warmXtrophic Project: Flowering Phenology Analysis

Moriah Young

July 09, 2021

Load in packages & data

```
# clear all existing data
rm(list = ls())

# Load packages
library(tidyverse)
library(ggplot2)
library(lme4)
library(lmerTest)
library(emmeans)
library(vegan)
library(car)
library(rstatix)
library(scales)
library(fitdistrplus)
library(moments)  # for calculating skewness of data
library(ggpubr)
library(jtools)  # summ() function
library(predictmeans)
library(olsrr)
library(car)
library(fitdistrplus)
library(ggpubr)
library(interactions)
library(sjPlot)
library(effects)
library(glmmTMB)
library(GGally)  # ggpairs() function
library(bbmle)  # AICtab() function

# Set ggplot2 plotting This code for ggplot2 sets the theme to mostly black and
# white (Arial font, and large font, base size=24)
theme_set(theme_bw(14))
theme_update(axis.text.x = element_text(size = 12, angle = 90), axis.text.y = element_text(size = 12))

# Set working directory
Sys.getenv("L1DIR")
```

```
## [1] "/Volumes/GoogleDrive/Shared drives/SpaCE_Lab_warmXtrophic/data/L1/"
```

```r
L1_dir <- Sys.getenv("L1DIR")
L2_dir <- Sys.getenv("L2DIR")
list.files(L1_dir)
```

```
##  [1] "ANPP"              "climate_data"      "CN"
##  [4] "greenup"           "herbivory"         "HOBO_data"
##  [7] "PAR"               "phenology"         "plant_composition"
## [10] "SLA"
```

```r
# Set ggplot2 plotting This code for ggplot2 sets the theme to mostly black and
# white (Arial font, and large font, base size=24)
theme_set(theme_bw(14))
theme_update(axis.text.x = element_text(size = 12, angle = 90), axis.text.y = element_text(size = 12))

# Read in data cleaned phenology data from L1
phen_data <- read.csv(file.path(L1_dir, "phenology/final_flwr_sd_L1.csv"))
phen_data$X <- NULL  # get rid of 'X' column that shows up
View(phen_data)  # take a look at the data to see if looks good

# Order warm and ambient so that warm shows up first in plotting (and is default
# is red = warm; blue = ambient). First make it a factor
phen_data$state <- as.factor(phen_data$state)
levels(phen_data$state)
```

```
## [1] "ambient" "warmed"
```

```r
# [1] 'ambient' 'warmed'
phen_data$state <- factor(phen_data$state, levels(phen_data$state)[c(2, 1)])
levels(phen_data$state)
```

```
## [1] "warmed"  "ambient"
```

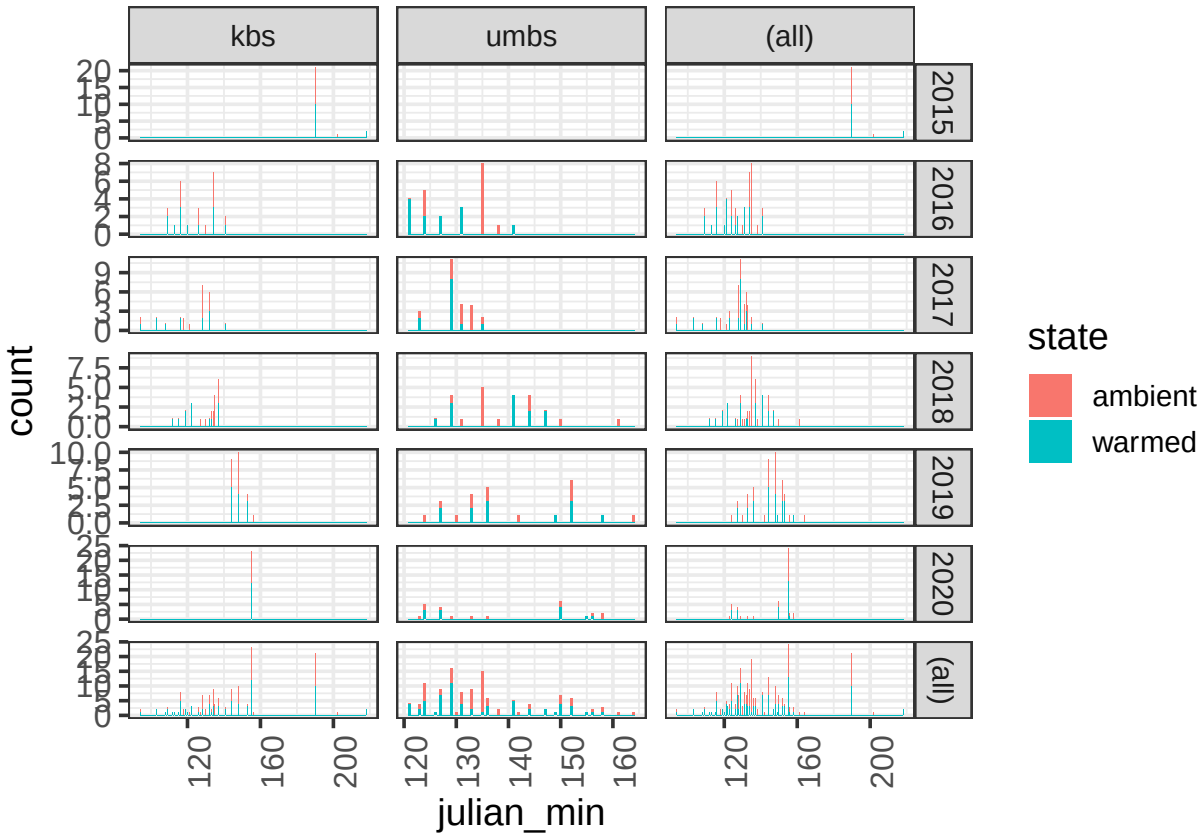
```r
# [1] 'warmed' 'ambient'

# Flowering data species level data for flowering
flwr_spp <- read.csv(file.path(L1_dir, "phenology/final_flwr_species_L1.csv"))
flwr_spp$X <- NULL

# plot level data for flowering
flwr_plot <- read_csv(file.path(L1_dir, "phenology/final_flwr_plot_L1.csv"))
```

```
## Warning: Missing column names filled in: 'X1' [1]
```

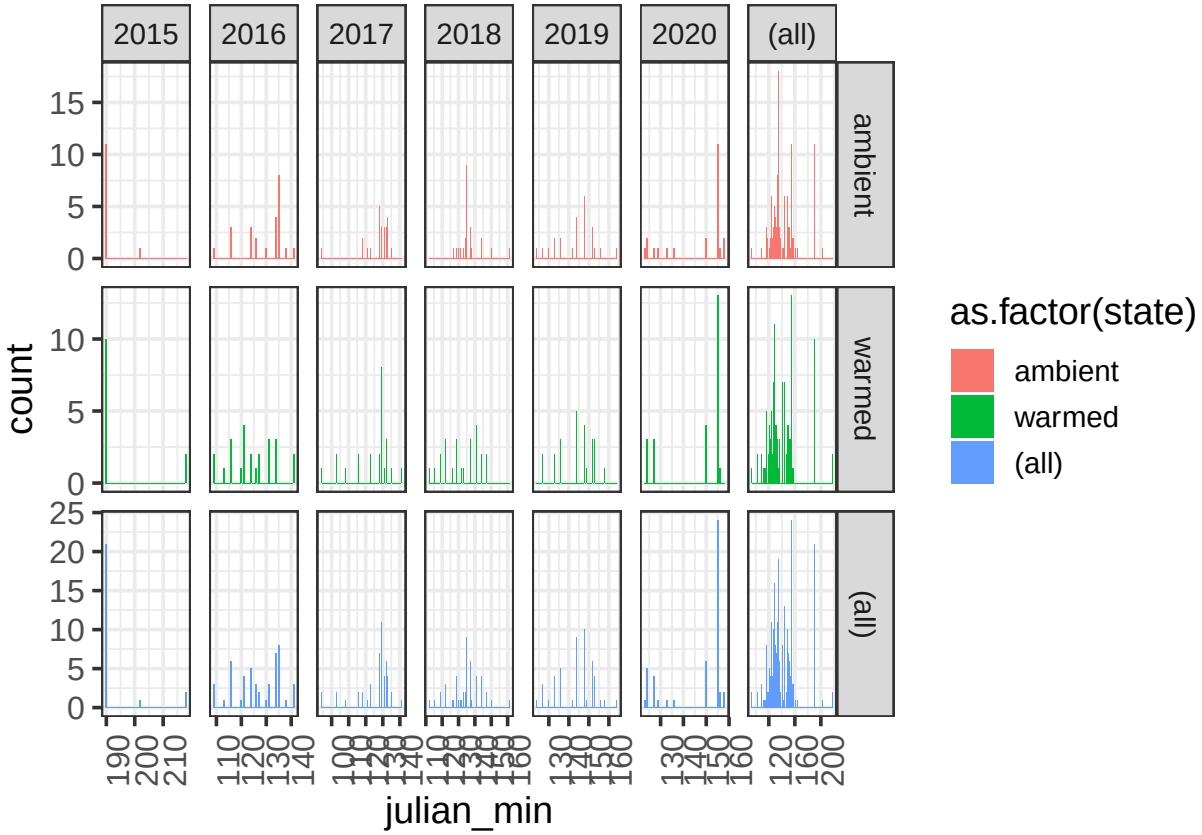
```r
flwr_plot$X1 <- NULL
```

Data exploration

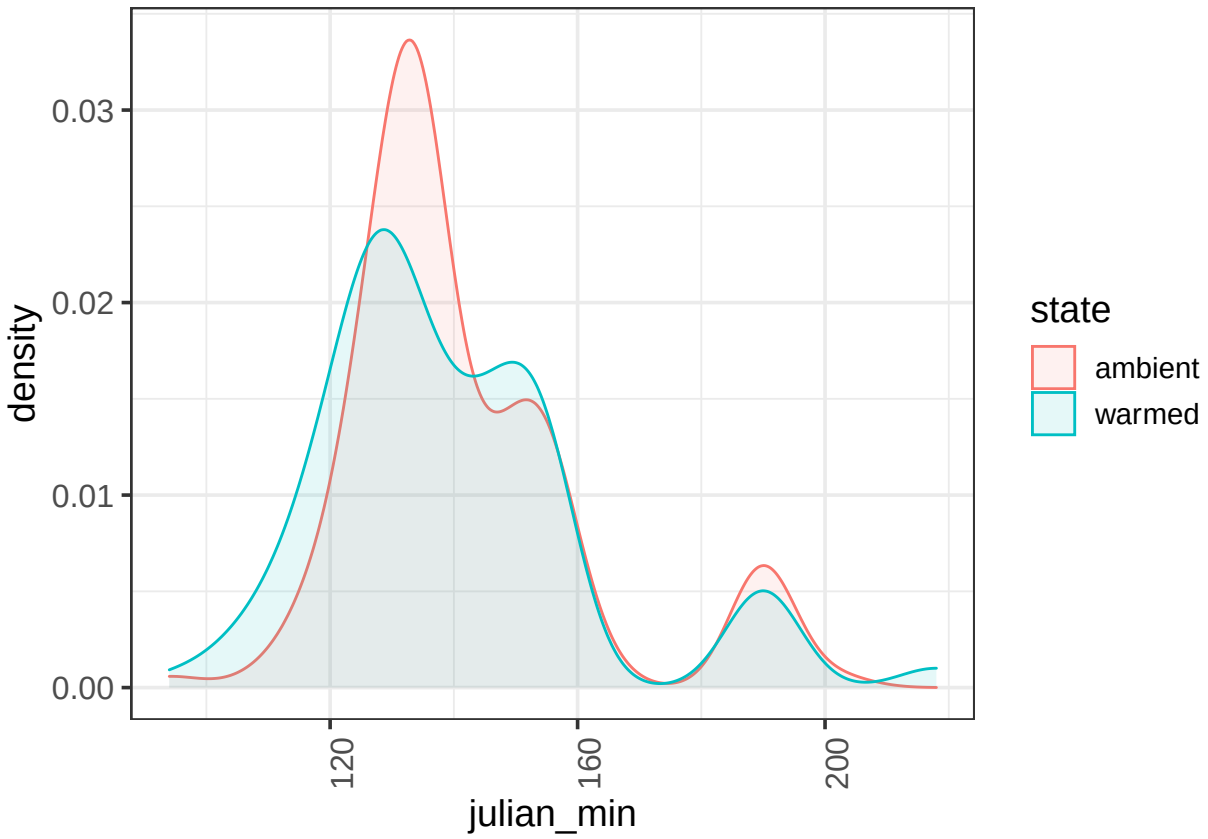
```
# Visualizing avg minimum julian date for both sites
ggplot(flwr_plot, aes(julian_min, fill = state)) + geom_histogram(binwidth = 0.5) +
    facet_grid(year ~ site, margins = TRUE, scales = "free")
```



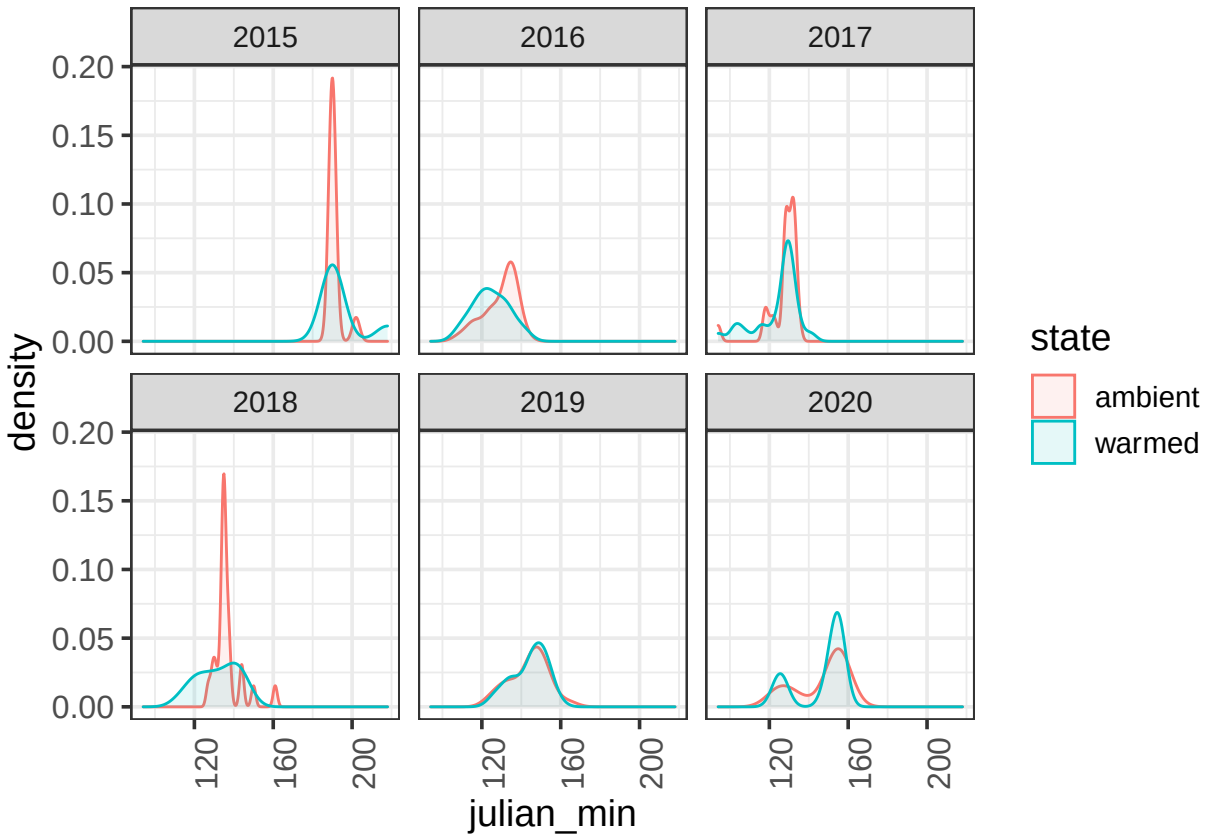

```
ggplot(flwr_plot, aes(julian_min, fill = as.factor(state))) + geom_histogram(binwidth = 0.5) +
    facet_grid(state ~ year, margins = TRUE, scales = "free")
```



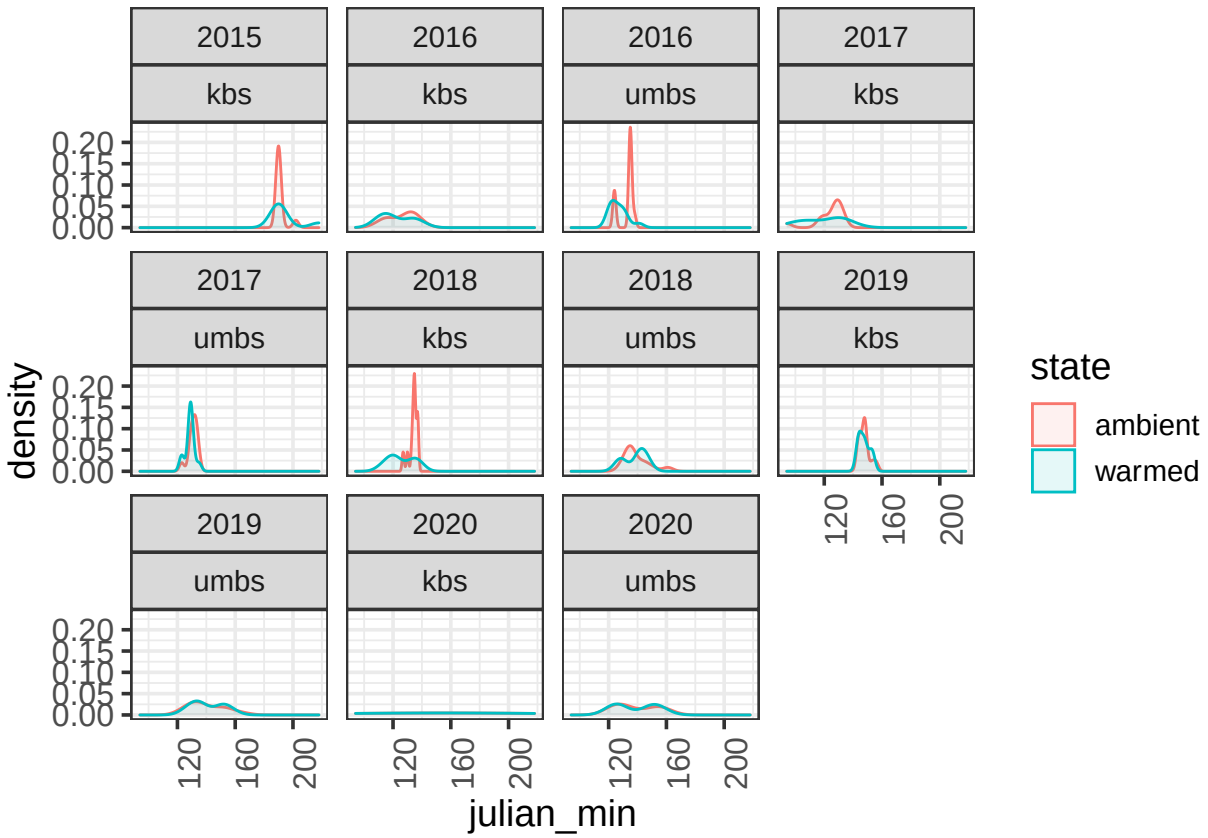

```
ggplot(flwr_plot, aes(julian_min, fill = state, color = state)) + geom_density(alpha = 0.1)
```



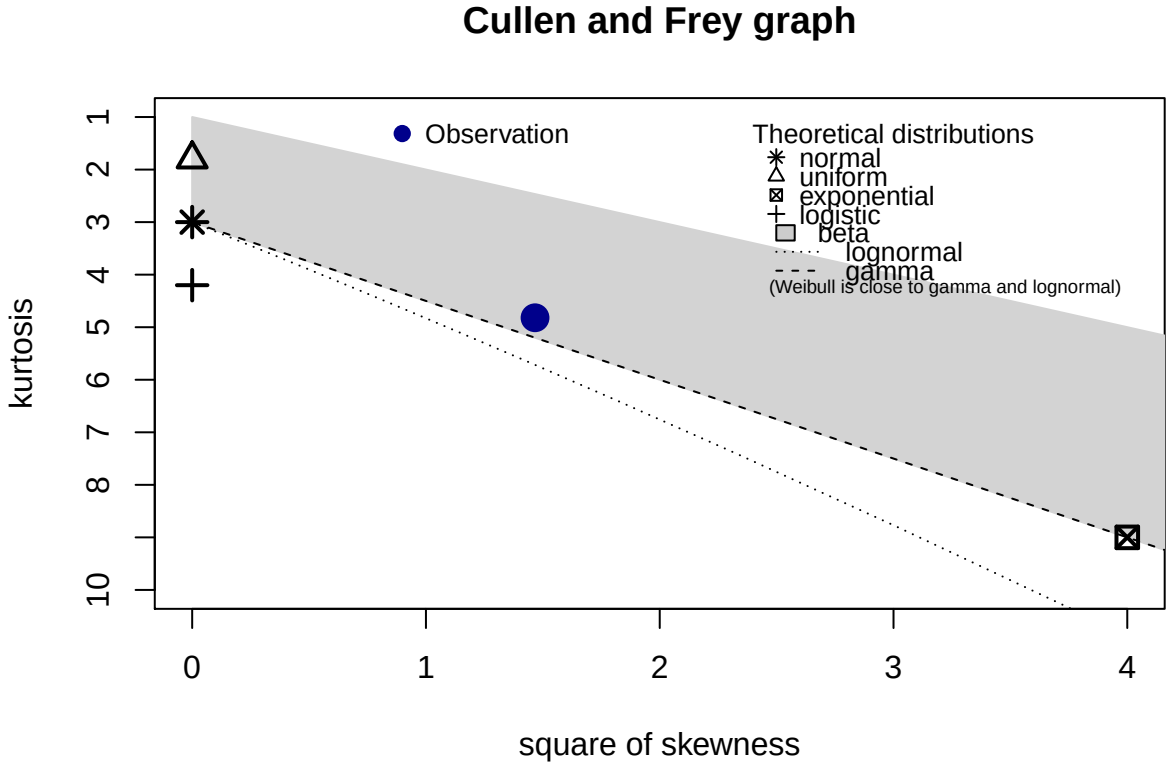

```
ggplot(flwr_plot, aes(julian_min, fill = state, color = state)) + geom_density(alpha = 0.1) +
    facet_wrap(~year)
```




```
ggplot(flwr_plot, aes(julian_min, fill = state, color = state)) + geom_density(alpha = 0.1) +
    facet_wrap(~year + site)
```

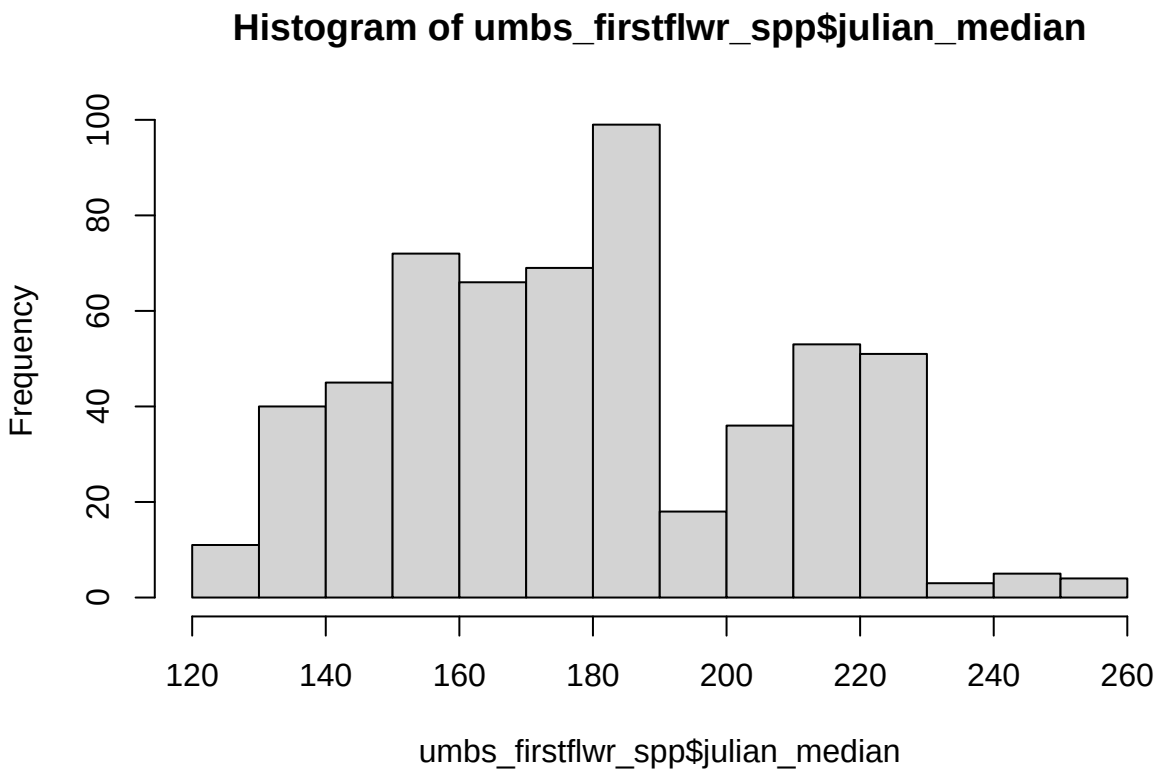



```
descdist(flwr_plot$julian_min, discrete = FALSE)
```

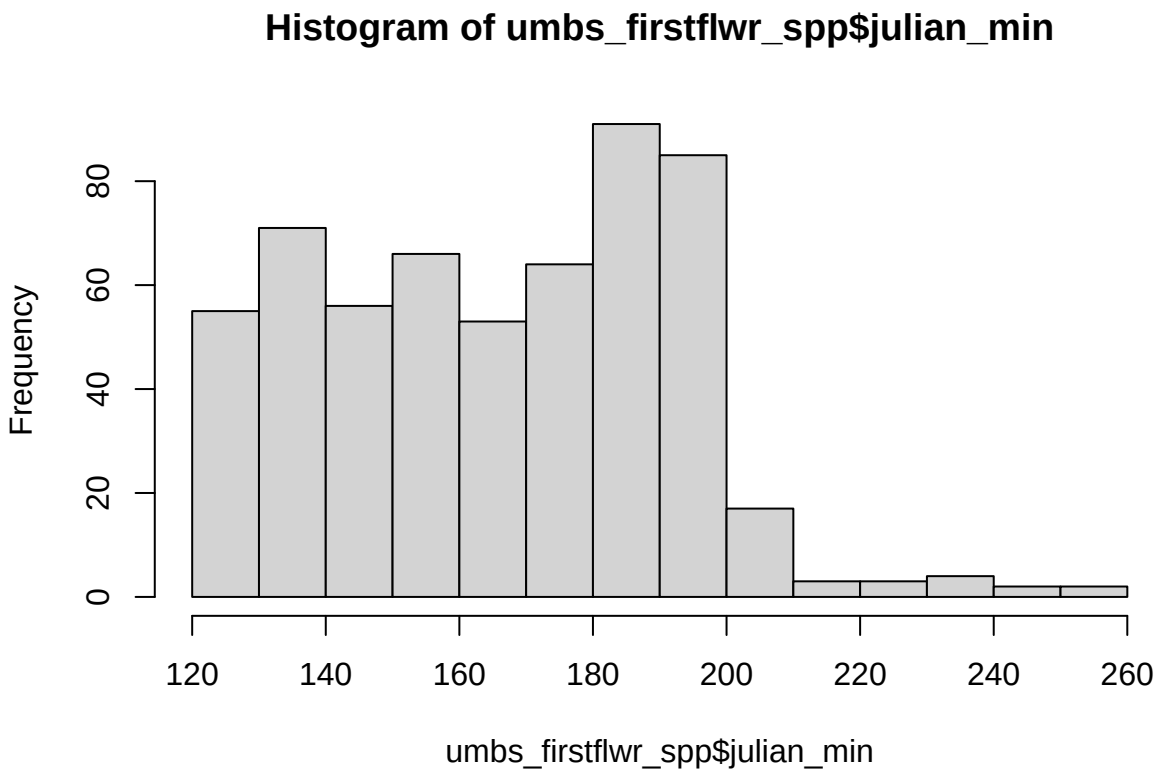


```
## summary statistics
## ------
## min:  94   max:  218
## median:  135
## mean:  140.7452
## estimated sd:  20.92644
## estimated skewness:  1.211022
## estimated kurtosis:  4.820602
```

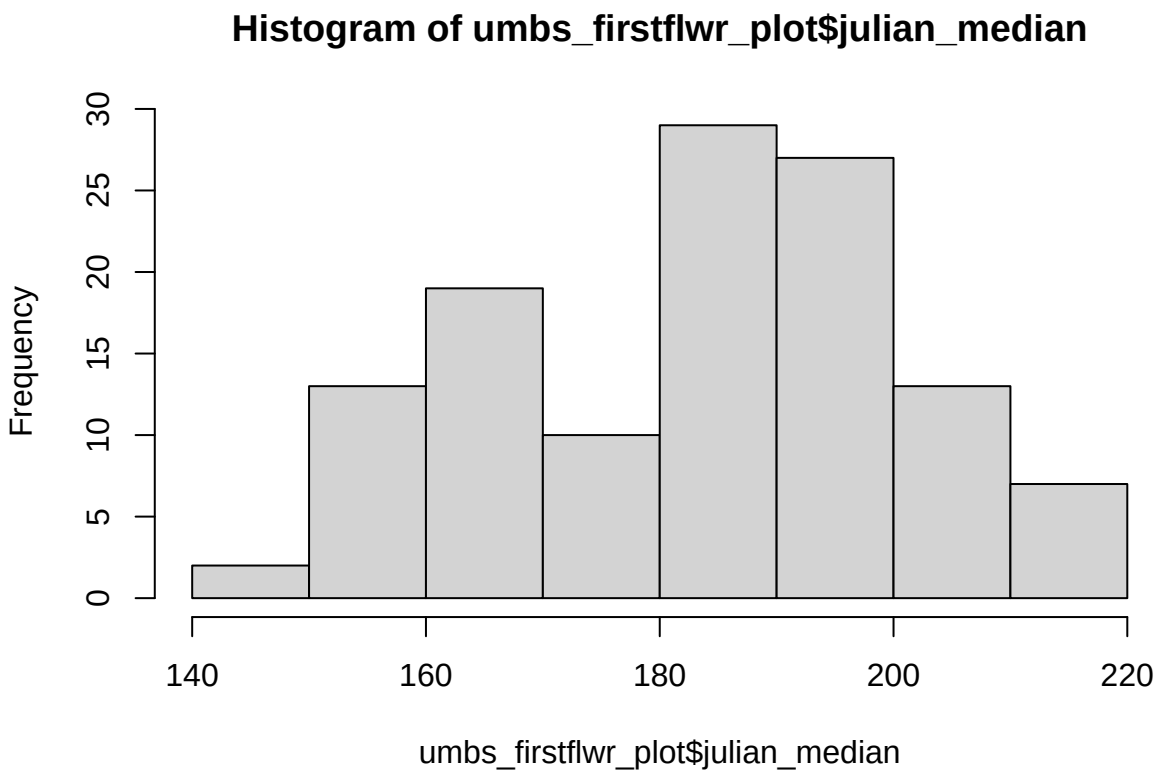
```r
### UMBS ###
umbs_firstflwr_spp <- subset(flwr_spp, site == "umbs")  # pull out umbs only data
hist(umbs_firstflwr_spp$julian_median)
```



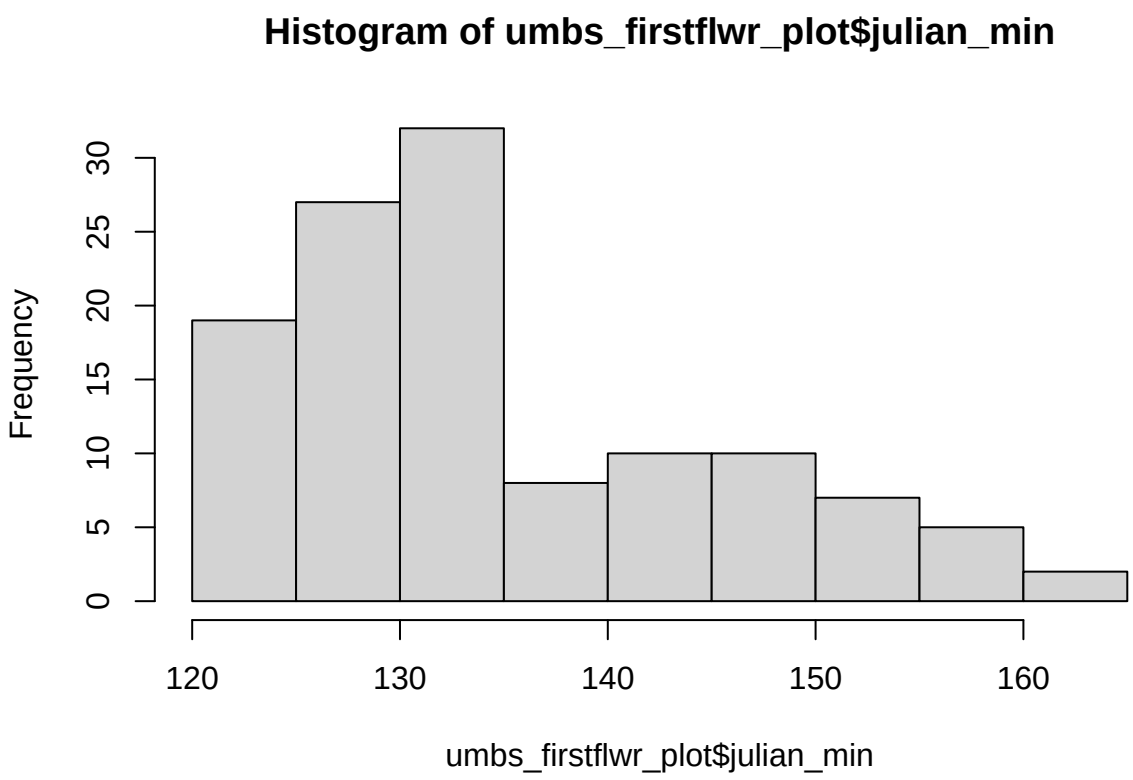

```r
hist(umbs_firstflwr_spp$julian_min)
```



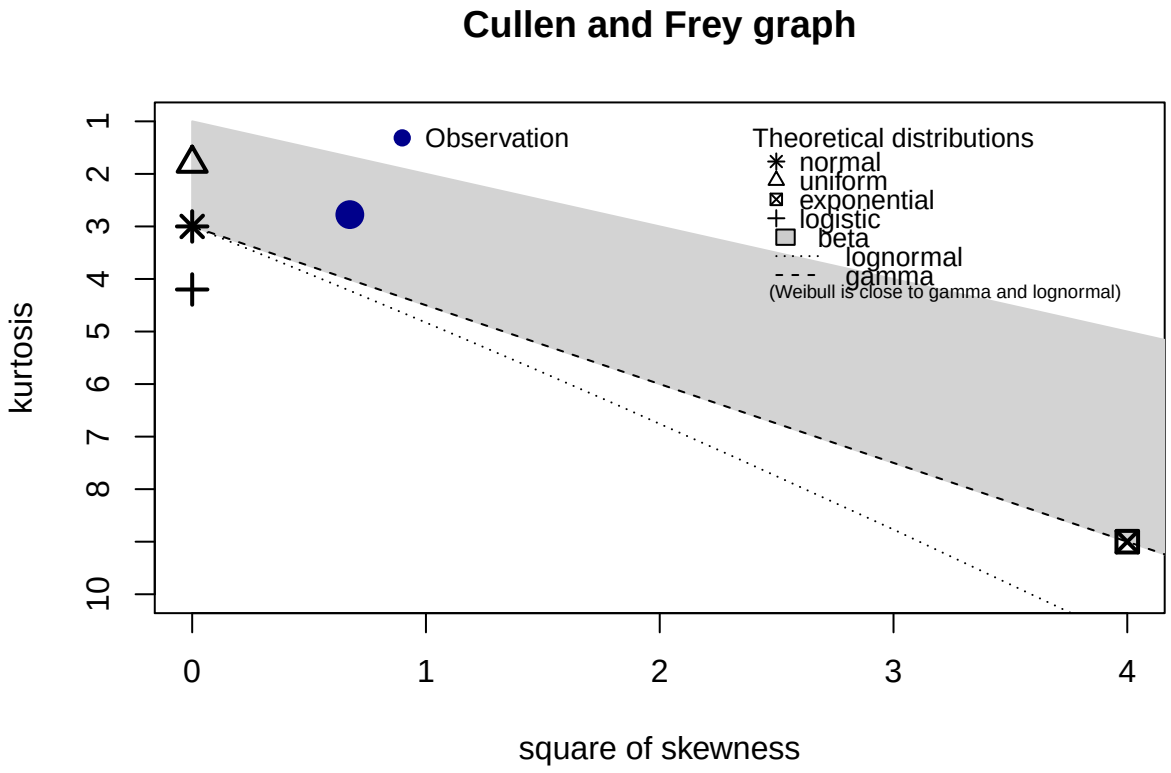

```
umbs_firstflwr_plot <- subset(flwr_plot, site == "umbs")  # pull out umbs only data
hist(umbs_firstflwr_plot$julian_median)
```



```r
hist(umbs_firstflwr_plot$julian_min)
```

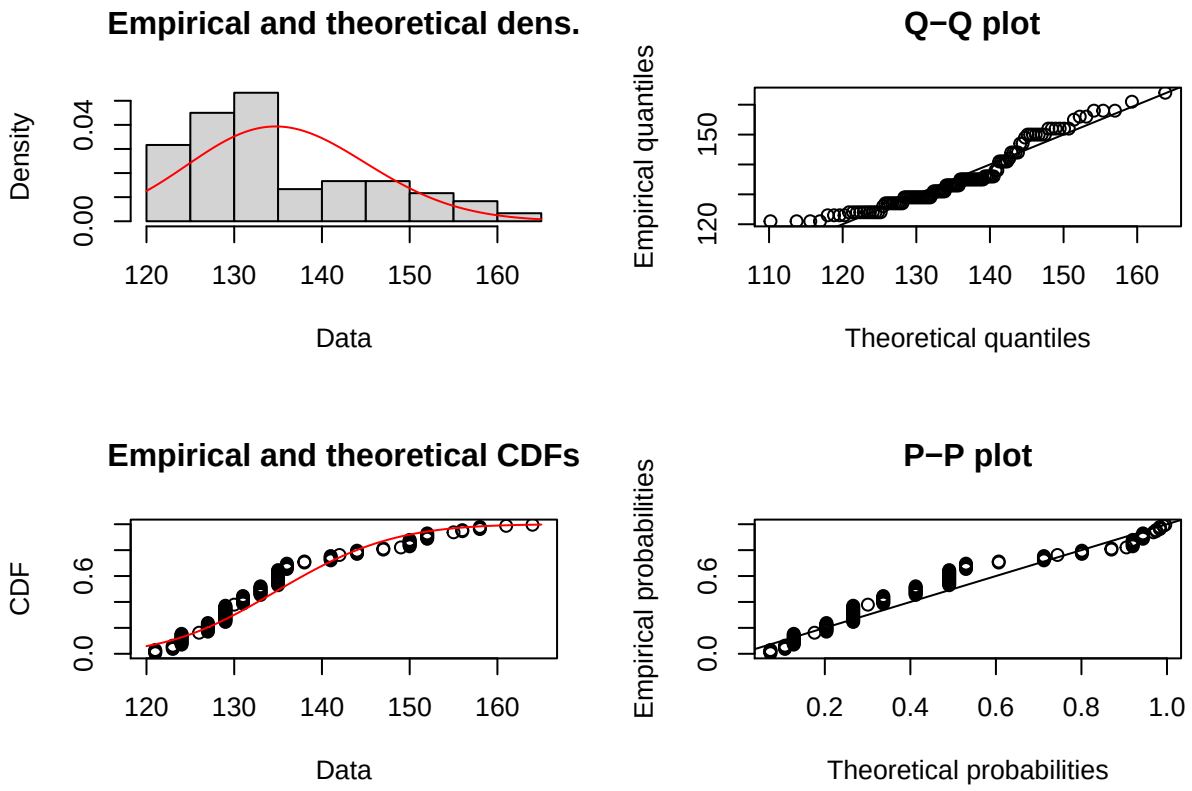



```r
# Exploring distributions for these left-skewed data:
descdist(umbs_firstflwr_plot$julian_min, discrete = FALSE)
```

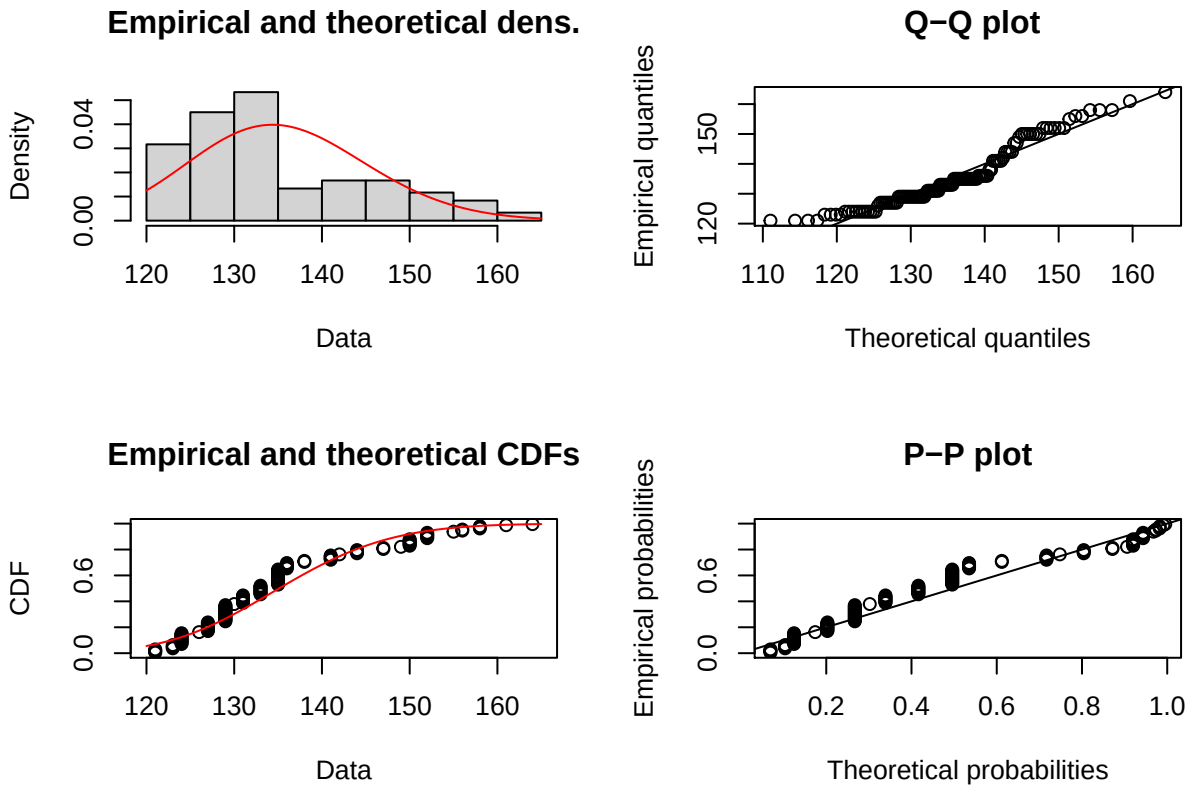


```
## summary statistics
## ------
## min:  121   max:  164
## median:  133
## mean:  135.4917
## estimated sd:  10.38745
## estimated skewness:  0.8212294
## estimated kurtosis:  2.775374
```

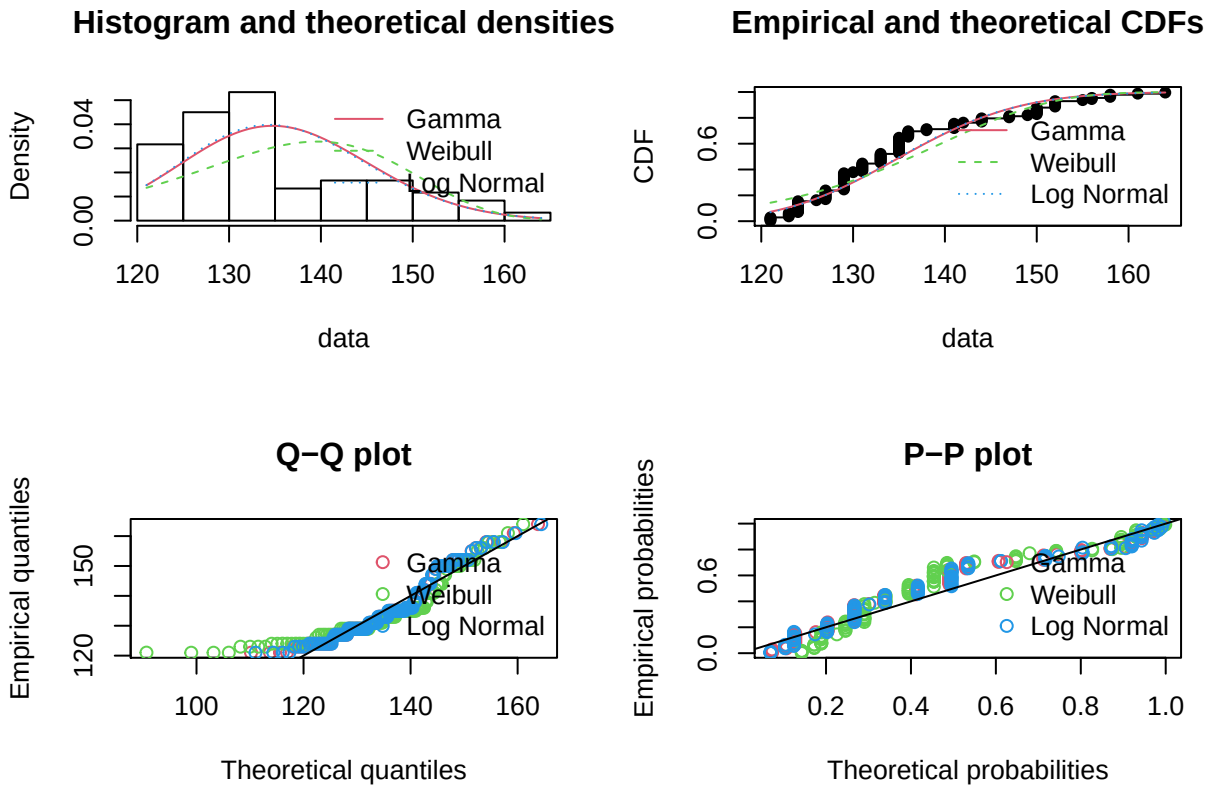
```r
# Beta distribution
fit.gamma <- fitdist(umbs_firstflwr_plot$julian_min, "gamma")
plot(fit.gamma)
```




```r
# Weibull distribution
fit.weibull <- fitdist(umbs_firstflwr_plot$julian_min, "weibull")

# Lognormal distribution
fit.ln <- fitdist(umbs_firstflwr_plot$julian_min, "lnorm")
plot(fit.ln)
```




```
par(mfrow = c(2, 2))
plot.legend <- c("Gamma", "Weibull", "Log Normal")
denscomp(list(fit.gamma, fit.weibull, fit.ln), legendtext = plot.legend)
cdfcomp(list(fit.gamma, fit.weibull, fit.ln), legendtext = plot.legend)
qqcomp(list(fit.gamma, fit.weibull, fit.ln), legendtext = plot.legend)
ppcomp(list(fit.gamma, fit.weibull, fit.ln), legendtext = plot.legend)
```



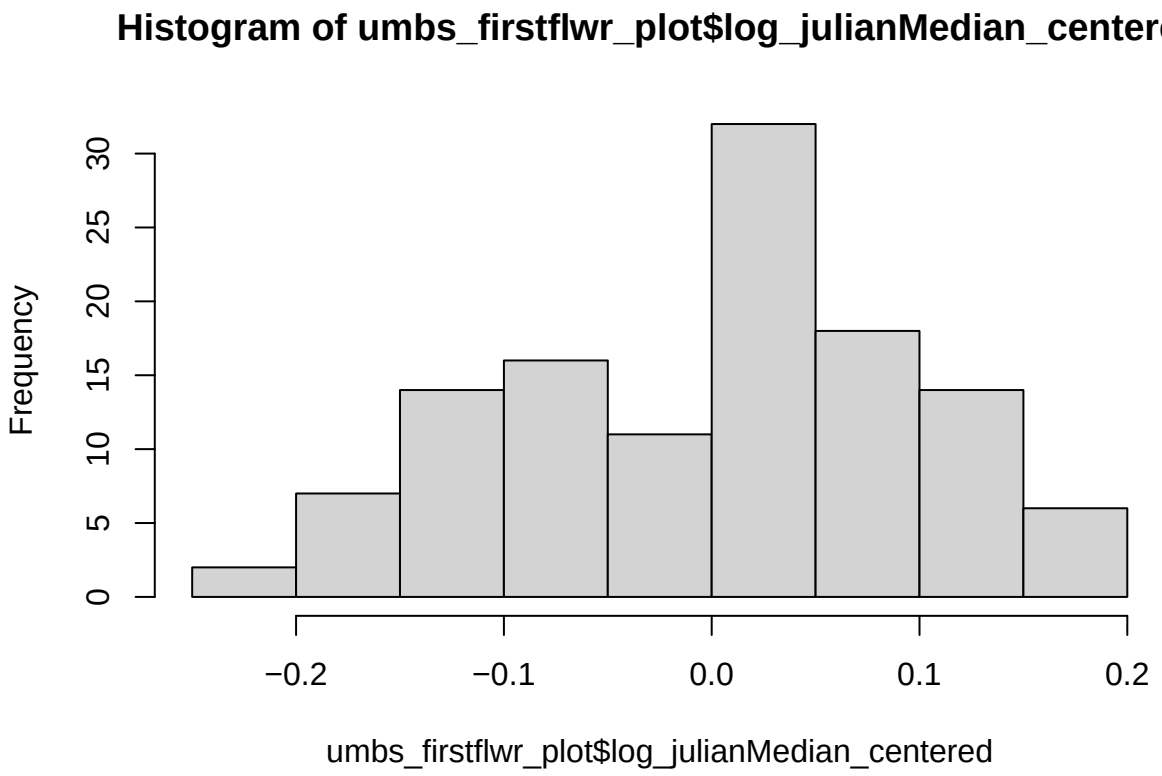

```
# Goodness of fit comparisons across fits (can't include the sqrt normal bc it
# becomes diff response values)
gofstat(list(fit.gamma, fit.weibull, fit.ln), fitnames = c("Gamma", "Weibull", "Log Normal"))
```

```
## Goodness-of-fit statistics
##                                 Gamma   Weibull Log Normal
## Kolmogorov-Smirnov statistic 0.1700250 0.2150911  0.1648747
## Cramer-von Mises statistic   0.5808497 1.0209433  0.5424176
## Anderson-Darling statistic   3.3369537 5.6394902  3.1360083
##
## Goodness-of-fit criteria
##                                   Gamma  Weibull Log Normal
## Akaike's Information Criterion 900.5729 933.5664   898.4475
## Bayesian Information Criterion 906.1479 939.1413   904.0225
```

```
# Lognormal is slightly better than gamma
```

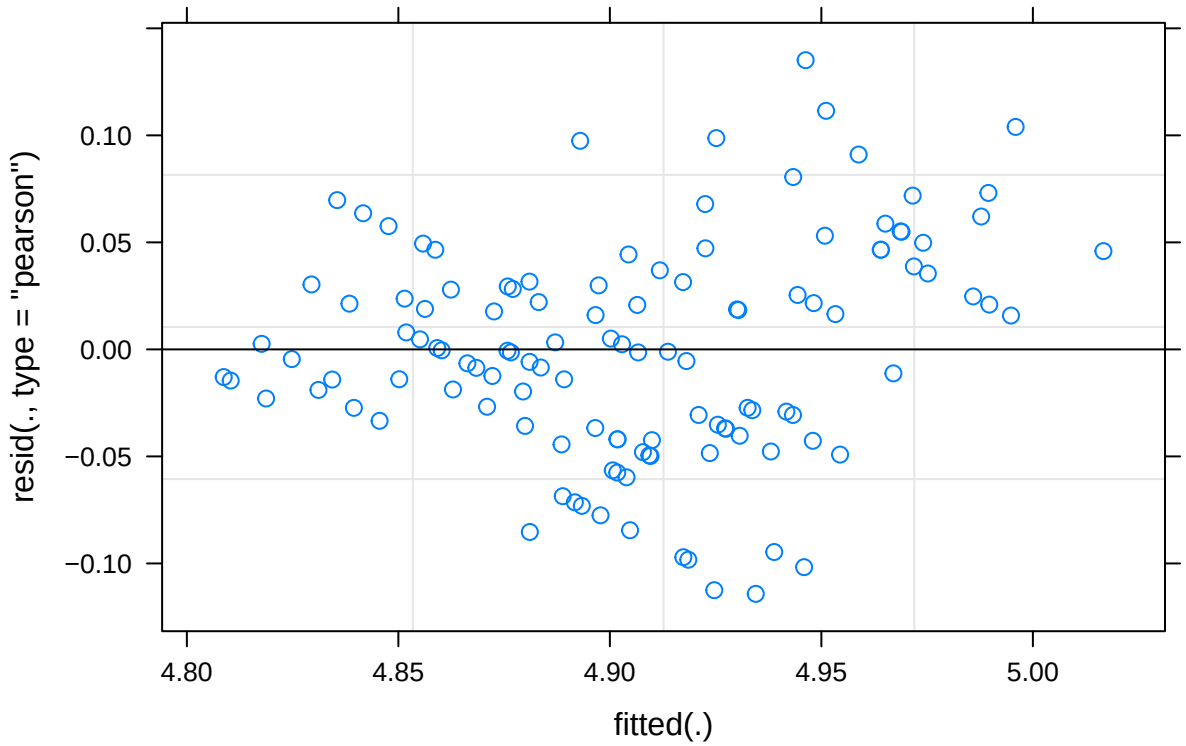
Centering and transforming data

```
umbs_firstflwr_plot$log_julian_median <- log(umbs_firstflwr_plot$julian_median)
umbs_firstflwr_plot$log_julianMedian_centered = umbs_firstflwr_plot$log_julian_median -
    mean(umbs_firstflwr_plot$log_julian_median)
hist(umbs_firstflwr_plot$log_julianMedian_centered)
```



Determining appropriate distribution in mixed effects model

```
# See Ben Bolker's site for details on fitting glmms:
# https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html Fixed effects: warming,
# year, insecticide Random effect = plot (bc observations are nested within plot)
# Fit with ML bc we are interested in estimating fixed effects more than random
# effects See
# https://uoftcoders.github.io/rcourse/lec08-linear-mixed-effects-models.html for
# more details
m1 <- lmer(log(julian_min) ~ state + year + insecticide + (1 | plot), data = umbs_firstflwr_plot,
    REML = FALSE)

# Check Assumptions: (1) Linearity: if covariates are not categorical (year
# isn't) (2) Homogeneity: Need to Check by plotting residuals vs predicted
# values.
par(mfrow = c(1, 2))
plot(m1, main = "Avg Min Julian Date")
```




```
# Homogeneity of variance is ok here (increasing variance in resids is not
# increasing with fitted values) Check for homogeneity of variances (true if
# p>0.05). If the result is not significant, the assumption of equal variances
# (homoscedasticity) is met (no significant difference between the group
# variances).
leveneTest(residuals(m1) ~ umbs_firstflwr_plot$state)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.

## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1   0.001 0.9744
##       118
```

```
# Assumption met
leveneTest(residuals(m1) ~ umbs_firstflwr_plot$plot)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.

## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group 23  0.7084 0.8265
##       96
```

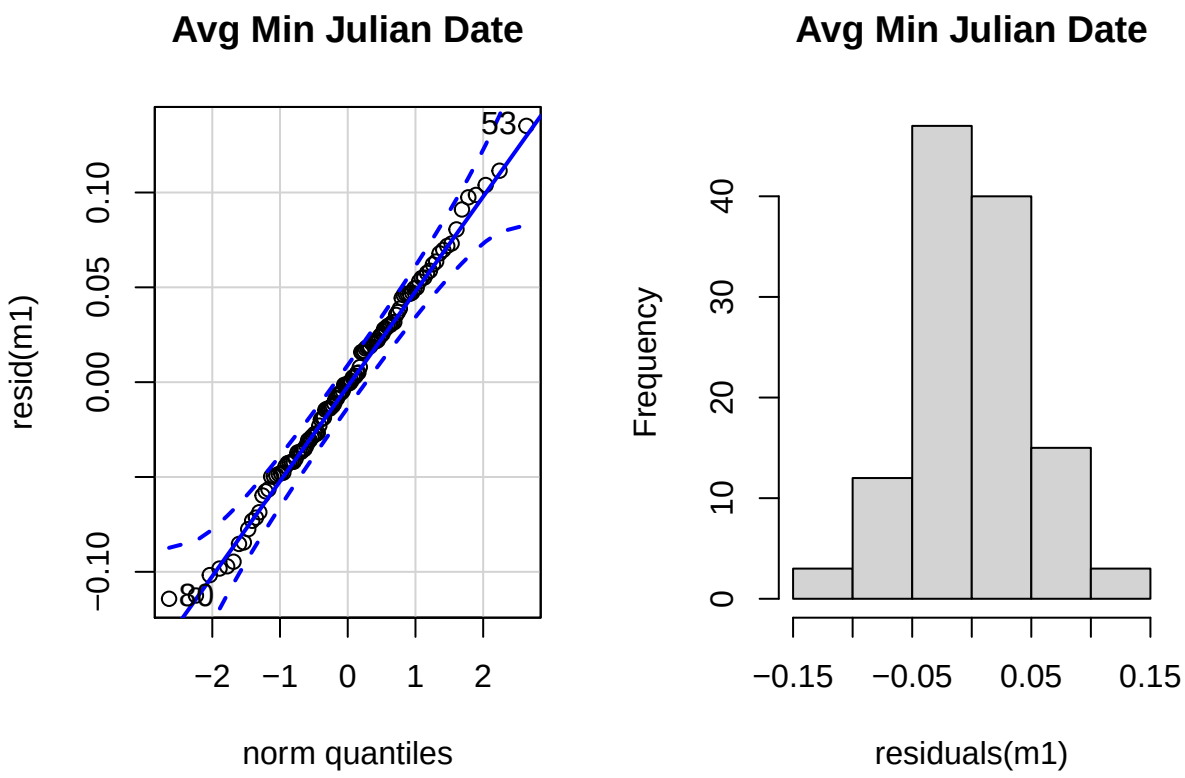
```
# Assumption met
leveneTest(residuals(m1) ~ umbs_firstflwr_plot$insecticide)
```

```
## Warning in leveneTest.default(y = y, group = group, ...): group coerced to
## factor.

## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value Pr(>F)
## group   1  1.5017 0.2228
##       118
```

```
# Assumption met

# (3) Normality of error term: need to check by histogram, QQplot of residuals,
# could do Kolmogorov-Smirnov test.  Check for normal residuals
qqPlot(resid(m1), main = "Avg Min Julian Date")
```

```
## [1] 53 80
```

```
hist(residuals(m1), main = "Avg Min Julian Date")
```



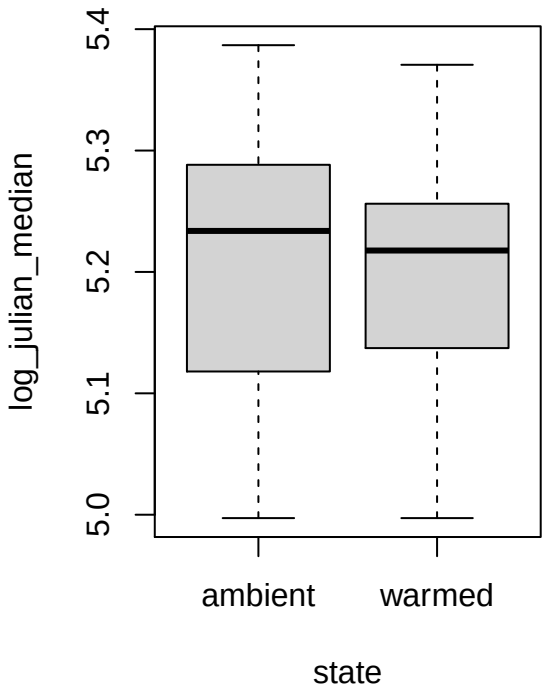

```
shapiro.test(resid(m1))  # not normally distributed resids bc p>0.05
```

```
##
##  Shapiro-Wilk normality test
##
## data:  resid(m1)
## W = 0.99448, p-value = 0.922
```

```
# Outliers sla[377,] sla[801,] Outlier test - yes, these are outliers bc p<0.05
outlierTest(m1)
```

```
## No Studentized residuals with Bonferroni p < 0.05
## Largest |rstudent|:
##    rstudent unadjusted p-value Bonferroni p
## 53 2.691335          0.0081977      0.98372
```

```
# Remove these 2 points sla1<-sla[-c(377, 801),] m1.nooutlier <-
# lmer(log(area_cm2) ~ state + species + year + insecticide + (1|plot), data =
# sla1, REML=FALSE) shapiro.test(resid(m1.nooutlier)) # normally distributed
# resids - good!

# (4) Normality of random effect: Get the estimate of random effect (e.g., random
# intercepts), and check them as you would check the residual. [***need to do for
# final model]

# Plot key relationships ```{r} Convert all columns to factor first
umbs_firstflwr_plot <- as.data.frame(unclass(umbs_firstflwr_plot), stringsAsFactors = TRUE)
plot(log_julian_median ~ state, data = umbs_firstflwr_plot)
```



Fit models

```
m0 <- lmer(log_julian_median ~ 1, data = umbs_firstflwr_plot)
```

```
## Error: No random effects terms specified in formula
```

```
m1 <- lmer(log_julian_median ~ state, data = umbs_firstflwr_plot)
```

```
## Error: No random effects terms specified in formula
```

```
m2 <- lmer(log_julian_median ~ state + insecticide, data = umbs_firstflwr_plot)
```

```
## Error: No random effects terms specified in formula
```

```
m3 <- lmer(log_julian_median ~ state * insecticide, data = umbs_firstflwr_plot)
```

```
## Error: No random effects terms specified in formula
```

```
m4 <- lmer(log_julian_median ~ state * insecticide + (1 | species), data = umbs_firstflwr_plot)
```

```
## Error in eval(predvars, data, env): object 'species' not found
```

```
m5 <- lmer(log_julian_median ~ state + insecticide + (1 | species), data = umbs_firstflwr_plot)
```

```
## Error in eval(predvars, data, env): object 'species' not found
```

```
m6 <- lmer(log_julian_median ~ state * insecticide + (1 | species) + (1 | year_factor),
    data = umbs_firstflwr_plot)
```

```
## Error in eval(predvars, data, env): object 'species' not found
```

```
m7 <- lmer(log_julian_median ~ state + insecticide + (1 | species) + (1 | year_factor),
    data = umbs_firstflwr_plot)
```

```
## Error in eval(predvars, data, env): object 'species' not found
```

```
m8 <- lmer(log_julian_median ~ state + insecticide + origin + (1 | species) + (1 |
    year_factor), data = umbs_firstflwr_plot)
```

```
## Error in eval(predvars, data, env): object 'origin' not found
```

```
m9 <- lmer(log_julian_median ~ state + insecticide + origin + (1 | species) + (1 |
    year_factor) + (1 | plot), data = umbs_firstflwr_plot)
```

```
## Error in eval(predvars, data, env): object 'origin' not found
```

Compare models

```
# AICtab(m0,m1,m2,m3,m4,m5, weights=TRUE)
```

Evaluate models using residuals