Eelgrass Phenology Graphics 2023

Curtis C. Bohlen

2023-09-14

Contents

Introduction

In 2023, Casco Bay Estuary Partnership (CBEP) funded a seasonal survey of the timing of flowering and seed set of eelgrass (*Zostera marina*) in Casco Bay. Scientists call the seasonal timing of flowering and other life history events "phenology". So this study is a study of eelgrass phenology in Casco Bay, Maine.

The field project was conducted by "Team Zostera", a growing coalition of people interested in eelgrass in the region led by Glenn Page, of SustainaMetrix.

Field observations were conducted at two eelgrass beds near Portland, Maine, one near East End Beach, in Portland, and off the north end of Mackworth Island, in Falmouth. The two sites are about 1.75 miles apart as the crow flies.

Team Zostera provided CBEP with a copy of the field data in early September of 2023. In this workbook, I assemble a "tidy" data frame based on their data and explore options for graphic presentation of the study's results.

I anticipate little need for heavy-handed statistical methods, as a graphical summary will convey the essential seasonal information. The nature of the data also limits options for statistical modelling. Generalized linear models (on ordered data) could be useful for statistical comparison or to quantify seasonal variability, but that is not essential for a preliminary study.

Load Libraries

```
library(tidyverse)
library(readxl)
#library(emmeans)   # For extracting useful "marginal" model summaries

theme_set(theme_bw())
```

Explanation of Data

Data was collected following a protocol ("SOPs", or "Standard Operating Procedures") developed by Jillian Carr (of the Massachusetts Bays Program) and Phil Colarusso (Of EPA's Region 1 office, in Boston.) A copy of the protocol is available by request to the Casco Bay Estuary Partnership.

For each sampling event, a minimum of five eelgrass flowering shoots were collected. For each shoot, each side-branch (formally called a "rhipidium") was examined, in order from the bottom of the plant (oldest) to the top of the plant (youngest).

Each side branch is a compound inflorescence, composed of multiple smaller branchlets. On each side branch (formally, a "spathe"), the flowering stage of each flowering branchlet was recorded. Order of branchlets (spathes) on the inflorescence (rhipidium) was not recorded.

Flowering stage was recorded as an ordered integer, with values from zero (spathes developed, but styles not yet appeared) to six (seeds released, shoot beginning to wither.)

Data Properties

The sampling protocol generates data with complex structure, including both nesting and order information that may be important.

Nesting

The nesting structure of the data includes the following:

- Spathe, nested within
- Rhodium, ordered within
- Shoot, nested within

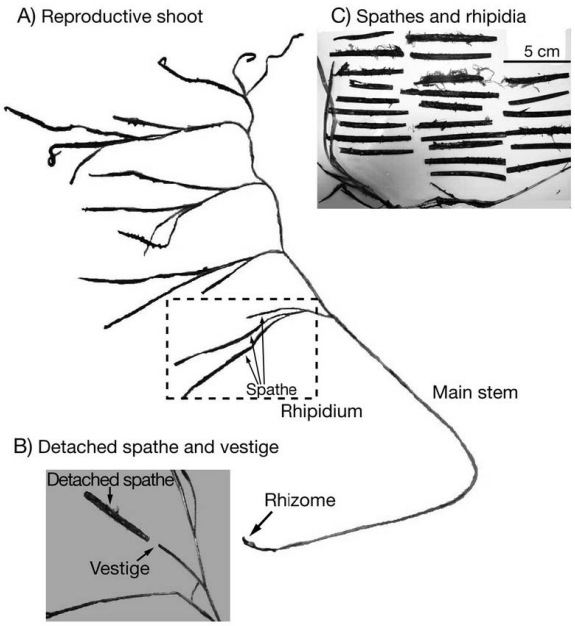


Figure 1: Eelgrass Flowering Shoot Morphology

- Date, nested within
- Site

Site and Date are not crossed, as only one site was sampled on most sampling days.

Note that Site and Date are naturally explanatory variables, best modeled here
as fixed effects. Shoots, on the other hand, are representative of each site, so Shoots are best modeled as random effects. Since the order of the rhipidia on each shoot is meaningful, we need to model them somehow as fixed effects, but it is not immediately clear how. Since shoots may vary in age or number of rhipidia, order is not likely comparable across shoots, so order should be modeled within shoots.

Wow. That data structure will generate complicated models for such a simple observational study!

Read Data

Data was provided in an Excel file ("Eelgrass Sampling Data.xlsx") with separate tabs for different sampling events. The date and sampling locations are coded in the names of the tabs.

Preparation of the Excel File

I hand edited the Excel file to address some data format inconsistencies, producing "Eelgrass Sampling Data Amended.xlsx".

My edits included the following:

*deleting two tabs containing no data

- replacing narrative data description ("two spathes at stage 0, one at one") with corresponding data in comma delimited lists ("0,0,1") and removing the word "and" from lists of values.

Data Quality Issues

Four data quality questions arose during the data review and editing:

1. A few rhipidia had no spathes. It is not clear how to code them, as the lack of spathes is useful information (suggesting either a non-flowering shoot or a flowering shoot too immature to show inflorescence development). Regardless, it is worth tracking this as distinct from a blank row in the data. Here I coded them as a single value, Stage -99, to be replaced with `na` when appropriate in data preparation

2. Sometimes a spathe was observed, but data was not recorded because the observers were uncertain how to score it. These are also treated as missing values, principally to retain correct spathe ordering.

3. In a few places, observers assigned spathes to intermediate stages, coding them with dashes to indicate a range (e.g., "stage 4-5"). That coding is liable to error, as it is very similar to separation of two numbers with a comma, so it should be avoided. For analysis purposes, we need to assign such intermediate stage observations to one of the six designated stages. Here, I arbitrarily assigned them to the higher stage, and added a Note in the Excel data. This mostly happened on one data sheet.

4. Excel appears to have a tendency to interpret comma delinted lists (or hyphen- delimited lists) of two or three values as dates. This was only evident after importing the data to R. Excel has many ways of representing dates, and some delimited lists are allowable formats. In Excel, cell contents appear indistinguishable from comma delimited lists, but internally and on import to R, they appear as large integers. I could search for these false dates in Excel by looking for cells formatted not as "General", but as "Custom". I corrected these cells by reformatting the cells and re-entering the data as a comma delimited list. Most occurred on the July 5 data sheet.

```
fn <- "Eelgrass Sampling Data Amended.xlsx"
sheets <- excel_sheets(fn)
sheets
#>  [1] "July 27 - Mackworth"   "July 19 - East End"    "July 15th - Mackworth"
#>  [4] "July 12th - East End"  "July 11th - Mackworth" "July 5th - East End"
#>  [7] "June 30 - Mackworth"   "June 22 - East End"    "June 20 - Mackworth"
#> [10] "June 18 - East End"    "June 13 - Mackworth"   "June 11 - Mackworth"
#> [13] "June 7 - Mackworth"
```

To create "tidy" data, we need to iterate over each sheet, split off the dates and locations, and convert the written dates into formal Date objects. Unfortunately, some dates include "th" suffixes, so we will have to handle this conversion directly.

The Logic of Extracting Data from Tab Names

(Note, similar data is recorded on each TAB, but not as part of the general data sheet format. I could also pull this data as part of a general function for accessing the data off each data sheet. For data quality reasons, I should probably do both and compare results.)

```
labs <- as_tibble(sheets) %>%
  filter(value != "Template")
labs
#> # A tibble: 13 x 1
#>    value
#>    <chr>
#>  1 July 27 - Mackworth
#>  2 July 19 - East End
```

```
#>  3 July 15th - Mackworth
#>  4 July 12th - East End
#>  5 July 11th - Mackworth
#>  6 July 5th - East End
#>  7 June 30 - Mackworth
#>  8 June 22 - East End
#>  9 June 20 - Mackworth
#> 10 June 18 - East End
#> 11 June 13 - Mackworth
#> 12 June 11 - Mackworth
#> 13 June 7 - Mackworth
```

```
(split_cols <- labs %>%
   separate_wider_delim(value, " - ",
                        names =c("date_str", "location")))
#> # A tibble: 13 x 2
#>    date_str  location
#>    <chr>     <chr>
#>  1 July 27   Mackworth
#>  2 July 19   East End
#>  3 July 15th Mackworth
#>  4 July 12th East End
#>  5 July 11th Mackworth
#>  6 July 5th  East End
#>  7 June 30   Mackworth
#>  8 June 22   East End
#>  9 June 20   Mackworth
#> 10 June 18   East End
#> 11 June 13   Mackworth
#> 12 June 11   Mackworth
#> 13 June 7    Mackworth
```

```
split_cols_2 <- split_cols %>%
  mutate(date_str = sub("th", "", date_str))
split_cols_2
#> # A tibble: 13 x 2
#>    date_str location
#>    <chr>    <chr>
#>  1 July 27  Mackworth
#>  2 July 19  East End
#>  3 July 15  Mackworth
#>  4 July 12  East End
#>  5 July 11  Mackworth
#>  6 July 5   East End
#>  7 June 30  Mackworth
#>  8 June 22  East End
#>  9 June 20  Mackworth
#> 10 June 18  East End
#> 11 June 13  Mackworth
#> 12 June 11  Mackworth
#> 13 June 7   Mackworth
```

Now, let's convert those dates. I use generalized code here instead of hard coding "June" and "July" in case in future we need to do this conversion for more than just two months. `month.name` is a built in R constant.

```
split_cols_3 <- split_cols_2 %>%
  separate_wider_delim(date_str, " ",
                       names =c("month_str", "day_str")) %>%
  mutate(month_num = as.character(match(month_str, month.name)),
         the_date = as.Date(paste0(month_num, "/", day_str, "/", "2023"),
                            format = "%m/%d/%Y")) %>%
  select(-month_str, -day_str, -month_num)
split_cols_3
#> # A tibble: 13 x 2
#>    location  the_date
#>    <chr>     <date>
#>  1 Mackworth 2023-07-27
#>  2 East End  2023-07-19
#>  3 Mackworth 2023-07-15
#>  4 East End  2023-07-12
#>  5 Mackworth 2023-07-11
#>  6 East End  2023-07-05
#>  7 Mackworth 2023-06-30
#>  8 East End  2023-06-22
#>  9 Mackworth 2023-06-20
#> 10 East End  2023-06-18
#> 11 Mackworth 2023-06-13
#> 12 Mackworth 2023-06-11
#> 13 Mackworth 2023-06-07
```

The challenge here, of course, is that we need to do this while we load the data, so it is worth constructing a function to handle the conversion.

A Function to Convert Dates and Times

Here is a small function to do the job, constructed for use in the Tidyverse. This handles the "Template" appropriately, by generating `NA`. It looks for the " - " and if it is missing, you end up with NA.

```
split_date_location = function(.data, tab_labels) {

  stopifnot(inherits(.data, 'data.frame'))

  tab_labels <- rlang::enquo(tab_labels)
  labs <- rlang::eval_tidy(tab_labels, .data)
  stopifnot(is.character(labs))


  df <- dplyr::tibble(labs = labs) %>%
    separate_wider_delim(labs, " - ",
                         names =c("date_str", "location"),
                         too_few = 'align_start') %>%
    mutate(date_str = sub("th", "", date_str)) %>%
    separate_wider_delim(date_str, " ",
                         names =c("month_str", "day_str"),
                         too_few = 'align_start') %>%
    mutate(month_num = as.character(match(month_str, month.name)),
           the_date = as.Date(paste0(month_num, "/", day_str, "/", "2023"),
```

```
                              format = "%m/%d/%Y")) %>%
    select(-month_str, -day_str, -month_num)

  return(df)
}
```

```
labs <- as_tibble(sheets)
labs
#> # A tibble: 13 x 1
#>    value
#>    <chr>
#>  1 July 27 - Mackworth
#>  2 July 19 - East End
#>  3 July 15th - Mackworth
#>  4 July 12th - East End
#>  5 July 11th - Mackworth
#>  6 July 5th - East End
#>  7 June 30 - Mackworth
#>  8 June 22 - East End
#>  9 June 20 - Mackworth
#> 10 June 18 - East End
#> 11 June 13 - Mackworth
#> 12 June 11 - Mackworth
#> 13 June 7 - Mackworth

split_date_location(labs, value)
#> # A tibble: 13 x 2
#>    location  the_date
#>    <chr>     <date>
#>  1 Mackworth 2023-07-27
#>  2 East End  2023-07-19
#>  3 Mackworth 2023-07-15
#>  4 East End  2023-07-12
#>  5 Mackworth 2023-07-11
#>  6 East End  2023-07-05
#>  7 Mackworth 2023-06-30
#>  8 East End  2023-06-22
#>  9 Mackworth 2023-06-20
#> 10 East End  2023-06-18
#> 11 Mackworth 2023-06-13
#> 12 Mackworth 2023-06-11
#> 13 Mackworth 2023-06-07
```

Iterating over each sheet

We can use `lapply()` to efficiently iterate over all the data sheets and assemble a list of data frames. We then convert that list into one large data frame using the Tidyverse's `bind_rows()` function.

The core coding idea is from Geeks for Geeks.

```
data_frame_list <- lapply(setNames(sheets, sheets),
                     function(x) read_excel(fn, sheet=x, range = "a9:c50",
                                            col_names = c("Shoot",
```

```
                                                    "Rhipidium",
                                                    "Spathe"),
                                        col_types = c("numeric",
                                                    "numeric",
                                                    "text")))
data_frame_list <- data_frame_list[-c(1, 9)]

data_frame <- bind_rows(data_frame_list, .id="Sample") %>%
  fill(Shoot, .direction = 'down') %>%
  filter(! is.na(Spathe)) %>%
  mutate(Spathe = na_if(Spathe, "-99"))
```

Separating Sampling Date and Sampling Locations

This shows why it was worth writing that short function. . . .

```
sample_info <- data_frame %>%
  split_date_location(Sample)

df <- sample_info %>%
  bind_cols(data_frame)
```

Split Spathe-level Data Into Separate Rows

```
the_data <- df %>%
  separate_longer_delim(Spathe, ",") %>%
  mutate(Spathe_num = as.numeric(Spathe))
#> Warning: There was 1 warning in `mutate()`.
#> i In argument: `Spathe_num = as.numeric(Spathe)`.
#> Caused by warning:
#> ! NAs introduced by coercion
```

Last (?) Data Correction

The warning arises because one entry, from July 19th, Shoot 1, Rhipidium 4, contained a comm-delimited list with two items that ends in a comma, implying there should be a third value. I have no way of determining whether the comma was an error or if a value is missing. Here, I delete that row, which is the same as assuming the comma should not have been there and here was not another Spathe to observe.

```
the_data <- the_data %>%
  filter(! (Sample == "July 19 - East End" &
              Shoot == 1 &
              Rhipidium == 4 &
              is.na(Spathe_num)))
```

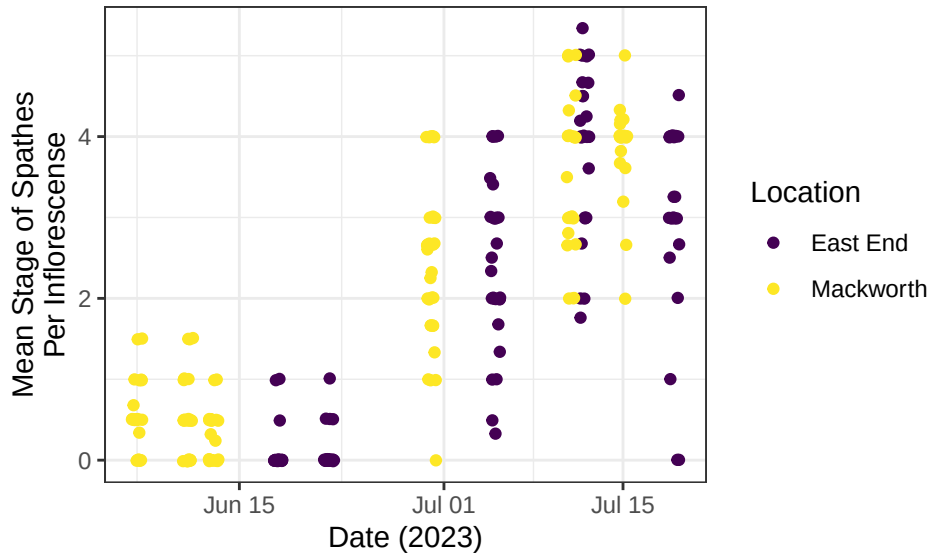
Prelimnary Graphics

Stage Graphics

Mean Stage of Inflorescences

```
the_data %>%
  group_by(location, the_date, Shoot, Rhipidium) %>%
  summarize(mean_stage = mean(Spathe_num, na.rm = TRUE)) %>%

  ggplot(aes(the_date, mean_stage, color = location)) +
  geom_jitter() +
  #geom_smooth(method = "gam", formula = y~s(x, k = 3)) +
  labs(y = "Mean Stage of Spathes\nPer Inflorescense") +
  xlab("Date (2023)") +
  labs(color = "Location") +
  scale_colour_viridis_d()
#> `summarise()` has grouped output by 'location', 'the_date', 'Shoot'. You can
#> override using the `.groups` argument.
#> Warning: Removed 2 rows containing missing values (`geom_point()`).
```

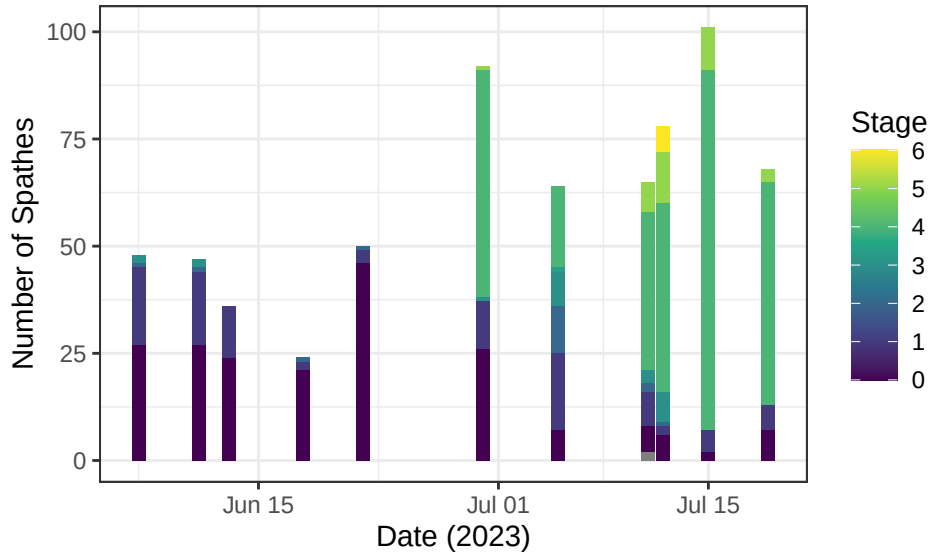



```
ggsave('mean_stage.png',
         width = 5, height = 3)
#> Warning: Removed 2 rows containing missing values (`geom_point()`).
```

It is not at all surprising to see the mean stage of the spathes observed climbing over the course of the season.

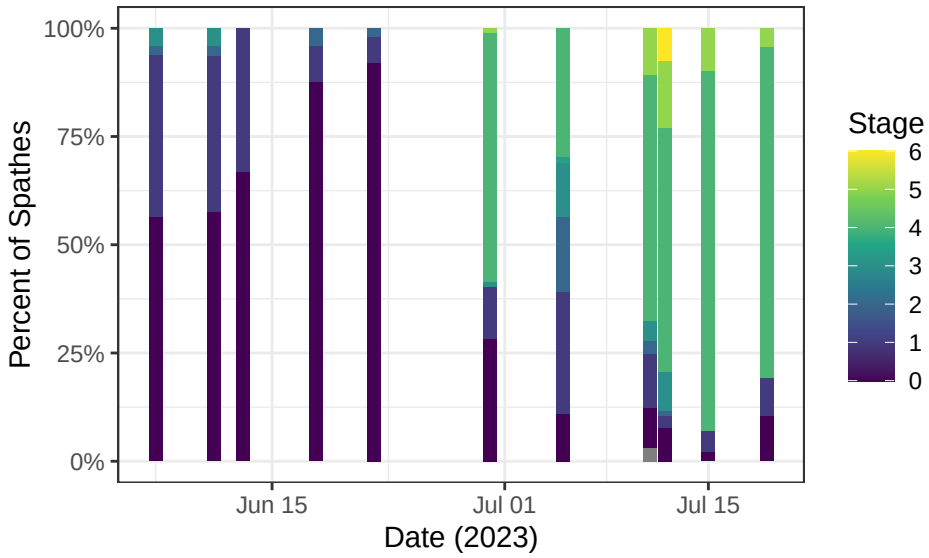
Bar Charts of Stage

```
the_data %>%
  ggplot(aes(x = the_date, fill = Spathe_num, group = -Spathe_num)) +
  geom_bar() +
  #facet_wrap(~location) +
  ylab("Number of Spathes") +
  xlab("Date (2023)") +
  labs(fill = "Stage") +
  scale_fill_viridis_c()
```




```
ggsave('stage_bar.png',
         width = 5, height = 3)
```

```
the_data %>%
  ggplot(aes(x = the_date, fill = Spathe_num, group = -Spathe_num)) +
  geom_bar(position = "fill") +
  #facet_wrap(~location) +
  ylab("Percent of Spathes") +
  xlab("Date (2023)") +
  labs(fill = "Stage") +
  scale_y_continuous(labels = scales::percent) +
  scale_fill_viridis_c()
```




```
ggsave('stage_bar_percent.png',
         width = 5, height = 3)
```

We start to see substantial numbers and a significant percentage of spathes in Stages four or five beginning in late June or July. Both percentage and numbers appear to increase as the season progresses. It would be interesting to extend the sampling season next year to gain understanding of the end of the season as well.

Frequency of Mature Spathes

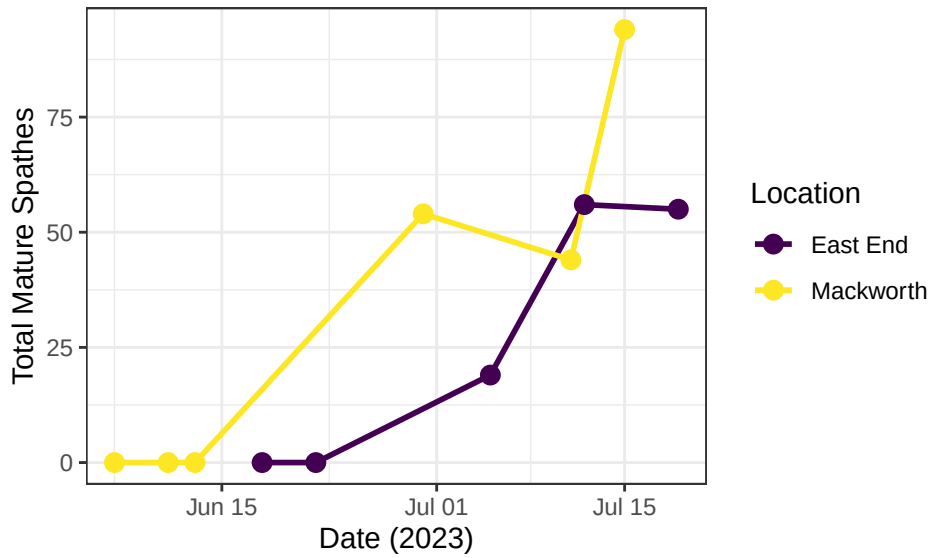
I defined spathes in stages four or five as "mature" spathes. The idea is those spathes are mature enough to potentially provide seeds when harvested.

Total Number of Mature Spathes

```
the_data %>%
  mutate(is_four_five = Spathe_num == 5 | Spathe_num == 4) %>%
  group_by(location, the_date) %>%
  summarize(num_four_five = sum(is_four_five, na.rm = TRUE)) %>%

  ggplot(aes(x = the_date, y = num_four_five, color = location)) +
  geom_line(size = 1) +
  geom_point(size = 3) +
  #facet_wrap(~location) +
  ylab("Total Mature Spathes") +
  xlab("Date (2023)") +
  labs(color = "Location") +
  scale_colour_viridis_d()
#> `summarise()` has grouped output by 'location'. You can override using the
#> `.groups` argument.
#> Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
#> i Please use `linewidth` instead.
```

```
#> This warning is displayed once every 8 hours.
#> Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
#> generated.
```




```
ggsave('spathes_in_45.png',
         width = 5, height = 3)
```

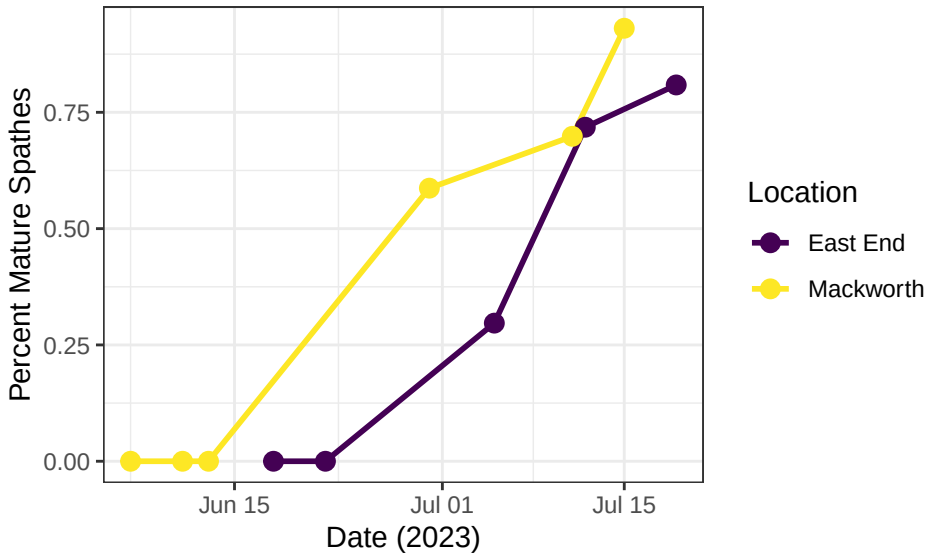
From a practical perspective, the total number of spathes that are at a stage where they might be harvested to collect seed is likely to be important for efficiency of harvest. It does not look like we sampled late enough in the season in 2023 to document a drop-off in availability of suitable spathes.

Overall Percent of Mature Spathes

This is an alternative graphic making more or less the same points.

```
the_data %>%
  filter( ! is.na(Spathe_num)) %>%
  mutate(is_four_five = Spathe_num == 5 | Spathe_num == 4) %>%
  group_by(location, the_date) %>%
  summarize(pct_four_five = sum(is_four_five)/length(is_four_five)) %>%

  ggplot(aes(x = the_date, y = pct_four_five, color = location)) +
  geom_line(size = 1) +
  geom_point(size = 3) +
  #facet_wrap(~location) +
  ylab("Percent Mature Spathes") +
  xlab("Date (2023)") +
  labs(color = "Location") +
  scale_colour_viridis_d()
#> `summarise()` has grouped output by 'location'. You can override using the
#> `.groups` argument.
```




```
ggsave('overall_percent_stage_45.png',
         width = 5, height = 3)
```

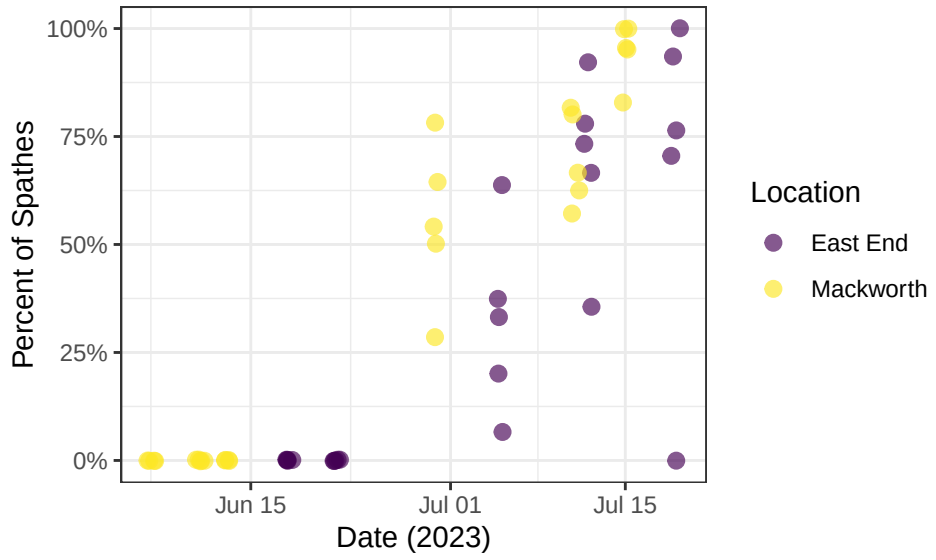
Shoot by Shoot Percent of Mature Spathes

Here the points have been moved around slightly and made partially transparent to make it clear there's a lot of shoots in June with no mature spathes.

```
  the_data %>%
  mutate(is_four_five = Spathe_num == 5 | Spathe_num == 4) %>%
  group_by(location, the_date, Shoot) %>%
  summarize(pct_45 = sum(is_four_five, na.rm = TRUE)/
                     (length(is_four_five) - sum(is.na(is_four_five)))) %>%

  ggplot(aes(the_date, pct_45, color = location)) +
  #geom_point(size = 2.5, alpha = 0.5) +
  geom_jitter(size = 2.5, alpha = 0.65) +

  #geom_smooth(method = 'gam', formula = y~s(x, k = 3)) +
  ylab("Percent of Spathes") +
  xlab("Date (2023)") +
  labs(color = "Location") +
  scale_y_continuous(labels = scales::percent) +
  scale_color_viridis_d()
#> `summarise()` has grouped output by 'location', 'the_date'. You can override
#> using the `.groups` argument.
```




```
ggsave('Shoot_percent_stage_45.png',
         width = 5, height = 3)
```

This offers a bit more understanding of variability among shoots. By mid-July, most shoots have at least 50% mature spathes, suggesting that's likely to be a good time to harvest. But the proportion of spathes that are mature varies quite a bit.

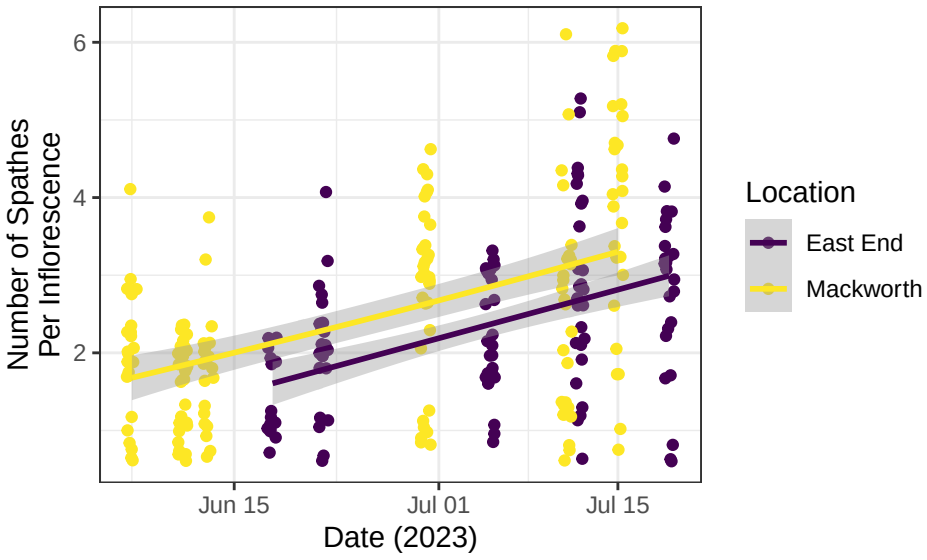
As before, the data does not extend late enough in the season to show a drop off in the number of mature spathes.

Number of Inflorescenece and Spathes Per Shoot

Number of Spathes per Inflorescence

If there are strong seasonal patterns in the number of spathes per inflorescence, or the number of inflorescence per shoot, that might also influence the ideal time for harvesting seed. It is going to be best to sample when we can get the most mature spathes, not the highest percentage of mature spathes.

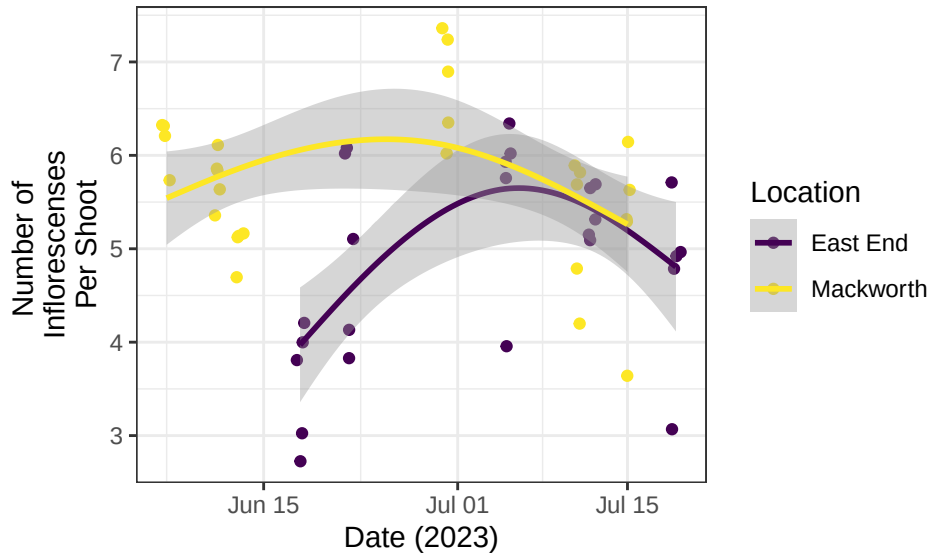
```
the_data %>%
  group_by(location, the_date, Shoot, Rhipidium) %>%
  summarize(num_spathes = n()) %>%
  ggplot(aes(the_date, num_spathes, color = location)) +
  geom_jitter() +
  geom_smooth(method = "gam", formula = y~s(x, k = 3)) +
  ylab("Number of Spathes\nPer Inflorescence") +
  xlab("Date (2023)") +
  labs(color = "Location") +
  scale_colour_viridis_d()
#> `summarise()` has grouped output by 'location', 'the_date', 'Shoot'. You can
#> override using the `.groups` argument.
```




```
ggsave('spathes_per_inf.png',
         width = 5, height = 3)
```

Number of Inflorescences Per Shoot

```
the_data %>%
  group_by(location, the_date, Shoot, Rhipidium) %>%
  summarize() %>%
  ungroup(Rhipidium) %>%
  summarize(num_Rhipidia = n()) %>%
  ggplot(aes(the_date, num_Rhipidia, color = location)) +
  geom_jitter() +
  geom_smooth(method = "gam", formula = y~s(x, k = 3)) +
  ylab("Number of \nInflorescenses\nPer Shoot") +
  xlab("Date (2023)") +
  labs(color = "Location") +
  scale_colour_viridis_d()
#> `summarise()` has grouped output by 'location', 'the_date', 'Shoot'. You can
#> override using the `.groups` argument.
#> `summarise()` has grouped output by 'location', 'the_date'. You can override
#> using the `.groups` argument.
```




```
ggsave('infl_per_shoot.png',
         width = 5, height = 3)
```

The number of spathes per inflorescence continued to increase through the summer, while the number of inflorescences per shoot appears to peak sometime in late June or early July. Given the limited data, those apparent patterns may reflect sampling variability (note the wide error bars). This deserves additional attention in future years.

Future Questions / Directions to Consider

1. Differences in Stage by order of Inflorescence on the Shoot
2. Hierarchical Generalized Linear Models of Stage, based on ordered data.