813, 750, 828, 812, 808, 822, 797…
## $ YLD         <dbl> 4613, 4069, 5111, 5252, 4022, 4365, 4943, 5467, 4358, 5360…
## $ HT          <dbl> 78, 82, 78, 80, 68, 76, 79, 80, 66, 80, 78, 76, 74, 80, 78…
## $ NOTES       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
```

We have a number of variables with different types in here. Some are character and some are numeric.

The first location has phenotype data for yield and height. The other locations have data for yield, test weight, and height.

One of the common issues we discussed with data cleaning was differences in trait names or abbreviations. Are the column headers in all the data frames the same?

```
#use the function from the janitor package to compare the columns
compare_cols <- compare_df_cols(df1, df2, df3, df4) %>%
  arrange(column_name)
compare_cols
```

```
##    column_name       df1       df2       df3       df4
## 1        BLOCK   numeric   numeric   numeric   numeric
## 2          COL   numeric   numeric   numeric   numeric
## 3  DESCRIPTION character character character character
## 4       DESIGN character character character character
## 5           HT   numeric   numeric      <NA>   numeric
## 6           ID character character      <NA> character
## 7          LOC character character character character
## 8        NOTES   logical   logical   logical   logical
## 9         PLOT   numeric   numeric   numeric   numeric
## 10       PLTHT      <NA>      <NA>   numeric      <NA>
## 11     PROGRAM character character character character
## 12         REP   numeric   numeric   numeric   numeric
## 13         ROW   numeric   numeric   numeric   numeric
## 14       STUDY character character character character
## 15          TW      <NA>   numeric      <NA>   numeric
## 16         TWT      <NA>      <NA>   numeric      <NA>
## 17        TYPE character character character character
## 18         VAR      <NA>      <NA> character      <NA>
## 19        YEAR      <NA>      <NA>   numeric      <NA>
## 20       YIELD      <NA>      <NA>   numeric      <NA>
## 21         YLD   numeric   numeric      <NA>   numeric
## 22          YR   numeric   numeric      <NA>   numeric
```

```
#look at the mismatched ones
compare_df_cols(df1, df2, df3, df4, return = "mismatch", bind_method = "rbind")
```

```
##    column_name       df1       df2       df3       df4
## 1           HT   numeric   numeric      <NA>   numeric
## 2           ID character character      <NA> character
## 3        PLTHT      <NA>      <NA>   numeric      <NA>
## 4           TW      <NA>   numeric      <NA>   numeric
## 5          TWT      <NA>      <NA>   numeric      <NA>
## 6          VAR      <NA>      <NA> character      <NA>
## 7         YEAR      <NA>      <NA>   numeric      <NA>
## 8        YIELD      <NA>      <NA>   numeric      <NA>
## 9          YLD   numeric   numeric      <NA>   numeric
## 10          YR   numeric   numeric      <NA>   numeric
```

We can see some differences in our column names right away. It looks like the data sheet for location three is a bit different from the others.
YR and YEAR
ID and VAR
TW and TWT
YLD and YIELD
HT and PLTHT

Adjust these headers and create a data set.
When using the rename function the syntax is new name = old name.

```
#make a list of our data frames
df_list <- list(df1, df2, df3, df4)

#bind them into a raw data set, renaming the columns as needed
#rename is new name = old name
raw_data <- map(df_list, ~ rename(.x, any_of(c("YR" = "YEAR", "ID" = "VAR", "TW" = "T
WT", "YLD" = "YIELD", "HT" = "PLTHT")))) %>%
  bind_rows()

#check the data - is this what you expected?
glimpse(raw_data)
```

```
## Rows: 1,738
## Columns: 17
## $ YR          <dbl> 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014, 2014…
## $ PROGRAM     <chr> "University of Nebraska", "University of Nebraska", "Unive…
## $ STUDY       <chr> "YldQtl-Val_2014_ClayCenter", "YldQtl-Val_2014_ClayCenter"…
## $ DESCRIPTION <chr> "Yield QTL Validation, Yield QTL Validation,", "Yield QTL …
## $ DESIGN      <chr> "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "CRD", "C…
## $ LOC         <chr> "Clay Center, NE", "Clay Center, NE", "Clay Center, NE", "…
## $ ID          <chr> "11_26", "11_73", "13_13", "13_59", "14_10", "14_2", "14_2…
## $ TYPE        <chr> "plot", "plot", "plot", "plot", "plot", "plot", "plot", "p…
## $ REP         <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1…
## $ BLOCK       <dbl> 10, 10, 20, 20, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 9, 9, 16, 16…
## $ PLOT        <dbl> 2097, 2098, 2191, 2192, 2035, 2032, 2039, 2038, 2037, 2033…
## $ ROW         <dbl> 10, 10, 20, 20, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 9, 9, 16, 16…
## $ COL         <dbl> 40, 41, 34, 35, 38, 35, 42, 41, 40, 36, 34, 39, 37, 43, 36…
## $ YLD         <dbl> 3773, 3679, 3295, 3423, 4654, 4459, 4277, 4096, 5071, 4708…
## $ HT          <dbl> 84, 84, 74, 78, 82, 82, 82, 80, 82, 92, 80, 80, 88, 82, 82…
## $ NOTES       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
## $ TW          <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA…
```

This looks a bit better. We have only one YEAR, TW, YLD, and HT. We've also resolved the ID and VAR column names and all our samples have ID.

```
#look at the data
view(raw_data)
```

If we look at the data we can see variables for year, the study name and some descriptive notes, information about the location, the ID of the sample, and our phenotype data.
We've got phenotype data for yield, test weight, and plant height.

Next let's check the number of samples in each location and rep.

A side note about pipes If you are not familiar with the tidyverse packages, you may not be familiar with the pipe. The pipe, %>%, comes from the magrittr package by Stefan Milton Bache. Packages in the tidyverse load %>% for you automatically. For a more detailed explanation of pipes and how they work see https://r4ds.had.co.nz/pipes.html (https://r4ds.had.co.nz/pipes.html). If you are reading the code aloud, the pipe can be read as "and then".

In the code below we are asking to take the raw data, and then group it by location and rep, and then count the number of samples.

```
#check the number of lines in each location and rep
raw_data %>%
  group_by(LOC, REP) %>%
  count()
```

```
## # A tibble: 8 × 3
## # Groups:   LOC, REP [8]
##   LOC               REP     n
##   <chr>           <dbl> <int>
## 1 Clay Center, NE     1   220
## 2 Clay Center, NE     2   220
## 3 Ithaca, NE          1   198
## 4 Ithaca, NE          2   220
## 5 Lincoln, NE         1   220
## 6 Lincoln, NE         2   220
## 7 Sidney, NE          1   220
## 8 Sidney, NE          2   220
```

We have four locations, each with 2 reps.
For the location variable, note that we have more than one thing in a cell.

There are some notes from the breeder in the DESCRIPTION field.

```
#what is in the notes? get the unique values
raw_data %>%
  select(DESCRIPTION) %>%
  unique(.)
```

```
## # A tibble: 2 × 1
##   DESCRIPTION
##   <chr>
## 1 Yield QTL Validation, Yield QTL Validation,
## 2 Yield QTL Validation, Yield QTL Validation, Planting error in 1st pass result…
```

When we read over this we can see that we have a note from the program that there was a mistake and some data should be discarded.

Next we can run a quick count of the sample names - here this is the ID variable. Based on our locations and reps we expect to see each sample eight times.

```
#run a count of the sample names
id_counts <- raw_data %>%
  group_by(ID) %>%
  count(name = "sample_count") %>%
  arrange(desc(sample_count))
head(id_counts) #show the top of the id_counts data frame
```

```
## # A tibble: 6 × 2
## # Groups:   ID [6]
##   ID    sample_count
##   <chr>        <int>
## 1 11_26            8
## 2 11_73            8
## 3 13_59            8
## 4 14_10            8
## 5 14_2             8
## 6 14_23            8
```

```
tail(id_counts) #show the bottom of the id_counts data frame
```

```
## # A tibble: 6 × 2
## # Groups:   ID [6]
##   ID         sample_count
##   <chr>             <int>
## 1 NE13695               7
## 2 NH13606               7
## 3 NW13575               7
## 4 NW13652               7
## 5 TX1112_11             7
## 6 TX1112_111            7
```

These samples are present 7-8 times, which makes sense.

Based on our initial assessment we have some cleaning to do.

Data processing

Let's tackle these issues from our data set.

1. Fix the location to have one thing in a cell.
2. Remove the data that the breeder said to discard.

```
#start a new data frame with our processed data
mydata <- raw_data

dim(mydata) #1738 rows and 17 columns
```

```
## [1] 1738   17
```

```
#for location we have more than one thing in a cell
#separate into variables
mydata <- mydata %>%
  separate(LOC, into = c("LOC", "STATE"), sep = ",")

#filter the data to remove anything with the text note "Data discarded"
#the ! = not, so here we are asking to filter for data where "Data discarded" is NOT
in the DESCRIPTION
mydata <- mydata %>%
  filter(!str_detect(DESCRIPTION, "Data discarded"))

dim(mydata) #1320 rows and 18 columns
```

```
## [1] 1320   18
```

```
#look at what we have now
mydata %>%
  group_by(LOC, REP) %>%
  count()
```

```
## # A tibble: 6 × 3
## # Groups:   LOC, REP [6]
##   LOC           REP     n
##   <chr>       <dbl> <int>
## 1 Clay Center     1   220
## 2 Clay Center     2   220
## 3 Lincoln         1   220
## 4 Lincoln         2   220
## 5 Sidney          1   220
## 6 Sidney          2   220
```

We've dropped one location by removing the "discard" data. Now we have three locations with two reps per location, and 220 samples per rep.

Create some summaries and visuals for the data

We will select the variables that we are interested in and pivot to long format data.
Then we will calculate the mean and sd for each trait in each location.

```
#select some columns and pivot the data
mydata <- mydata %>%
  select(YR, PROGRAM, LOC, STATE, ID, REP, BLOCK, ROW, COL, PLOT, YLD, TW, HT) %>%
  pivot_longer(., cols = c("YLD", "TW", "HT"), names_to = "Trait", values_to = "Measu
rement")

#calculate the means for the traits by location
mydata %>%
  group_by(LOC, Trait) %>%
  summarize(mean = mean(Measurement), sd = sd(Measurement)) %>%
  arrange(LOC, Trait)
```

```
## `summarise()` has grouped output by 'LOC'. You can override using the `.groups`
## argument.
```

```
## # A tibble: 9 × 4
## # Groups:   LOC [3]
##   LOC         Trait   mean     sd
##   <chr>       <chr>  <dbl>  <dbl>
## 1 Clay Center HT      81.7   6.99
## 2 Clay Center TW      NA    NA
## 3 Clay Center YLD   3242.  709.
## 4 Lincoln     HT      35.6   3.45
## 5 Lincoln     TW     769.   69.0
## 6 Lincoln     YLD   4498.  651.
## 7 Sidney      HT      81.0   9.20
## 8 Sidney      TW     787.   31.5
## 9 Sidney      YLD   4172.  939.
```

One location is missing test weight data, which we already knew from looking at our data before.

Looks like something's up with our height data. The mean height for Lincoln is much lower than the mean height for Clay Center or Sidney.

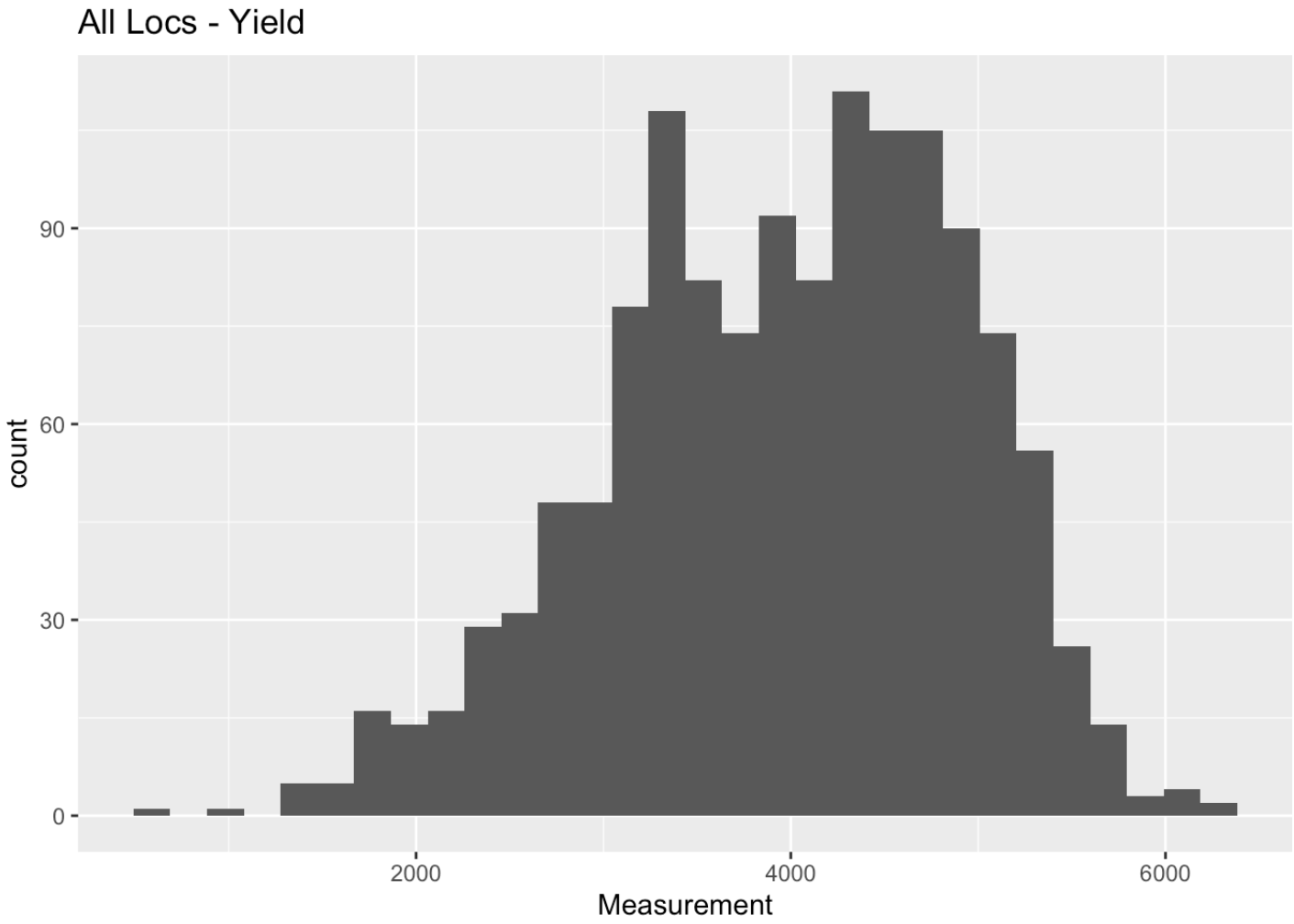
Look at the distributions of the data.

This can tell us about general trends and help us see any obvious problems.

Distribution for yield - create a histogram for all the data, and then for each location.

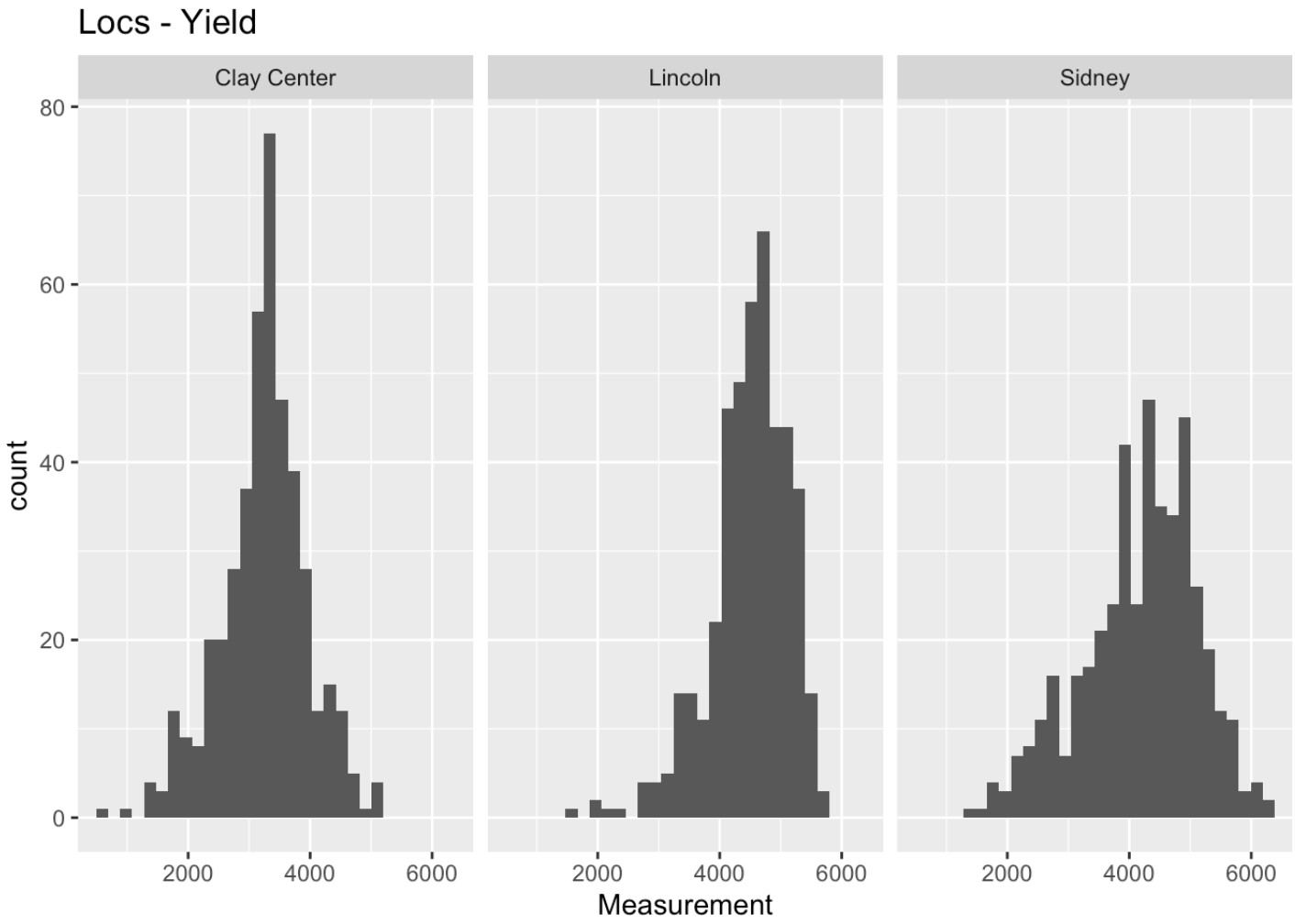
```
#histogram with all the data over locations
ggplot(mydata %>% filter(Trait == "YLD"), aes(x = Measurement)) +
  geom_histogram() +
  labs(title = "All Locs - Yield")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#facet over locations
ggplot(mydata %>% filter(Trait == "YLD"), aes(x = Measurement)) +
  geom_histogram() +
  facet_wrap(vars(LOC)) +
  labs(title = "Locs - Yield")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

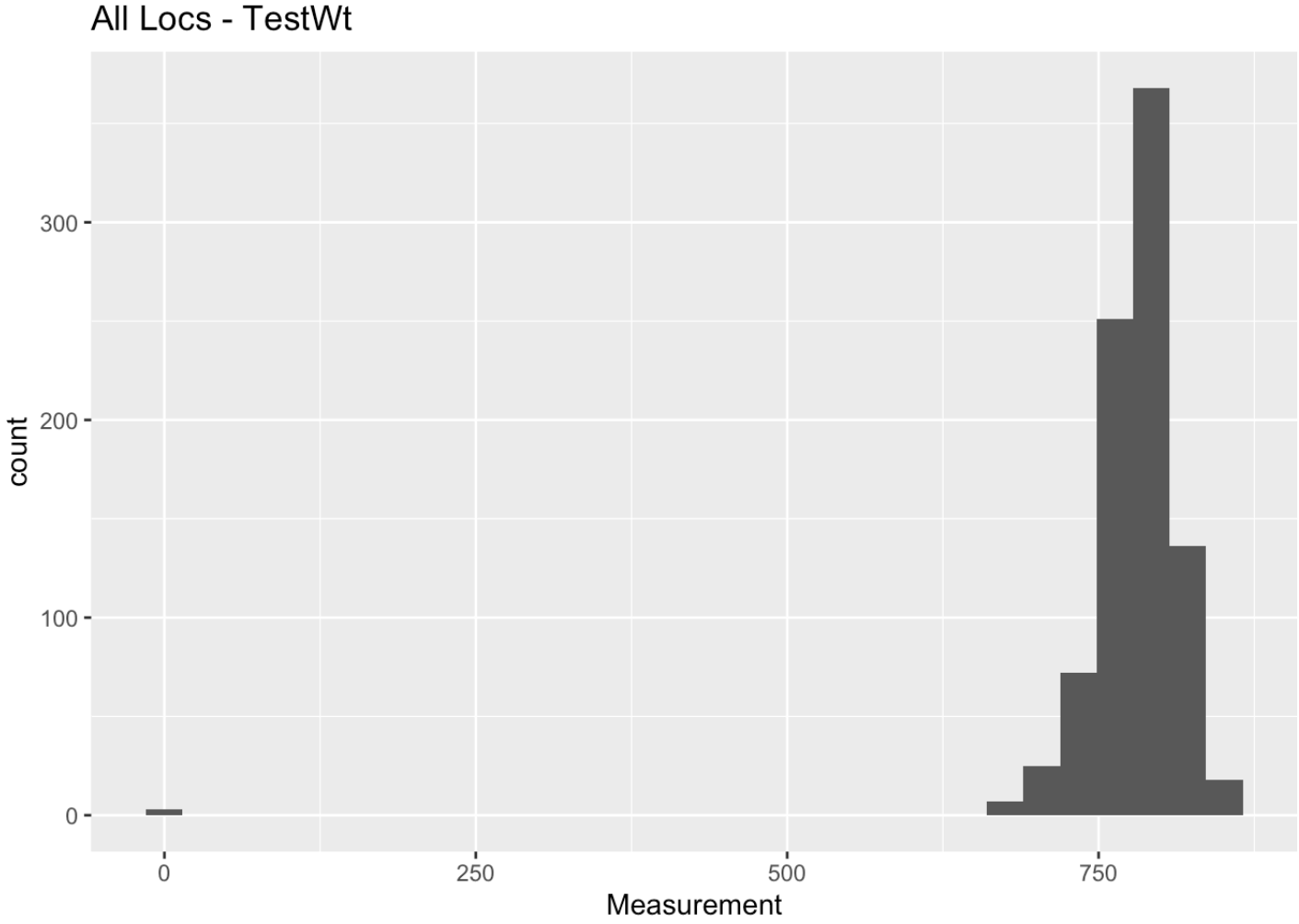


This looks pretty good for yield.

Distribution for test weight - same thing.

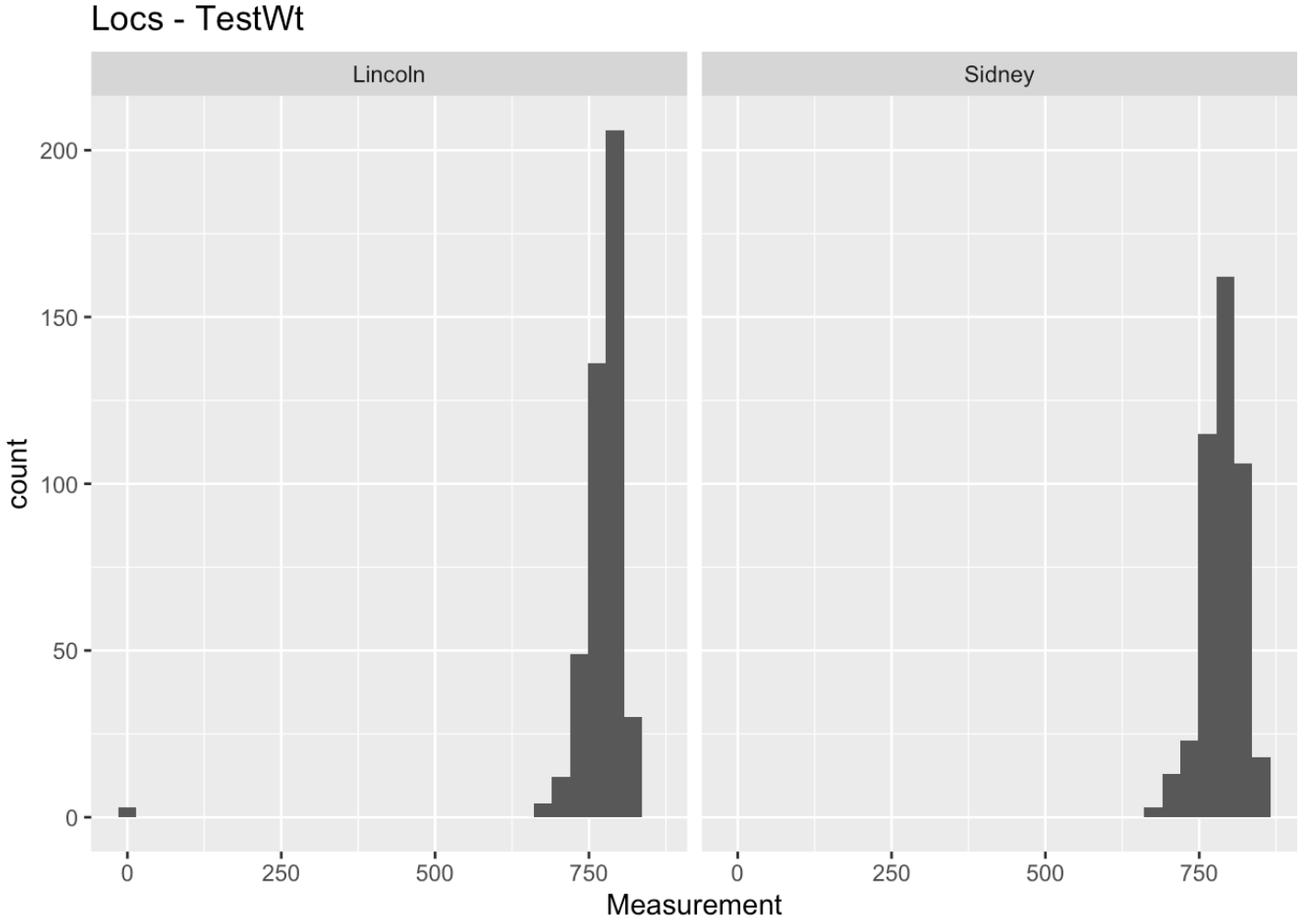
```
#histogram with all the data over locations
#if we run this without dropping the missing location data we get a warning
#we can add to our filter statement to drop the missing values
ggplot(mydata %>% filter(Trait == "TW", !is.na(Measurement)), aes(x = Measurement)) +
  geom_histogram() +
  labs(title = "All Locs - TestWt")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#facet over locations
ggplot(mydata %>% filter(Trait == "TW", !is.na(Measurement)), aes(x = Measurement)) +
  geom_histogram() +
  facet_wrap(vars(LOC)) +
  labs(title = "Locs - TestWt")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



There are a couple of zero values in here for the Lincoln location. Zero may sneak in as missing data. Investigate these IDs to see what is going on here. Sometimes zero is meaningful.

First, let's get the low test weight data and see which samples these are.

```
low_tw <- mydata %>%
  filter(Trait == "TW" & Measurement < 10) %>%
  select(LOC, ID, Trait, Measurement)
low_tw
```

```
## # A tibble: 3 × 4
##   LOC     ID    Trait Measurement
##   <chr>   <chr> <chr>       <dbl>
## 1 Lincoln 14_83 TW              0
## 2 Lincoln 14_10 TW              0
## 3 Lincoln CT199 TW              0
```

```
#we have three zero values
```

We have three IDs with 0 for test weight in the Lincoln location.

```
#is there data for these IDs for the other variables?
low_tw <- mydata %>%
  filter(LOC %in% "Lincoln" & ID %in% low_tw$ID) %>%
  select(LOC, REP, ID, Trait, Measurement) %>%
  arrange(Trait, ID)
low_tw
```

```
## # A tibble: 18 × 5
##    LOC       REP ID    Trait Measurement
##    <chr>   <dbl> <chr> <chr>       <dbl>
##  1 Lincoln     1 14_10 HT           33.9
##  2 Lincoln     2 14_10 HT           33.9
##  3 Lincoln     1 14_83 HT           35.4
##  4 Lincoln     2 14_83 HT           37.0
##  5 Lincoln     1 CT199 HT           33.1
##  6 Lincoln     2 CT199 HT           34.2
##  7 Lincoln     1 14_10 TW          803
##  8 Lincoln     2 14_10 TW            0
##  9 Lincoln     1 14_83 TW            0
## 10 Lincoln     2 14_83 TW          790
## 11 Lincoln     1 CT199 TW          781
## 12 Lincoln     2 CT199 TW            0
## 13 Lincoln     1 14_10 YLD        5219
## 14 Lincoln     2 14_10 YLD        4849
## 15 Lincoln     1 14_83 YLD        5548
## 16 Lincoln     2 14_83 YLD        5172
## 17 Lincoln     1 CT199 YLD        4048
## 18 Lincoln     2 CT199 YLD        4082
```

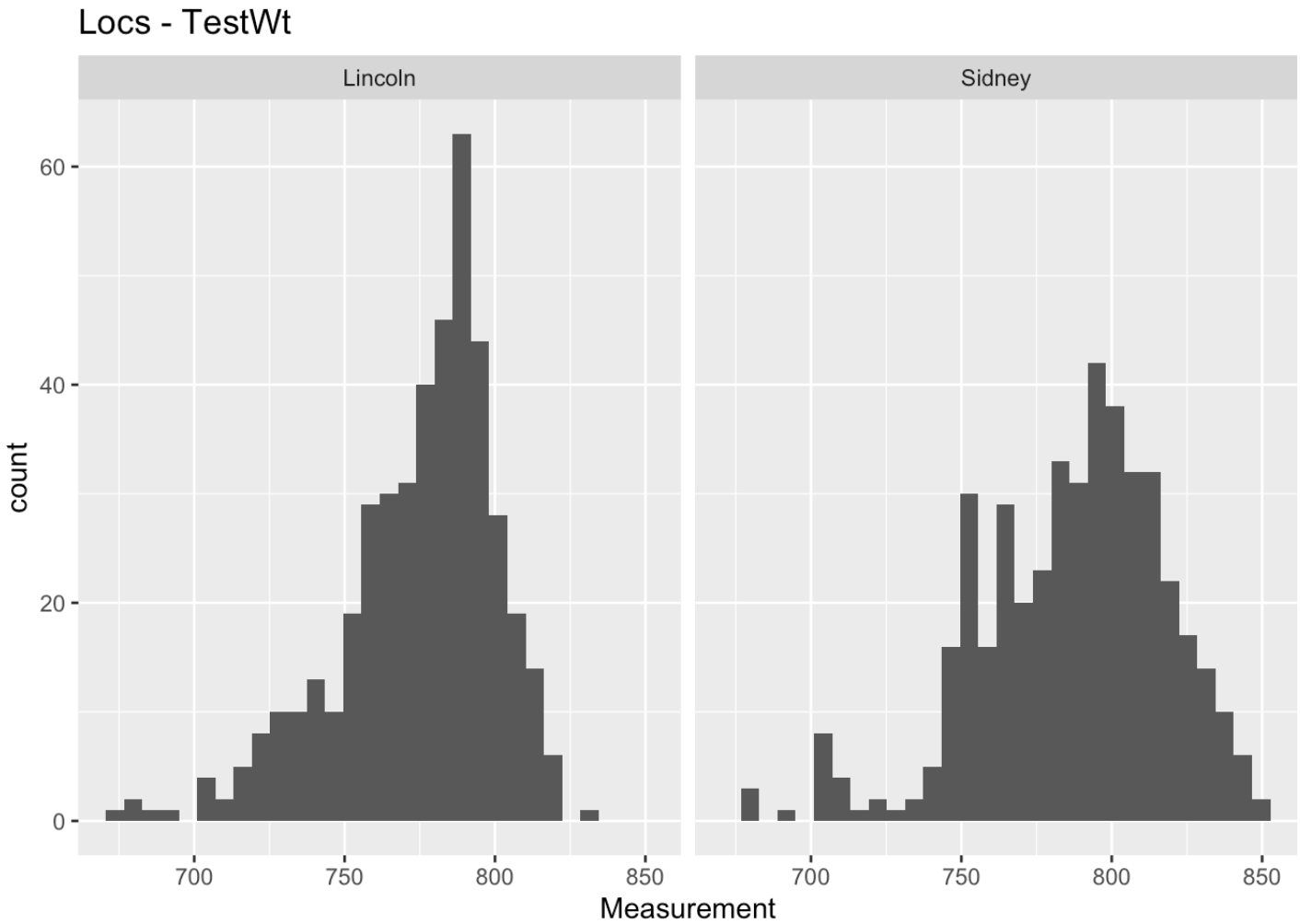
```
#there is yield and height data for these low tw IDs
```

If we get all the data for these IDs in Lincoln we can see that there is yield and height data. It doesn't make sense to have a value for yield with 0 test weight, so it's likely that 0 has been used in this location as the code for missing data (always a good idea to double check).

```
#replace these 0 values with missing data
mydata <- mydata %>%
  mutate(Measurement = ifelse(Trait %in% c("TW") & Measurement == 0, NA, Measurement)
) #replace with NA

#recheck
ggplot(mydata %>% filter(Trait == "TW", !is.na(Measurement)), aes(x = Measurement)) +
  geom_histogram() +
  facet_wrap(vars(LOC)) +
  labs(title = "Locs - TestWt")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

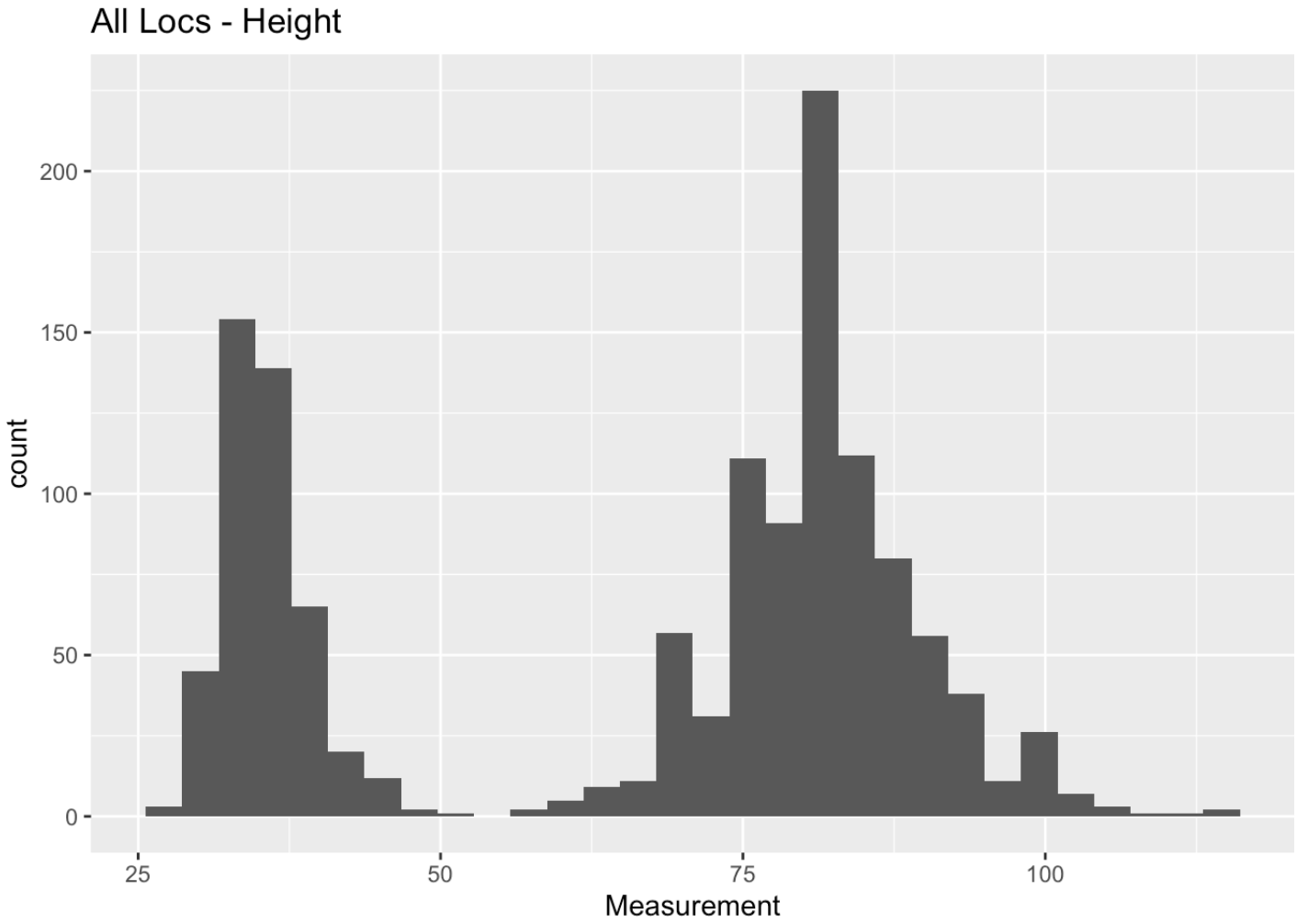


That looks better. The zero values are gone and overall this looks ok.

Distribution for height.

```
#histogram with all the data over locations
ggplot(mydata %>% filter(Trait == "HT"), aes(x = Measurement)) +
  geom_histogram() +
  labs(title = "All Locs - Height")
```

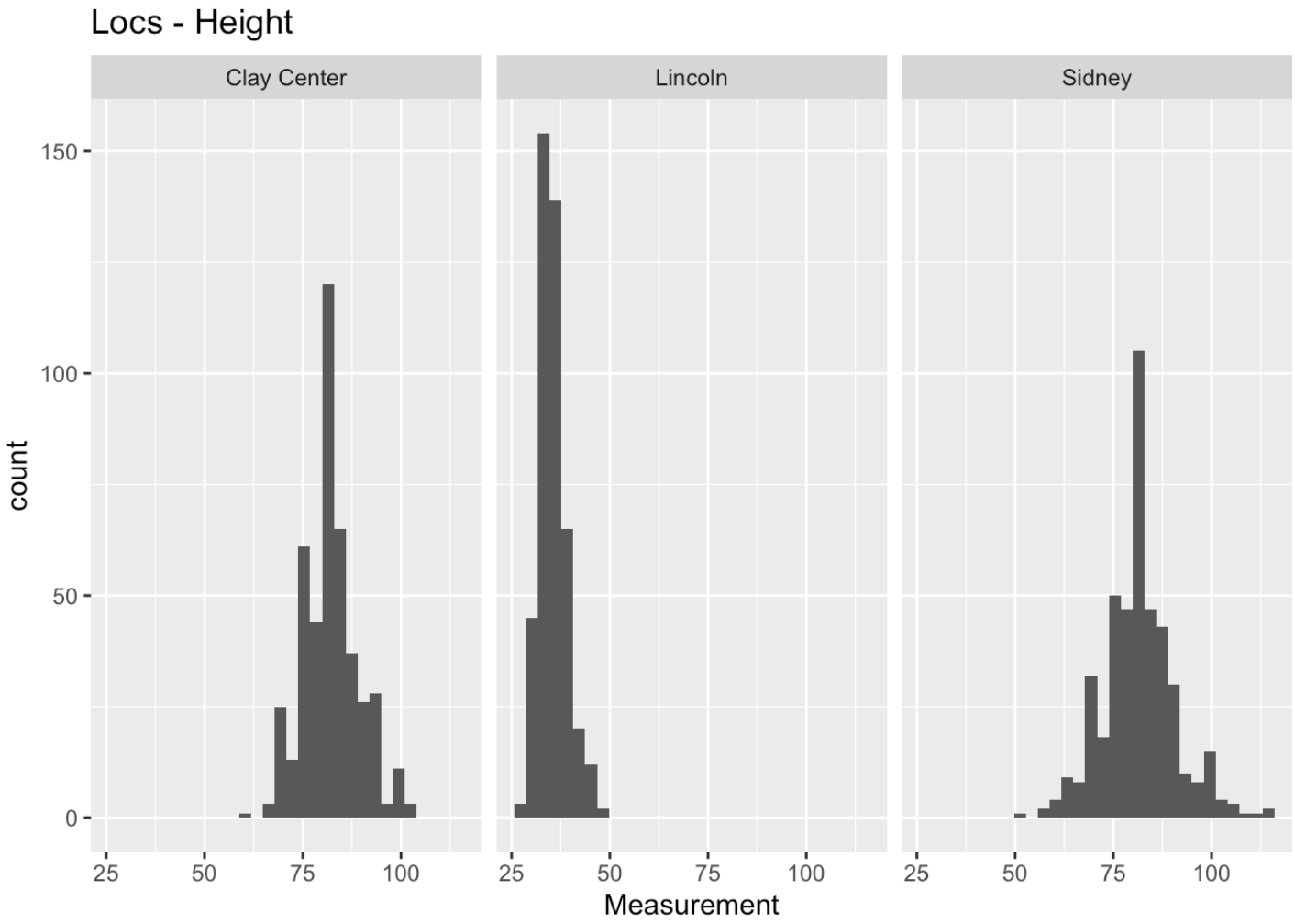
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
#something is up here

#facet over locations
ggplot(mydata %>% filter(Trait == "HT"), aes(x = Measurement)) +
  geom_histogram() +
  facet_wrap(vars(LOC)) +
  labs(title = "Locs - Height")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



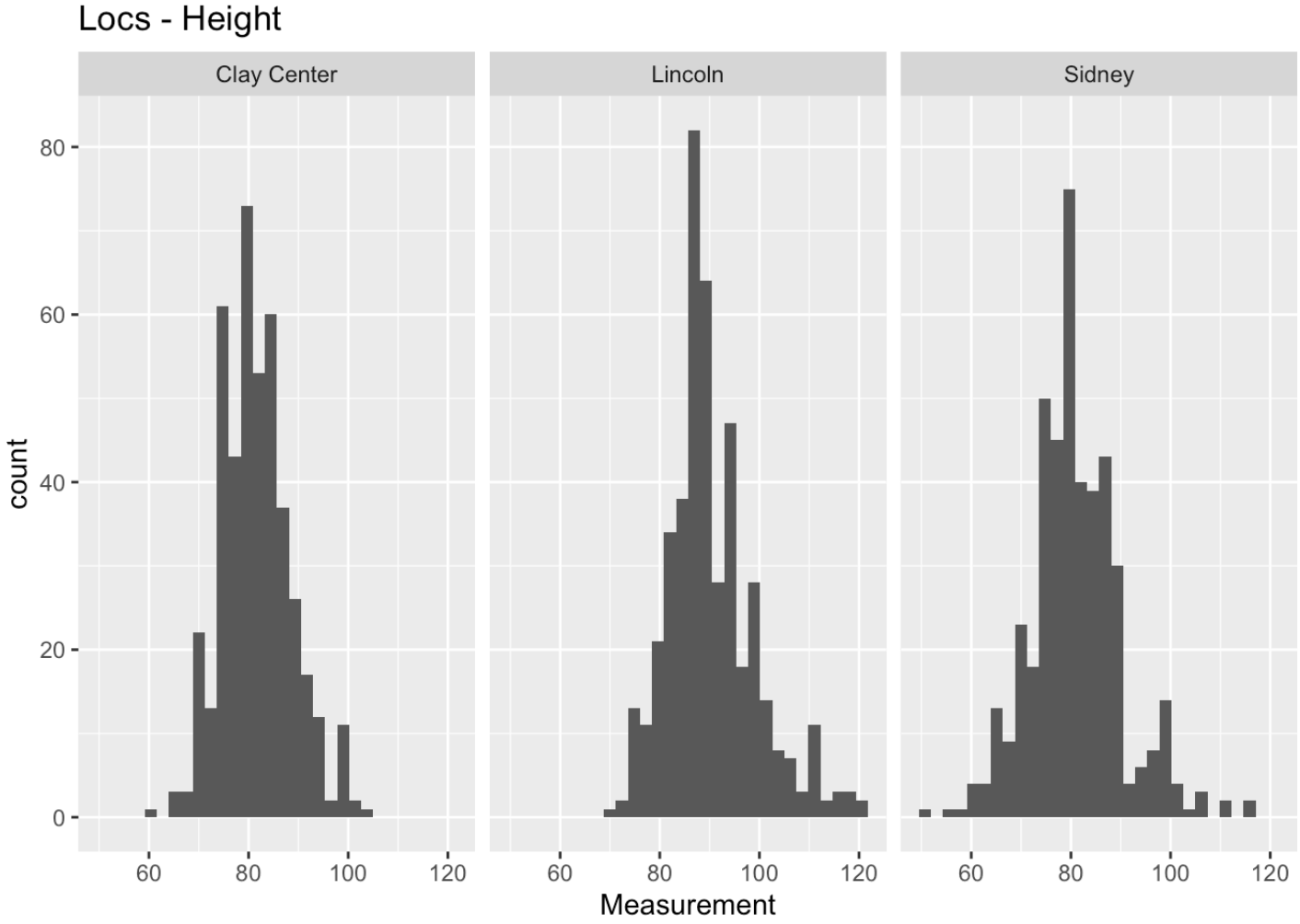
The distribution for height is odd. When we plot the data for each location we can see that one of the locations has height data reported on a different scale.

Clay Center and Sidney have height reported in centimeters, but Lincoln has height reported in inches. We will convert this data to be on the same scale as the other locations.

```
#convert the height data for Lincoln
mydata <- mydata %>%
  mutate(Measurement = ifelse(Trait %in% c("HT") & LOC %in% c("Lincoln"), Measurement
*2.54, Measurement))

#recheck
ggplot(mydata %>% filter(Trait == "HT"), aes(x = Measurement)) +
  geom_histogram() +
  facet_wrap(vars(LOC)) +
  labs(title = "Locs - Height")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



That looks correct.

Export data

At this point we have a long format data set where we've looked at the distributions for each of our phenotype traits, corrected for odd values, and adjusted values that were not on the same scale.

We can export this data in the long format for downstream processing.
We can also convert back to the wide format if that is needed for reporting.

```
#save the file
write.csv(mydata, file = here::here("data_clean", "YLDQTLVAL_clean.csv"), row.names =
F)

#select report columns, pivot to wide format, and arrange by rep and ID
mydata_report <- mydata %>%
  select(YR, PROGRAM, LOC, STATE, ID, REP, Trait, Measurement) %>%
  arrange(Trait, LOC) %>%
  pivot_wider(., names_from = c(Trait, LOC), values_from = Measurement) %>%
  arrange(REP, ID)
write.csv(mydata_report, file = here::here("data_clean", "YLDQTLVAL_report.csv"), row
.names = F)
```

```
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      janitor_2.1.0   forcats_0.5.1   stringr_1.5.0
##  [5] dplyr_1.1.2     purrr_1.0.1     readr_2.1.0     tidyr_1.3.0
##  [9] tibble_3.2.1    ggplot2_3.4.2   tidyverse_1.3.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.7       lubridate_1.8.0  assertthat_0.2.1 rprojroot_2.0.2
##  [5] digest_0.6.29    utf8_1.2.3       R6_2.5.1         cellranger_1.1.0
##  [9] backports_1.3.0  reprex_2.0.1     evaluate_0.14    highr_0.9
## [13] httr_1.4.2       pillar_1.9.0     rlang_1.1.1      readxl_1.3.1
## [17] rstudioapi_0.14  jquerylib_0.1.4  rmarkdown_2.11   labeling_0.4.2
## [21] bit_4.0.4        munsell_0.5.0    broom_1.0.4      compiler_4.1.2
## [25] modelr_0.1.8     xfun_0.28        pkgconfig_2.0.3  htmltools_0.5.3
## [29] tidyselect_1.2.0 fansi_1.0.4      crayon_1.4.2     tzdb_0.3.0
## [33] dbplyr_2.1.1     withr_2.5.0      grid_4.1.2       jsonlite_1.7.2
## [37] gtable_0.3.0     lifecycle_1.0.3  DBI_1.1.1        magrittr_2.0.3
## [41] scales_1.2.1     cli_3.6.1        stringi_1.7.5    vroom_1.5.6
## [45] farver_2.1.0     fs_1.6.1         snakecase_0.11.0 xml2_1.3.2
## [49] bslib_0.3.1      ellipsis_0.3.2   generics_0.1.3   vctrs_0.6.2
## [53] tools_4.1.2      bit64_4.0.5      glue_1.6.2       hms_1.1.1
## [57] parallel_4.1.2   fastmap_1.1.0    yaml_2.2.1       colorspace_2.0-2
## [61] rvest_1.0.2      knitr_1.36       haven_2.4.3      sass_0.4.1
```