Chapter3 - Evaluation of variability

Erika Nishida

2023-1-14

Install packages

```
library(tidyverse)
library(agricolae) #for LSD.test()
library(gt) #for 表の出力
library(car) #for TypeII ANOVA
library(lme4) #for 混合効果モデル
library(lmerTest) #for 混合効果モデル
library(emmeans) #for 多重比較
library(ineq) # for ジニ係数
library(boot) #for 95%信頼区間の算出
library(Hmisc) # for ジニ平均差
```

Data importing & preprocessing

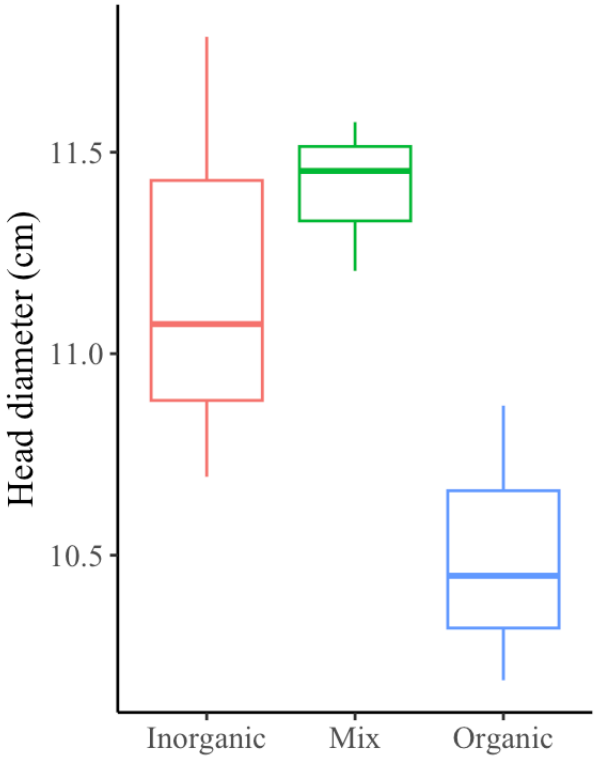
2022年3月19日の花蕾直径(2700個)のデータがある。3肥料処理 x 3反復 = 9個の大プロットが存在する。大プロットを4 x 4 = 16区画に分けたもの = 小プロットと定義した。

```
#データをインポート
#dt：2022年3月19日の花蕾直径(2700個)データ
dt <- read.csv("headdiameter_20220319.csv") %>%
  select(-`X`)

#処理区（無機肥料、混合肥料、有機肥料の）を示すベクトルを追加する
#各処理3反復
#3処理 x 3反復 = 9個の大プロットがある
dt2 <- dt %>%
  mutate("tre" = case_when(
    large_plot == 1 ~ "Inorganic fertilizer",
    large_plot == 2 ~ "Mix fertilizer",
    large_plot == 3 ~ "Organic fertilizer",
    large_plot == 4 ~ "Organic fertilizer",
    large_plot == 5 ~ "Inorganic fertilizer",
    large_plot == 6 ~ "Mix fertilizer",
    large_plot == 7 ~ "Mix fertilizer",
    large_plot == 8 ~ "Organic fertilizer",
    large_plot == 9 ~ "Inorganic fertilizer"
  ))

#乱塊法として東西方向を3ブロックに分ける
dt3 <- dt2 %>%
  mutate("Block" = case_when(
    large_plot == 1|large_plot == 4|large_plot == 7 ~ "1",
    large_plot == 2|large_plot == 5|large_plot == 8 ~ "2",
    large_plot == 3|large_plot == 6|large_plot == 9 ~ "3"
  ))
```

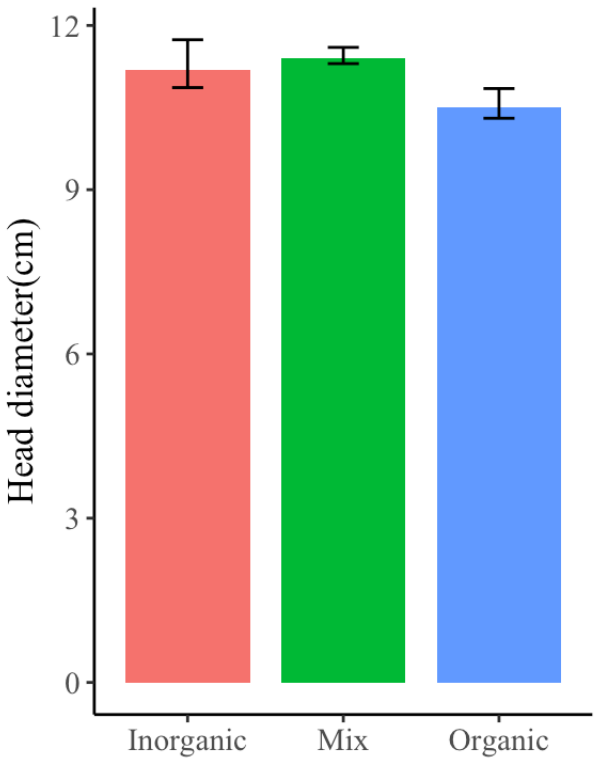
[3.3.1. 平均花蕾直径] - Fig 3-3のレシピ

1. 可視化

```
#データフレーム作り
hd_largeplot <- dt3 %>%
  group_by(`large_plot`, `tre`, `Block`) %>%
  dplyr::summarize("Ave_HD" = mean(`head.diameter`))

#箱ひげ図
p <- ggplot(data = hd_largeplot,
            mapping = aes(x = tre, y = Ave_HD)) +
  geom_boxplot(mapping = aes(color = `tre`)) +
  labs(y = "Head diameter (cm)",
       x = "") +
  guides(color = "none") +
  scale_x_discrete(labels= c("Inorganic", "Mix", "Organic")) +
  theme_classic() +
  theme(text = element_text(family = "Times New Roman"),
        axis.text = element_text(size = 11),
        axis.title = element_text(size = 13))

print(p)
```



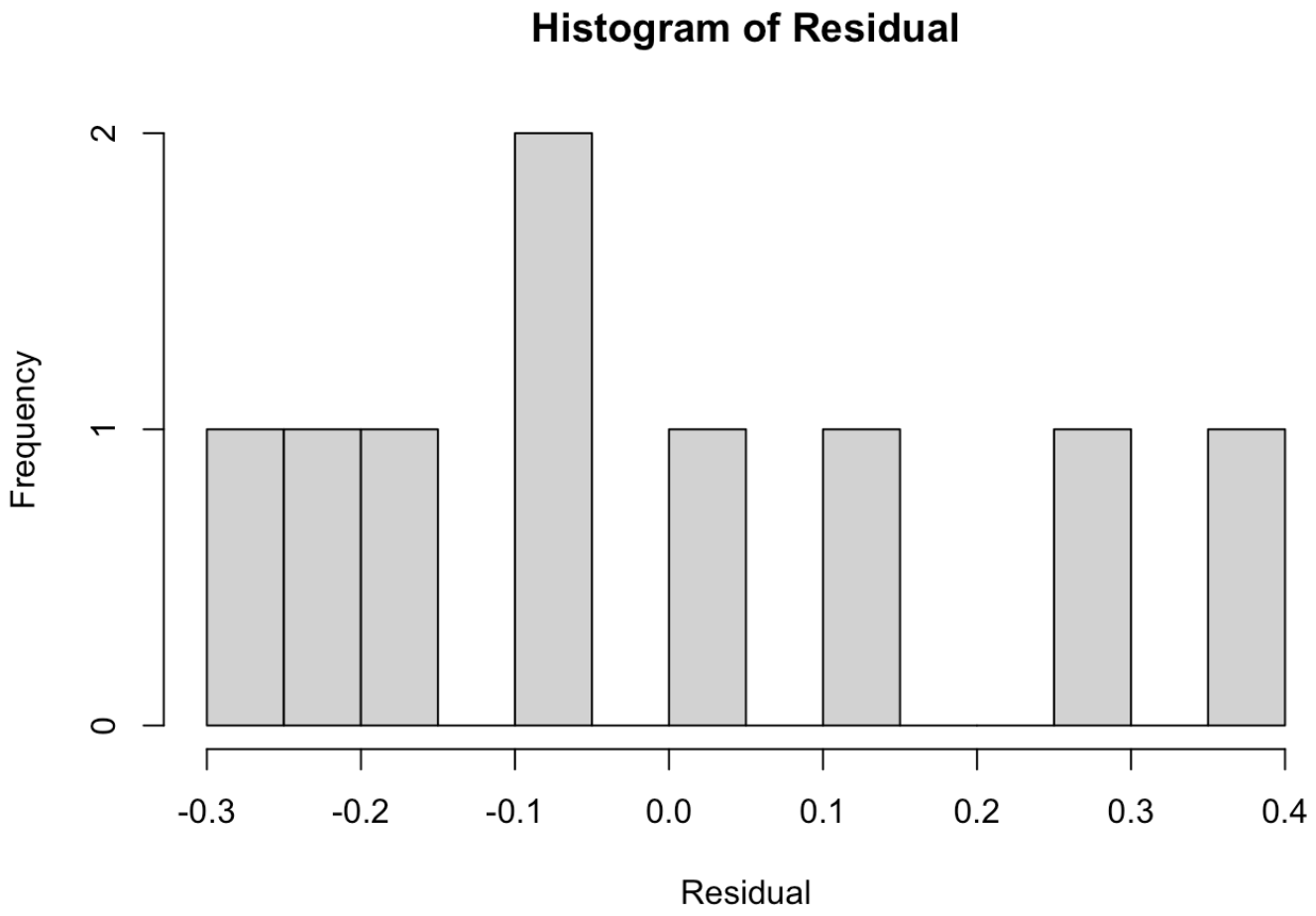
```
#棒グラフ
#→生データを入力しても、自動で棒グラフにしてはくれない。棒グラフが示す平均値や標準誤差は、事前に計算してデータフレーム化しておく必要がある。

#1. 標準誤差を示すエラーバーを追加するために、事前に標準誤差を計算する
se <- hd_largeplot %>%
  group_by(tre) %>%
  summarise(mean = mean(Ave_HD),
                sd = sd(Ave_HD),
                se = sd(Ave_HD)/sqrt(length(Ave_HD)))

#2.グラフ用にデータフレームを作る
boxplot <- hd_largeplot %>%
  group_by(tre) %>%
  dplyr::summarise("y" = mean(Ave_HD))

p2 <- ggplot(se, mapping = aes(x = tre, y = mean)) +
  geom_col(mapping = aes(fill = tre),
           width = 0.8) +
  geom_errorbar(aes(ymin = mean - se, ymax = mean + sd), width = .2) +
  labs(x = "",
       y = "Head diameter(cm)") +
  guides(fill = "none") +
  scale_x_discrete(labels= c("Inorganic", "Mix", "Organic")) +
  theme_classic() +
  theme(text = element_text(family = "Times New Roman"),
        axis.text = element_text(size = 11),
        axis.title = element_text(size = 13))

print(p2)
```



2. 統計解析

想定するモデル式：y = μ(総平均) + Block + 肥料処理 + ε(誤差項)

```
#解析用にデータを成形する
hd_largeplot <- dt3 %>%
  group_by(`large_plot`, `tre`, `Block`) %>%
  dplyr::summarize("Ave_HD" = mean(`head.diameter`))

#誤差項の正規性の確認
glm_avehd <- glm(data = hd_largeplot,
           Ave_HD ~ Block + tre)
predicted_value <- predict(glm_avehd)
Residual <- hd_largeplot$Ave_HD - predicted_value
hist(Residual, nclass = 10)
```




```
shapiro <- shapiro.test(Residual)
shapiro #W = 0.95917, p-value = 0.7896
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Residual
## W = 0.95917, p-value = 0.7896
```

```
#等分散性の確認
bartlett.test(Residual~hd_largeplot$tre)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Residual by hd_largeplot$tre
## Bartlett's K-squared = 1.5489, df = 2, p-value = 0.4609
```

```
#Bartlett's K-squared = 1.5489, df = 2, p-value = 0.4609

#分散分析
car::Anova(glm_avehd) #tre：0.000932, Block：0.060784
```

```
## Analysis of Deviance Table (Type II tests)
##
## Response: Ave_HD
##       LR Chisq Df Pr(>Chisq)
## Block   5.6009  2   0.060784 .
## tre    13.9563  2   0.000932 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

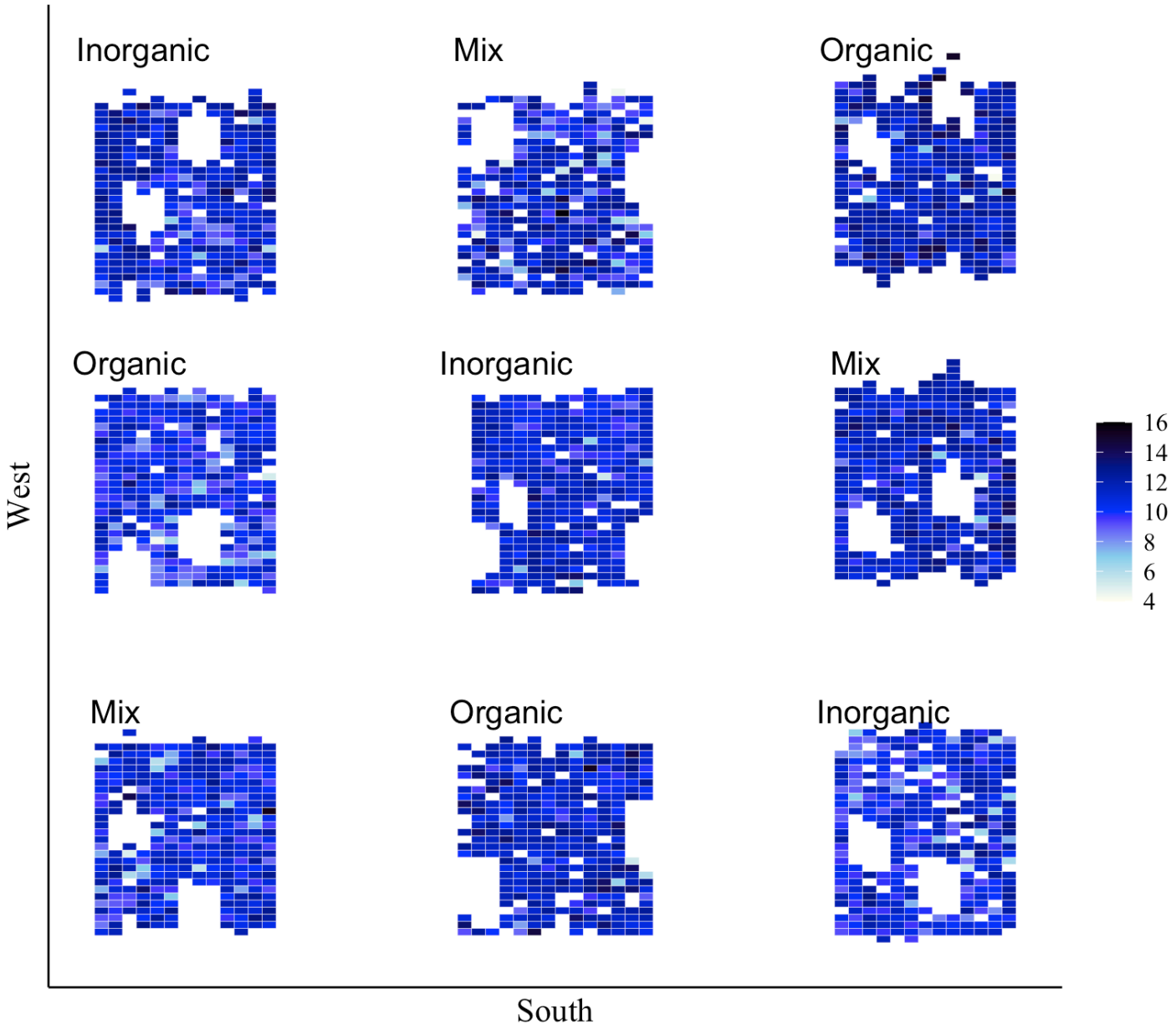
```
#LSD test
lsd_glm_avehd <- LSD.test(glm_avehd, "tre")
lsd_glm_avehd
```

```
## $statistics
##      MSerror Df    Mean       CV  t.value       LSD
##   0.09604569  4 11.0331 2.808933 2.776445 0.7025583
##
## $parameters
##         test p.ajusted name.t ntr alpha
##   Fisher-LSD      none    tre   3  0.05
##
## $means
##                          Ave_HD       std r      LCL      UCL      Min      Max
## Inorganic fertilizer 11.18484 0.5543505 3 10.68806 11.68162 10.69462 11.78641
## Mix fertilizer       11.41126 0.1882620 3 10.91447 11.90804 11.20544 11.57476
## Organic fertilizer   10.50321 0.3439674 3 10.00642 10.99999 10.18963 10.87110
##                             Q25      Q50      Q75
## Inorganic fertilizer 10.88405 11.07349 11.42995
## Mix fertilizer       11.32951 11.45358 11.51417
## Organic fertilizer   10.31926 10.44888 10.65999
##
## $comparison
## NULL
##
## $groups
##                          Ave_HD groups
## Mix fertilizer       11.41126      a
## Inorganic fertilizer 11.18484     ab
## Organic fertilizer   10.50321      b
##
## attr(,"class")
## [1] "group"
```

[3.3.3.1. 個体間差異]

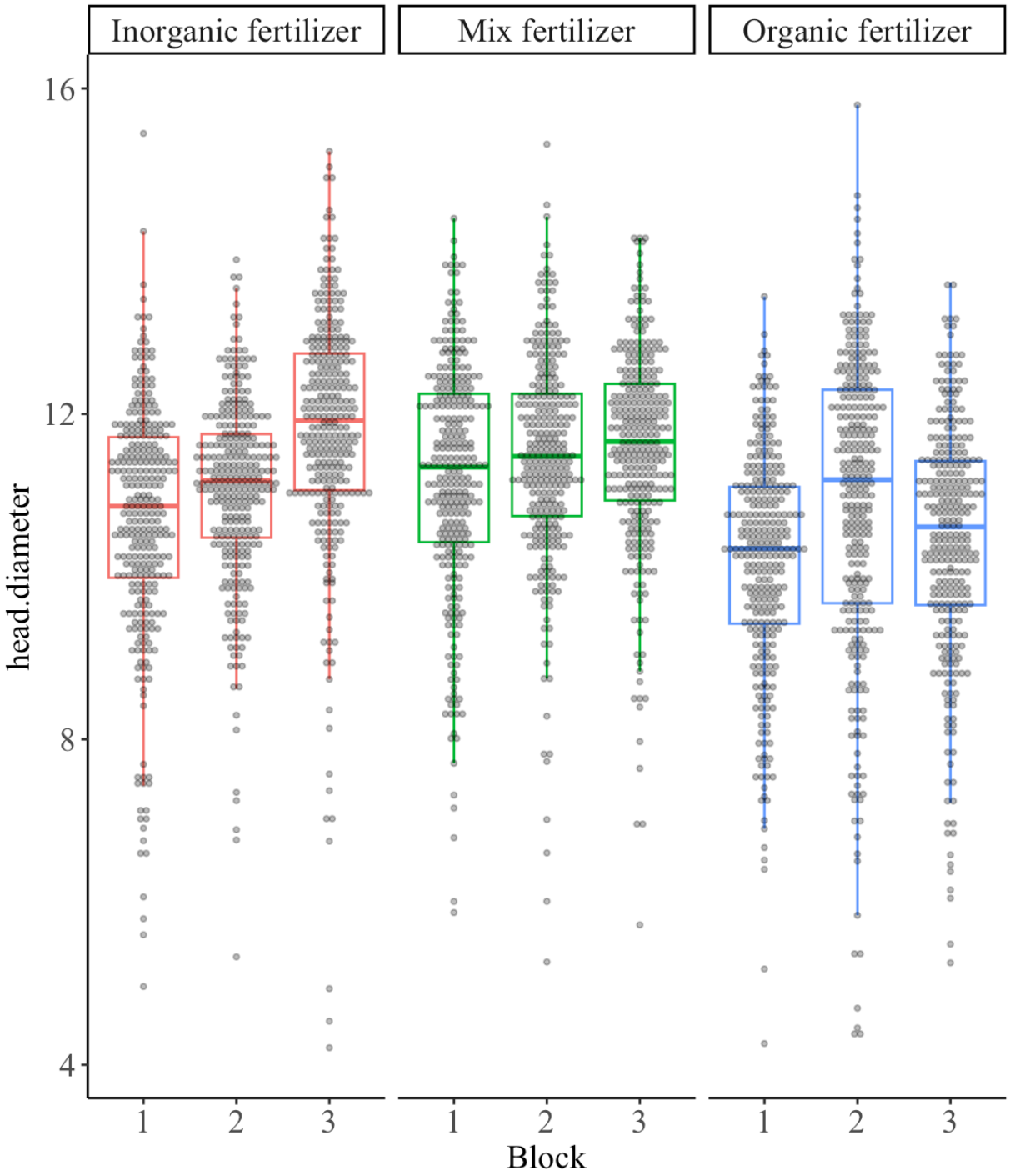
1. 可視化(ヒートマップ) **- Fig 3-5**のレシピ

```
#ヒートマップ
p3 <- ggplot(data = dt3, mapping = aes(x = EW_info, y = NS_info, fill = `head.diame
ter`)) +
  geom_tile(colour = "white") +
  scale_fill_gradientn(colours=c("ivory", "skyblue", "blue", "darkblue", "black"),
                       limits=c(4,16),
                       name ="") +
  labs(x = "South", y = "West") +
  annotate("text", x = 11, y = 135, label = "Inorganic", size = 5.5) +
  annotate("text", x = 35, y = 135, label = "Mix", size = 5.5) +
  annotate("text", x = 63.5, y = 135, label = "Organic", size = 5.5) +
  annotate("text", x = 10, y = 91, label = "Organic", size = 5.5) +
  annotate("text", x = 37, y = 91, label = "Inorganic", size = 5.5) +
  annotate("text", x = 62, y = 91, label = "Mix", size = 5.5) +
  annotate("text", x = 9, y = 42, label = "Mix", size = 5.5) +
  annotate("text", x = 37, y = 42, label = "Organic", size = 5.5) +
  annotate("text", x = 64, y = 42, label = "Inorganic", size = 5.5) +
  theme_classic() +
  theme(text = element_text("Times New Roman"),
        axis.ticks = element_blank(),
        axis.text.y = element_blank(),
        axis.text.x = element_blank(),
        axis.title = element_text(size = 16),
        legend.text = element_text(size = 12))

print(p3)
```



2. 可視化(箱ひげ図) - Fig 3-5のレシピ

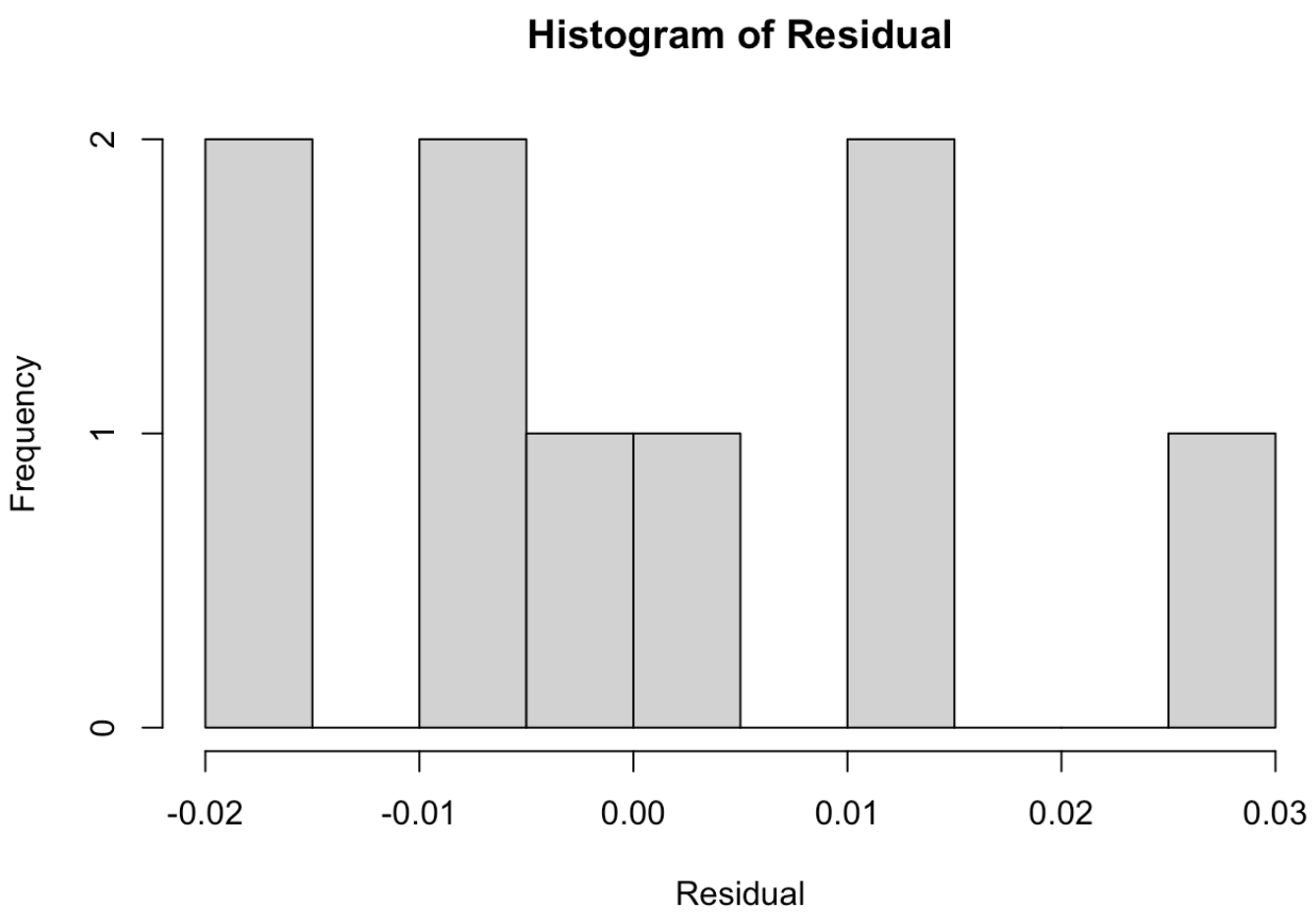
```
#箱ひげ図
p4 <- ggplot(dt3, aes(x = Block, y = `head.diameter`)) +
  geom_boxplot(aes(color = tre), outlier.color = NA) +
  geom_dotplot(binaxis = "y", binwidth = 0.07, stackdir = "center", alpha = 0.3) +
  facet_grid( .~tre) +
  guides(color = "none") +
  theme_classic() +
  theme(text = element_text(family = "Times New Roman"),
        axis.text = element_text(size = 14),
        axis.title = element_text(size = 14),
        strip.text = element_text(size = 14))
print(p4)
```



3. 統計解析(CV)

大プロット内のばらつきを肥料処理間で比較する。大プロットごとの変動係数(以下、CV)を算出する
想定するモデル式：CV = 総CV平均μ + 肥料処理 + (Block)*ランダム効果として

```
#各大プロットのCV(300個のばらつきを評価)を算出する
cv_in_largeplot <- dt3 %>%
  group_by(large_plot, tre, Block) %>%
  dplyr::summarize(CV = sd(`head.diameter`)/mean(`head.diameter`))

#誤差項の正規性の確認
lmer_cv_in_largeplot <- lme4::lmer(data = cv_in_largeplot,
           CV ~ tre + (1|Block))
predicted_value <- predict(lmer_cv_in_largeplot)
Residual <- cv_in_largeplot$CV - predicted_value
hist(Residual, nclass = 10)
```




```
shapiro <- shapiro.test(Residual)
shapiro #W = 0.95433, p-value = 0.7376
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Residual
## W = 0.95433, p-value = 0.7376
```

```
#等分散性の確認
bartlett.test(Residual~cv_in_largeplot$tre)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Residual by cv_in_largeplot$tre
## Bartlett's K-squared = 0.73673, df = 2, p-value = 0.6919
```

```
#Bartlett's K-squared = 0.73673, df = 2, p-value = 0.6919

#分散分析
car::Anova(lmer_cv_in_largeplot) #0.04392
```

```
## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: CV
##      Chisq Df Pr(>Chisq)
## tre 6.2506  2    0.04392 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#多重比較
emmeans(lmer_cv_in_largeplot,list(pairwise ~ tre), adjust = "tukey")
```

```
## $`emmeans of tre`
##  tre                   emmean     SE df lower.CL upper.CL
##  Inorganic fertilizer  0.125 0.0101  6   0.0999    0.150
##  Mix fertilizer        0.118 0.0101  6   0.0933    0.143
##  Organic fertilizer    0.152 0.0101  6   0.1271    0.177
##
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $`pairwise differences of tre`
##  1                                          estimate     SE df t.ratio p.value
##  Inorganic fertilizer - Mix fertilizer       0.00668 0.0143  4   0.466  0.8902
##  Inorganic fertilizer - Organic fertilizer -0.02714 0.0143  4  -1.894  0.2542
##  Mix fertilizer - Organic fertilizer       -0.03382 0.0143  4  -2.360  0.1568
##
## Degrees-of-freedom method: kenward-roger
## P value adjustment: tukey method for comparing a family of 3 estimates
```

4. ジニ係数

```
#データセット作り
dt3_I <- dt3 %>%
  filter(tre == "Inorganic fertilizer") %>%
  select(`head.diameter`)

dt3_IO <- dt3 %>%
  filter(tre == "Mix fertilizer") %>%
  select(`head.diameter`)

dt3_O <- dt3 %>%
  filter(tre == "Organic fertilizer") %>%
  select(`head.diameter`)

#ジニ係数を算出する
Gini_I <- ineq(dt3_I$`head.diameter`, type = "Gini")
Gini_I
```

```
## [1] 0.07037303
```

```
Gini_IO <- ineq(dt3_IO$`head.diameter`, type = "Gini")
Gini_IO
```

```
## [1] 0.06449897
```

```
Gini_O <- ineq(dt3_O$`head.diameter`, type = "Gini")
Gini_O
```

```
## [1] 0.086118
```

```
#95%信頼区間を算出する
#関数を定義
g.boot.fun <- function(data, i){
  data <- data[i]
  Gini(data)
}

g.boot_I <- boot(data = dt3_I$`head.diameter`,
                 statistic = g.boot.fun,
                 R = 5000,
                 sim  = "ordinary")
boot.ci(g.boot_I, type = "bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = g.boot_I, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 0.0661,  0.0756 )
## Calculations and Intervals on Original Scale
```

```
g.boot_IO <- boot(data = dt3_IO$`head.diameter`,
                 statistic = g.boot.fun,
                 R = 5000,
                 sim  = "ordinary")
boot.ci(g.boot_IO, type = "bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = g.boot_IO, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 0.0609,  0.0691 )
## Calculations and Intervals on Original Scale
```

```
g.boot_O <- boot(data = dt3_O$`head.diameter`,
                 statistic = g.boot.fun,
                 R = 5000,
                 sim  = "ordinary")
boot.ci(g.boot_O, type = "bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = g.boot_O, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 0.0814,  0.0916 )
## Calculations and Intervals on Original Scale
```

5. ジニ平均差

```
GiniMd_I <- GiniMd(dt3_I$`head.diameter`, na.rm = TRUE)
GiniMd_I
```

```
## [1] 1.575973
```

```
GiniMd_IO <- GiniMd(dt3_IO$`head.diameter`, na.rm = TRUE)
GiniMd_IO
```

```
## [1] 1.473666
```

```
GiniMd_O <- GiniMd(dt3_O$`head.diameter`, na.rm = TRUE)
GiniMd_O
```

```
## [1] 1.811043
```

```
#95%信頼区間を算出する
#関数を定義する
gm.boot.fun <- function(data, i){
  data <- data[i]
  GiniMd(data)
}

gm.boot_I <- boot(data = dt3_I$`head.diameter`,
                  statistic = gm.boot.fun,
                  R = 5000,
                  sim  = "ordinary")
boot.ci(gm.boot_I, type = "bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = gm.boot_I, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 1.481,  1.689 )
## Calculations and Intervals on Original Scale
```

```
gm.boot_IO <- boot(data = dt3_IO$`head.diameter`,
                  statistic = gm.boot.fun,
                  R = 5000,
                  sim  = "ordinary")
boot.ci(gm.boot_IO, type = "bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = gm.boot_IO, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 1.394,  1.567 )
## Calculations and Intervals on Original Scale
```

```
gm.boot_O <- boot(data = dt3_O$`head.diameter`,
                  statistic = gm.boot.fun,
                  R = 5000,
                  sim  = "ordinary")
boot.ci(gm.boot_O, type = "bca")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 5000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = gm.boot_O, type = "bca")
##
## Intervals :
## Level       BCa
## 95%   ( 1.724,  1.917 )
## Calculations and Intervals on Original Scale
```

6. ローレンツ非対称係数 - Fig 3-6のレシピ

```
#ローレンツ非対称係数を算出する
Lasym(dt3_I$`head.diameter`)
```

```
## [1] 0.8553155
```

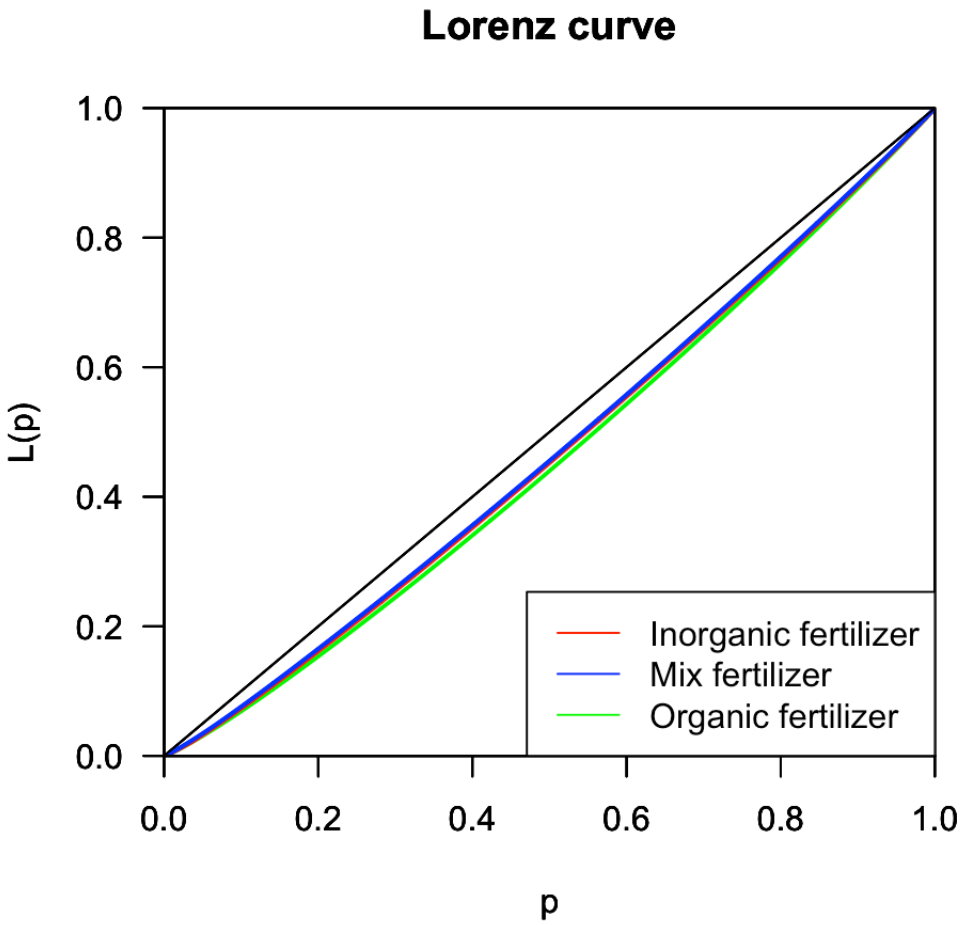
```
Lasym(dt3_IO$`head.diameter`)
```

```
## [1] 0.895952
```

```
Lasym(dt3_O$`head.diameter`)
```

```
## [1] 0.8659711
```

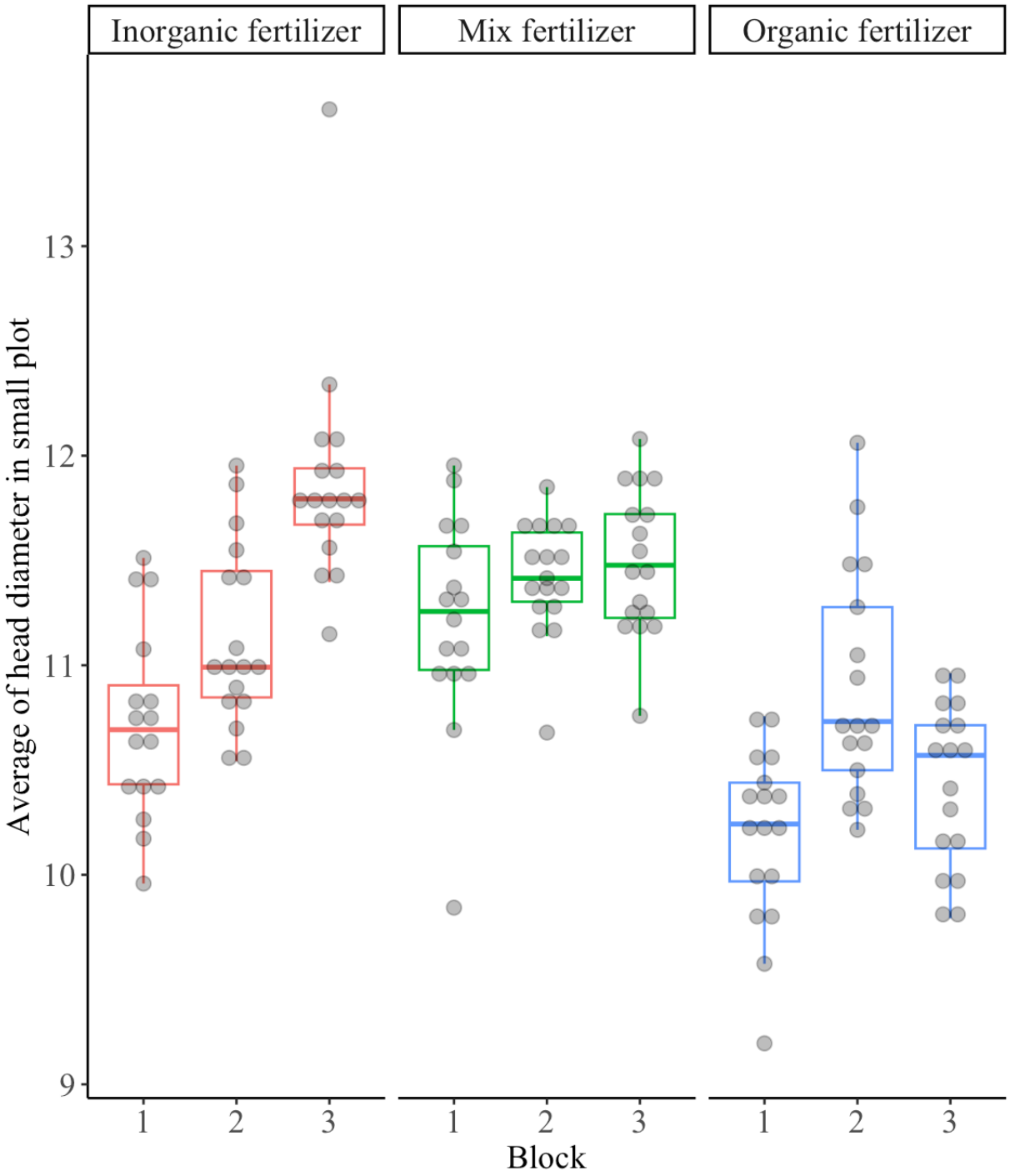
```
#ローレンツ曲線を描く
plot(Lc(dt3_IO$`head.diameter`), col = "blue")
par(new = T)
plot(Lc(dt3_O$`head.diameter`), col = "green")
par(new = T)
plot(Lc(dt3_I$`head.diameter`), col = "red")
par(new = T)
plot(Lc(dt3_IO$`head.diameter`), col = "blue")
legend(
  "bottomright",
  legend = c("Inorganic fertilizer", "Mix fertilizer", "Organic fertilizer"),
  lty = c("solid", "solid", "solid"),
  col = c("red", "blue", "green")
)
```



[3.3.3.2. 圃場内の空間変異]

1. 可視化(箱ひげ図) - Fig 3-7のレシピ

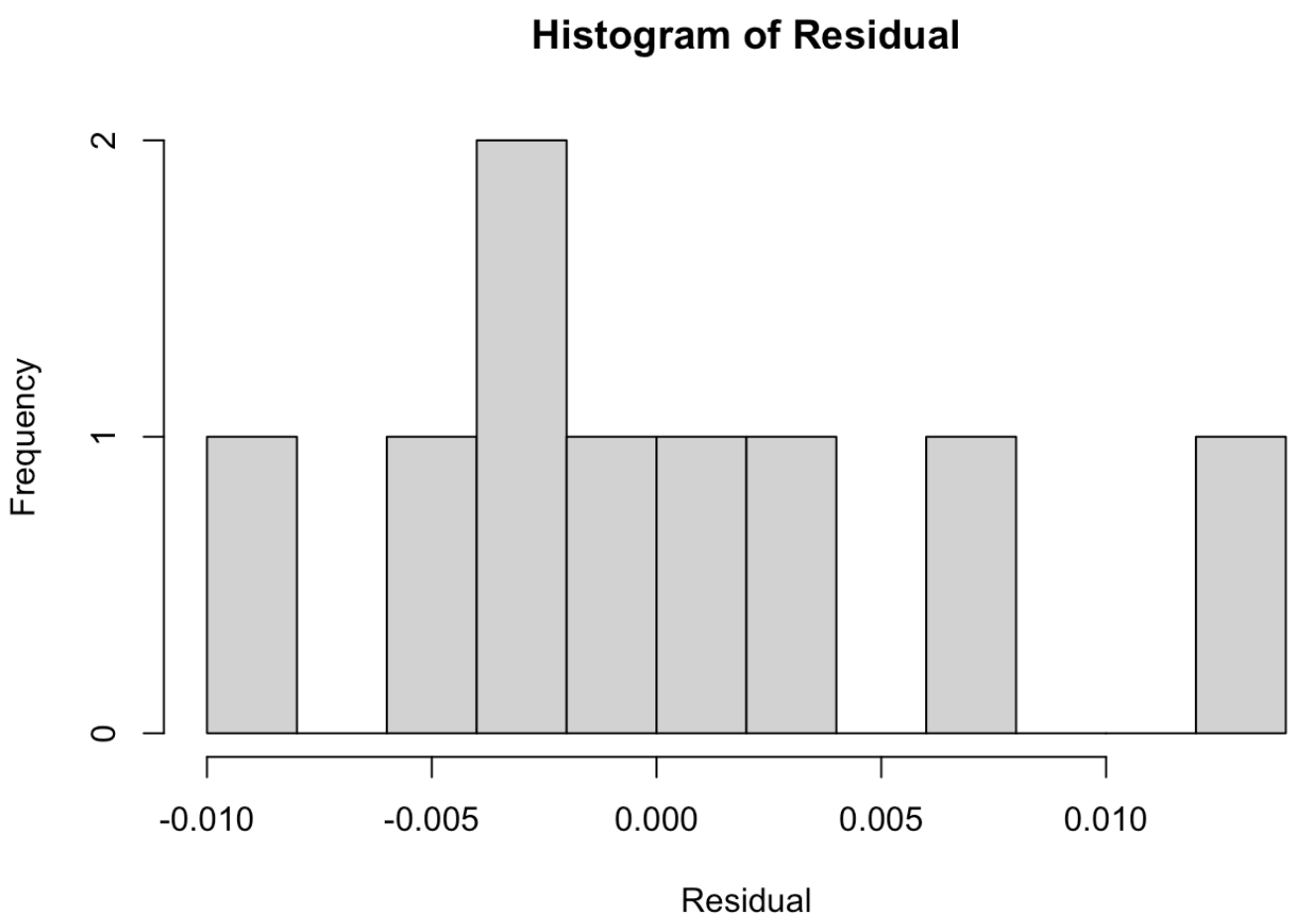
```
#データセットを作る
hd_smallplot <- dt3 %>%
  group_by(tre, Block, small_plot) %>%
  dplyr::summarize(Ave_hd = mean(`head.diameter`))

#箱ひげ図
p4 <- ggplot(hd_smallplot, aes(x = Block, y = `Ave_hd`)) +
  geom_boxplot(aes(color = tre), outlier.color = NA) +
  geom_dotplot(binaxis = "y", binwidth = 0.07, stackdir = "center", alpha = 0.3) +
  facet_grid( .~tre) +
  guides(color = "none") +
  labs(y = "Average of head diameter in small plot") +
  theme_classic() +
  theme(text = element_text(family = "Times New Roman"),
        axis.text = element_text(size = 14),
        axis.title = element_text(size = 14),
        strip.text = element_text(size = 14))
print(p4)
```



2. 統計解析(CV)

大プロット内の局所的なばらつきを肥料処理間で比較する。小プロットごとのCVを算出する。 ###### 想定するモデル式：CV = 総CV平均μ + 肥料処理 + (Block)*ランダム効果

```
#データフレームを作る
cv_bw_smallplot <- hd_smallplot %>%
  group_by(tre, Block) %>%
  dplyr::summarize(CV = sd(Ave_hd)/mean(Ave_hd))

#誤差項の正規性を確認
lmer_cv_bw_smallplot <- lmer(data = cv_bw_smallplot,
           CV ~ tre + (1|Block))
predicted_value <- predict(lmer_cv_bw_smallplot)
Residual <- cv_bw_smallplot$CV - predicted_value
hist(Residual, nclass = 10)
```




```
shapiro <- shapiro.test(Residual)
shapiro #W = 0.96067, p-value = 0.8053
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Residual
## W = 0.96067, p-value = 0.8053
```

```
#等分散性の確認
bartlett.test(Residual~cv_bw_smallplot$tre)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  Residual by cv_bw_smallplot$tre
## Bartlett's K-squared = 2.3489, df = 2, p-value = 0.309
```

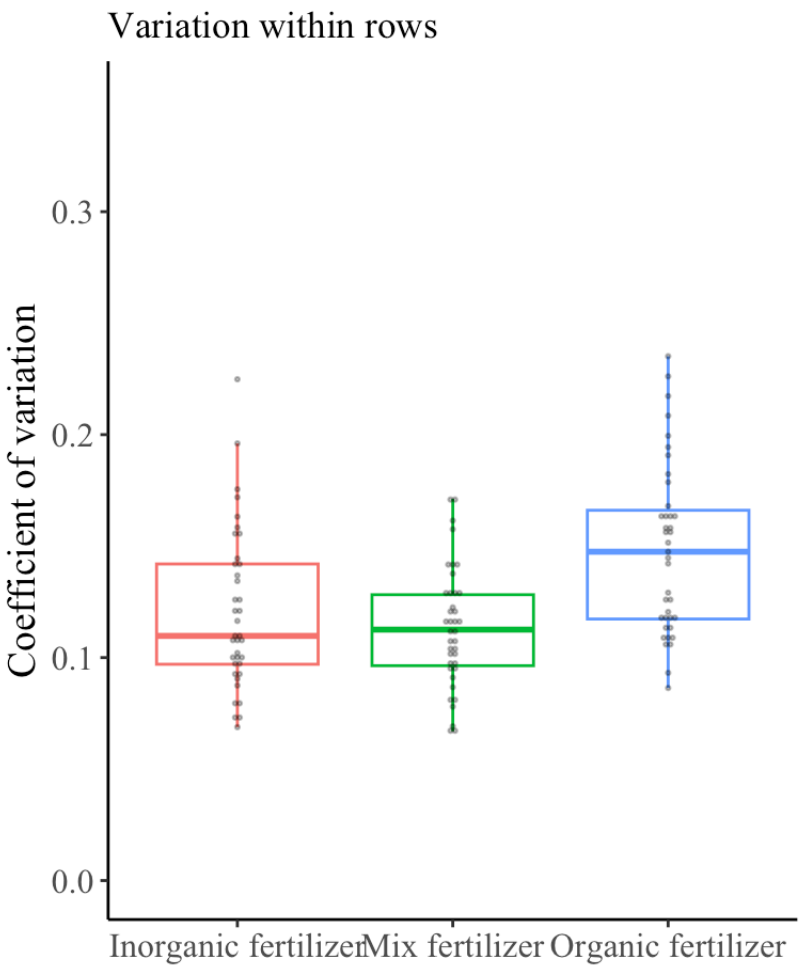
```
#Bartlett's K-squared = 2.3489, df = 2, p-value = 0.309

#分散分析
car::Anova(lmer_cv_bw_smallplot) #0.2374
```

```
## Analysis of Deviance Table (Type II Wald chisquare tests)
##
## Response: CV
##      Chisq Df Pr(>Chisq)
## tre 2.8757  2     0.2374
```

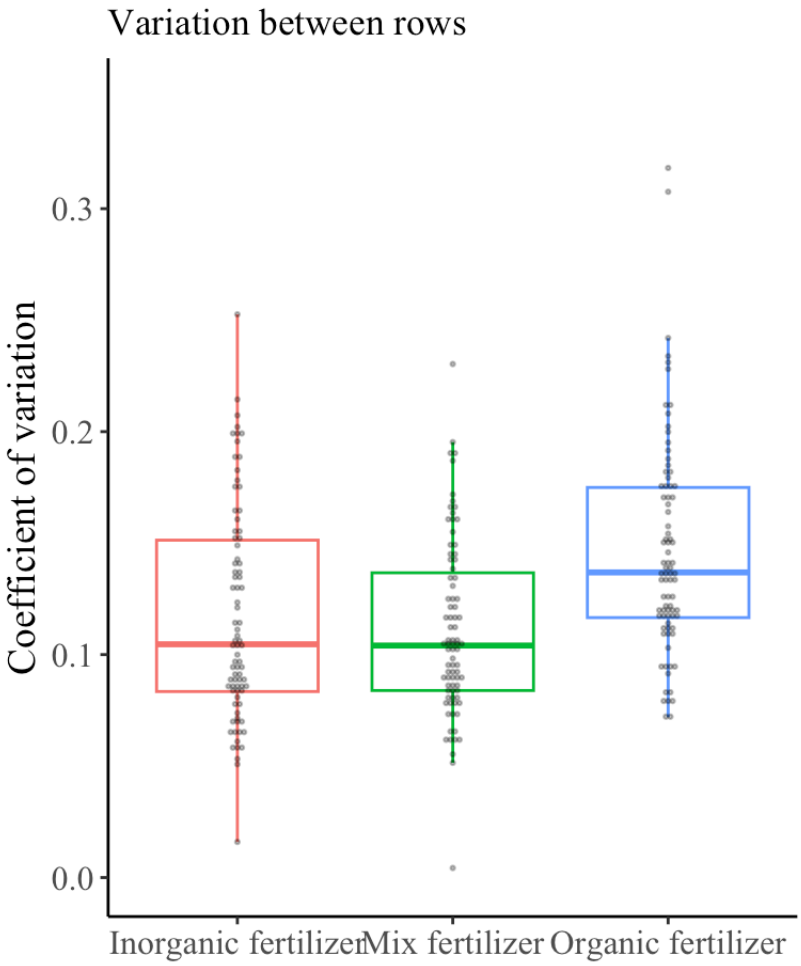
[3.3.3.3&4 列内・列間の個体間差異] - Fig 3-8&9のレシピ

1. 可視化(箱ひげ図)

```r
#列間(横列)の数を揃える
dt4 <- dt3 %>%
  filter(`NS_info` != 10,
         `NS_info` != 39,
         `NS_info` != 40,
         `NS_info` != 59,
         `NS_info` != 88,
         `NS_info` != 89,
         `NS_info` != 90,
         `NS_info` != 91,
         `NS_info` != 100,
         `NS_info` != 101,
         `NS_info` != 130,
         `NS_info` != 131,
         `NS_info` != 132,
         `NS_info` != 134)

#EW_infoの7と20、個体が破壊試験により存在しないことが判明。→除く
dt5 <- dt4 %>%
  filter(EW_info != 7,
         EW_info != 20)

#以下このdt5を使って解析
#データフレームを作る for 列内
#各列のCVを算出
cv_in_row <- dt5 %>%
  group_by(EW_info, tre, Block) %>%
  dplyr::summarize(CV = sd(`head.diameter`)/mean(`head.diameter`))

#列内の個体間差異を可視化
p5 <- ggplot(cv_in_row, aes(x = tre, y = `CV`)) +
  geom_boxplot(aes(color = tre), outlier.color = NA) +
  geom_dotplot(binaxis = "y", binwidth = 0.002, stackdir = "center", alpha = 0.3) +
  guides(color = "none") +
  labs(x = "",
       y = "Coefficient of variation") +
  ggtitle("Variation within rows") +
   ylim(c(0, 0.35)) +
  theme_classic() +
  theme(text = element_text(family = "Times New Roman"),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 14),
        strip.text = element_text(size = 12))
print(p5)
```




```
#データフレームを作る for 列間
cv_bw_row <- dt5 %>%
  group_by(NS_info, tre, Block) %>%
  dplyr::summarize(CV = sd(`head.diameter`)/mean(`head.diameter`)) %>%
  filter(NS_info != 129)

#列間の個体間差異を可視化
p6 <- ggplot(cv_bw_row, aes(x = tre, y = `CV`)) +
  geom_boxplot(aes(color = tre), outlier.color = NA) +
  geom_dotplot(binaxis = "y", binwidth = 0.002, stackdir = "center", alpha = 0.3) +
  guides(color = "none") +
  labs(x = "",
       y = "Coefficient of variation") +
  ggtitle("Variation between rows") +
  ylim(c(0, 0.35)) +
  theme_classic() +
  theme(text = element_text(family = "Times New Roman"),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 14),
        strip.text = element_text(size = 12))
print(p6)
```



2. 統計解析(CV)

列内/列間のばらつきを肥料処理間で比較する。縦列/横列ごとのCVを算出する。 ###### 想定するモデル式：CV = 総CV平均μ + 肥料処理 + (Block|EW_info)*ランダム効果として。EW_infoはBlockの入れ子になっている

Chapter4 - Prediction of Optimal Harvest Date

Erika Nishida

2023-01-24

[4.2.3.1. 花蕾の生育予測モデル] - Fig 4-3 & Fig 4-4のレシピ

必要なパッケージをインストールする

```
library(tidyverse)
library(patchwork)
```

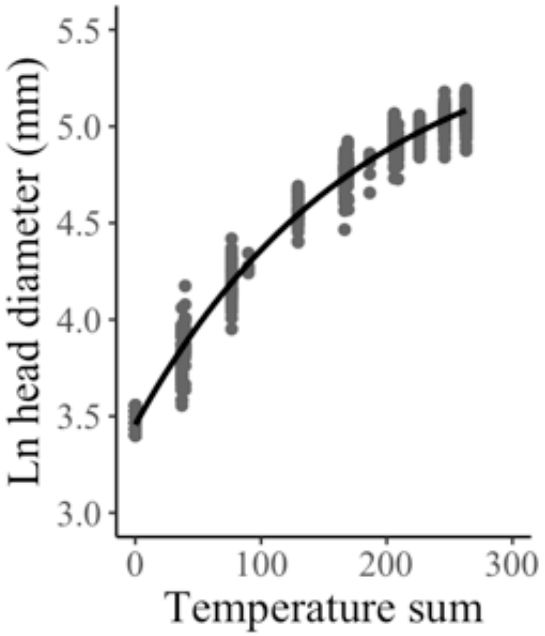
＊Parametarization by 2021 data & Validation by 2020 data

1. 手測定データを用いて、非線形回帰によるモデルのパラメータを推定する

```
#データをインポートする
#"2.30_35mm.2021dt.csv"：直径30-35 mmを含む手測定データ in 2021
dt <- read_csv("2.30_35mm.2021dt.csv")
head(dt)
```

```
## # A tibble: 6 × 2
##   `Tempsum(ii)` `Ln(HD)`
##           <dbl>    <dbl>
## 1             0     3.40
## 2             0     3.40
## 3             0     3.40
## 4             0     3.40
## 5             0     3.40
## 6             0     3.40
```

```
#Fig 4-3-bを描く
p <- ggplot(data = dt, mapping = aes(x = `Tempsum(ii)`, y = `Ln(HD)` )) +
  geom_point(color = "gray40") +
  geom_smooth(method = "nls", formula = y ~ a-b*exp(-c*x), method.args = list(star
t = list(a = 3.5, b = 2, c = 0.0074)), se = FALSE, color = "black") +
  xlim(0, 300) +
  ylim(3, 5.5) +
  labs(x = "Temperature sum",
       y = "Ln head diameter (mm)") +
  theme_classic() +
  theme(text = element_text("Times New Roman", size = 16))
print(p)
```




```
#パラメータを推定する
mod1 <- nls(`Ln(HD)`~a-b*exp(-c*`Tempsum(ii)`),
            data = dt,
            start = list(a = 3.5, b = 2, c = 0.0074),
            trace = FALSE)
summary(mod1)
```

```
##
## Formula: `Ln(HD)` ~ a - b * exp(-c * `Tempsum(ii)`)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 5.5702743  0.0411187  135.47   <2e-16 ***
## b 2.1143457  0.0380109   55.62   <2e-16 ***
## c 0.0055824  0.0002122   26.31   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08545 on 527 degrees of freedom
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 1.63e-06
```

```
a <- 5.5702744
b <- 2.1143457
c <- 0.0055824

#R^2を求める→97.89
summary.aov(lm(dt$`Ln(HD)`~1))
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## Residuals   529  182.8  0.3456
```

```
#Sum Sq = SSY
SSY = 182.8
SSE = 527*0.08545^2
100*(SSY - SSE)/SSY
```

```
## [1] 97.89497
```

2. UAVデータを用いて、花蕾の生育予測モデルを構築する

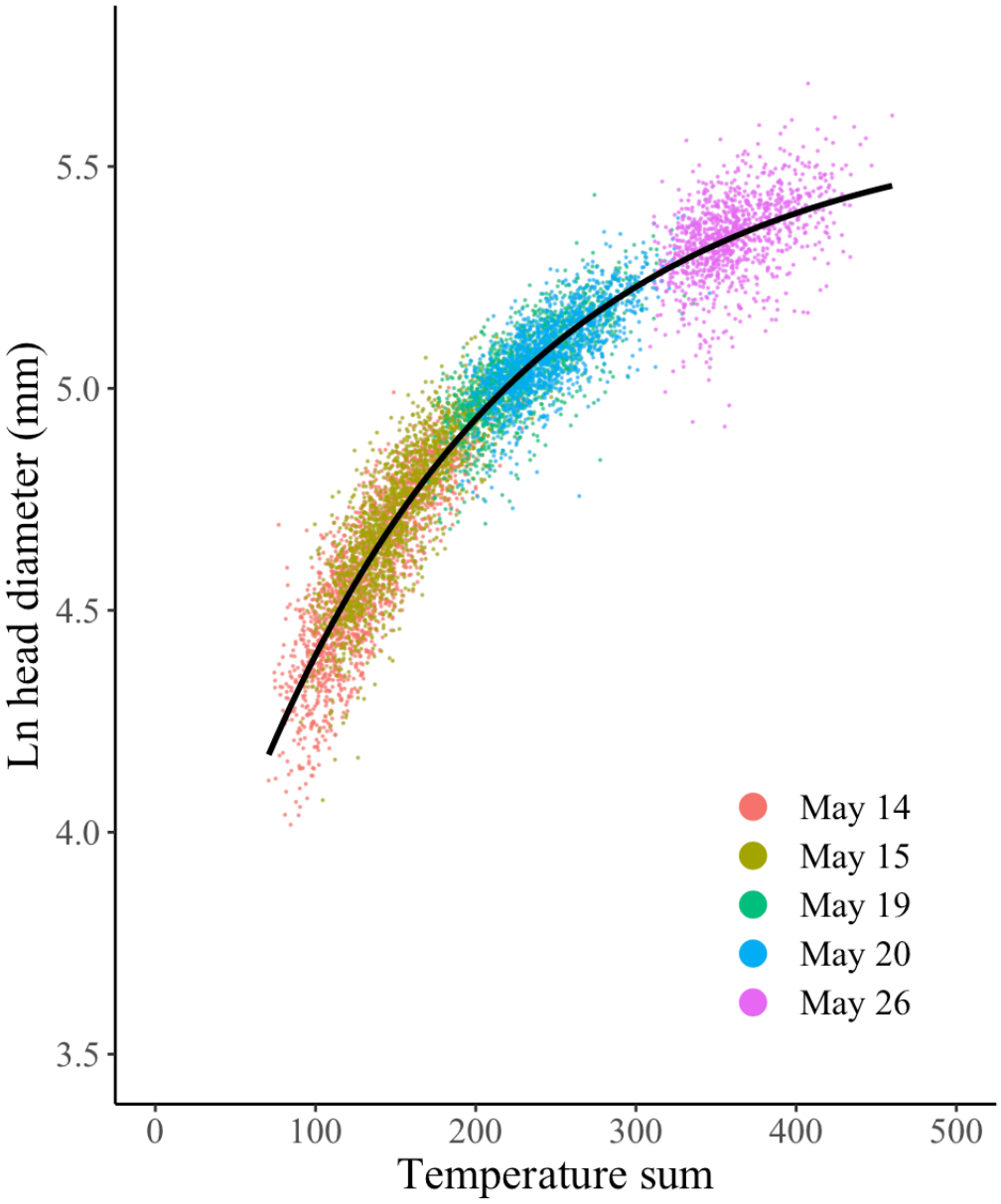
```
#"3.2021_broccoli.csv"：ドローン空撮画像から推定した花蕾直径 in 2021
dat1_tidy <- read_csv("3.2021_broccoli.csv")
head(dat1_tidy)
```

```
## # A tibble: 6 × 13
##       date field label pos   `area(mm2)` conve…¹ eccen…² equiv…³ major…⁴ minor…
⁵
##      <dbl> <chr> <dbl> <chr>       <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl
>
## 1 20210512 24-2    285 West         2371    2529   0.485    54.9    59.2    51.
8
## 2 20210514 24-2    285 West         4424    4565   0.439    75.1    79.4    71.
3
## 3 20210515 24-2    285 West         5680    5766   0.315    85.0    88.0    83.
5
## 4 20210519 24-2    285 West        12854   13470   0.501   128.    139.    120.
## 5 20210520 24-2    285 West        14110   14488   0.490   134.    145.    126.
## 6 20210526 24-2    285 West        20576   21540   0.504   162.    176.    152.
## # … with 3 more variables: `perimeter(mm)` <dbl>, circularity <dbl>,
## #   weight_pred <dbl>, and abbreviated variable names ¹`convex_area(mm2)`,
## #   ²eccentricity, ³`equivalent_diameter(mm)`, ⁴`major_axis_length(mm)`,
## #   ⁵`minor_axis_length(mm)`
```

```
dat1_tidy <- dat1_tidy %>% rename(Diametermm = "major_axis_length(mm)", ID = label
, Date = date)

#使うデータ（date, ID, major_axis_length）をピックアップする
dat2_tidy <-
  dat1_tidy %>%
  select(1,3,4,9)

#破壊試験周り、周囲、マーカー周りのブロッコリーは生育の仕方が通常と異なる可能性があるので、それらのデータを抜いた。
#エクセル上でMapと照らし合わせて抜く方が正確なので、一旦書き出して、編集した上で再度インポートする
dat2_wide <- pivot_wider(dat2_tidy, names_from = "Date", values_from = "Diametermm
")
write.csv(dat2_wide, file = "dat2_wide.csv")
dat3_wide <- read.csv(file = "4.dat2_wide_updated2021.csv") %>%
  rename("20210512" = X20210512, "20210514" = X20210514,
         "20210515" = X20210515, "20210519" = X20210519,
         "20210520" = X20210520, "20210526" = X20210526)
```

```
#明らかな外れ値（継時的に花蕾直径が小さくなる個体）を除く
dat4_wide <- dat3_wide %>%
  mutate(NG = if_else(`20210512`>`20210514`|`20210514`>`20210515`|
                  `20210515`>`20210519`|`20210519`>`20210520`|
                  `20210520`>`20210526`, "NG", "OK")) %>%
  replace_na(list(NG = "NA"))

dat4_wide <- dat4_wide %>%
  filter(NG != "NG") %>%
  filter(ID != 346, ID != 2275,
         ID != 3464, ID != 1189,
         ID != 2634) %>%
  select(-`NG`)

#最初のドローン空撮日(2021年5月12日)における、UAVデータの積算温度を算出する
dat5_wide <- dat4_wide %>%
  mutate("0512" = log((a - log(`20210512`))/b)/-c, .after = `20210512`)

#その後のドローン空撮日の積算温度を算出する
tempsum <- dat5_wide %>%
  select(1,4) %>%
  mutate("20210514" = `0512` + 17.4 + 15.7) %>%  #気温実測値
  mutate("20210515" = `20210514` + 20) %>%
  mutate("20210519" = `20210515` + 20 + 19.5 + 20 + 20) %>%
  mutate("20210520" = `20210519` + 17.1) %>%
  mutate("20210526" = `20210520` + 18.2 + 20 + 19.7 + 19.3 + 20 + 20) %>%
  select(-`0512`) %>%
  gather(-ID, key = "Date", value = "Temp.sum")

#log(直径)データを用意する
HD <- dat5_wide %>%
  select(-2, -3, -4) %>%
  gather(-ID, key = "Date", value = "HD") %>%
  mutate("log(HD)" = log(`HD`))

#tempsumとHDのデータフレームをくっつける
dat6_tidy <- left_join(tempsum, HD, by = c("Date","ID"))

#可視化
#Fig 4-3-dを描く
p1 <- ggplot(data = dat6_tidy, mapping = aes(x = `Temp.sum`, y = `log(HD)`)) +
  geom_point(mapping = aes(color = factor(Date)), size = 0.06, alpha = 0.6) +
  labs(x = "Temperature sum",
       y = "Ln head diameter (mm)",
       colour = "") +
  scale_color_discrete(labels = c("May 14", "May 15", "May 19", "May 20", "May 26"
)) +
  xlim(0, 500) +
  ylim(3.5, 5.75) +
  geom_smooth(method = "nls", formula = y ~ a-b*exp(-c*x), method.args = list(star
t = list(a = 3.5, b = 2, c = 0.0074)),se = FALSE, color = "black") +
  theme_classic() +
  theme(text = element_text("Times New Roman", size = 16),
        legend.position = c(0.8, 0.21)) +
```

```
  guides(colour=guide_legend(override.aes = list(alpha=1,size=4)))

print(p1)
```




```
#パラメタライズ用にデータ加工
dat7_tidy <- dat6_tidy %>%
  select(3,5)
#3母数モデル, 30-35mmdataを元に
mod3 <- nls(`log(HD)`~a-b*exp(-c*`Temp.sum`),
            data = dat7_tidy,
            start = list(a = 3.5, b = 2, c = 0.0074),
            trace = TRUE)
```

```
## 27147.65    (2.27e+01): par = (3.5 2 0.0074)
## 66.93283    (5.21e-01): par = (5.579852 2.097985 0.005522424)
## 52.63135    (4.77e-03): par = (5.604958 2.156843 0.005817305)
## 52.63016    (6.68e-06): par = (5.606283 2.159589 0.005814555)
```

```
summary(mod3)
```

```
##
## Formula: `log(HD)` ~ a - b * exp(-c * Temp.sum)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 5.606e+00  9.490e-03   590.7   <2e-16 ***
## b 2.160e+00  9.083e-03   237.8   <2e-16 ***
## c 5.815e-03  8.717e-05    66.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08366 on 7519 degrees of freedom
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 6.684e-06
##   ( 6118 個の観測値が欠損のため削除されました )
```

```
a2 <- 5.606e+00
b2 <- 2.160e+00
c2 <- 5.815e-03

#R^2を求める
summary.aov(lm(dat7_tidy$`log(HD)`~1))
```

```
##               Df Sum Sq Mean Sq F value Pr(>F)
## Residuals   8054  755.9 0.09385
##  5585 個の観測値が欠損のため削除されました
```

```
#Sum Sq = SSY
SSY = 755.9
SSE = 7519*0.08391^2
100*(SSY - SSE)/SSY
```

```
## [1] 92.99637
```

3. 生育予測モデルの精度検証

構築した2021年生育予測モデルを用いて、2020年度の花蕾直径を予測

まず予測値を用意する

```
#上記で推定したパラメータを入力する
a <- a2
b <- b2
c <- c2

#データをインポートする
dat1v_tidy <- read_csv("6.2020_broccoli.csv")
head(dat1v_tidy)
```

```
## # A tibble: 6 × 12
##       date label  area convex_…¹ eccen…² equiv…³ major…⁴ minor…⁵ min_a…⁶ min_a…
⁷
##      <dbl> <dbl> <dbl>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl
>
## 1 20200520   292  1276      1344   0.817    40.3    54.1    31.2    53.3    30.
3
## 2 20200520     1  3606      3709   0.520    67.8    73.5    62.8    69.7    66.
0
## 3 20200520   148  4273      4434   0.679    73.8    87.1    63.9    82.4    66.
7
## 4 20200520   293  5143      5282   0.450    80.9    85.8    76.6    83.2    76.
3
## 5 20200520     2  4276      4490   0.521    73.8    80.4    68.7    77.7    69.
2
## 6 20200520   294  8421      8726   0.487   104.    111.     97.3   103.    102.
## # … with 2 more variables: perimeter <dbl>, circularity <dbl>, and abbreviated
## #   variable names ¹convex_area, ²eccentricity, ³equivalent_diameter,
## #   ⁴major_axis_length, ⁵minor_axis_length, ⁶min_area_rect_max,
## #   ⁷min_area_rect_min
```

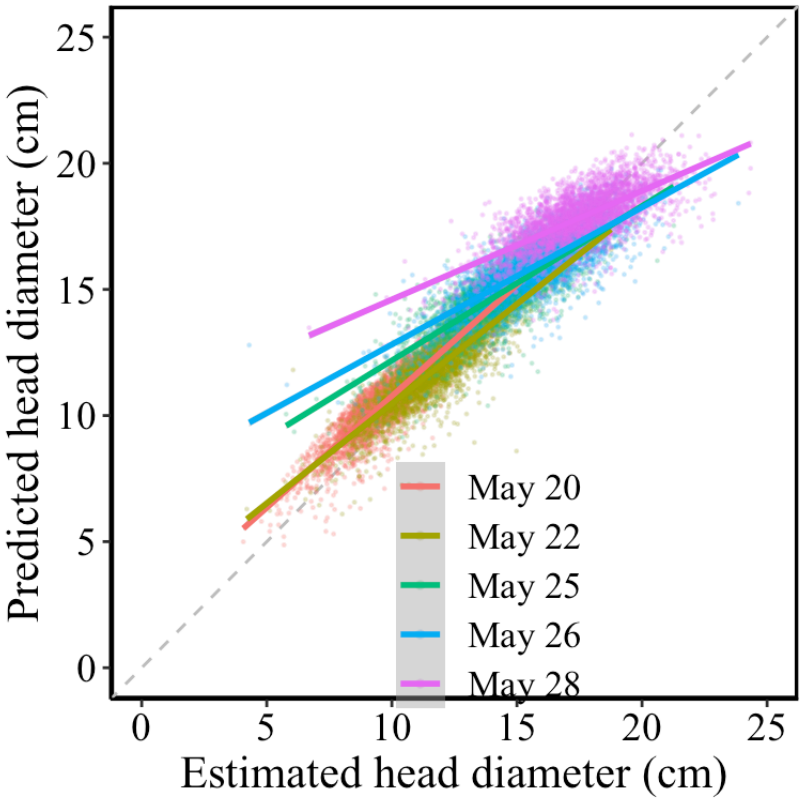
```
dat1v_tidy <- dat1v_tidy %>% rename(Diametermm = "major_axis_length", ID = label,
Date = date)

#使うデータ（date, ID, major_axis_length）をピックアップする
dat2v_tidy <-
  dat1v_tidy %>%
  select(1,2,7)

dat2v_wide <- pivot_wider(dat2v_tidy, names_from = "Date", values_from = "Diameter
mm")

#破壊試験周り、周囲、マーカー周りのブロッコリーは生育の仕方が通常と異なる可能性があるので、それらの
データを抜いた。
#エクセル上でMapと照らし合わせて抜く方が正確なので、一旦書き出して、編集した上で再度インポートする
write_csv(dat2v_wide, file = "dat2v_wide.csv")
dat3v_wide <- read.csv(file = "7.dat2v_wide_updated2020.csv") %>%
  rename("20200520" = X20200520, "20200522" = X20200522,
         "20200525" = X20200525, "20200526" = X20200526,
         "20200528" = X20200528)

#5/18のデータを追加する
"0518" <- dat2v_wide %>%
```

```
  select("ID", "20200518")
dat3v_wide2 <- left_join(dat3v_wide, `0518`, by = "ID")

#明らかな外れ値（継時的に花蕾直径が小さくなる個体）を除く
dat4v_wide <- dat3v_wide2 %>%
  mutate(NG = if_else(`20200518`>`20200520`|`20200520`>`20200522`|`20200522`>`2020
0525`|
                      `20200525`>`20200526`|`20200526`>`20200528`, "NG", "OK")) %>%
  replace_na(list(NG = "NA"))

dat4v_wide <- dat4v_wide %>%
  filter(NG != "NG") %>%
  filter(ID != 548, ID != 604,
         ID != 689, ID != 2377,
         ID != 4875, ID != 5686,
         ID != 5686, ID != 6787,
         ID != 1472, ID != 1564,
         ID != 2249, ID != 2466,
         ID != 4937, ID != 6721,
         ID != 7378)

dat5v_wide <- dat4v_wide %>%
  subset(!(is.na(dat4v_wide$`20200518`)))

#1. ドローン空撮初日(5/18)までの積算温度を算出する
dat6v_wide <- dat5v_wide %>%
  mutate("0518" = log((a - log(`20200518`))/b)/-c, .after = `20200518`)

#2. その後の積算温度を算出する
temp_sumv_tidy <- dat6v_wide %>%
  select(`ID`,`0518`) %>%
  mutate("0519" = `0518`+ 19) %>%
  mutate("0520" = `0519`+ 17.3) %>%
  mutate("0521" = `0520`+ 12.4) %>%
  mutate("0522" = `0521`+ 11.6) %>%
  mutate("0523" = `0522`+ 15.4) %>%
  mutate("0524" = `0523`+ 18.9) %>%
  mutate("0525" = `0524`+ 20) %>%
  mutate("0526" = `0525`+ 20) %>%
  mutate("0527" = `0526`+ 20) %>%
  mutate("0528" = `0527`+ 20) %>%
  gather(-ID, key = `Date`, value = Tempsum)

#3. 花蕾直径を生育予測モデルを用いて予測する
pred_valuev_tidy <- temp_sumv_tidy %>%
  mutate("pred_HD" = exp(a - b*exp(-c*`Tempsum`))) %>%
  mutate("pred_HD_cm" = `pred_HD`/10) %>%
  select(-`Tempsum`, -`pred_HD`)

pred_valuev_wide <- pred_valuev_tidy %>%
  spread(key = Date, value = pred_HD_cm) %>%
  select(`ID`, `0520`, `0522`, `0525`, `0526`, `0528`)
```

(生育予測モデルによる)予測値と(ドローン空撮画像からの)推定値を比較する

```
#4. 推定値と実際の数値を並べたデータフレームを作る
dat6v_wide <- dat5v_wide %>%
  select(-`20200518`, -NG)

HDv_pred_true <- dat6v_wide %>%
  inner_join(pred_valuev_wide, by = "ID") %>%
  mutate("True0520" = `20200520`/10,
         "True0522" = `20200522`/10,
         "True0525" = `20200525`/10,
         "True0526" = `20200526`/10,
         "True0528" = `20200528`/10,
         "Pred0520" = `0520`,
         "Pred0522" = `0522`,
         "Pred0525" = `0525`,
         "Pred0526" = `0526`,
         "Pred0528" = `0528`)

HDv_pred_true_2_true <- HDv_pred_true %>%
  select(`ID`,`True0520`, `True0522`, `True0525`, `True0526`, `True0528`) %>%
  rename("May 20" = `True0520`,
         "May 22" = `True0522`,
         "May 25" = `True0525`,
         "May 26" = `True0526`,
         "May 28" = `True0528`) %>%
  gather(-ID, key = Date, value = TrueHD)

HDv_pred_true_2_pred <- HDv_pred_true %>%
  select(`ID`,`Pred0520`, `Pred0522`, `Pred0525`, `Pred0526`, `Pred0528`) %>%
  rename("May 20" = `Pred0520`,
         "May 22" = `Pred0522`,
         "May 25" = `Pred0525`,
         "May 26" = `Pred0526`,
         "May 28" = `Pred0528`) %>%
  gather(-ID, key = Date, value = PredHD)

HDv_pred_true_2 <- full_join(HDv_pred_true_2_true, HDv_pred_true_2_pred, by = c("I
D", "Date"))

#図示
#散布図＋近似直線
#Fig 4-4-aを描く

p2 <- ggplot(data = HDv_pred_true_2) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", colour = "grey") +
  geom_point(mapping = aes(x = `TrueHD`, y = `PredHD`, colour = `Date`), size = 0.
2, alpha = 0.2) +
  geom_smooth(mapping = aes(x = `TrueHD`, y = `PredHD`, colour = `Date`), method =
"lm", size = 1) +
  xlim(0,25) +
  ylim(0,25) +
  labs(x = "Estimated head diameter (cm)",
       y = "Predicted head diameter (cm)",
       colour = "") +
```

```
  theme_classic() +
  guides(colour = guide_legend(override.aes = list(size = 1))) +
  theme(
        axis.text = element_text("Times New Roman",size = 14, colour = "black"), #
目盛の数字
        legend.text = element_text("Times New Roman",size = 13),
        axis.title = element_text("Times New Roman",size = 16),#軸タイトル
        legend.title = element_text("Times New Roman",size = 16),
        legend.position = c(0.55, 0.21),
        legend.background = element_blank(),
        legend.key = element_blank(),
        panel.grid.major = element_blank(),   #グリッドは入れない
        panel.grid.minor = element_blank(),   #グリッドは入れない
        panel.background = element_rect(fill = "white", colour = "black", size = 1
.2)
#fillは枠内の色、colourは枠線の色、sizeは枠線の太さ
        )
print(p2)
```




```
#r^2を求める
model0520 <- lm(HDv_pred_true$Pred0520~HDv_pred_true$`True0520`)
summary(model0520) #R^2→0.8174
```

```
##
## Call:
## lm(formula = HDv_pred_true$Pred0520 ~ HDv_pred_true$True0520)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4058 -0.3330  0.0445  0.4170  1.4499
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              1.933909   0.067102   28.82   <2e-16 ***
## HDv_pred_true$True0520 0.883380   0.006757  130.73   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6083 on 3814 degrees of freedom
##   ( 2772 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.8175, Adjusted R-squared:  0.8175
## F-statistic: 1.709e+04 on 1 and 3814 DF,  p-value: < 2.2e-16
```

```
model0522 <- lm(HDv_pred_true$Pred0522~HDv_pred_true$`True0522`)
summary(model0522) #R^2→0.7773
```

```
##
## Call:
## lm(formula = HDv_pred_true$Pred0522 ~ HDv_pred_true$True0522)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -5.8117 -0.3671  0.0430  0.4434  4.4548
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              2.574481   0.083542   30.82   <2e-16 ***
## HDv_pred_true$True0522 0.789486   0.006951  113.57   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6688 on 3692 degrees of freedom
##   ( 2894 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.7775, Adjusted R-squared:  0.7774
## F-statistic: 1.29e+04 on 1 and 3692 DF,  p-value: < 2.2e-16
```

```
model0525 <- lm(HDv_pred_true$Pred0525~HDv_pred_true$`True0525`)
summary(model0525) #R^2→0.684
```

```
##
## Call:
## lm(formula = HDv_pred_true$Pred0525 ~ HDv_pred_true$True0525)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1426 -0.4283  0.0127  0.4700  3.0173
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              6.057719   0.100011   60.57   <2e-16 ***
## HDv_pred_true$True0525 0.612336   0.006848   89.41   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.733 on 3693 degrees of freedom
##   ( 2893 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.684,  Adjusted R-squared:  0.6839
## F-statistic:  7995 on 1 and 3693 DF,  p-value: < 2.2e-16
```

```
model0526 <- lm(HDv_pred_true$Pred0526~HDv_pred_true$`True0526`)
summary(model0526) #→0.623
```

```
##
## Call:
## lm(formula = HDv_pred_true$Pred0526 ~ HDv_pred_true$True0526)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8524 -0.4379  0.0098  0.4736  3.0731
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)              7.383392   0.111235   66.38   <2e-16 ***
## HDv_pred_true$True0526 0.543272   0.007014   77.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7613 on 3631 degrees of freedom
##   ( 2955 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.623,  Adjusted R-squared:  0.6229
## F-statistic:  6000 on 1 and 3631 DF,  p-value: < 2.2e-16
```

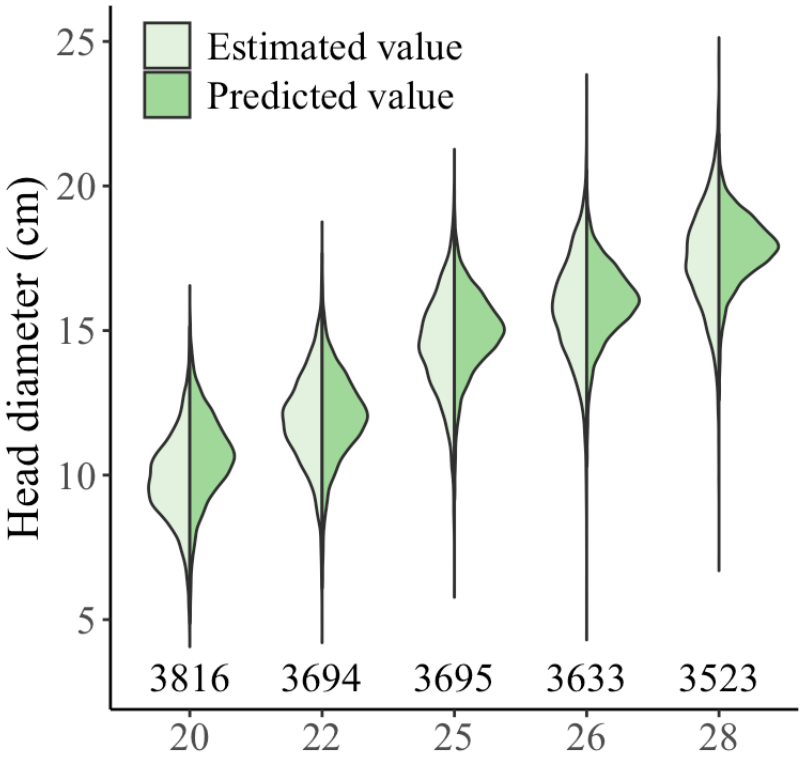
```
model0528 <- lm(HDv_pred_true$Pred0528~HDv_pred_true$`True0528`)
summary(model0528) #→0.5187
```

```
##
## Call:
## lm(formula = HDv_pred_true$Pred0528 ~ HDv_pred_true$True0528)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7656 -0.4492  0.0176  0.4993  2.4044
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             10.300914   0.122741   83.92   <2e-16 ***
## HDv_pred_true$True0528   0.430143   0.006983   61.59   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7633 on 3521 degrees of freedom
##   ( 3065 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.5187, Adjusted R-squared:  0.5185
## F-statistic:  3794 on 1 and 3521 DF,  p-value: < 2.2e-16
```

```
#violin plot
#Fig4-4-cを描く

#必要なデータのみ抽出する
HDv_pred_true_D<- HDv_pred_true %>%
  select(c("ID","True0520","Pred0520", "True0522","Pred0522","True0525","Pred0525"
,
           "True0526","Pred0526","True0528","Pred0528"))

HDv_pred_true_Dtidy <- HDv_pred_true_D %>%
  gather(-ID, key = `Date`,value = HD)

#violin plot描写に都合の良いようにデータを成形
HDv_pred_true_Dtidy_PT <- HDv_pred_true_Dtidy %>%
  mutate(TF = if_else(Date == "Pred0520"|Date == "Pred0522"|Date == "Pred0525"|
                      Date == "Pred0526"|Date == "Pred0528",
                      "Predicted value", "Estimated value")) %>%
  mutate(date = case_when(
    Date == "Pred0520" ~ "20",
    Date == "True0520" ~ "20",
    Date == "Pred0522" ~ "22",
    Date == "True0522" ~ "22",
    Date == "Pred0525" ~ "25",
    Date == "True0525" ~ "25",
    Date == "Pred0526" ~ "26",
    Date == "True0526" ~ "26",
    Date == "Pred0528" ~ "28",
    Date == "True0528" ~ "28",))

#violin plotを描く関数を拡張する
#拡張した結果使い方は...
#ggplot(my_data, aes(x, y, fill = m)) + geom_split_violin()
```

```
GeomSplitViolin <- ggproto("GeomSplitViolin", GeomViolin,
                           draw_group = function(self, data, ..., draw_quantiles =
NULL) {
                             data <- transform(data, xminv = x - violinwidth * (x
- xmin), xmaxv = x + violinwidth * (xmax - x))
                             grp <- data[1, "group"]
                             newdata <- plyr::arrange(transform(data, x = if (grp
%% 2 == 1) xminv else xmaxv), if (grp %% 2 == 1) y else -y)
                             newdata <- rbind(newdata[1, ], newdata, newdata[nrow(
newdata), ], newdata[1, ])
                             newdata[c(1, nrow(newdata) - 1, nrow(newdata)), "x"]
<- round(newdata[1, "x"])

                             if (length(draw_quantiles) > 0 & !scales::zero_range(
range(data$y))) {
                               stopifnot(all(draw_quantiles >= 0), all(draw_quanti
les <=
                                                                       1))
                               quantiles <- ggplot2:::create_quantile_segment_fram
e(data, draw_quantiles)
                               aesthetics <- data[rep(1, nrow(quantiles)), setdiff
(names(data), c("x", "y")), drop = FALSE]
                               aesthetics$alpha <- rep(1, nrow(quantiles))
                               both <- cbind(quantiles, aesthetics)
                               quantile_grob <- GeomPath$draw_panel(both, ...)
                               ggplot2:::ggname("geom_split_violin", grid::grobTre
e(GeomPolygon$draw_panel(newdata, ...), quantile_grob))
                             }
                             else {
                               ggplot2:::ggname("geom_split_violin", GeomPolygon$d
raw_panel(newdata, ...))
                             }
                           })

geom_split_violin <- function(mapping = NULL, data = NULL, stat = "ydensity", posi
tion = "identity", ...,
                              draw_quantiles = NULL, trim = TRUE, scale = "area",
na.rm = FALSE,
                              show.legend = NA, inherit.aes = TRUE) {
  layer(data = data, mapping = mapping, stat = stat, geom = GeomSplitViolin,
        position = position, show.legend = show.legend, inherit.aes = inherit.aes,
        params = list(trim = trim, scale = scale, draw_quantiles = draw_quantiles,
na.rm = na.rm, ...))
}

#Fig 4-4-cを描く
p3 <- ggplot(data = HDv_pred_true_Dtidy_PT, aes(`date`, `HD`, fill = `TF`)) +
  geom_split_violin() +
  labs(x = "",
       y = "Head diameter (cm)",
       fill = "") +
  annotate("text", x = 1, y = 3, label = "3816", family = "Times New Roman", size
= 5) +
  annotate("text", x = 2, y = 3, label = "3694", family = "Times New Roman", size
```

```
= 5) +
  annotate("text", x = 3, y = 3, label = "3695", family = "Times New Roman", size
= 5) +
  annotate("text", x = 4, y = 3, label = "3633", family = "Times New Roman", size
= 5) +
  annotate("text", x = 5, y = 3, label = "3523", family = "Times New Roman", size
= 5) +
  scale_fill_brewer(palette = "YIOrRd") +
  theme_classic() +
  theme(axis.text.y = element_text("Times New Roman", size = 14),
        axis.title.y = element_text("Times New Roman", size = 16),
        axis.text.x = element_text("Times New Roman", size = 14),
        legend.text = element_text("Times New Roman", size = 14),
        legend.position = c(0.28, 0.94))
print(p3)
```



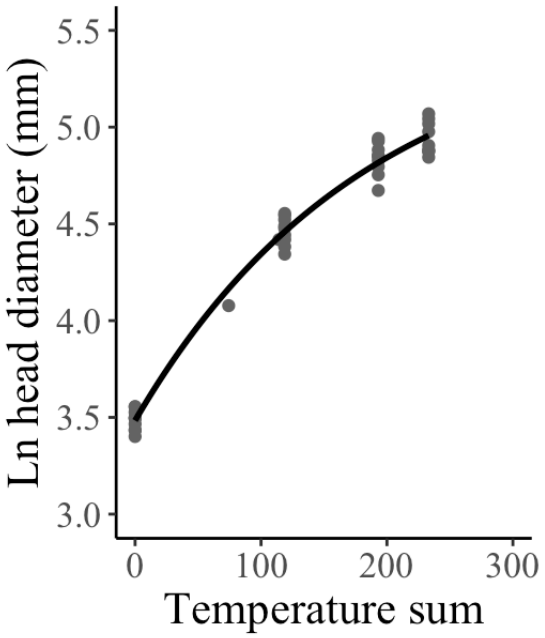
＊Parametarization by 2020 data & Validation by 2021 data

1. 手測定データを用いて、非線形回帰によるモデルのパラメータを推定する

```
#データをインポートする
#"2020.30_35mm.2020dt.csv"：直径30-35 mmを含む手測定データ in 2020
dt_2020 <- read_csv("2020.2.30-35mm.2020dt.csv")
head(dt_2020)
```

```
## # A tibble: 6 × 2
##   Temp.sum `Ln(HD)`
##      <dbl>    <dbl>
## 1        0     3.40
## 2        0     3.43
## 3        0     3.43
## 4        0     3.47
## 5        0     3.50
## 6        0     3.50
```

```
#Fig 4-3-aを描く
p4 <- ggplot(data = dt_2020, mapping = aes(x = `Temp.sum`, y = `Ln(HD)` )) +
  geom_point(color = "gray40") +
  geom_smooth(method = "nls", formula = y ~ a-b*exp(-c*x), method.args = list(star
t = list(a = 5.5, b = 2, c = 0.0055)), se = FALSE, color = "black") +
  xlim(0, 300) +
  ylim(3, 5.5) +
  labs(x = "Temperature sum",
       y = "Ln head diameter (mm)") +
  theme_classic() +
  theme(text = element_text("Times New Roman", size = 16))
print(p4)
```




```
#パラメータを推定する
mod1_2020 <- nls(`Ln(HD)`~a-b*exp(-c*`Temp.sum`),
            data = dt_2020,
            start = list(a = 5.5, b = 2, c = 0.0055),
            trace = TRUE)
```

```
## 0.1974826   (1.51e-01): par = (5.5 2 0.0055)
## 0.1930858   (2.07e-03): par = (5.53023 2.0466 0.005453053)
## 0.1930850   (1.11e-05): par = (5.530072 2.046453 0.005455088)
## 0.1930850   (2.75e-07): par = (5.530082 2.046462 0.005455039)
```

```
summary(mod1_2020)
```

```
##
## Formula: `Ln(HD)` ~ a - b * exp(-c * Temp.sum)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 5.5300824  0.1540893  35.889  < 2e-16 ***
## b 2.0464624  0.1509853  13.554 1.04e-15 ***
## c 0.0054550  0.0007323   7.449 8.53e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07324 on 36 degrees of freedom
##
## Number of iterations to convergence: 3
## Achieved convergence tolerance: 2.749e-07
```

```
#パラメータGET
d <- 5.5300823
e <- 2.0464624
f <- 0.0054550

#R^2を求める→98.52
summary.aov(lm(dt_2020$`Ln(HD)`~1))
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## Residuals   38  13.02  0.3427
```

```
#Sum Sq = SSY
SSY = 13.02
SSE = 36*0.07324^2
100*(SSY - SSE)/SSY
```

```
## [1] 98.51684
```

2. UAVデータを用いて、花蕾の生育予測モデルを構築する

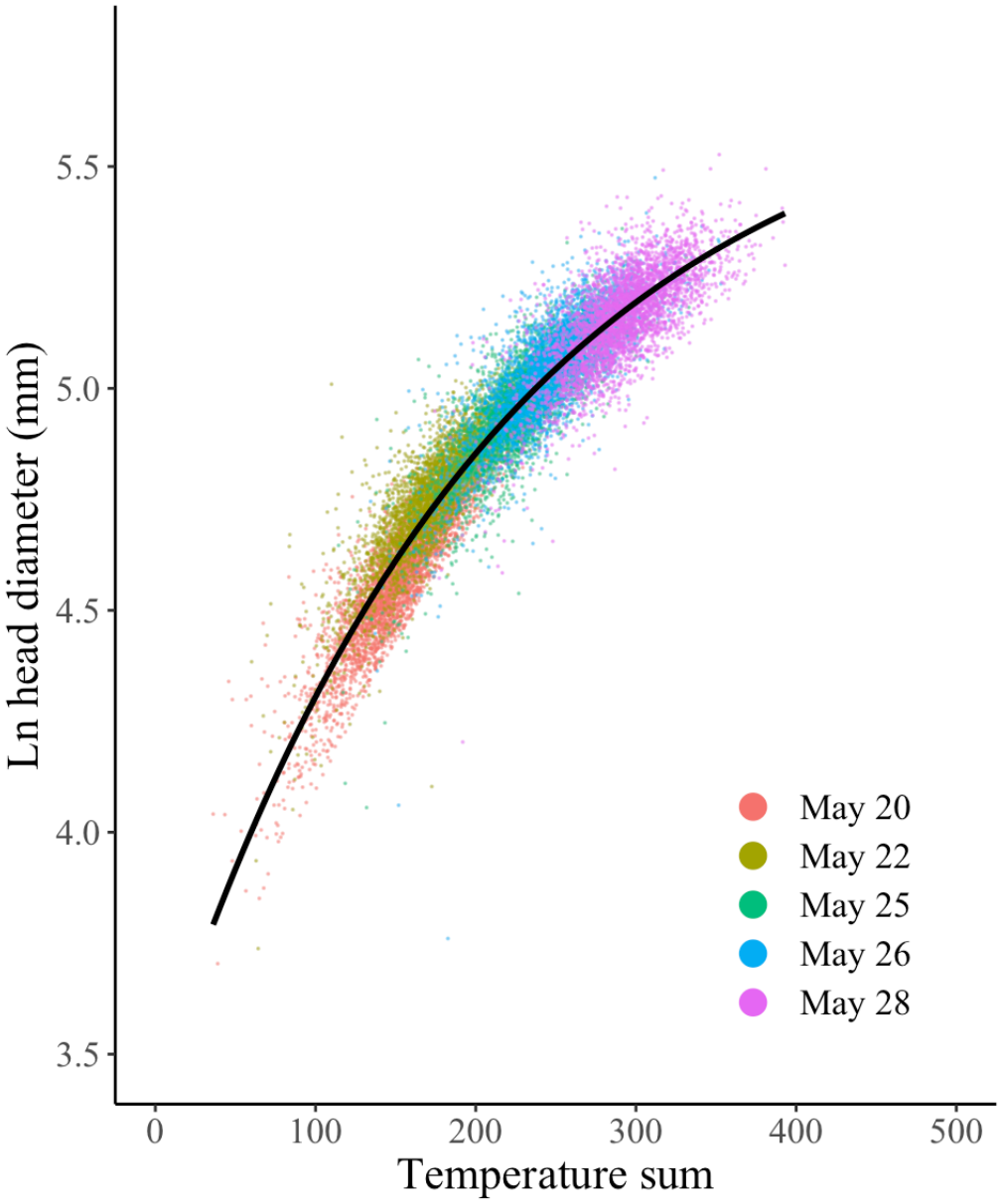
```
#import the data
dat1_tidy_2020 <- read_csv("6.2020_broccoli.csv")
head(dat1_tidy_2020)
```

```
## # A tibble: 6 × 12
##       date label  area convex_…¹ eccen…² equiv…³ major…⁴ minor…⁵ min_a…⁶ min_a…
⁷
##      <dbl> <dbl> <dbl>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl
>
## 1 20200520   292  1276      1344   0.817    40.3    54.1    31.2    53.3    30.
3
## 2 20200520     1  3606      3709   0.520    67.8    73.5    62.8    69.7    66.
0
## 3 20200520   148  4273      4434   0.679    73.8    87.1    63.9    82.4    66.
7
## 4 20200520   293  5143      5282   0.450    80.9    85.8    76.6    83.2    76.
3
## 5 20200520     2  4276      4490   0.521    73.8    80.4    68.7    77.7    69.
2
## 6 20200520   294  8421      8726   0.487   104.    111.     97.3   103.    102.
## # … with 2 more variables: perimeter <dbl>, circularity <dbl>, and abbreviated
## #   variable names ¹convex_area, ²eccentricity, ³equivalent_diameter,
## #   ⁴major_axis_length, ⁵minor_axis_length, ⁶min_area_rect_max,
## #   ⁷min_area_rect_min
```

```
dat1_tidy_2020 <- dat1_tidy_2020 %>% rename(Diametermm = "major_axis_length", ID =
label, Date = date)

#使うデータ（date, ID, major_axis_length）をピックアップする
dat2_tidy_2020 <-
  dat1_tidy_2020 %>%
  select(1,2,7)

#破壊試験周り、周囲、マーカー周りのブロッコリーは生育の仕方が通常と異なる可能性があるので、それらの
データを抜いた。
#エクセル上でMapと照らし合わせて抜く方が正確なので、一旦書き出して、編集した上で再度インポートする
dat2_wide_2020 <- pivot_wider(dat2_tidy_2020, names_from = "Date", values_from = "
Diametermm")
write_csv(dat2_wide_2020, file = "dat2v_wide.csv")
dat3_wide_2020 <- read.csv(file = "7.dat2v_wide_updated2020.csv") %>%
  rename("20200520" = X20200520, "20200522" = X20200522,
         "20200525" = X20200525, "20200526" = X20200526,
         "20200528" = X20200528)

#5/18のデータを追加する
"0518" <- dat2_wide_2020 %>%
  select("ID", "20200518")
dat3_wide2_2020 <- left_join(dat3_wide_2020, `0518`, by = "ID")

#明らかな外れ値（継時的に花蕾直径が小さくなる個体）を除く
dat4_wide_2020 <- dat3_wide2_2020 %>%
  mutate(NG = if_else(`20200518`>`20200520`|`20200520`>`20200522`|`20200522`>`2020
0525`|
                      `20200525`>`20200526`|`20200526`>`20200528`, "NG", "OK")) %>%
  replace_na(list(NG = "NA"))
```

```
dat4_wide_2020 <- dat4_wide_2020 %>%
  filter(NG != "NG") %>%
  filter(ID != 548, ID != 604,
         ID != 689, ID != 2377,
         ID != 4875, ID != 5686,
         ID != 5686, ID != 6787,
         ID != 1472, ID != 1564,
         ID != 2249, ID != 2466,
         ID != 4937, ID != 6721,
         ID != 7378)

dat4.5_wide_2020 <- dat4_wide_2020 %>%
  subset(!(is.na(dat4_wide_2020$`20200518`)))

#最初のドローン空撮日(2020年5月18日)における、UAVデータの積算温度を算出する
dat5_wide_2020 <- dat4.5_wide_2020 %>%
  mutate("0518" = log((d - log(`20200518`))/e)/-f, .after = `20200518`)

#その後のドローン空撮日の積算温度を算出する
tempsum_2020 <- dat5_wide_2020 %>%
  select(`ID`,`0518`) %>%
  mutate("20200520" = `0518` + 19 + 17.3) %>%
  mutate("20200522" = `20200520` + 12.4 + 11.6) %>%
  mutate("20200525" = `20200522` + 15.4 + 18.9 + 20) %>%
  mutate("20200526" = `20200525` + 20) %>%
  mutate("20200528" = `20200526` + 20 + 20) %>%
  select(-`0518`) %>%
  gather(-ID, key = "Date", value = "Tempsum")

#log(直径)データを用意する
HD_2020 <- dat5_wide_2020 %>%
  select(-`0518`, -`NG`, -`20200518`) %>%
  gather(-ID, key = "Date", value = "HD") %>%
  mutate("log(HD)" = log(`HD`))

#tempsumとHDのデータフレームをくっつける
dat6_tidy_2020 <- full_join(HD_2020, tempsum_2020, by = c("Date","ID"))

#可視化
#Fig 4-3-cを描く
p5 <- ggplot(data = dat6_tidy_2020, mapping = aes(`Tempsum`, `log(HD)`)) +
  geom_point(mapping = aes(color = factor(Date)), size = 0.03, alpha = 0.4) +
  labs(x = "Temperature sum",
       y = "Ln head diameter (mm)",
       colour = "") +
  scale_colour_discrete(labels = c("May 20", "May 22", "May 25", "May 26", "May 28
")) +
  xlim(0, 500) +
  ylim(3.5, 5.75) +
  geom_smooth(method = "nls", formula = y ~ a-b*exp(-c*x), method.args = list(star
t = list(a = 6, b= 2, c =0.003)), se = FALSE, colour = "black") +
  theme_classic() +
  theme(text = element_text("Times New Roman", size = 16),
        legend.position = c(0.8, 0.21)) +
```

```
  guides(colour = guide_legend(override.aes = list(alpha = 1, size =4)))

print(p5)
```




```
#パラメタライズ用にデータを用意する
dat7_tidy_2020 <- dat6_tidy_2020 %>%
  select(`log(HD)`,`Tempsum`)

#パラメータを推定する
mod3_2020 <- nls(`log(HD)`~a-b*exp(-c*`Tempsum`),
            data = dat7_tidy_2020,
            start = list(a = 6, b = 2, c = 0.003),
            trace = TRUE)
```

```
## 200.1667    (9.50e-01): par = (6 2 0.003)
## 132.0407    (5.03e-01): par = (5.708294 1.988348 0.004189321)
## 106.9792    (1.21e-01): par = (5.73159 2.308622 0.004891219)
## 105.4339    (8.80e-03): par = (5.755271 2.328966 0.004739372)
## 105.4257    (6.13e-05): par = (5.754496 2.329848 0.004749437)
## 105.4257    (3.41e-06): par = (5.754606 2.329865 0.004748848)
```

```
summary(mod3_2020)
```

```
##
## Formula: `log(HD)` ~ a - b * exp(-c * Tempsum)
##
## Parameters:
##    Estimate Std. Error t value Pr(>|t|)
## a 5.7546056  0.0135739  423.95   <2e-16 ***
## b 2.3298645  0.0057414  405.80   <2e-16 ***
## c 0.0047488  0.0000741   64.09   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07578 on 18358 degrees of freedom
##
## Number of iterations to convergence: 5
## Achieved convergence tolerance: 3.407e-06
##   ( 14579 個の観測値が欠損のため削除されました )
```

```
d2 <- 5.7546056
e2 <- 2.3298645
f2 <- 0.0047488

#R^2を求める→90.5%
summary.aov(lm(dat7_tidy_2020$`log(HD)`~1))
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## Residuals   18360   1114 0.06069
##  14579 個の観測値が欠損のため削除されました
```

```
#Sum Sq = SSY
SSY = 1114
SSE = 18358*0.07578^2
100*(SSY - SSE)/SSY
```

```
## [1] 90.53655
```

3. 生育予測モデルの精度検証

構築した2020年生育予測モデルを用いて、2021年度の花蕾直径を予測

まず予測値を用意する

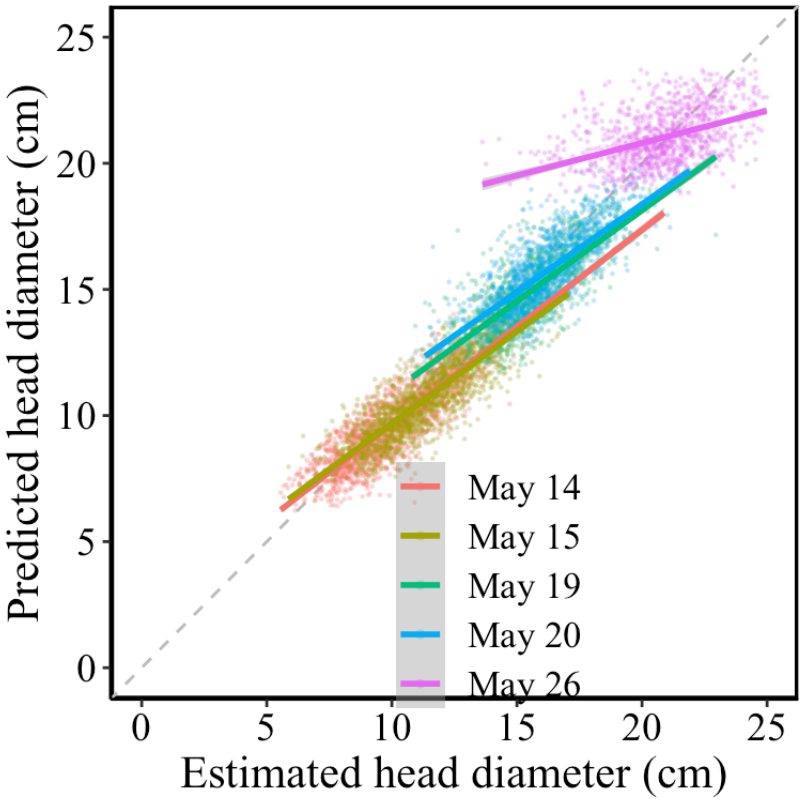
```
#設定項目↓(上記でGETしたパラメータを入力する)
d <- d2
e <- e2
f <- f2

#データをインポートする
dat1v_tidy_2021 <- read_csv("3.2021_broccoli.csv")
dat1v_tidy_2021 <- dat1v_tidy_2021 %>% rename(Diametermm = "major_axis_length(mm)"
, ID = label, Date = date)

#使うデータ（date, ID, major_axis_length）をピックアップする
dat2v_tidy_2021 <-
  dat1v_tidy_2021 %>%
  select(1,3,4,9)

dat2v_wide_2021 <- pivot_wider(dat2v_tidy_2021, names_from = "Date", values_from =
"Diametermm")

#破壊試験周り、周囲、マーカー周りのブロッコリーは生育の仕方が通常と異なる可能性があるので、それらの
データを抜いた。
#エクセル上でMapと照らし合わせて抜く方が正確なので、一旦書き出して、編集した上で再度インポートする
write.csv(dat2v_wide_2021, file = "dat2v_wide_2021.csv")
dat3v_wide_2021 <- read.csv(file = "4.dat2_wide_updated2021.csv") %>%
  rename("20210512" = X20210512, "20210514" = X20210514,
         "20210515" = X20210515, "20210519" = X20210519,
         "20210520" = X20210520, "20210526" = X20210526)

#明らかな外れ値（継時的に花蕾直径が小さくなる個体）を除く
dat4v_wide_2021 <- dat3v_wide_2021 %>%
  mutate(NG = if_else(`20210512`>`20210514`|`20210514`>`20210515`|
                  `20210515`>`20210519`|`20210519`>`20210520`|
                  `20210520`>`20210526`, "NG", "OK")) %>%
  replace_na(list(NG = "NA"))

dat4v_wide_2021 <- dat4v_wide_2021 %>%
  filter(NG != "NG") %>%
  filter(ID != 346, ID != 2275,
         ID != 3464, ID != 1189,
         ID != 2634)

dat5v_wide_2021 <- subset(dat4v_wide_2021, !(is.na(dat4v_wide_2021$`20210512`))) %
>%
  select(-pos, -NG)

#1. ドローン空撮初日(5/18)までの積算温度を算出する
dat6v_wide_2021 <- dat5v_wide_2021 %>%
  mutate("0512" = log((d - log(`20210512`))/e)/-f, .after = `20210512`)

#2. その後の積算温度を算出する
temp_sumv_tidy_2021 <- dat6v_wide_2021 %>%
  select(`ID`, `0512`) %>%
  mutate("0513" = `0512`+ 17.4) %>%
  mutate("0514" = `0513`+ 15.7) %>%
  mutate("0515" = `0514`+ 20) %>%
```

```
  mutate("0516" = `0515`+ 20) %>%
  mutate("0517" = `0516`+ 19.5) %>%
  mutate("0518" = `0517`+ 20) %>%
  mutate("0519" = `0518`+ 20) %>%
  mutate("0520" = `0519`+ 17.1) %>%
  mutate("0521" = `0520`+ 18.2) %>%
  mutate("0522" = `0521`+ 20) %>%
  mutate("0523" = `0522`+ 19.7) %>%
  mutate("0524" = `0523`+ 19.3) %>%
  mutate("0525" = `0524`+ 20) %>%
  mutate("0526" = `0525`+ 20) %>%
  gather(-ID, key = `Date`, value = Tempsum)

#3. 花蕾直径を生育予測モデルを用いて予測する
pred_valuev_tidy_2021 <- temp_sumv_tidy_2021 %>%
  mutate("pred_HD" = exp(d - e*exp(-f*`Tempsum`))) %>%
  mutate("pred_HD_cm" = `pred_HD`/10) %>%
  select(-`Tempsum`, -`pred_HD`)

pred_valuev_wide_2021 <- pred_valuev_tidy_2021 %>%
  spread(key = Date, value = pred_HD_cm) %>%
  select(`ID`, `0514`, `0515`, `0519`, `0520`, `0526`)
```

(生育予測モデルによる)予測値と(ドローン空撮画像からの)推定値を比較する

```
#4. 推定値と実際の数値を並べたデータフレームを作る
dat6v_wide_2021 <- dat5v_wide_2021 %>%
  select(!`20210512`)

HDv_pred_true_2021 <- dat6v_wide_2021 %>%
  inner_join(pred_valuev_wide_2021, by = "ID") %>%
  mutate("True0514" = `20210514`/10,
         "True0515" = `20210515`/10,
         "True0519" = `20210519`/10,
         "True0520" = `20210520`/10,
         "True0526" = `20210526`/10,
         "Pred0514" = `0514`,
         "Pred0515" = `0515`,
         "Pred0519" = `0519`,
         "Pred0520" = `0520`,
         "Pred0526" = `0526`)

HDv_pred_true_2021_true <- HDv_pred_true_2021 %>%
  select(`ID`,`True0514`, `True0515`, `True0519`, `True0520`, `True0526`) %>%
  rename("May 14" = `True0514`,
         "May 15" = `True0515`,
         "May 19" = `True0519`,
         "May 20" = `True0520`,
         "May 26" = `True0526`) %>%
  gather(-ID, key = Date, value = TrueHD)

HDv_pred_true_2021_pred <- HDv_pred_true_2021 %>%
  select(`ID`,`Pred0514`, `Pred0515`, `Pred0519`, `Pred0520`, `Pred0526`) %>%
  rename("May 14" = `Pred0514`,
```

```
         "May 15" = `Pred0515`,
         "May 19" = `Pred0519`,
         "May 20" = `Pred0520`,
         "May 26" = `Pred0526`) %>%
  gather(-ID, key = Date, value = PredHD)

HDv_pred_true_2_2021 <- full_join(HDv_pred_true_2021_true, HDv_pred_true_2021_pred
, by = c("ID", "Date"))

#図示
#散布図＋近似直線
#Fig 4-4-bを描く

p6 <- ggplot(data = HDv_pred_true_2_2021) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed", colour = "grey") +
  geom_point(mapping = aes(x = `TrueHD`, y = `PredHD`, colour = `Date`), size = 0.
2, alpha = 0.2) +
  geom_smooth(mapping = aes(x = `TrueHD`, y = `PredHD`, colour = `Date`), method =
"lm", size = 1) +
  xlim(0,25) +
  ylim(0,25) +
  labs(x = "Estimated head diameter (cm)",
       y = "Predicted head diameter (cm)",
       colour = "") +
  theme_classic() +
  guides(colour = guide_legend(override.aes = list(size =1))) +
  theme(
        axis.text = element_text("Times New Roman",size = 14, colour = "black"), #
目盛の数字
        legend.text = element_text("Times New Roman",size = 13),
        axis.title = element_text("Times New Roman",size = 16),#軸タイトル
        legend.title = element_text("Times New Roman",size = 16),
        legend.position = c(0.55, 0.21),
        legend.background = element_blank(),
        legend.key = element_blank(),
        panel.grid.major = element_blank(),   #グリッドは入れない
        panel.grid.minor = element_blank(),   #グリッドは入れない
        panel.background = element_rect(fill = "white", colour = "black", size = 1
.2)
#fillは枠内の色、colourは枠線の色、sizeは枠線の太さ
        )
print(p6)
```




```
#r^2を求める
model <- lm(HDv_pred_true_2021$Pred0514~HDv_pred_true_2021$`True0514`)
summary(model) #R^2→0.7637
```

```
##
## Call:
## lm(formula = HDv_pred_true_2021$Pred0514 ~ HDv_pred_true_2021$True0514)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8207 -0.5356 -0.0052  0.5509  2.4495
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   1.981985   0.099894   19.84   <2e-16 ***
## HDv_pred_true_2021$True0514 0.769005   0.009706   79.23   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8037 on 1942 degrees of freedom
##   ( 15 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.7637, Adjusted R-squared:  0.7636
## F-statistic:  6277 on 1 and 1942 DF,  p-value: < 2.2e-16
```

```
model0515 <- lm(HDv_pred_true_2021$Pred0515~HDv_pred_true_2021$`True0515`)
summary(model0515) #R^2→0.743
```

```
##
## Call:
## lm(formula = HDv_pred_true_2021$Pred0515 ~ HDv_pred_true_2021$True0515)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.73276 -0.54562 -0.00017  0.53296  2.67804
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   2.39930    0.12075   19.87   <2e-16 ***
## HDv_pred_true_2021$True0515  0.72879    0.01038   70.18   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8252 on 1704 degrees of freedom
##   ( 253 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.743,  Adjusted R-squared:  0.7428
## F-statistic:  4925 on 1 and 1704 DF,  p-value: < 2.2e-16
```

```
model0519 <- lm(HDv_pred_true_2021$Pred0519~HDv_pred_true_2021$`True0519`)
summary(model0519) #→0.5932
```

```
##
## Call:
## lm(formula = HDv_pred_true_2021$Pred0519 ~ HDv_pred_true_2021$True0519)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.1063 -0.7018 -0.0350  0.6523  4.5155
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   3.70527    0.24251   15.28   <2e-16 ***
## HDv_pred_true_2021$True0519  0.72149    0.01546   46.67   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9794 on 1494 degrees of freedom
##   ( 463 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.5932, Adjusted R-squared:  0.5929
## F-statistic:  2178 on 1 and 1494 DF,  p-value: < 2.2e-16
```

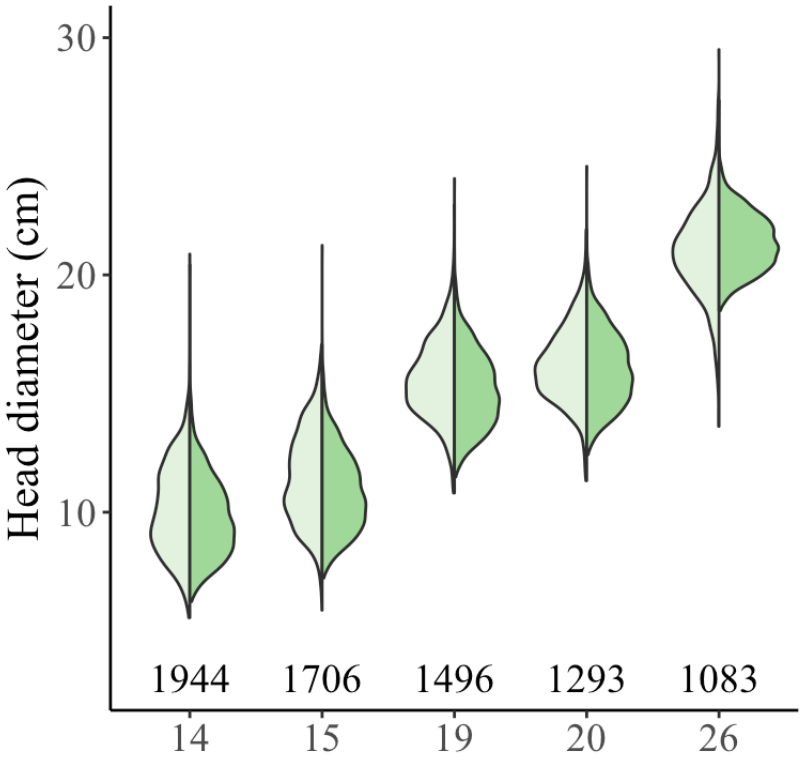
```
model0520 <- lm(HDv_pred_true_2021$Pred0520~HDv_pred_true_2021$`True0520`)
summary(model0520) #→0.571
```

```
##
## Call:
## lm(formula = HDv_pred_true_2021$Pred0520 ~ HDv_pred_true_2021$True0520)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8219 -0.6540 -0.0067  0.6366  4.1189
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                   4.47094    0.27307   16.37   <2e-16 ***
## HDv_pred_true_2021$True0520  0.69531    0.01677   41.46   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9469 on 1291 degrees of freedom
##   ( 666 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.571,  Adjusted R-squared:  0.5707
## F-statistic:  1719 on 1 and 1291 DF,  p-value: < 2.2e-16
```

```
model0526 <- lm(HDv_pred_true_2021$Pred0526~HDv_pred_true_2021$`True0526`)
summary(model0526) #→0.2247
```

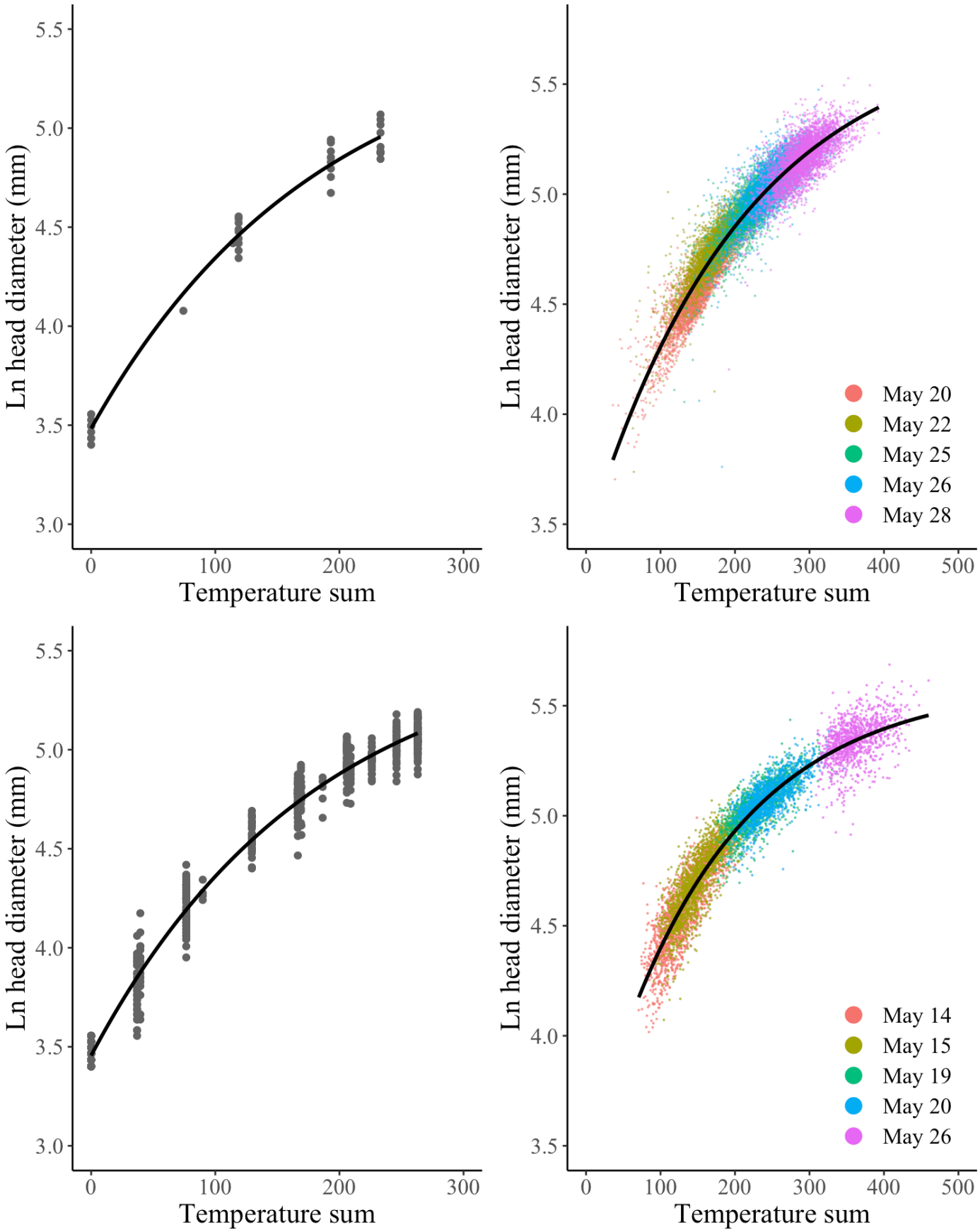
```
##
## Call:
## lm(formula = HDv_pred_true_2021$Pred0526 ~ HDv_pred_true_2021$True0526)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6730 -0.6990 -0.0755  0.6588  2.9167
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  15.59959    0.30971   50.37   <2e-16 ***
## HDv_pred_true_2021$True0526  0.26002    0.01469   17.70   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9552 on 1081 degrees of freedom
##   ( 876 個の観測値が欠損のため削除されました )
## Multiple R-squared:  0.2247, Adjusted R-squared:  0.224
## F-statistic: 313.3 on 1 and 1081 DF,  p-value: < 2.2e-16
```

```
#violin plot
#Fig4-4-dを描く

#必要なデータのみ抽出する
HDv_pred_true_D_2021<- HDv_pred_true_2021 %>%
  select(c("ID","True0514","Pred0514","True0515","Pred0515",
           "True0519","Pred0519","True0520","Pred0520",
           "True0526", "Pred0526"))

HDv_pred_true_Dtidy_2021 <- HDv_pred_true_D_2021 %>%
  gather(-ID, key = `Date`,value = HD)

HDv_pred_true_Dtidy_PT_2021 <- HDv_pred_true_Dtidy_2021 %>%
  mutate(TF = if_else(Date == "Pred0514"|Date == "Pred0515"|
                      Date == "Pred0519"|Date == "Pred0520"|
                      Date == "Pred0526",
                      "Predicted value", "Estimated value")) %>%
  mutate(date = case_when(
    Date == "Pred0514" ~ "14",
    Date == "True0514" ~ "14",
    Date == "Pred0515" ~ "15",
    Date == "True0515" ~ "15",
    Date == "Pred0519" ~ "19",
    Date == "True0519" ~ "19",
    Date == "Pred0520" ~ "20",
    Date == "True0520" ~ "20",
    Date == "Pred0526" ~ "26",
    Date == "True0526" ~ "26"))

#図示
p7 <- ggplot(data = HDv_pred_true_Dtidy_PT_2021, aes(`date`, `HD`, fill = `TF`)) +
  geom_split_violin() +
  ylim(3,30) +
  labs(x = "",
       y = "Head diameter (cm)",
       fill = "") +
  guides(fill = "none") +
  annotate("text", x = 1, y = 3, label = "1944", family = "Times New Roman", size
= 5) +
  annotate("text", x = 2, y = 3, label = "1706", family = "Times New Roman", size
= 5) +
  annotate("text", x = 3, y = 3, label = "1496", family = "Times New Roman", size
= 5) +
  annotate("text", x = 4, y = 3, label = "1293", family = "Times New Roman", size
= 5) +
  annotate("text", x = 5, y = 3, label = "1083", family = "Times New Roman", size
= 5) +
  scale_fill_brewer(palette = "YIOrRd") +
  theme_classic() +
  theme(axis.text.y = element_text("Times New Roman", size = 14),
        axis.title.y = element_text("Times New Roman", size = 16),
        axis.text.x = element_text("Times New Roman", size = 14))
print(p7)
```



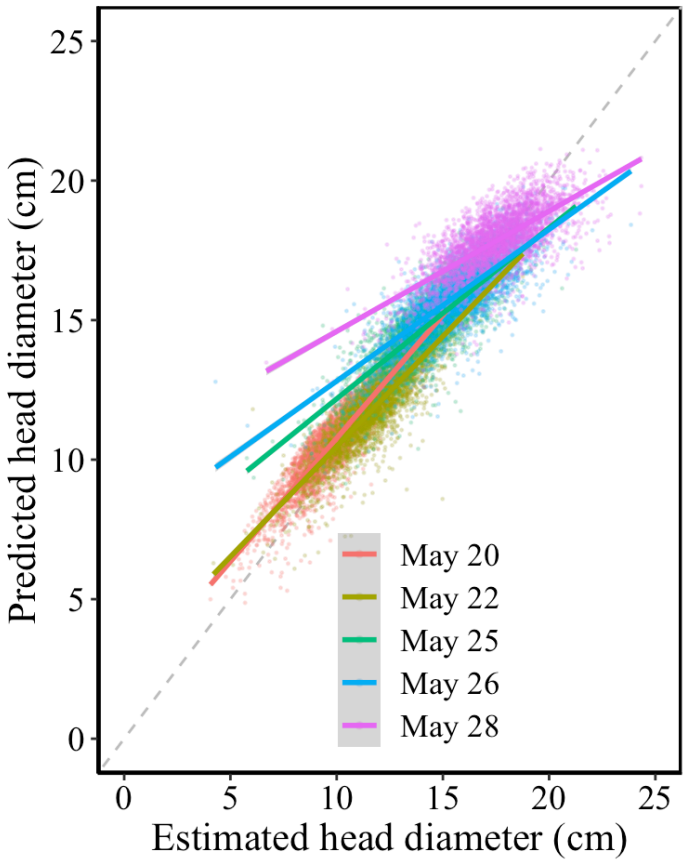
4. Fig4-3, 4-4をOutput！

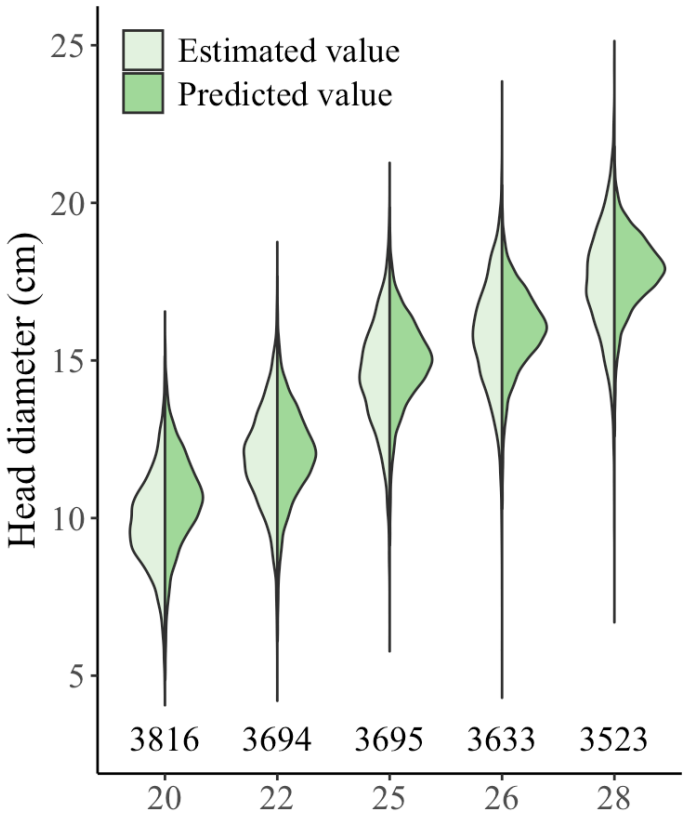
```
(p4 + p5) / (p + p1)
```

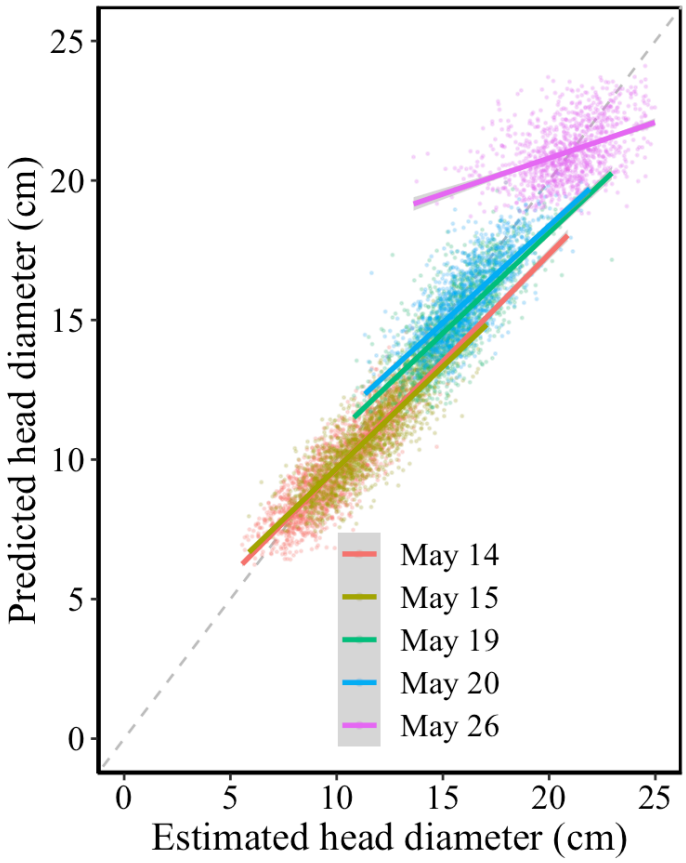


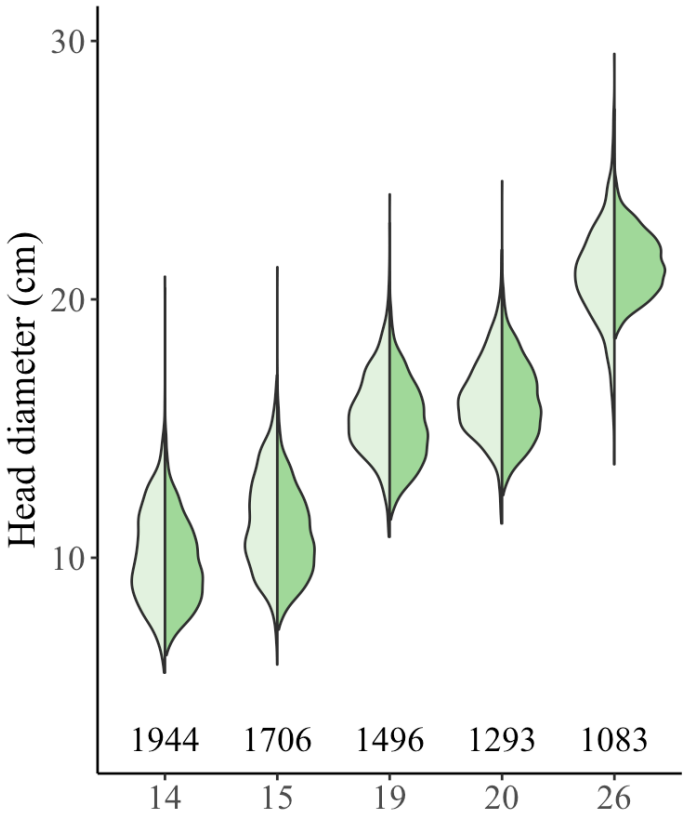

```
(p2 + p3) / (p6 + p7)
```

```
## `geom_smooth()` using formula = 'y ~ x'
## `geom_smooth()` using formula = 'y ~ x'
```


Predicted head diameter (cm)
Estimated head diameter (cm)
May 20
May 22
May 25
May 26
May 28


Estimated value
Predicted value
Head diameter (cm)
3816
3694
3695
3633
3523
20
22
25
26
28


Predicted head diameter (cm)
Estimated head diameter (cm)
May 14
May 15
May 19
May 20
May 26


Head diameter (cm)
1944
1706
1496
1293
1083
14
15
19
20
26

Chapter4 - Prediction of Optimal Harvest Date

Erika Nishida

2023-01-26

[4.2.3.2. ブロッコリーの価格データの取得] - Fig 4-5 & Fig 4-6 のレシピ

必要なパッケージをインストールする

```
library(tidyverse)
library(patchwork)
```

データをインポートする

```
broccoli_price <- read_csv("broccoli_price_csv/broccoli_price.csv")
head(broccoli_price)
```

```
## # A tibble: 6 × 8
##   Yr.Mon.Season Size  Price  Year Month Season    Ratio Mon.Season
##   <chr>         <chr> <dbl> <dbl> <dbl> <chr>     <dbl>      <dbl>
## 1 2019.1.1      ALL    96.1  2019     1 beginning 1.01         1.1
## 2 2019.1.2      ALL   122.   2019     1 middle    0.988        1.2
## 3 2019.1.3      ALL   110.   2019     1 end       0.948        1.3
## 4 2019.2.1      ALL   102.   2019     2 beginning 0.972        2.1
## 5 2019.2.2      ALL    82.6  2019     2 middle    1.00         2.2
## 6 2019.2.3      ALL    59.3  2019     2 end       0.912        2.3
```

```
broccoli_price_A <- read_csv("broccoli_price_csv/broccoli_price_A.csv")
head(broccoli_price_A)
```

```
## # A tibble: 6 × 8
##   Yr.Mon.Season Size  Price  Year Month Season    Ratio Mon.Season
##   <chr>         <chr> <dbl> <dbl> <dbl> <chr>     <dbl>      <dbl>
## 1 2019.1.1      ALL    96.1  2019     1 beginning 1.01         1.1
## 2 2019.1.2      ALL   122.   2019     1 middle    0.988        1.2
## 3 2019.1.3      ALL   110.   2019     1 end       0.948        1.3
## 4 2019.2.1      ALL   102.   2019     2 beginning 0.972        2.1
## 5 2019.2.2      ALL    82.6  2019     2 middle    1.00         2.2
## 6 2019.2.3      ALL    59.3  2019     2 end       0.912        2.3
```

```
broccoli_price_W <- read_csv("broccoli_price_csv/broccoli_price_White.csv")
head(broccoli_price_W)
```

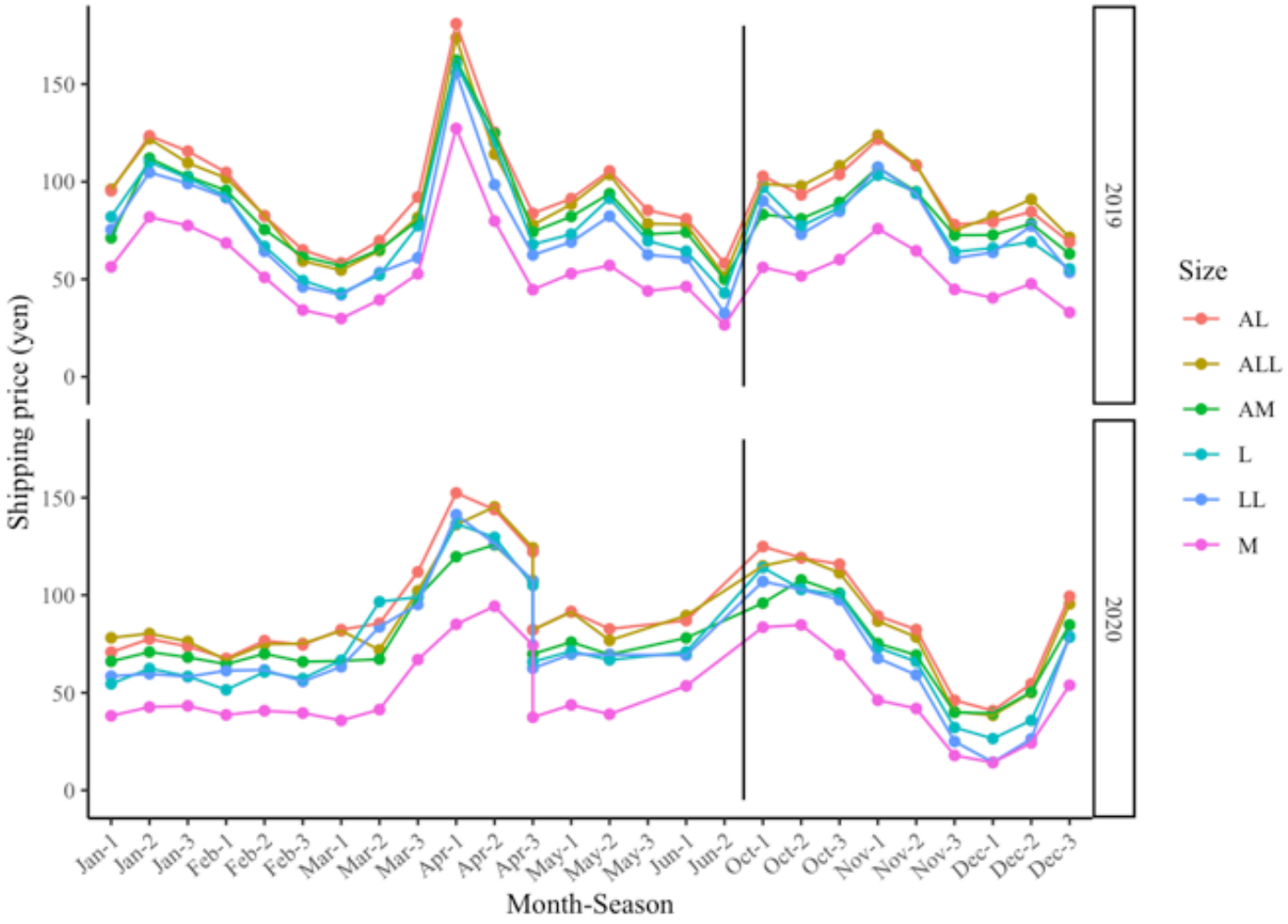
```
## # A tibble: 6 × 8
##   Yr.Mon.Season Size  Price  Year Month Season    Ratio Mon.Season
##   <chr>         <chr> <dbl> <dbl> <dbl> <chr>     <dbl>      <dbl>
## 1 2019.1.1      LL     75.5  2019     1 beginning 0.920        1.1
## 2 2019.1.1      L      82.1  2019     1 beginning 1            1.1
## 3 2019.1.1      M      56.4  2019     1 beginning 0.687        1.1
## 4 2019.1.2      LL    105.   2019     1 middle    0.951        1.2
## 5 2019.1.2      L     110.   2019     1 middle    1            1.2
## 6 2019.1.2      M      81.9  2019     1 middle    0.744        1.2
```

```
broccoli_price_AL <- read_csv("broccoli_price_csv/broccoli_price(AL as standard).c
sv")
head(broccoli_price_AL)
```

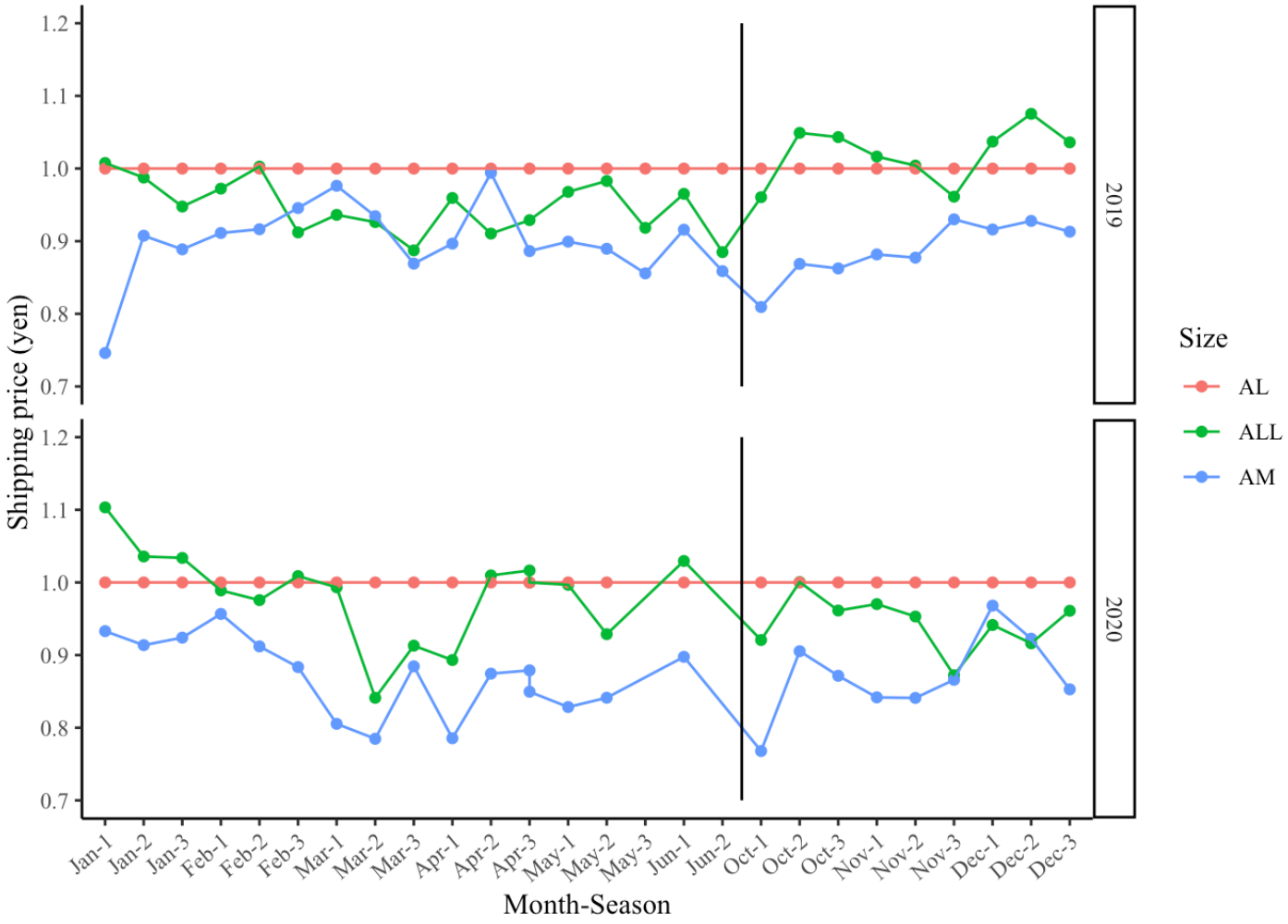
```
## # A tibble: 6 × 8
##   Yr.Mon.Season Size  Price  Year Month Season    Ratio Mon.Season
##   <chr>         <chr> <dbl> <dbl> <dbl> <chr>     <dbl>      <dbl>
## 1 2019.1.1      ALL    96.1  2019     1 beginning 1.01         1.1
## 2 2019.1.2      ALL   122.   2019     1 middle    0.988        1.2
## 3 2019.1.3      ALL   110.   2019     1 end       0.948        1.3
## 4 2019.2.1      ALL   102.   2019     2 beginning 0.972        2.1
## 5 2019.2.2      ALL    82.6  2019     2 middle    1.00         2.2
## 6 2019.2.3      ALL    59.3  2019     2 end       0.912        2.3
```

Fig.4-5&6を描く

```
# Fig.4-5
p1 <- ggplot(data = broccoli_price, mapping = aes(x = factor(Mon.Season), y=Price,
group = Size, colour = Size)) +
  geom_point() +
  geom_line() +
  labs(x = "Month-Season",
       y = "Shipping price (yen)") +
  annotate("segment", x = 17.5, xend = 17.5, y = -5, yend = 180,
           alpha = 1, color = "black")  +
  facet_grid(Year~.) +
  scale_x_discrete(labels = c("Jan-1", "Jan-2", "Jan-3", "Feb-1", "Feb-2", "Feb-3"
,
                              "Mar-1", "Mar-2", "Mar-3", "Apr-1", "Apr-2", "Apr-3"
,
                              "May-1", "May-2", "May-3", "Jun-1", "Jun-2",
                              "Oct-1", "Oct-2", "Oct-3", "Nov-1", "Nov-2", "Nov-3"
,
                              "Dec-1", "Dec-2", "Dec-3")) +
  theme_classic() +
  theme(text = element_text("Times New Roman"),
        axis.text.x = element_text(angle = 40, hjust = 1))

print(p1)
```




```
#Fig.4-6
p2 <- ggplot(data = broccoli_price_A, mapping = aes(x = factor(Mon.Season), y =Rat
io, group = Size, colour = Size)) +
  geom_point() +
  geom_line() +
  labs(x = "Month-Season",
       y = "Shipping price (yen)") +
  annotate("segment", x = 17.5, xend = 17.5, y = 0.7, yend = 1.2,
           alpha = 1, color = "black")  +
  facet_grid(Year~.) +
  scale_x_discrete(labels = c("Jan-1", "Jan-2", "Jan-3", "Feb-1", "Feb-2", "Feb-3"
,
                              "Mar-1", "Mar-2", "Mar-3", "Apr-1", "Apr-2", "Apr-3"
,
                              "May-1", "May-2", "May-3", "Jun-1", "Jun-2",
                              "Oct-1", "Oct-2", "Oct-3", "Nov-1", "Nov-2", "Nov-3"
,
                              "Dec-1", "Dec-2", "Dec-3")) +
  theme_classic() +
  theme(text = element_text("Times New Roman"),
        axis.text.x = element_text(angle = 40, hjust = 1))
print(p2)
```


Shipping price (yen)
1.2
1.1
1.0
0.9
0.8
0.7
2019
2020
Jan-1
Jan-2
Jan-3
Feb-1
Feb-2
Feb-3
Mar-1
Mar-2
Mar-3
Apr-1
Apr-2
Apr-3
May-1
May-2
May-3
Jun-1
Jun-2
Oct-1
Oct-2
Oct-3
Nov-1
Nov-2
Nov-3
Dec-1
Dec-2
Dec-3
Month-Season
Size
AL
ALL
AM

Chapter4 - Prediction of Optimal Harvest Date

Erika Nishida

2022/05/20

[4.2.3.3.最適収穫日の推定 & 4.2.4.予測] - Fig.4-7/8/9/11のレシピ

in 2021ver.

Install packages

```
library(tidyverse)
```

Import data frame

Data pre-processing

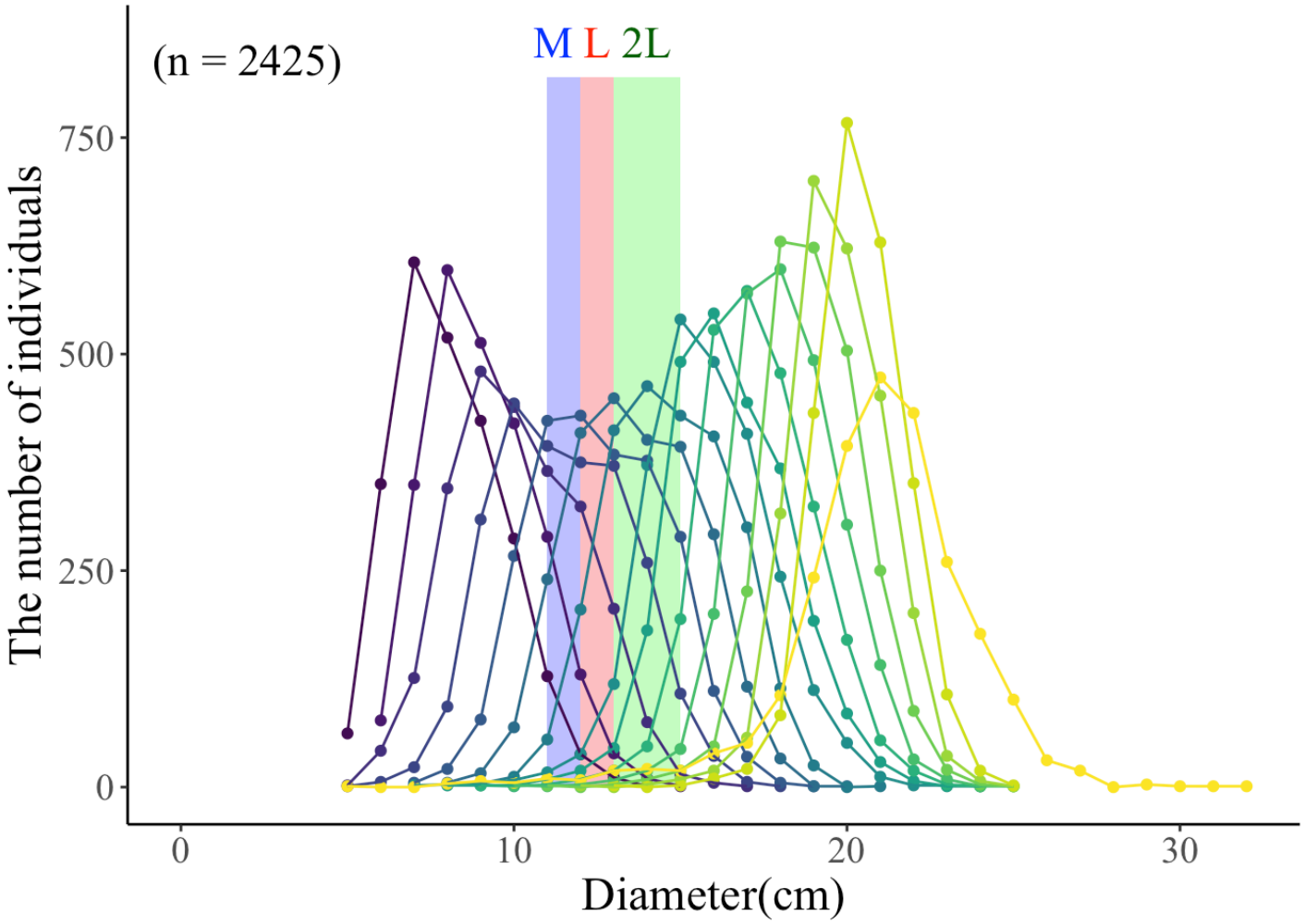
```
#Import data frame
dat1_tidy_2021 <- read_csv("2021_broccoli.csv")

#Data preprocessing
#使うデータ（date, ID, major_axis_length）をピックアップする
dat2_tidy_2021 <- dat1_tidy_2021 %>%
  rename(Diametermm = "major_axis_length(mm)", ID = label, Date = date) %>%
  select(1,3,9)

#欠損データを含む個体を除く
dat2_wide_2021 <- pivot_wider(dat2_tidy_2021, names_from = "Date", values_from = "
Diametermm")
dat3_wide_2021 <- dat2_wide_2021 %>%
  filter(complete.cases(dat2_wide_2021==F))
```

“Fig.4-7：花蕾直径のサイズ分布の推移”のレシピ

```
#ドローン空撮を行っていない(＝UAVデータがない)日付の各個体の花蕾直径を、内挿に基づき(同年の生育予
測モデルと気象データから）推定する

#Input parameters of growth model constructed by 2021 data
a <- 5.606e+00
b <- 2.160e+00
c <- 5.815e-03

#直近のUAVデータの積算温度を算出する
dat_true_2021 <- dat3_wide_2021 %>%
  rename("12" = `20210512`,
         "14" = `20210514`,
```

```
          "15" = `20210515`,
          "19" = `20210519`,
          "20" = `20210520`,
          "26" = `20210526`) %>%
  mutate("tempsum_by5/11" = log({a - log(`12`)}/b)/-c, .after = `12`) %>%
  mutate("tempsum_by5/14" = log({a - log(`15`)}/b)/-c, .after = `15`) %>%
  mutate("tempsum_by5/19" = log({a - log(`20`)}/b)/-c, .after = `20`)

#直近のUAVデータから算出した積算温度 + 気象データから、UAVデータがない日の積算温度を算出する
#後に生育モデルを用いて、その積算温度から花蕾直径を逆算する

dat_true2_2021 <- dat_true_2021 %>%
  mutate("tempsum_by5/12" = `tempsum_by5/11` + 17.4, .after = `12`) %>%
  select(-`tempsum_by5/11`) %>%
  mutate("tempsum_by5/15" = `tempsum_by5/14` + 20, .after = `15`) %>%
  mutate("tempsum_by5/16" = `tempsum_by5/14` + 20 + 19.5, .after = `tempsum_by5/15
`) %>%
  mutate("tempsum_by5/17" = `tempsum_by5/14` + 20 + 19.5 + 20, .after = `tempsum_b
y5/16`) %>%
  select(-`tempsum_by5/14`) %>%
  mutate("tempsum_by5/20" = `tempsum_by5/19` + 18.2, .after = `20`) %>%
  mutate("tempsum_by5/21" = `tempsum_by5/19` + 18.2 + 20, .after = `tempsum_by5/20
`) %>%
  mutate("tempsum_by5/22" = `tempsum_by5/19` + 18.2 + 20 + 19.7, .after = `tempsum
_by5/21`) %>%
  mutate("tempsum_by5/23" = `tempsum_by5/19` + 18.2 + 20 + 19.7 + 19.3, .after = `
tempsum_by5/22`) %>%
  mutate("tempsum_by5/24" = `tempsum_by5/19` + 18.2 + 20 + 19.7 + 19.3 + 20, .afte
r = `tempsum_by5/23`) %>%
  select(-`tempsum_by5/19`)

#生育モデルで、積算温度から花蕾直径を逆算する
dat_true3_2021 <- dat_true2_2021 %>%
  mutate("13" = exp(a - b*exp(-c*`tempsum_by5/12`)), .after = `12`) %>%
  select(-`tempsum_by5/12`) %>%
  mutate("16" = exp(a - b*exp(-c*`tempsum_by5/15`)), .after = `15`) %>%
  select(-`tempsum_by5/15`) %>%
  mutate("17" = exp(a - b*exp(-c*`tempsum_by5/16`)), .after = `16`) %>%
  select(-`tempsum_by5/16`) %>%
  mutate("18" = exp(a - b*exp(-c*`tempsum_by5/17`)), .after = `17`) %>%
  select(-`tempsum_by5/17`) %>%
  mutate("21" = exp(a - b*exp(-c*`tempsum_by5/20`)), .after = `20`) %>%
  select(-`tempsum_by5/20`) %>%
  mutate("22" = exp(a - b*exp(-c*`tempsum_by5/21`)), .after = `21`) %>%
  select(-`tempsum_by5/21`) %>%
  mutate("23" = exp(a - b*exp(-c*`tempsum_by5/22`)), .after = `22`) %>%
  select(-`tempsum_by5/22`) %>%
  mutate("24" = exp(a - b*exp(-c*`tempsum_by5/23`)), .after = `23`) %>%
  select(-`tempsum_by5/23`) %>%
  mutate("25" = exp(a - b*exp(-c*`tempsum_by5/24`)), .after = `24`) %>%
  select(-`tempsum_by5/24`)


#Change unit from mm to cm
```

```
i = 12
for (i in seq_along(dat_true3_2021)){
  dat_true3_2021[[i]] <- dat_true3_2021[[i]]/10
}
dat_true3_2021 <- dat_true3_2021 %>%
  mutate(`ID` = `ID`*10)

#Fig.4-7用にデータを成形する
#縦長データへ
dat_true3_tidy2021 <- dat_true3_2021 %>%
  gather(-ID, key = "Date", value = "HD")

#四捨五入する
add_datint_tidy2021 <-
  dat_true3_tidy2021 %>%
  mutate(Diameter = round(HD, digits = 0))

#花蕾直径を1cm単位でクラス分けし、クラスごとに個数を集計する
for_histogram_add2021 <- add_datint_tidy2021 %>%
  group_by(Date,Diameter) %>%
  summarize(number=n()) %>%
  complete(
    Diameter = full_seq(Diameter, period = 1),
    fill = list(number = 0)
  )

#Fig.4-7を描く
p1 <- ggplot(data = for_histogram_add2021, mapping = aes(x = Diameter, y = number,
group = factor(Date), colour = factor(Date))) +
  annotate("rect", xmin = 11, xmax = 12, ymin = 0, ymax = 820, alpha = 0.3, fill =
"blue") +
  annotate("rect", xmin = 12, xmax = 13, ymin = 0, ymax = 820, alpha = 0.3, fill =
"red") +
  annotate("rect", xmin = 13, xmax = 15, ymin = 0, ymax = 820, alpha = 0.3, fill =
"green") +
  annotate("text", x = 11.2, y = 860, label = "M",
           colour = "blue", size = 6, family = "Times New Roman") +
  annotate("text", x = 12.5, y = 860, label = "L",
           colour = "red", size = 6, family = "Times New Roman") +
  annotate("text", x = 14, y = 860, label = "2L",
           colour = "darkgreen", size = 6, family = "Times New Roman") +
  annotate("text", x = 2, y = 840, label = "(n = 2425)",
           size = 6, family = "Times New Roman") +
  geom_point() +
  geom_line() +
  xlim(0,32) + ylim(0,860) +
  labs(x = "Diameter(cm)", y = "The number of individuals") +
  scale_color_viridis_d(name = "Date") +
  theme_classic() +
  theme(text = element_text("Times New Roman", size = 18),
        legend.position = "none")

print(p1)
```



“Fig.4-8：一斉収穫を想定した時の収入と規格外廃棄率の推移”のレシピ

1.日毎の収入を算出する

```
#まずはグループ化する
add_data_tidy_grouped2021 <- dat_true3_tidy2021 %>%
  group_by(Date)

#LL, L, Mサイズを満たす個数をそれぞれ集計する
sales_Msize2021 <- add_data_tidy_grouped2021 %>%
  filter(between(HD, 11, 12)) %>%
  summarize(number=n()) %>%
  mutate(Size = "M")
sales_Lsize2021 <- add_data_tidy_grouped2021 %>%
  filter(between(HD, 12, 13)) %>%
  summarize(number=n()) %>%
  mutate(Size = "L")
sales_2Lsize2021 <- add_data_tidy_grouped2021 %>%
  filter(between(HD, 13, 15)) %>%
  summarize(number=n()) %>%
  mutate(Size = "2L")

#上3つのデータをマージする
sales_size2021 <- merge(sales_Msize2021, sales_Lsize2021, all = T)
sales_size2021 <- merge(sales_size2021, sales_2Lsize2021, all = T)

#"implicit missing value"を補う→ただし今回はN/Aがないので不要
```

```
sales_size2021 <- sales_size2021 %>%
  complete(Date,Size,
           fill = list(number = 0))

#Lサイズを1としたサイズごとの相対価格をインプットする
#Case1：階級間の開きが小さい時
M_c1 = 0.956662856
L_c1 = 1
"2L_c1" = 0.989022542148134

#Case2：階級間の開きが大きい時
M_c2 = 0.784841275
L_c2 = 1
"2L_c2" = 0.841168164

#サイズ別個数と相対価格を掛け合わせて、日付別に収入を算出する
sales_size2021$number <- as.numeric(sales_size2021$number)
sales_dat2021 <- sales_size2021 %>%
  mutate("Relative Price case1" = case_when(
    Size == "2L" ~  number*`2L_c1`,
    Size == "L" ~ number,
    Size == "M" ~ number*M_c1
  )) %>%
  mutate("Relative Price case2" = case_when(
    Size == "2L" ~  number*`2L_c2`,
    Size == "L" ~ number,
    Size == "M" ~ number*M_c2
  ))

sales_dat_grouped2021 <- sales_dat2021 %>%
  group_by(Date) %>%
  summarise("Case1" = sum(`Relative Price case1`),"Case2" = sum(`Relative Price ca
se2`))

#図示用に、縦長データに変換する
sales_dat_grouped_tidy2021 <- pivot_longer(sales_dat_grouped2021, cols = c(Case1,
Case2), names_to = "Cases", values_to = "Totalprice") %>%
  relocate(Cases)
```

2. 日毎の規格外廃棄率を算出する

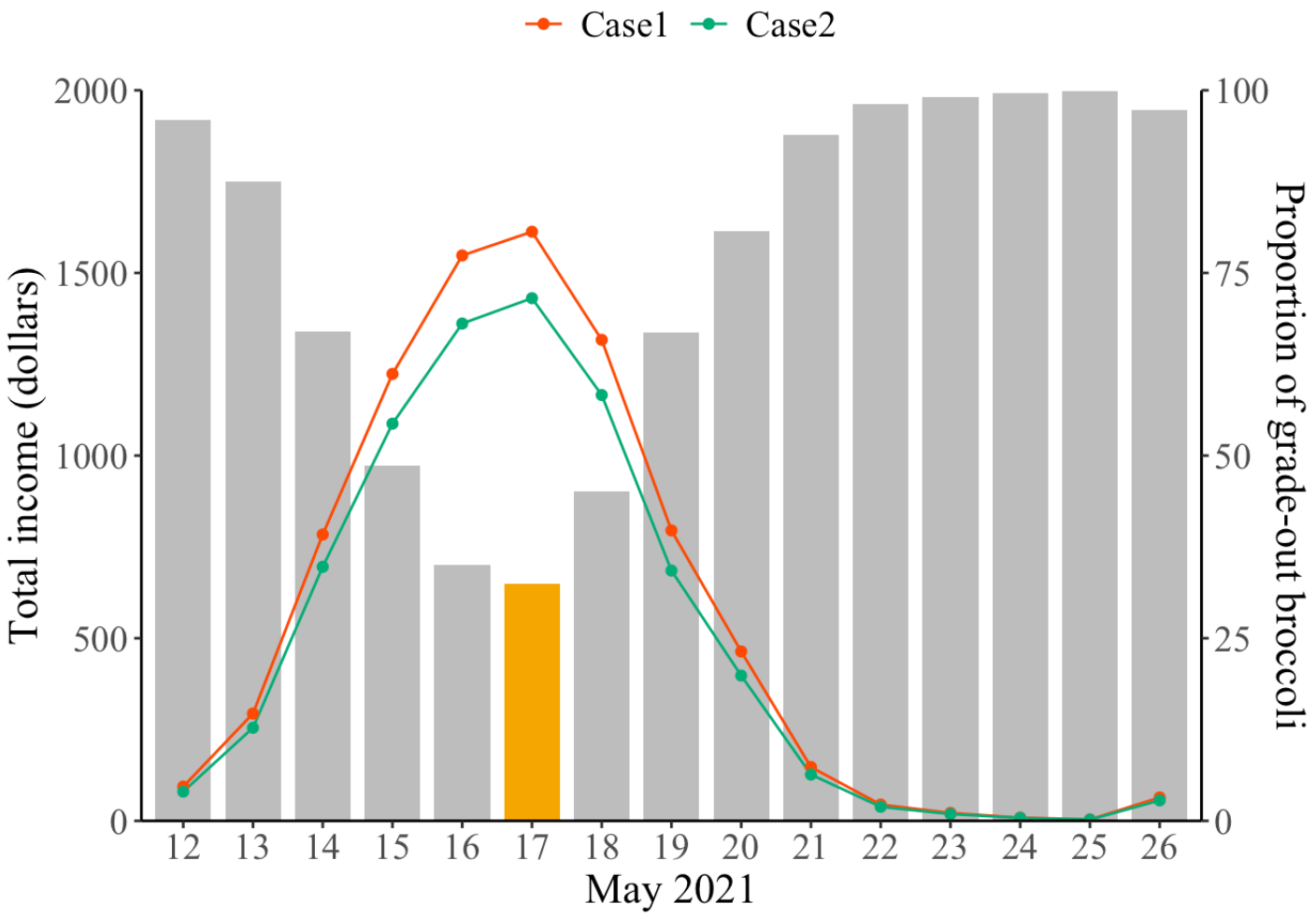
```
#全ブロッコリー数(欠損データなしに限る)＝2425として、規格外廃棄りつを算出
nonstandard_dat2021 <- sales_dat2021 %>%
  group_by(Date) %>%
  summarise("Num_Standard" = sum(`number`)) %>%
  mutate("Num_Nonstandard" = 2425 - `Num_Standard`, "Percentage" = `Num_Nonstandar
d`/2425)
```

3. 図示する

```
#ggplotでは、2軸グラフを書く時、y軸のスケールはあくまで共通になってしまう。
#そこで2つ目の変数を1つ目の変数に合わせて縮めたり伸ばしたりして収めて、プロットし、
#sec.axisというオプションで、左軸を好きなように変換した軸を右側に設定することで、
```

```
#異なるスケールのy軸を右側におくことを実現させる。
#ゆえに、y2の値をy1軸にスケールした値をデータフレーム上に作る必要がある

#1. まずは各軸の範囲を決める
yaxis1_2021 <- c(0, 2000)
yaxis2_2021 <- c(0, 100)

#2. y2の値をy1軸にスケールした値をデータフレーム上に作る
#-------------------------------------------------------
# variabel_scaler(y2, yaxis1, yaxis2)：y2のプロットの位置をy1のスケールに合わせる
#   y2              :  y2軸に示すデータ
#   yaxis1, yaxis2 :  y1軸とy2軸の対応
#   戻り値          :  y2の値をy1軸相当の位置にした値
#-------------------------------------------------------
variable_scaler_2021 <- function(y2_2021, yaxis1_2021, yaxis2_2021){
  a_2021 <- (yaxis1_2021[2] - yaxis1_2021[1]) / (yaxis2_2021[2] - yaxis2_2021[1])
  b_2021 <- (yaxis1_2021[1] * yaxis2_2021[2] - yaxis1_2021[2] * yaxis2_2021[1]) /
(yaxis2_2021[2] - yaxis2_2021[1])
  ret_2021 <- a_2021 * y2_2021 + b_2021
  ret_2021
}

#y2B：左軸のスケールに合わせる前のy2の値(before)
dt2021 <- full_join(sales_dat_grouped2021, nonstandard_dat2021, by = "Date") %>%
  mutate("y2B" = `Percentage`*100) %>%
  mutate("y2A" = variable_scaler_2021(y2B, yaxis1_2021,yaxis2_2021))


#3. 最適収穫日をオレンジ色で強調するための処理
dt2_2021 <- dt2021 %>%
  select(Date, Case1, Case2, y2A) %>%
  gather(-Date, -y2A, key = "Cases", value = "Income") %>%
  mutate("y2A_2" = case_when(
    Cases == "Case1" ~ `y2A`,
    Cases == "Case2" ~ 0
  )) %>%
  select(-y2A) %>%
  mutate(hl = recode(Date, "12" = "no", "13" = "no", "14" = "no", "15" = "no",
                     "16" = "no", "17" = "yes", "18" = "no", "19" = "no", "20" = "
no",
                     "21" = "no", "22" = "no", "23" = "no", "24" = "no", "25" = "n
o", "26" = "no",))

#4. 図示する
#-------------------------------------------------------
# axis_scaler(y1, yaxis1, yaxis2)：右軸のメモリ値を左軸のスケールに調整する
#   y1              :  y1軸の値（sec_axis()の"."）
#   yaxis1, yaxis2 :  y1軸とy2軸の対応
#   戻り値          :  y2の値をy1軸相当の位置にした値
#-------------------------------------------------------
axis_scaler_2021 <- function(y1_2021, yaxis1_2021, yaxis2_2021){
  c_2021 <- (yaxis2_2021[2] - yaxis2_2021[1]) / (yaxis1_2021[2] - yaxis1_2021[1])
  d_2021 <- (yaxis2_2021[1] * yaxis1_2021[2] - yaxis2_2021[2] * yaxis1_2021[1]) /
(yaxis1_2021[2] - yaxis1_2021[1])
```

```
  ret_2021 <- c_2021 * y1_2021 + d_2021
  ret_2021
}

p2 <- ggplot(dt2_2021, mapping = aes(x = Date, group = 1)) +
  geom_bar(mapping = aes(x = Date, y = y2A_2, fill = hl), stat = "identity", width
= 0.8) +
  geom_line(mapping = aes(y = Income, group = Cases, colour = Cases)) +
  geom_point(mapping = aes(x = Date, y = Income, group = Cases, colour = Cases)) +
  scale_y_continuous(limit=yaxis1_2021, expand = c(0, 0),
                     sec.axis = sec_axis( ~(axis_scaler_2021(., yaxis1_2021, yaxis
2_2021)),
                     #左の軸の何分の1の値にするか, .は各y1を指している
                                          name = "Proportion of grade-out broccoli"
)) +
  labs(x = "May 2021", y = "Total income (dollars)", colour = "") +
  scale_color_manual(values = c("#ff4b00", "#03af7a")) +
  scale_fill_manual(values = c("grey", "#f6aa00"), guide = "none") +
  theme_classic() +
  theme(legend.position = "top",
        legend.text = element_text("Times New Roman", size = 14),
        axis.text.x = element_text("Times New Roman", size =14),
        axis.text.y = element_text("Times New Roman", size = 14),
        axis.title.y = element_text("Times New Roman", size = 16),
        axis.title.x = element_text("Times New Roman", size = 16))
print(p2)
```



“Fig.4-9 一斉収穫を想定した時の収入推移の予測”のレシピ

2020年の生育予測モデル、気象データ、ドローン空撮初日(5/12)のUAVデータを用いて、外挿に基づく最適収穫日の予測を行う。

1. 予測値を用意する

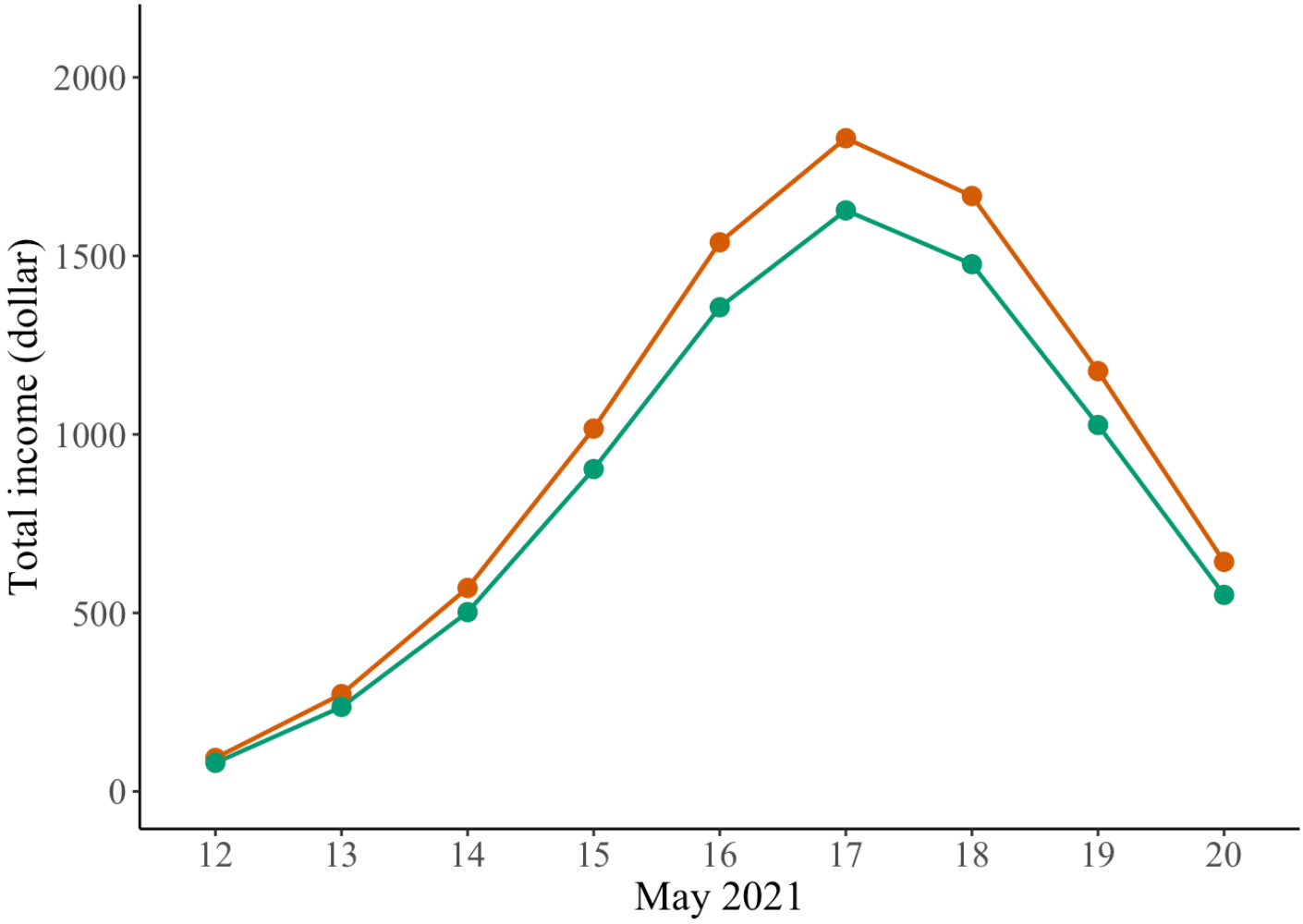
```
#2020年生育モデルのパラメータをインプットする
a <- 5.7546056
b <- 2.3298645
c <- 0.0047488

#1. 5/11までの積算温度をGETする。
dat4v_wide <- dat3_wide_2021 %>%
  mutate("12" = log((a- log(`20210512`))/b)/-c, .after = `20210512`)

#2.5/11-5/25までの積算温度を並べたデータを作る
temp_sumv <- dat4v_wide %>%
  select(1,3) %>%
  mutate("13" = `12`+ 17.4) %>% #UAV撮影初日なのでこの日のみ実測値
  mutate("14" = `13`+ 15.7) %>%
  mutate("15" = `14`+ 20) %>%
  mutate("16" = `15`+ 20) %>%
  mutate("17" = `16`+ 19.5) %>%
  mutate("18" = `17`+ 20) %>%
  mutate("19" = `18`+ 20) %>%
  mutate("20" = `19`+ 17.1)

#縦長データにする
temp_sumv_tidy <- temp_sumv %>%
  gather(-ID, key = `Date`, value = Tempsum)

#3. Pred_HDを算出
pred_data_2021 <- temp_sumv_tidy %>%
  mutate("pred_HD" = exp(a - b*exp(-c*`Tempsum`))) %>%
  select(1,2,4) %>%
  mutate("pred_HD_cm" = `pred_HD`/10) %>%
  select(-`pred_HD`)
```

2. 日毎の収入を算出する

```
pred_tidy_grouped <- pred_data_2021 %>%
  group_by(Date)

#サイズ別に個数を集計する
pred_sales_Msize <- pred_tidy_grouped %>%
  filter(between(pred_HD_cm, 11, 12)) %>%
  summarize(number=n()) %>%
  mutate(Size = "M")
pred_sales_Lsize <- pred_tidy_grouped %>%
  filter(between(pred_HD_cm, 12, 13)) %>%
  summarize(number=n()) %>%
  mutate(Size = "L")
pred_sales_2Lsize <- pred_tidy_grouped %>%
  filter(between(pred_HD_cm, 13, 15)) %>%
  summarize(number=n()) %>%
  mutate(Size = "2L")

#3つのデータをマージする
pred_sales_size <- merge(pred_sales_Msize, pred_sales_Lsize, all = T)
pred_sales_size <- merge(pred_sales_size, pred_sales_2Lsize, all = T)

#収入に換算する
pred_sales_dat <- pred_sales_size %>%
  mutate("Relative Price case1" = case_when(
    Size == "2L" ~  as.numeric(number)*`2L_c1`,
    Size == "L" ~ as.numeric(number),
    Size == "M" ~ as.numeric(number)*M_c1
  )) %>%
  mutate("Relative Price case2" = case_when(
    Size == "2L" ~  as.numeric(number)*`2L_c2`,
    Size == "L" ~ as.numeric(number),
    Size == "M" ~ as.numeric(number)*M_c2
  ))

pred_sales_dat_grouped <- pred_sales_dat %>%
  group_by(Date) %>%
  summarise("Case1" = sum(`Relative Price case1`),"Case2" = sum(`Relative Price ca
se2`))

#図示用に縦長データに変換する
pred_sales_dat_grouped_tidy <- pivot_longer(pred_sales_dat_grouped,
                                            cols = c(Case1, Case2),
                                            names_to = "Cases",
                                            values_to = "Totalprice") %>%
  relocate(Cases)
```

3. 図示する

```
p5 <- ggplot(pred_sales_dat_grouped_tidy, aes(x = Date, y = Totalprice)) +
  geom_line(aes(group = Cases, colour = Cases), linewidth = 0.8) +
  geom_point(aes(colour = Cases), size = 3) +
  labs(
  y = "Total income (dollar)",
  x = "May 2021") +
  ylim(0, 2100) +
  theme_classic() +
  scale_color_manual(values = c("Case1" = "#D55E00", "Case2" = "#009E73")) +
  theme(axis.text.x = element_text("Times New Roman", size = 14),
        axis.text.y = element_text("Times New Roman", size = 14),
        axis.title.x = element_text("Times New Roman", size = 16),
        axis.title.y = element_text("Times New Roman", size = 16),
        legend.text = element_text("Times New Roman", size = 14),
        legend.position = "none")
print(p5)
```



Chapter4 - Prediction of Optimal Harvest Date

Erika Nishida

2022/05/20

[4.2.3.3.最適収穫日の推定 & 4.2.4.予測] - Fig.4-7/8/9のレシピ in 2020ver.

Install packages

```
library(tidyverse)
```

Import data frame

Data pre-processing

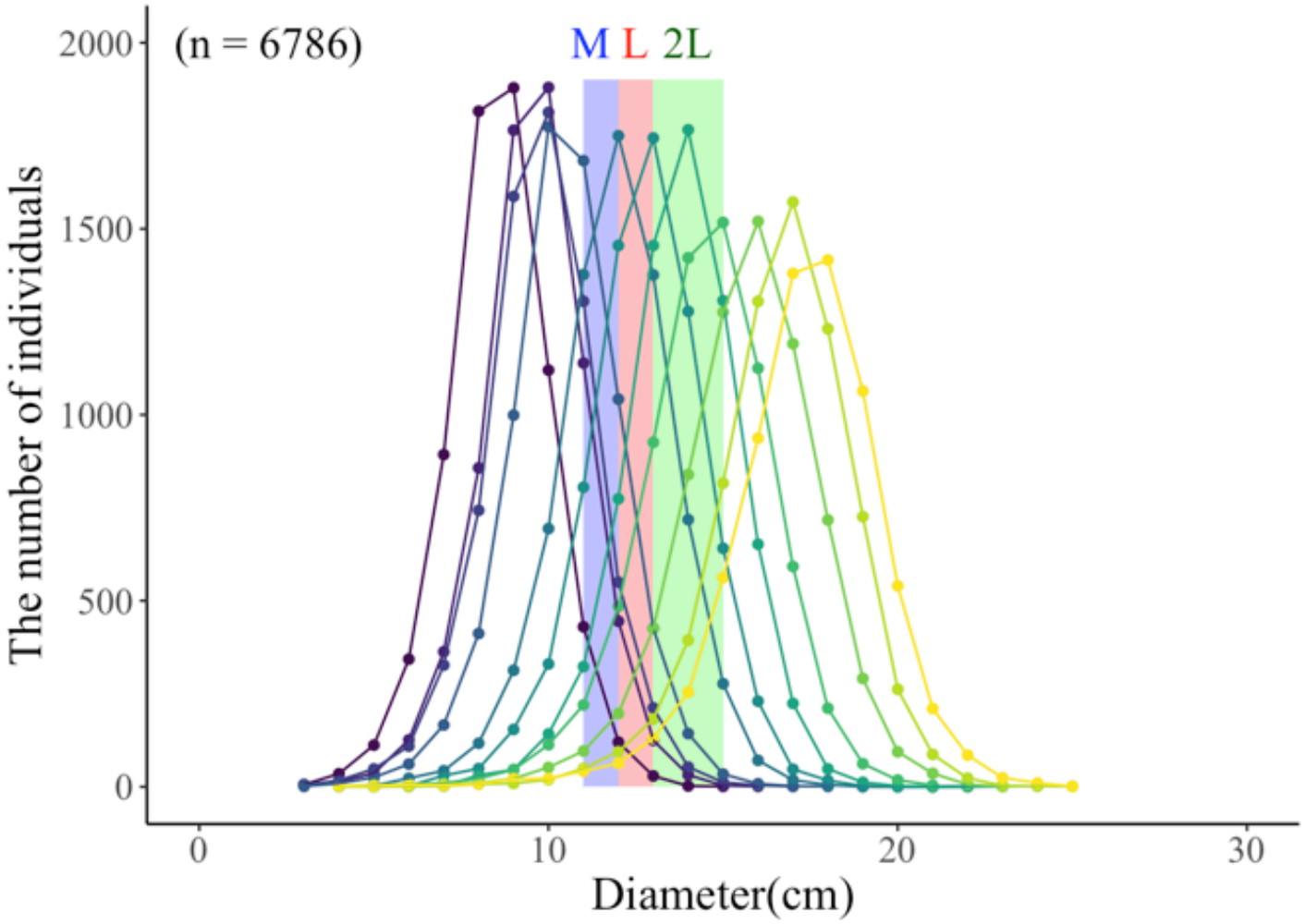
```
#Import data frame
dat1_tidy <- read_csv("2020_broccoli.csv")

#Data preprocessing
#使うデータ（date, ID, major_axis_length）をピックアップする
dat2_tidy <- dat1_tidy %>%
  rename(Diametermm = "major_axis_length", ID = label, Date = date) %>%
  select(1,2,7)

#欠損データを含む個体を除く
dat2_wide <- pivot_wider(dat2_tidy, names_from = "Date", values_from = "Diametermm
")
dat3_wide <-dat2_wide %>%
  filter(complete.cases(dat2_wide==F))
```

“Fig.4-7：花蕾直径のサイズ分布の推移”のレシピ

```
#ドローン空撮を行っていない(＝UAVデータがない)日付の各個体の花蕾直径を、内挿に基づき(同年の生育予測モデルと気象データから）推定する

#Input parameters of growth model constructed by 2020 data
a <- 5.7546056
b <- 2.3298645
c <- 0.0047488

#直近のUAVデータの積算温度を算出する
dat_true <- dat3_wide %>%
  rename("18" = `20200518`,
         "20" = `20200520`,
         "22" = `20200522`,
         "25" = `20200525`,
```

```
         "26" = `20200526`,
         "28" = `20200528`) %>%
  mutate("tempsum_by5/17" = log({a - log(`18`)}/b)/-c, .after = `18`) %>%
  mutate("tempsum_by5/19" = log({a - log(`20`)}/b)/-c, .after = `20`) %>%
  mutate("tempsum_by5/21" = log({a - log(`22`)}/b)/-c, .after = `22`) %>%
  mutate("tempsum_by5/25" = log({a - log(`26`)}/b)/-c, .after = `26`)

#直近のUAVデータから算出した積算温度 + 気象データから、UAVデータがない日の積算温度を算出する
#後に生育モデルを用いて、その積算温度から花蕾直径を逆算する
dat_true2 <- dat_true %>%
  mutate("tempsum_by5/18" = `tempsum_by5/17` + 19, .after = `18`) %>%
  select(-`tempsum_by5/17`) %>%
  mutate("tempsum_by5/20" = `tempsum_by5/19` + 12.4, .after = `20`) %>%
  select(-`tempsum_by5/19`) %>%
  mutate("tempsum_by5/22" = `tempsum_by5/21` + 15.4, .after = `22`) %>%
  mutate("tempsum_by5/23" = `tempsum_by5/21` + 15.4 + 18.9, .after = `tempsum_by5/
22`) %>%
  select(-`tempsum_by5/21`) %>%
  mutate("tempsum_by5/26" = `tempsum_by5/25` + 20, .after =`26`) %>%
  select(-`tempsum_by5/25`)

#生育モデルで、積算温度から花蕾直径を逆算する
dat_true3 <- dat_true2 %>%
  mutate("19" = exp(a - b*exp(-c*`tempsum_by5/18`)), .after = `18`) %>%
  select(-`tempsum_by5/18`) %>%
  mutate("21" = exp(a - b*exp(-c*`tempsum_by5/20`)), .after = `20`) %>%
  select(-`tempsum_by5/20`) %>%
  mutate("23" = exp(a - b*exp(-c*`tempsum_by5/22`)), .after = `22`) %>%
  select(-`tempsum_by5/22`) %>%
  mutate("24" = exp(a - b*exp(-c*`tempsum_by5/23`)), .after = `23`) %>%
  select(-`tempsum_by5/23`) %>%
  mutate("27" = exp(a - b*exp(-c*`tempsum_by5/26`)), .after = `26`) %>%
  select(-`tempsum_by5/26`)

#Change unit from mm to cm
i = 18
for (i in seq_along(dat_true3)){
  dat_true3[[i]] <- dat_true3[[i]]/10
}
dat_true3 <- dat_true3 %>%
  mutate(`ID` = `ID`*10)

#Fig.4-7用にデータを成形する
#縦長データへ
dat_true3_tidy <- dat_true3 %>%
  gather(-ID, key = "Date", value = "HD")

#四捨五入する
add_datint_tidy <-
  dat_true3_tidy %>%
  mutate(Diameter = round(HD, digits = 0))

#Regrouping by date&diameter
add_datint_tidy_grouped <- add_datint_tidy %>%
```

```
  group_by(Date,Diameter)

#花蕾直径を1cm単位でクラス分けし、クラスごとに個数を集計する
for_histogram_add <- add_datint_tidy_grouped %>%
  summarize(number=n()) %>%
  complete(
    Diameter = full_seq(Diameter, period = 1),
    fill = list(number = 0)
  )

#Fig.4-7を描く
p1 <- ggplot(data = for_histogram_add, mapping = aes(x = Diameter, y = number, gro
up = factor(Date), colour = factor(Date))) +
  annotate("rect", xmin = 11, xmax = 12, ymin = 0, ymax = 1900, alpha = 0.3, fill
= "blue") +
  annotate("rect", xmin = 12, xmax = 13, ymin = 0, ymax = 1900, alpha = 0.3, fill
= "red") +
  annotate("rect", xmin = 13, xmax = 15, ymin = 0, ymax = 1900, alpha = 0.3, fill
= "green") +
  annotate("text", x = 11.2, y = 2000, label = "M",
           colour = "blue", size = 6, family = "Times New Roman") +
  annotate("text", x = 12.5, y = 2000, label = "L",
           colour = "red", size = 6, family = "Times New Roman") +
  annotate("text", x = 14, y = 2000, label = "2L",
           colour = "darkgreen", size = 6, family = "Times New Roman") +
  annotate("text", x = 2, y = 2000, label = "(n = 6786)",
           size = 6, family = "Times New Roman") +
  geom_point() +
  geom_line() +
  xlim(0,30) + ylim(0,2000) +
  labs(x = "Diameter(cm)", y = "The number of individuals") +
  scale_color_viridis_d(name = "Date") +
  theme_classic() +
  theme(text = element_text("Times New Roman", size = 18),
        legend.position = "none")
print(p1)
```



“Fig.4-8：一斉収穫を想定した時の収入と規格外廃棄率の推移”のレシピ

1.日毎の収入を算出する

```
#まずはグループ化する
add_data_tidy_grouped <- dat_true3_tidy %>%
  group_by(Date)

#LL, L, Mサイズを満たす個数をそれぞれ集計する
sales_Msize <- add_data_tidy_grouped %>%
  filter(between(HD, 11, 12)) %>%
  summarize(number=n()) %>%
  mutate(Size = "M")
sales_Lsize <- add_data_tidy_grouped %>%
  filter(between(HD, 12, 13)) %>%
  summarize(number=n()) %>%
  mutate(Size = "L")
sales_2Lsize <- add_data_tidy_grouped %>%
  filter(between(HD, 13, 15)) %>%
  summarize(number=n()) %>%
  mutate(Size = "2L")

#上3つのデータをマージする
sales_size <- merge(sales_Msize, sales_Lsize, all = T)
sales_size <- merge(sales_size, sales_2Lsize, all = T)

#"implicit missing value"を補う→今回はN/Aがないので不要
```

```
sales_size <- sales_size %>%
  complete(Date,Size,
           fill = list(number = 0))

#Lサイズを1としたサイズごとの相対価格をインプットする
#Case1：階級間の開きが小さい時
M_c1 = 0.956662856
L_c1 = 1
"2L_c1" = 0.989022542148134

#Case2：階級間の開きが大きい時
M_c2 = 0.784841275
L_c2 = 1
"2L_c2" = 0.841168164

#サイズ別個数と相対価格を掛け合わせて、日付別に収入を算出する
sales_size$number <- as.numeric(sales_size$number)
sales_dat <- sales_size %>%
  mutate("Relative Price case1" = case_when(
    Size == "2L" ~  number*`2L_c1`,
    Size == "L" ~ number,
    Size == "M" ~ number*M_c1
  )) %>%
  mutate("Relative Price case2" = case_when(
    Size == "2L" ~  number*`2L_c2`,
    Size == "L" ~ number,
    Size == "M" ~ number*M_c2
  ))

sales_dat_grouped <- sales_dat %>%
  group_by(Date) %>%
  summarise("Case1" = sum(`Relative Price case1`),"Case2" = sum(`Relative Price ca
se2`))

#図示用に、縦長データに変換する
sales_dat_grouped_tidy <- pivot_longer(sales_dat_grouped, cols = c(Case1, Case2),
names_to = "Cases", values_to = "Totalprice") %>%
  relocate(Cases)
```

2. 日毎の規格外廃棄率を算出する

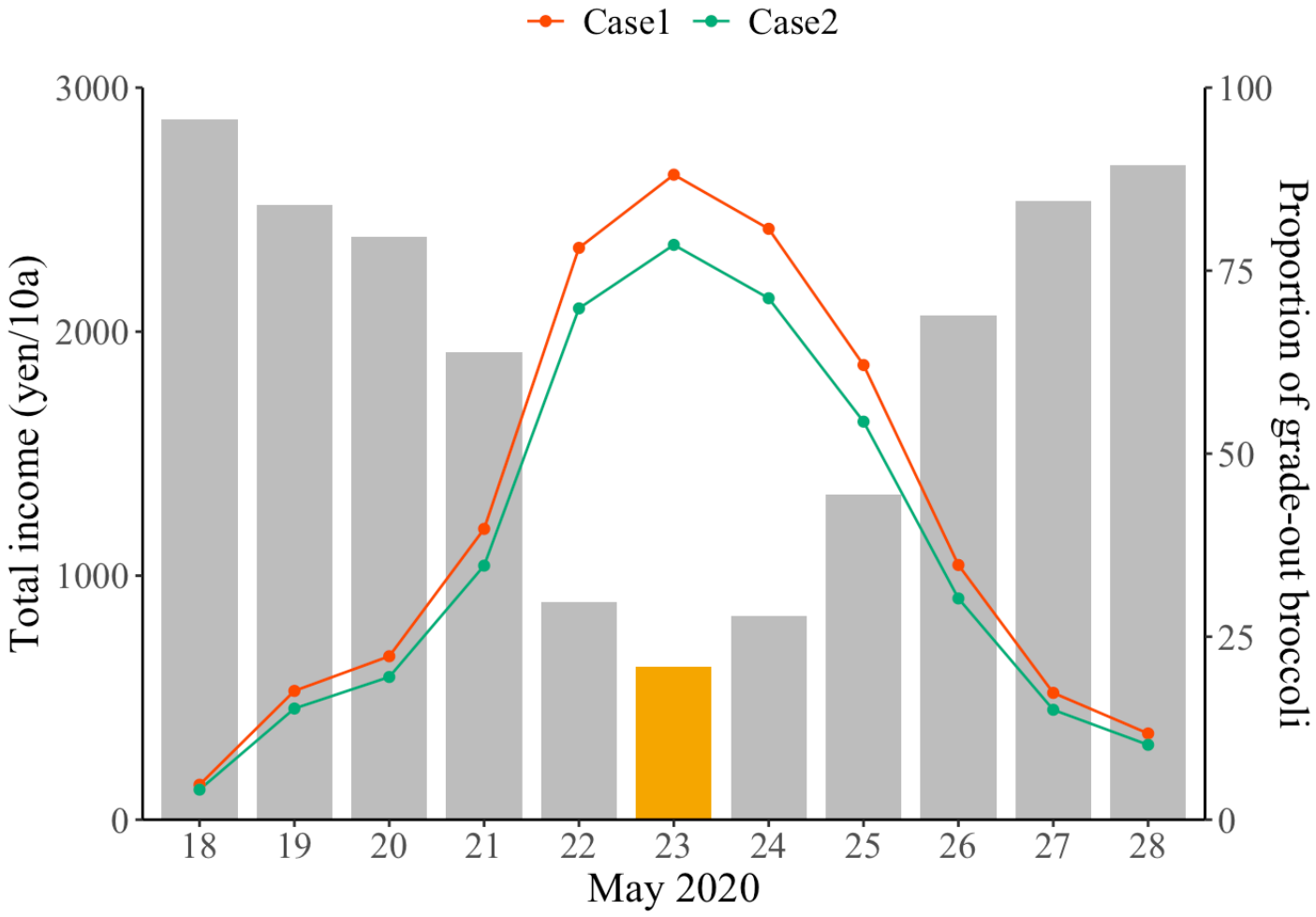
```
#全ブロッコリー数(欠損データなしに限る)＝6786として、規格外廃棄率を算出する
nonstandard_dat <- sales_dat %>%
  group_by(Date) %>%
  summarise("Num_Standard" = sum(`number`)) %>%
  mutate("Num_Nonstandard" = 6786 - `Num_Standard`, "Percentage" = `Num_Nonstandar
d`/6786)
```

3. 図示する

```
#ggplotでは、2軸グラフを書く時、y軸のスケールはあくまで共通になってしまう。
#そこで2つ目の変数を1つ目の変数に合わせて縮めたり伸ばしたりして収めて、プロットし、
#sec.axisというオプションで、左軸を好きなように変換した軸を右側に設定することで、
```

```
#異なるスケールのy軸を右側におくことを実現させる。
#ゆえに、y2の値をy1軸にスケールした値をデータフレーム上に作る必要がある

#1. まずは各軸の範囲を決める
yaxis1 <- c(0, 3000)
yaxis2 <- c(0, 100)

#2. y2の値をy1軸にスケールした値をデータフレーム上に作る
#-------------------------------------------------------
# variabel_scaler(y2, yaxis1, yaxis2)：y2のプロットの位置をy1のスケールに合わせる
#   y2             :  y2軸に示すデータ
#   yaxis1, yaxis2 :  y1軸とy2軸の対応
#   戻り値         :  y2の値をy1軸相当の位置にした値
#-------------------------------------------------------
variable_scaler <- function(y2, yaxis1, yaxis2){
  a <- (yaxis1[2] - yaxis1[1]) / (yaxis2[2] - yaxis2[1])
  b <- (yaxis1[1] * yaxis2[2] - yaxis1[2] * yaxis2[1]) / (yaxis2[2] - yaxis2[1])
  ret <- a * y2 + b
  ret
}

#y2B：左軸のスケールに合わせる前のy2の値(before)
dt <- full_join(sales_dat_grouped, nonstandard_dat, by = "Date") %>%
  mutate("y2B" = `Percentage`*100) %>%
  mutate("y2A" = variable_scaler(y2B, yaxis1,yaxis2))

#3. 最適収穫日をオレンジ色で強調するための処理
dt2 <- dt %>%
  select(Date, Case1, Case2, y2A) %>%
  gather(-Date, -y2A, key = "Cases", value = "Income") %>%
  mutate("y2A_2" = case_when(
    Cases == "Case1" ~ `y2A`,
    Cases == "Case2" ~ 0
  )) %>%
  select(-y2A) %>%
  mutate(hl = recode(Date, "18" = "no", "19" = "no", "20" = "no", "21" = "no",
                     "22" = "no", "23" = "yes", "24" = "no", "25" = "no", "26" = "
no",
                     "27" = "no", "28" = "no")) %>%
  mutate("Income10a" = Income/2)

#4. 図示する
#-------------------------------------------------------
# axis_scaler(y1, yaxis1, yaxis2)：右軸のメモリ値を左軸のスケールに調整する
#   y1             :  y1軸の値（sec_axis()の"."）
#   yaxis1, yaxis2 :  y1軸とy2軸の対応
#   戻り値         :  y2の値をy1軸相当の位置にした値
#-------------------------------------------------------
axis_scaler <- function(y1, yaxis1, yaxis2){
  c <- (yaxis2[2] - yaxis2[1]) / (yaxis1[2] - yaxis1[1])
  d <- (yaxis2[1] * yaxis1[2] - yaxis2[2] * yaxis1[1]) / (yaxis1[2] - yaxis1[1])
  ret <- c * y1 + d
  ret
}
```

```
p2 <- ggplot(dt2, mapping = aes(x = Date, group = 1)) +
  geom_bar(mapping = aes(x = Date, y = y2A_2, fill = hl), stat = "identity", width
= 0.8) +
  geom_line(mapping = aes(y = Income10a, group = Cases, colour = Cases)) +
  geom_point(mapping = aes(x = Date, y = Income10a, group = Cases, colour = Cases)
) +
  scale_y_continuous(limit=yaxis1, expand = c(0, 0),
                     sec.axis = sec_axis( ~(axis_scaler(., yaxis1, yaxis2)),
                     #左の軸の何分の1の値にするか, .は各y1を指している
                                          name = "Proportion of grade-out broccoli"
)) +
  labs(x = "May 2020", y = "Total income (yen/10a)", colour = "") +
  scale_color_manual(values = c("#ff4b00", "#03af7a")) +
  scale_fill_manual(values = c("grey", "#f6aa00"), guide = "none") +
  theme_classic() +
  theme(legend.position = "top",
        legend.text = element_text("Times New Roman", size = 14),
        axis.text.x = element_text("Times New Roman", size =14),
        axis.text.y = element_text("Times New Roman", size = 14),
        axis.title.y = element_text("Times New Roman", size = 16),
        axis.title.x = element_text("Times New Roman", size = 16))

print(p2)
```



“Fig.4-9 一斉収穫を想定した時の収入推移の予測”のレシピ

2020年の生育予測モデル、気象データ、ドローン空撮初日(5/12)のUAVデータを用いて、外挿に基づく最適収穫日の予測を行う。

1. 予測値を用意する

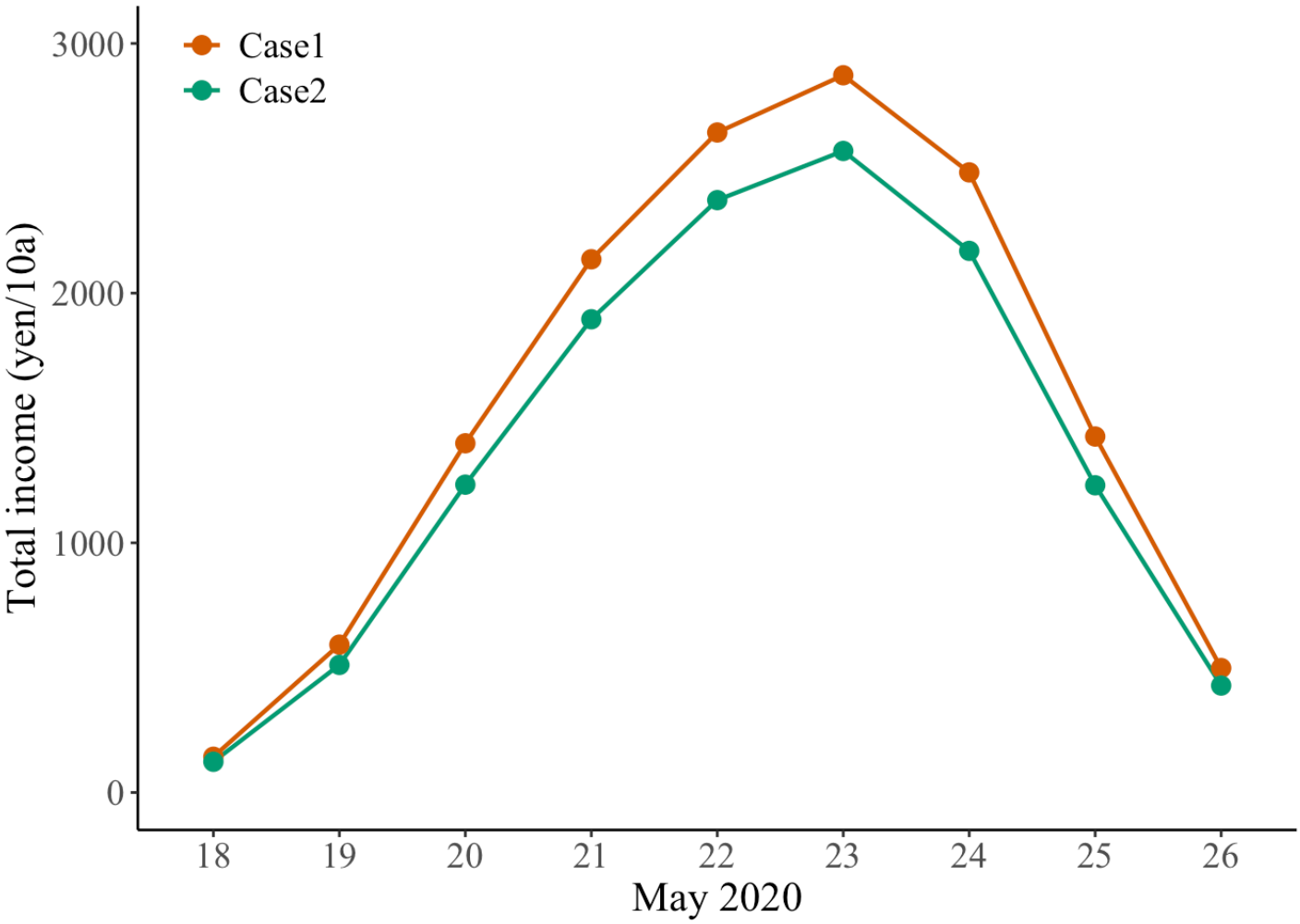
```
#2021年モデルを適用
a <- 5.580e+00
b <- 1.935e+00
c <- 6.164e-03

#1. 5/19までの積算温度をGETする。
dat4v_wide <- dat3_wide %>%
  mutate("18" = log({a - log(`20200518`)}/b)/-c, .after = `20200518`) %>%
  select(`ID`,`18`)

#2.5/11-5/25までの積算温度を並べたデータを作る
#  T = 平均気温, T(MAX) = 20, T(BASE) = 0
#tempsum_by5/19は、5/19の平均気温も含む
#5/21のHDを推定したい時は、Tempsum_5/20を使う
temp_sumv <- dat4v_wide %>%
  mutate("19" = `18` + 19) %>% #実測値(UAVを撮影した日)
  mutate("20" = `19` + 17.3) %>%
  mutate("21" = `20`+ 12.4) %>%
  mutate("22" = `21`+ 11.6) %>%
  mutate("23" = `22`+ 15.4) %>%
  mutate("24" = `23`+ 18.9) %>%
  mutate("25" = `24`+ 20) %>%
  mutate("26" = `25`+ 20)

#↑縦長データにする
temp_sumv_tidy <- temp_sumv %>%
  gather(-ID, key = `Date`, value = Tempsum)

#3. Pred_HDを算出
pred_valuev <- temp_sumv_tidy %>%
  mutate("pred_HD" = exp(a - b*exp(-c*`Tempsum`))) %>%
  select(1,2,4) %>%
  mutate("pred_HD_cm" = `pred_HD`/10) %>%
  select(-`pred_HD`)
```

2. 日毎の収入を算出する

```
pred_tidy_grouped <- pred_valuev %>%
  group_by(Date)

#サイズ別に個数を集計する
pred_sales_Msize <- pred_tidy_grouped %>%
  filter(between(pred_HD_cm, 11, 12)) %>%
  summarize(number=n()) %>%
  mutate(Size = "M")
pred_sales_Lsize <- pred_tidy_grouped %>%
  filter(between(pred_HD_cm, 12, 13)) %>%
  summarize(number=n()) %>%
  mutate(Size = "L")
pred_sales_2Lsize <- pred_tidy_grouped %>%
  filter(between(pred_HD_cm, 13, 15)) %>%
  summarize(number=n()) %>%
  mutate(Size = "2L")

#3つのデータをマージ
pred_sales_size <- merge(pred_sales_Msize, pred_sales_Lsize, all = T)
pred_sales_size <- merge(pred_sales_size, pred_sales_2Lsize, all = T)

#"implicit missing value"を補う
pred_sales_size <- pred_sales_size %>%
  complete(Date,Size,
           fill = list(number = 0))

#収入に換算する
pred_sales_size$number <- as.numeric(pred_sales_size$number)
pred_sales_dat <- pred_sales_size %>%
  mutate("Relative Price case1" = case_when(
    Size == "2L" ~  number*`2L_c1`,
    Size == "L" ~ number,
    Size == "M" ~ number*M_c1
  )) %>%
  mutate("Relative Price case2" = case_when(
    Size == "2L" ~  number*`2L_c2`,
    Size == "L" ~ number,
    Size == "M" ~ number*M_c2
  ))

pred_sales_dat_grouped <- pred_sales_dat %>%
  group_by(Date) %>%
  dplyr::summarise("Case1" = sum(`Relative Price case1`),"Case2" = sum(`Relative P
rice case2`))

#図示用に縦長データに変換する
pred_sales_dat_grouped_tidy <- pivot_longer(pred_sales_dat_grouped,
                                            cols = c(Case1, Case2),
                                            names_to = "Cases",
                                            values_to = "Totalprice") %>%
  relocate(Cases) %>%
  mutate("Income10a" = Totalprice/2)
```

3. 図示する

```
p3 <- ggplot(pred_sales_dat_grouped_tidy, aes(x = Date, y = `Income10a`)) +
  geom_line(aes(group = Cases, colour = Cases), size = 0.8) +
  geom_point(aes(colour = Cases), size = 3) +
  labs(
  y = "Total income (yen/10a)",
  x = "May 2020",
  colour = "") +
  ylim(0, 3000) +
  theme_classic() +
  scale_color_manual(values = c("Case1" = "#D55E00", "Case2" = "#009E73")) +
  theme(
        axis.text.x = element_text("Times New Roman", size = 14),
        axis.text.y = element_text("Times New Roman", size = 14),
        axis.title.y = element_text("Times New Roman", size = 16),
        axis.title.x = element_text("Times New Roman", size = 16),
        legend.text = element_text("Times New Roman", size = 14),
        legend.position = c(0.1, 0.95))

print(p3)
```



Chapter5 - Test in Fukushima

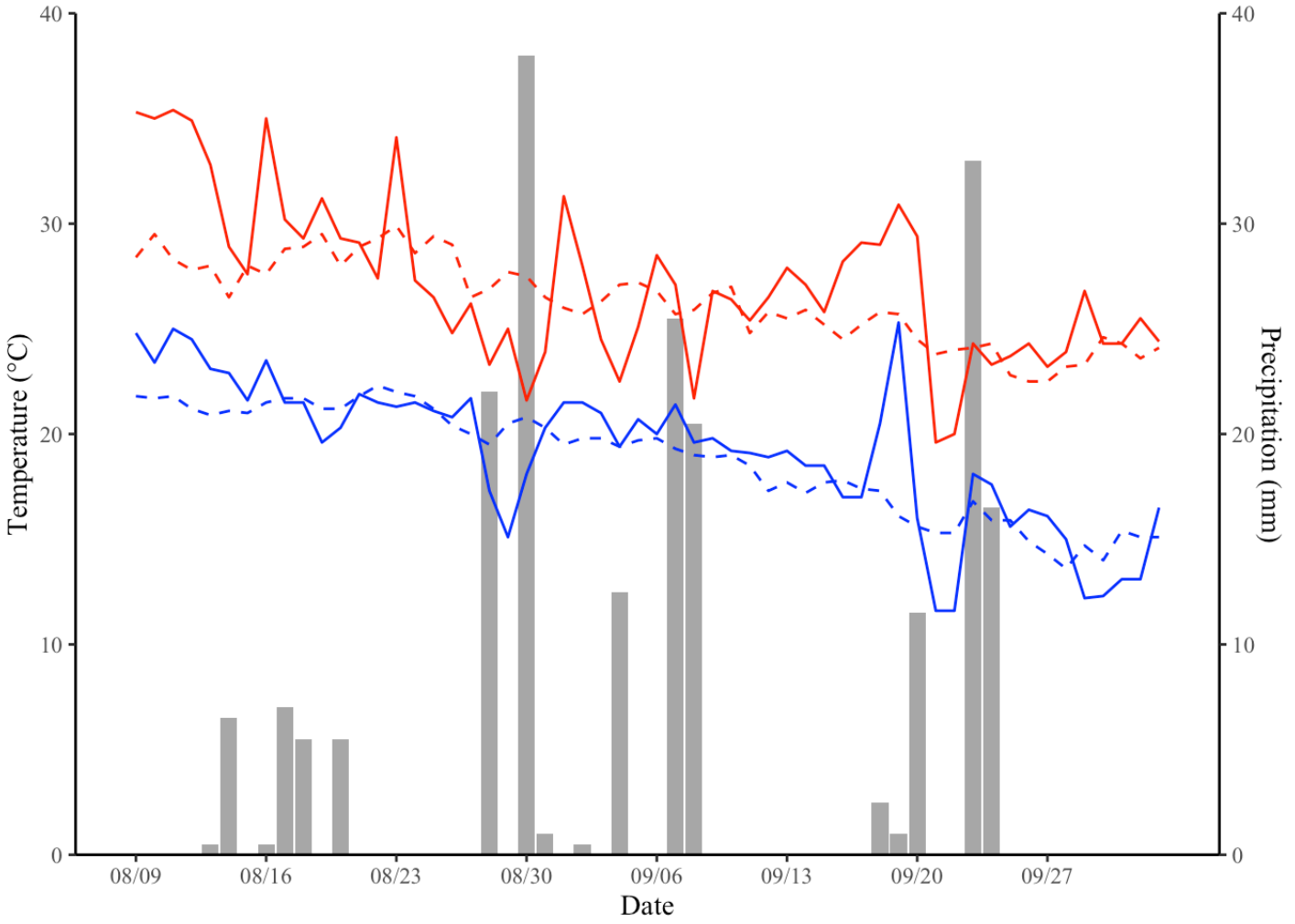
Erika Nishida

2023-01-31

- Fig 5-1 (気温と降水量の図示)のレシピ

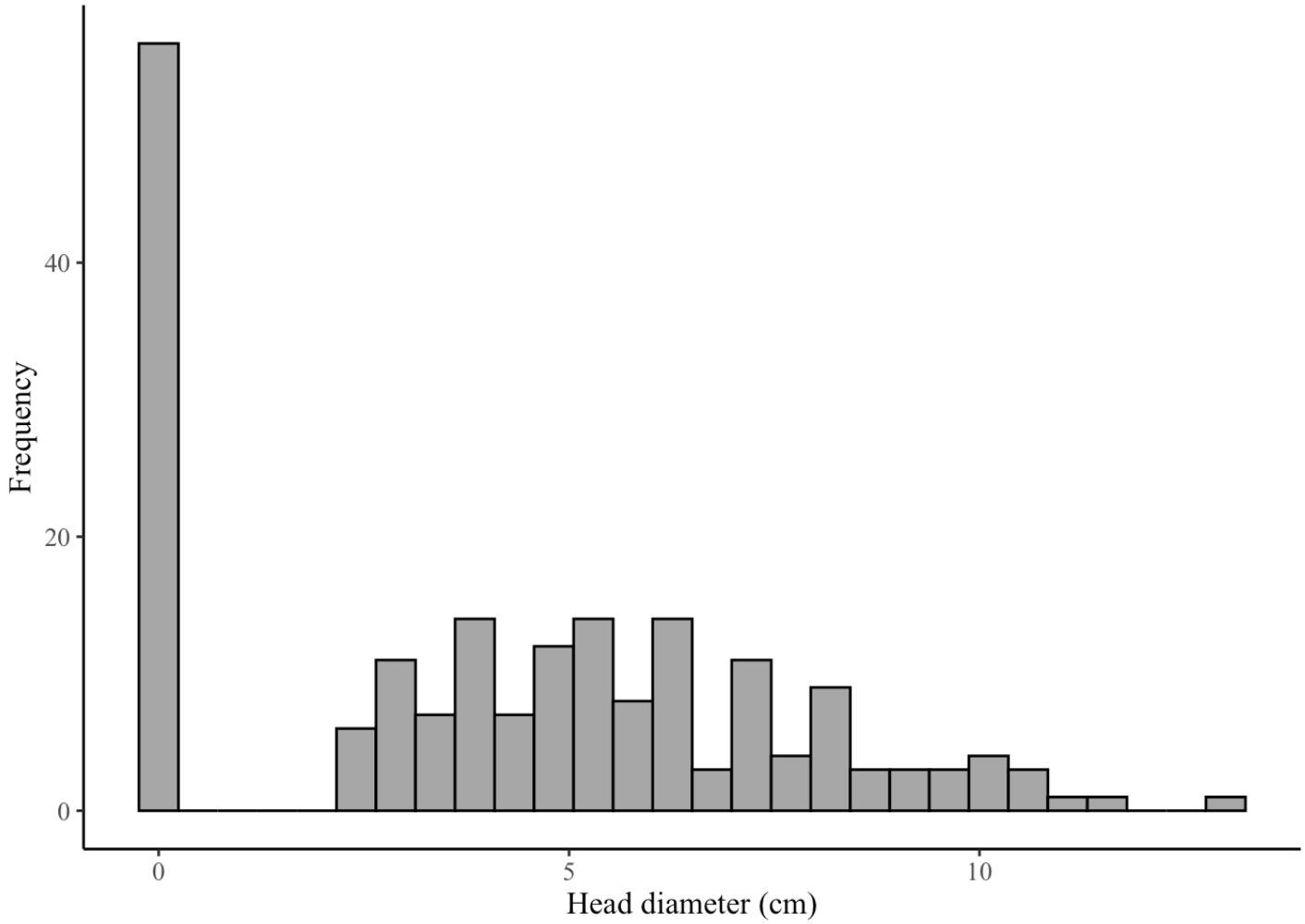
```
#Install packages
library(tidyverse)
library(scales)

#import data
dt <- read.csv("/Users/erika/Desktop/Fukushima.csv")

#ggplotでは、2軸グラフを書く時、y軸のスケールはあくまで共通になってしまう。
#そこで2つ目の変数を1つ目の変数に合わせて縮めたり伸ばしたりして収めて、プロットし、
#sec.axisというオプションで、左軸を好きなように変換した軸を右側に設定することで、
#異なるスケールのy軸を右側におくことを実現させる。
#故に、y2の値をy1軸にスケールした値をデータフレーム上に作る必要がある。

#1. まずは各軸の範囲を決める
yaxis1_2021 <- c(0, 40)
yaxis2_2021 <- c(0, 40)

#2. y2の値をy1軸にスケールした値をデータフレーム上に作る
#-------------------------------------------------------
# variabel_scaler(y2, yaxis1, yaxis2)：y2のプロットの位置をy1のスケールに合わせる
#   y2              :  y2軸に示すデータ
#   yaxis1, yaxis2 :  y1軸とy2軸の対応
#   戻り値          :  y2の値をy1軸相当の位置にした値
#-------------------------------------------------------
variable_scaler_2021 <- function(y2_2021, yaxis1_2021, yaxis2_2021){
  a_2021 <- (yaxis1_2021[2] - yaxis1_2021[1]) / (yaxis2_2021[2] - yaxis2_2021[1])
  b_2021 <- (yaxis1_2021[1] * yaxis2_2021[2] - yaxis1_2021[2] * yaxis2_2021[1]) / (
yaxis2_2021[2] - yaxis2_2021[1])
  ret_2021 <- a_2021 * y2_2021 + b_2021
  ret_2021
}

#y2B：左軸のスケールに合わせる前のy2の値(before)
dt2 <- dt %>%
  mutate("y2A" = variable_scaler_2021(rain, yaxis1_2021, yaxis2_2021))

#3. 図示する
#-------------------------------------------------------
# axis_scaler(y1, yaxis1, yaxis2)：右軸のメモリ値を左軸のスケールに調整する
#   y1              :  y1軸の値（sec_axis()の"."）
#   yaxis1, yaxis2 :  y1軸とy2軸の対応
#   戻り値          :  y2の値をy1軸相当の位置にした値
#-------------------------------------------------------
axis_scaler_2021 <- function(y1_2021, yaxis1_2021, yaxis2_2021){
```

```
  c_2021 <- (yaxis2_2021[2] - yaxis2_2021[1]) / (yaxis1_2021[2] - yaxis1_2021[1])
  d_2021 <- (yaxis2_2021[1] * yaxis1_2021[2] - yaxis2_2021[2] * yaxis1_2021[1]) / (
yaxis1_2021[2] - yaxis1_2021[1])
  ret_2021 <- c_2021 * y1_2021 + d_2021
  ret_2021
}

#日付のx軸を用意
datebreaks <- seq(as.Date("2022/8/9"), as.Date("2022/10/3"), by = "7 day")

#図示
p <- ggplot(data = dt2, mapping = aes(x = as.Date(Date))) +
  geom_bar(mapping = aes(y = y2A), stat = "identity", fill = "darkgrey") +
  geom_line(mapping = aes(y = max_temp), color = "red") +
  geom_line(mapping = aes(y = min_temo), color = "blue") +
  geom_line(mapping = aes(y = nrm10_max_temp), color = "red", linetype = "dashed")
+
  geom_line(mapping = aes(y = nrm10_min_temp), color = "blue", linetype = "dashed")
+
  scale_y_continuous(limit = yaxis1_2021, expand = c(0, 0),
                     sec.axis = sec_axis( ~(axis_scaler_2021(., yaxis1_2021, yaxis2
_2021)),
                                          name = "Precipitation (mm)")) +
  scale_x_date(breaks = datebreaks, labels = date_format("%m/%d")) +
  labs(x = "Date",
       y = "Temperature (℃)") +
  theme_classic() +
  theme(text = element_text(family = "Times New Roman"))

print(p)
```



- Fig 5-2 (手測定した花蕾直径のヒストグラム)のレシピ

```
#import data
dt3 <- read.csv("/Users/erika/Desktop/Fukushima_handmeasure.csv")

#draw histogram
p2 <- ggplot(data = dt3) +
  geom_histogram(mapping = aes(x = hd_cm),
                 fill = "darkgray",
                 color = "black",
                 bins = 28,
                 right = TRUE) +
  labs(x = "Head diameter (cm)",
       y = "Frequency") +
  theme_classic() +
  theme(text = element_text(family = "Times New Roman"),
        axis.text = element_text(size = 10),
        axis.title = element_text(size = 12))
print(p2)
```


Frequency
40
20
0
0
5
10
Head diameter (cm)