A Model to Connect Hummingbird Phenotypes to Flight Performance

Rosalee Elting

Table of contents

Introduction

For my project, I propose methods to understand the social dominance of hummingbirds and how social hierarchies are maintained. To evaluate this, I will use both field and lab work to create a model that reflects the phenotype > performance > fitness paradigm presented by Arnold in 1983 (Arnold, 1983). The actual variables I will measure are outlined below in the [###Measurement Variables] section. Overall, I will set these experiments up with dyadic combinations of trials between two male hummingbirds where a limiting resource is introduced (nectar or perches) to establish which individual maintains access to this resource and chases away an opponent. I will then connect these wins to traits about the individual that are variable. These will be phenotypic morphometric traits, as well as performance measurements from metabolic and flight kinematic measures. Lastly, with these characteristics I hope to understand how phenotypes and fitness connect to performance in an ecological landscape. I hope to do this by combining 1. fight data from the field and those winning individuals' phenotypes and 2. more fine-tuned phenotypic data and one-on-one fight results from lab trials. With this, I seek to establish what phenotypes and fitness performance traits are most predictable of social dominance hierarchies in hummingbirds.

Questions

Based on the above introduction, I ask the following two questions:

1. What variables in a complex suite of hummingbird morphological traits and flight maneuvers predict flight characteristics that are associated with increased speed and maneuverability?

Method

To investigate this, I hope to use the Arnold's Paradigm proposed in his 1983 paper. The framework below (Bergmann and McElroy, 2014) illustrates this experiment method graphically (below.)

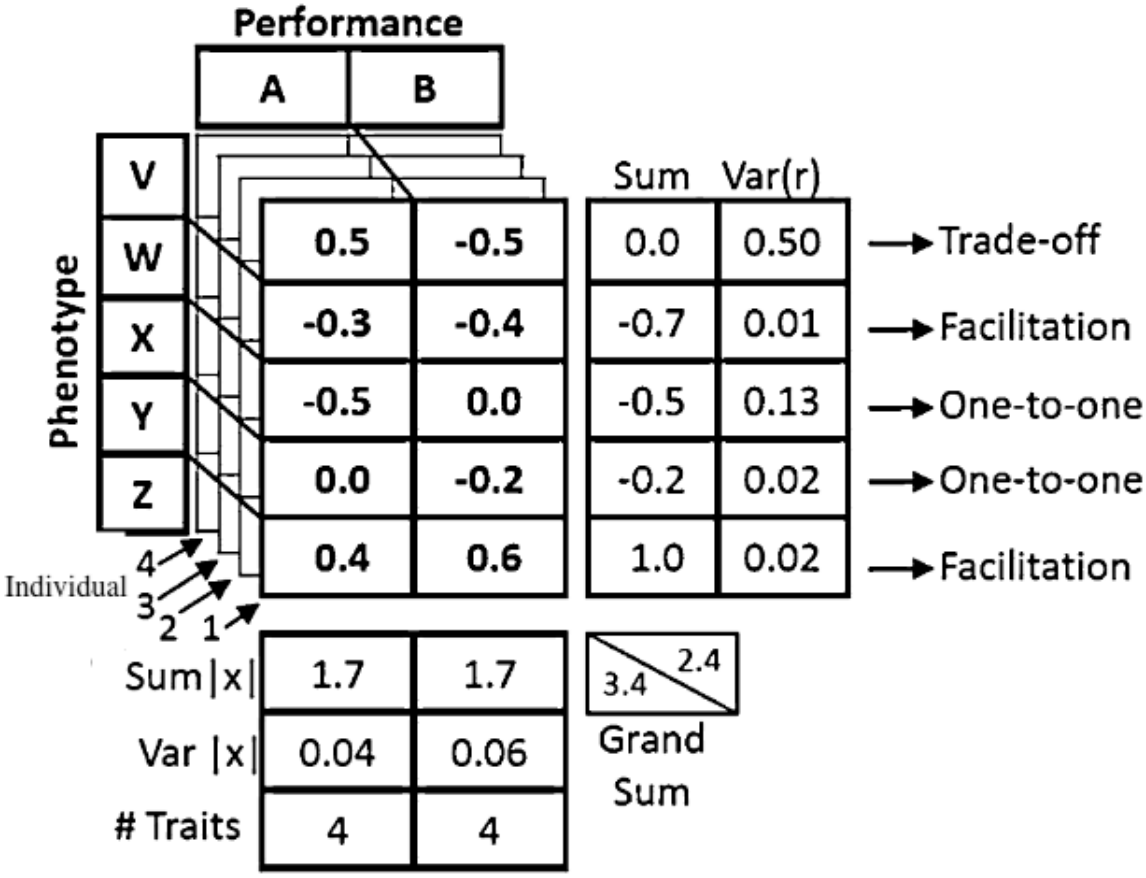


Figure 1: (Bergmann and McElroy, 2014)

Measurement Variables

I will be using a modified version of this Arnold framework in [Method] , but instead of comparing the phenotype-performance relationship of each species, I will be evaluating this on an individual level. My measurements, and their categories, will be:

Phenotypic Measures:

- Wing Aspect Ratio
- Body Mass
- Burst Muscle Capacity

Performance Measures:

- Flight accelerations and deceleration
- Arcing turn speed and radii
- Percentage of turns that are complex pitch-roll turns

Data and Model

I will be holding 6 Calliope Hummingbirds (*Selasphorus calliope*) captive in the spring of 2023. As I do not yet have data, I will find public repositories of data similar to that which I will collect. The data I am looking for will likely come from multiple sources and will be of two forms:

1. One-on-one dyadic fight results, which I can integrate into a Bayes framework to use as priors to predict winners of unique dyads. I would then like to integrate morphometrics to these predictors to see if they co-vary. Data from Márquez-Luna et al. 2022. provides dyadic win-loss data, but only as sums and it is interspecific, so I am hopeful to find another source with intraspecific competitions results.
2. I can use prior data collected by the Tobalske lab, potentially supplemented with other published data on Calliope hummingbirds to provide morphometric data. I can then hopefully find open source data on the fitness (metabolic rate and body composition) measures so that I can model the regressions between each phenotype and fitness combination in the matrix. If I have difficulty finding this data for Calliope hummingbirds, I will likely find it for Calypte anna instead, as it is more greatly studied, metabolic measures have already been published,and I have collaborators studying this species.

Public Repository Data for *Calypte anna*

Below is code I have written to explore data collected in Segre et al., 2015, an *eLife* publication exploring phenotypes and flight performance. The authors evaluated this data using an **information-theoretic approach** which compares the several possible scenarios and weights their relative predictive importance. The authors evaluated these data using a canned R function, nlme, therefore I would like to write a model to evaluate the same data and assess if I come to the same conclusions.

Exploring Data for *Calypte anna*

Loading Libraries and Data

First, I will load the necessary libraries

```
library(tidyverse)
```

```
-- Attaching packages --------------------------------------- tidyverse 1.3.2 --
v ggplot2 3.4.0     v purrr   1.0.1
v tibble  3.1.8     v dplyr   1.1.0
v tidyr   1.3.0     v stringr 1.5.0
v readr   2.1.3     v forcats 0.5.2
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
```

```
library(dplyr)
library(rstatix)
```

```
Attaching package: 'rstatix'

The following object is masked from 'package:stats':

    filter
```

```
library(ggpubr)
library(rstan)
```

```
Loading required package: StanHeaders
rstan (Version 2.21.8, GitRev: 2e1f913d3ca3)
For execution on a local, multicore CPU with excess RAM we recommend calling
options(mc.cores = parallel::detectCores()).
To avoid recompilation of unchanged Stan programs, we recommend calling
rstan_options(auto_write = TRUE)

Attaching package: 'rstan'

The following object is masked from 'package:tidyr':

    extract
```

```
library(tidybayes)
```

I will then load the csv of data from this publication

```
annas <- read_csv("annasairdata.csv")
```

```
Rows: 52 Columns: 40
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (3): bird, experiment, solocomp
dbl (37): vidid, capture_doy, trial_doy, daysafter, daysafterSTD, wing.lengt...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(annas)
```

```
# A tibble: 6 x 40
  bird   vidid experim~1 soloc~2 captu~3 trial~4 daysa~5 daysa~6 wing.~7 wing.ar
  <chr>  <dbl> <chr>     <chr>     <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
1 GAS01 143321 Van       solo         24      51      27  -0.756    51.6    7.13
2 GAS01 163920 Van       comp         24      51      27  -0.756    51.6    7.13
3 GAS02 150149 Van       solo        -20      56      76   0.447    52.9    7.83
4 GAS02 163920 Van       comp        -20      51      71   0.324    52.9    7.83
5 GAS03 102128 Van       solo         51      58       7  -1.25     55.4    7.59
6 GAS03  94818 Van       comp         51      61      10  -1.17     55.4    7.59
# ... with 30 more variables: mass <dbl>, w.lifted <dbl>, massSTD <dbl>,
```

```
#   wing.lengthSTD <dbl>, wing.arSTD <dbl>, w.liftedSTD <dbl>,
#   ss.total_vel <dbl>, ss.hor_acel <dbl>, ss.hor_decel <dbl>,
#   ss.pitch_down <dbl>, ss.pitch_up <dbl>, ss.vert_down_acel <dbl>,
#   ss.vert_up_acel <dbl>, ss.yaw <dbl>, ss.arc <dbl>, ss.pitch_roll <dbl>,
#   PRTpercent <dbl>, mean.total_vel <dbl>, mean.hor_acel <dbl>,
#   mean.hor_decel <dbl>, mean.pitch_down <dbl>, mean.pitch_up <dbl>, ...
```

I next am filtering and cleaning this data. I am removing data from solo trials, and keeping only competitive trials. The assumption is that birds perform flight maneuvers with increases speed or maneuverability in the presence of another individual. Therefore, I only include her data from competitive trials. Further I am selecting data I am interested in from this original data set.

In this code chunk, I also create a summary data frame with means and variances for each variable.

```
annas_clean <- annas %>%
  filter(solocomp == "comp") %>%
  select(c("mass", "massSTD", "w.lifted", "w.liftedSTD", "wing.ar", "wing.arSTD", "wing.le

annas_mean <-  as.data.frame(annas_clean) %>%
  summarise(across(colnames(annas_clean), ~mean(.x, na.rm=TRUE))) %>%
  pivot_longer(cols = colnames(annas_clean)) %>%
  select(-c("name"))
annas_var <- as.data.frame(annas_clean) %>%
  summarise(across(colnames(annas_clean), ~var(.x, na.rm = TRUE))) %>%
  pivot_longer(cols = colnames(annas_clean)) %>%
  select(-c("name"))

annas_summ <- data.frame(matrix( NA,
                                 ncol = 3,
                                 nrow = 17))

names <- data.frame(rownames = colnames(annas_clean))

annas_summ[,1] <- names
annas_summ[,2] <- annas_mean
annas_summ[,3] <- annas_var
names(annas_summ) <- c("variable", "mean", "var")
```

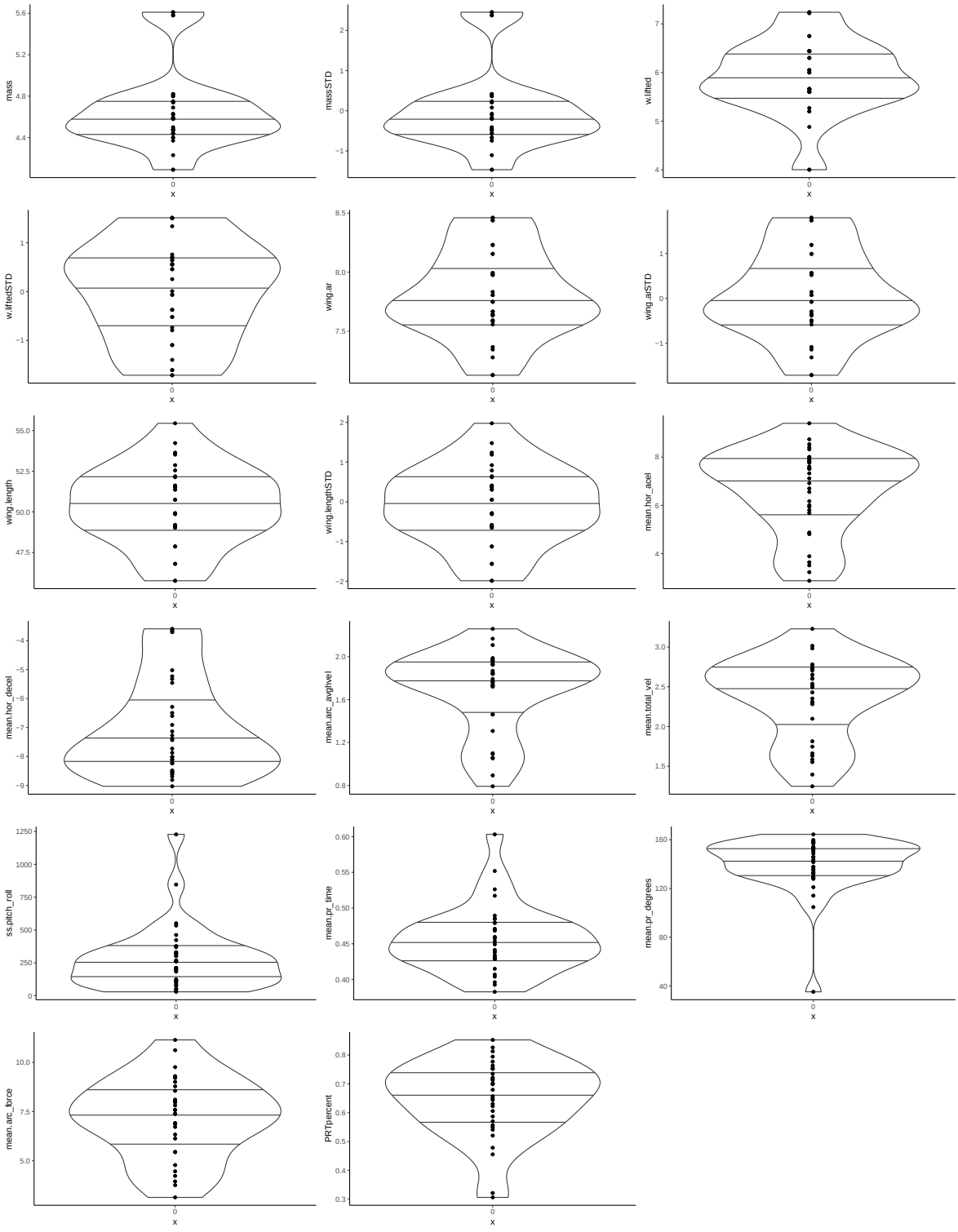
Visualizing Original Data

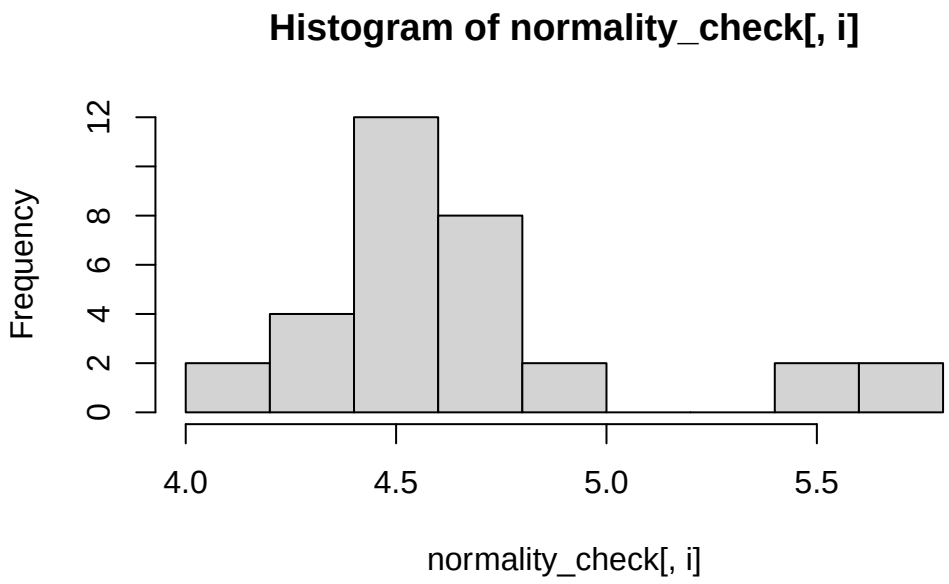
```
x <- as.factor(0)
 viz <- annas_clean %>%
   cross_join(x, copy = TRUE)

 colNames <- names(viz)[1:17]

 for (i in colNames) {
   plot <- ggplot(viz, aes_string(x= viz$y, y = i))+
     geom_violin(draw_quantiles = c(0.25, 0.5, 0.75))+
     geom_point() +
     theme_classic()
  print(plot)
  Sys.sleep(2)

 }
```

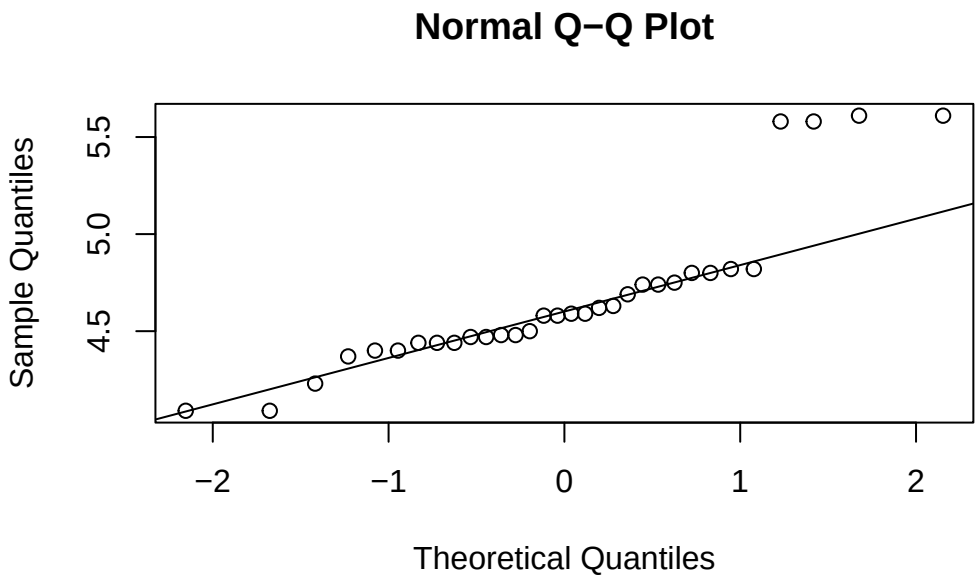
```
Warning: `aes_string()` was deprecated in ggplot2 3.0.0.
i Please use tidy evaluation ideoms with `aes()`
```

Investigating Distributions of Original Data

```
normality_check <- as.data.frame(annas_clean)
columns <- names(normality_check)

for (i in columns) {
  plot <- hist(x = normality_check[,i])
  plot2 <- qqnorm(normality_check[,i])
    qqline(normality_check[,i])
  print(plot)
  print(plot2)
}
```

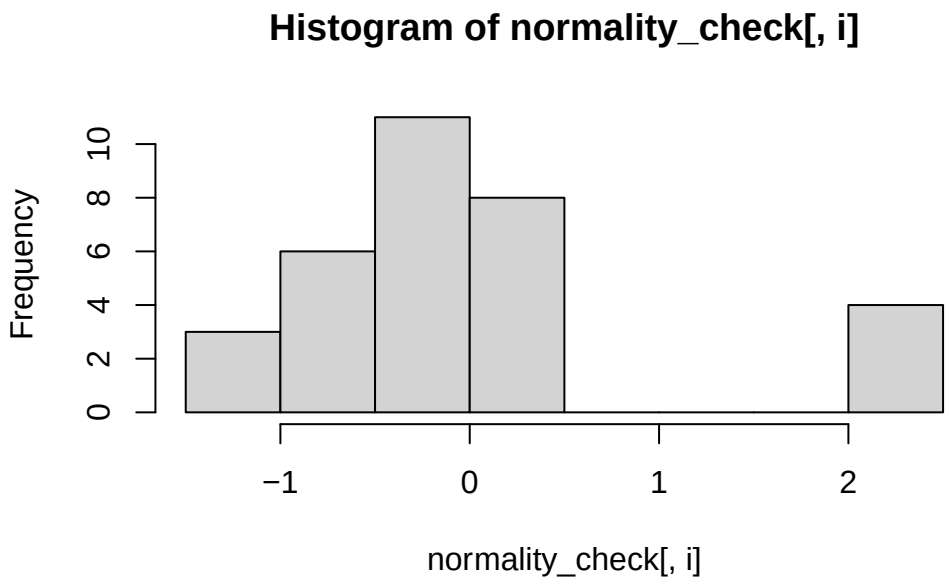

mass
massSTD
w.lifted
w.liftedSTD
wing.ar
wing.arSTD
wing.length
wing.lengthSTD
mean.hor_aoel
mean.hor_decel
mean.arc_avghvel
mean.total_vel
ss.pitch_roll
mean.pr_time
mean.pr_degrees
mean.arc_force
PRTpercent
x

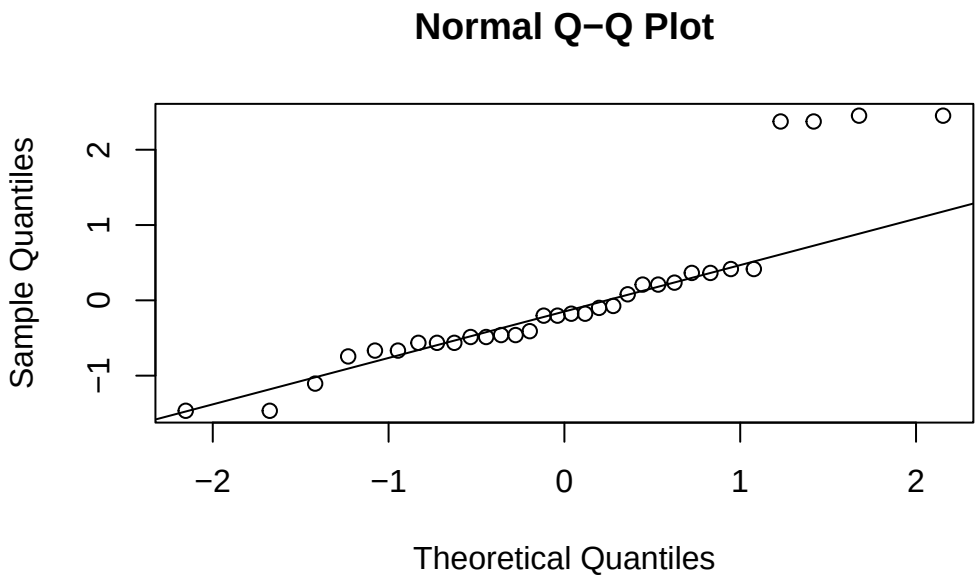





```
$breaks
 [1] 4.0 4.2 4.4 4.6 4.8 5.0 5.2 5.4 5.6 5.8
```

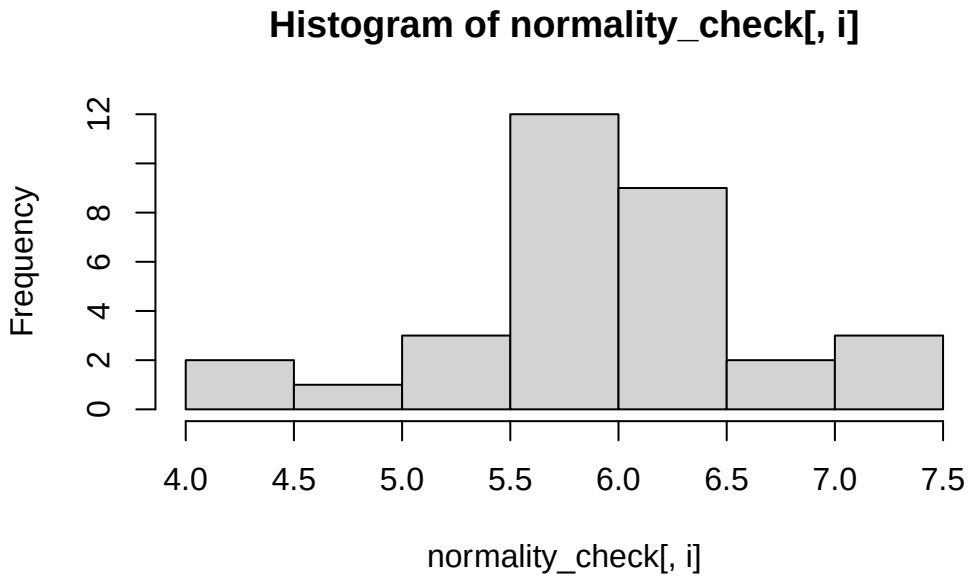
```
$counts
[1]  2  4 12  8  2  0  0  2  2

$density
[1] 0.3125 0.6250 1.8750 1.2500 0.3125 0.0000 0.0000 0.3125 0.3125

$mids
[1] 4.1 4.3 4.5 4.7 4.9 5.1 5.3 5.5 5.7

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -0.19709908 -1.22985876  0.19709908  0.27769044 -1.41779714  0.62609901
 [7] -0.83051088  0.36012989  0.72451438  0.83051088 -0.53340971 -0.44509652
[13] -0.72451438 -0.62609901  0.44509652  0.53340971 -1.07751557 -0.94678176
[19] -0.36012989 -0.27769044 -2.15387469 -1.67593972  0.94678176  1.07751557
[25]  1.22985876  1.41779714 -0.11776987 -0.03917609  0.03917609  0.11776987
[31]  1.67593972  2.15387469

$y
 [1] 4.50 4.37 4.62 4.63 4.23 4.75 4.44 4.69 4.80 4.80 4.47 4.47 4.44 4.44 4.74
[16] 4.74 4.40 4.40 4.48 4.48 4.09 4.09 4.82 4.82 5.58 5.58 4.58 4.58 4.59 4.59
[31] 5.61 5.61
```

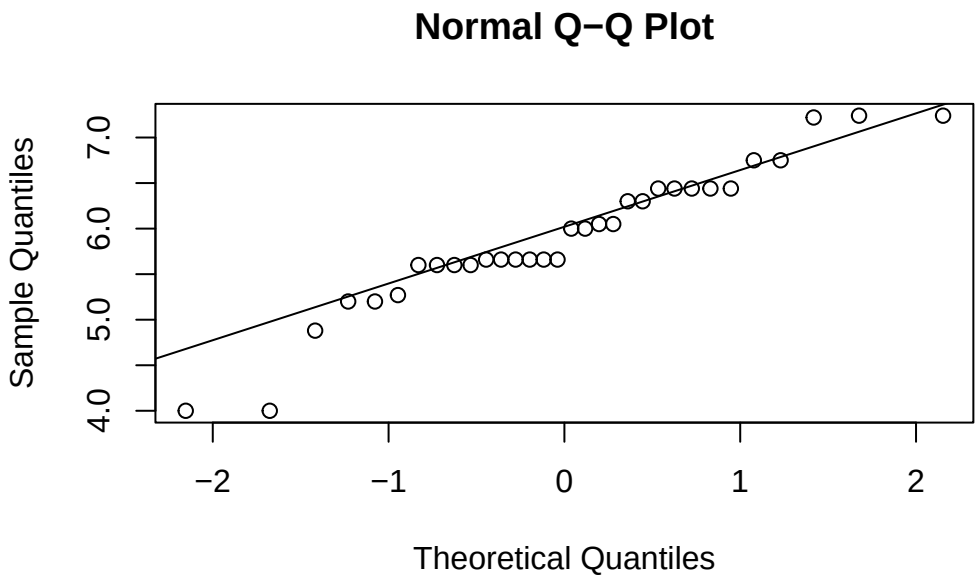





```
$breaks
[1] -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5
```

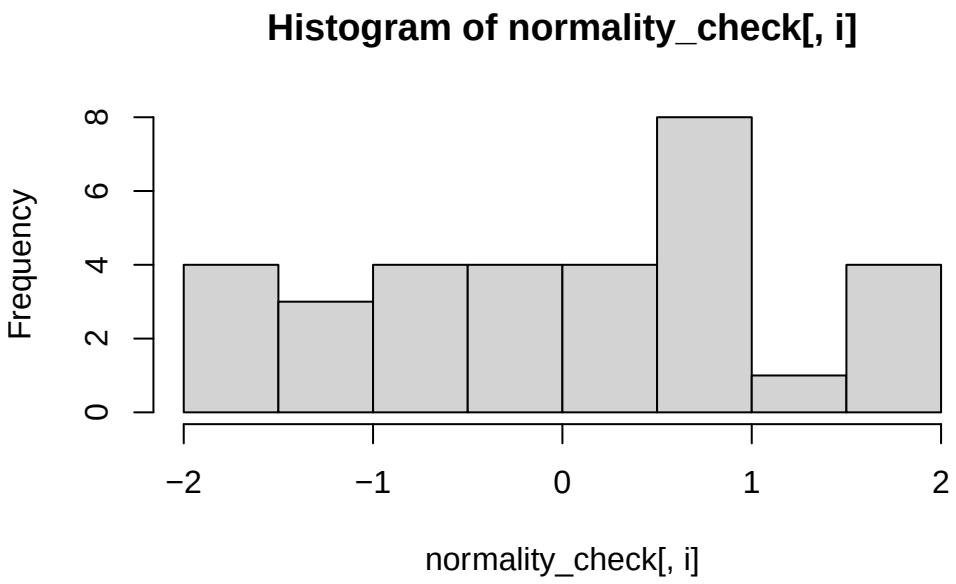
```
$counts
[1]  3  6 11  8  0  0  0  4

$density
[1] 0.1875 0.3750 0.6875 0.5000 0.0000 0.0000 0.0000 0.2500

$mids
[1] -1.25 -0.75 -0.25  0.25  0.75  1.25  1.75  2.25

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -0.19709908 -1.22985876  0.19709908  0.27769044 -1.41779714  0.62609901
 [7] -0.83051088  0.36012989  0.72451438  0.83051088 -0.53340971 -0.44509652
[13] -0.72451438 -0.62609901  0.44509652  0.53340971 -1.07751557 -0.94678176
[19] -0.36012989 -0.27769044 -2.15387469 -1.67593972  0.94678176  1.07751557
[25]  1.22985876  1.41779714 -0.11776987 -0.03917609  0.03917609  0.11776987
[31]  1.67593972  2.15387469

$y
 [1] -0.40953637 -0.74470172 -0.10015296 -0.07437101 -1.10564903  0.23501239
 [7] -0.56422807  0.08032069  0.36392215  0.36392215 -0.48688222 -0.48688222
[13] -0.56422807 -0.56422807  0.20923044  0.20923044 -0.66735587 -0.66735587
[19] -0.46110027 -0.46110027 -1.46659633 -1.46659633  0.41548605  0.41548605
[25]  2.37491428  2.37491428 -0.20328076 -0.20328076 -0.17749881 -0.17749881
[31]  2.45226013  2.45226013
```

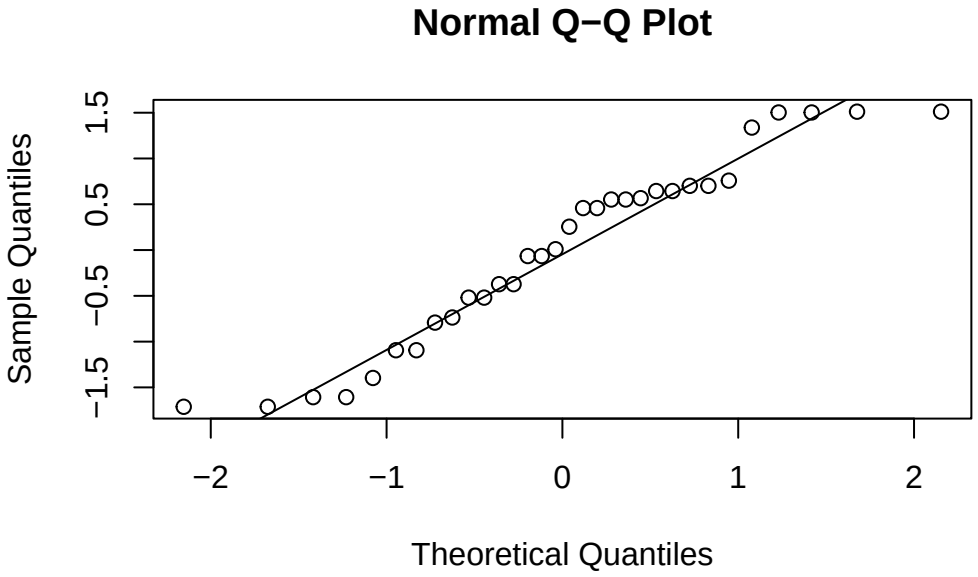





```
$breaks
[1] 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5
```

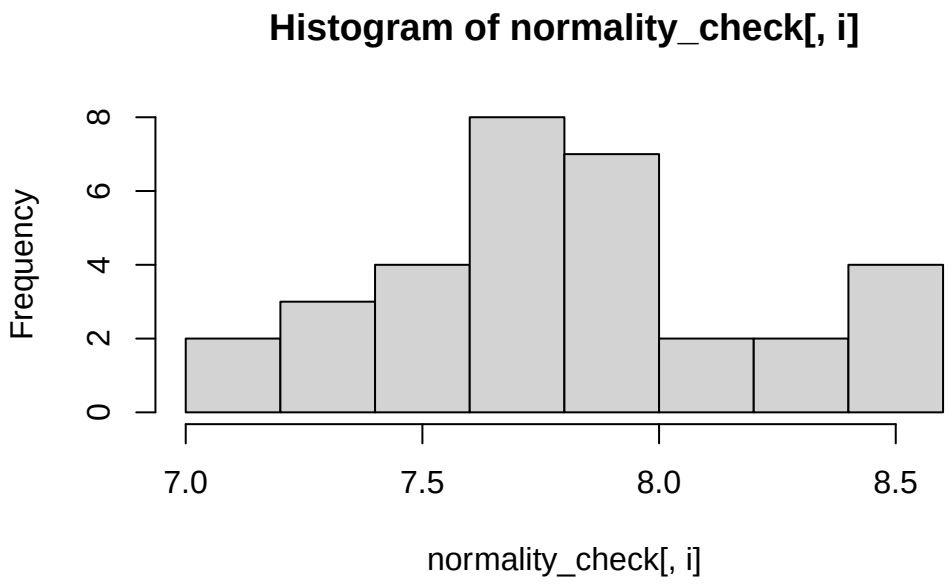
```
$counts
[1]  2  1  3 12  9  2  3

$density
[1] 0.1250 0.0625 0.1875 0.7500 0.5625 0.1250 0.1875

$mids
[1] 4.25 4.75 5.25 5.75 6.25 6.75 7.25

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1]  1.41779714 -1.41779714 -0.44509652 -0.36012989  0.53340971 -0.94678176
 [7]  0.62609901  0.72451438 -1.22985876 -1.07751557 -0.83051088 -0.72451438
[13] -2.15387469 -1.67593972 -0.62609901 -0.53340971  0.19709908  0.27769044
[19] -0.27769044 -0.19709908  0.83051088  0.94678176  1.67593972  2.15387469
[25] -0.11776987 -0.03917609  1.07751557  1.22985876  0.36012989  0.44509652
[31]  0.03917609  0.11776987

$y
 [1] 7.22 4.88 5.66 5.66 6.44 5.27 6.44 6.44 5.20 5.20 5.60 5.60 4.00 4.00 5.60
[16] 5.60 6.05 6.05 5.66 5.66 6.44 6.44 7.24 7.24 5.66 5.66 6.75 6.75 6.30 6.30
[31] 6.00 6.00
```

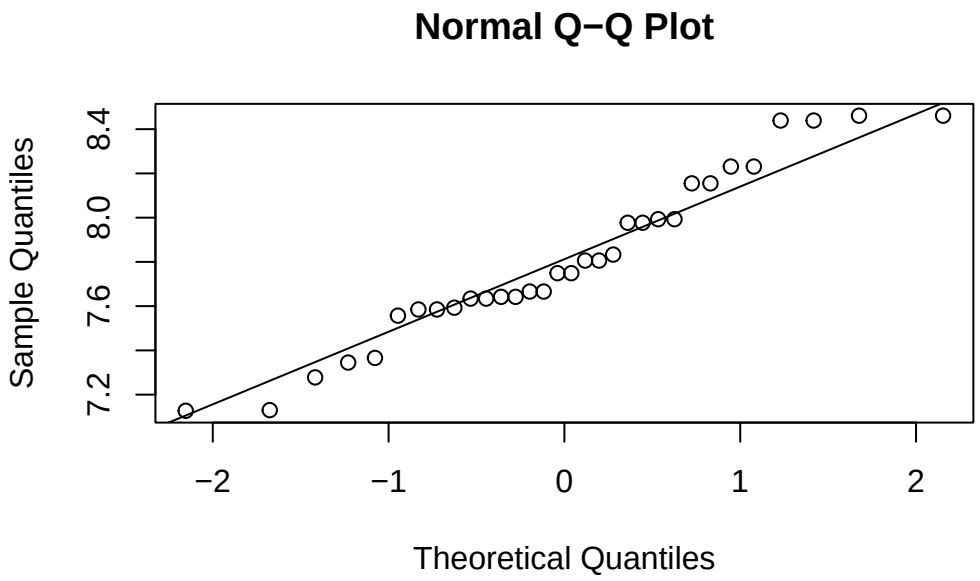





```
$breaks
[1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0
```

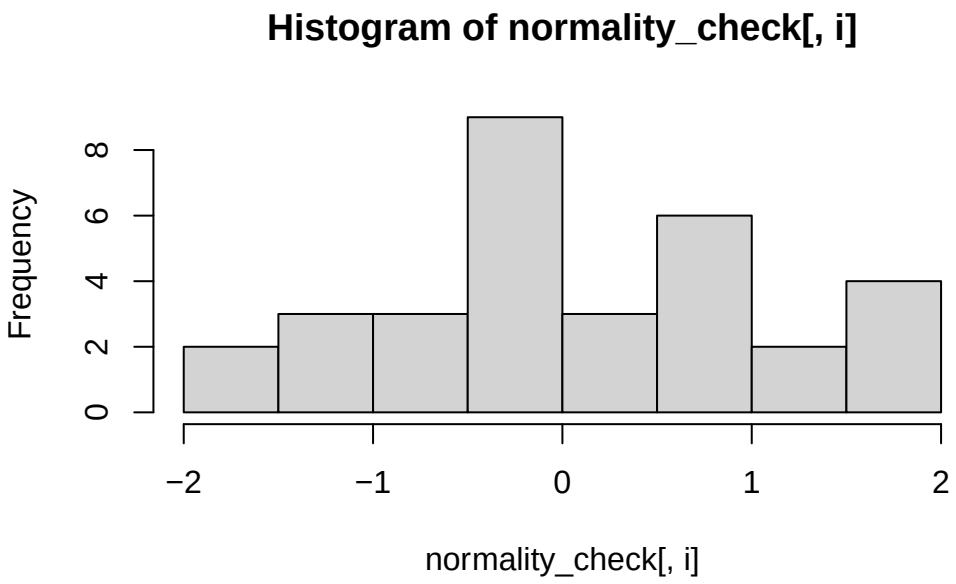
```
$counts
[1] 4 3 4 4 4 8 1 4

$density
[1] 0.2500 0.1875 0.2500 0.2500 0.2500 0.5000 0.0625 0.2500

$mids
[1] -1.75 -1.25 -0.75 -0.25  0.25  0.75  1.25  1.75

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1]  1.07751557 -0.72451438 -0.03917609 -1.07751557  0.03917609 -0.62609901
 [7]  0.44509652  0.94678176  0.72451438  0.83051088  0.27769044  0.36012989
[13] -2.15387469 -1.67593972  0.11776987  0.19709908 -0.94678176 -0.83051088
[19] -1.41779714 -1.22985876  0.53340971  0.62609901  1.67593972  2.15387469
[25] -0.53340971 -0.44509652  1.22985876  1.41779714 -0.36012989 -0.27769044
[31] -0.19709908 -0.11776987

$y
 [1]  1.33771804 -0.79240898  0.00912714 -1.39696563  0.25468224 -0.73668168
 [7]  0.56631868  0.75821018  0.70125357  0.70125357  0.55201105  0.55201105
[13] -1.71174522 -1.71174522  0.45848060  0.45848060 -1.09417237 -1.09417237
[19] -1.60648094 -1.60648094  0.64463562  0.64463562  1.51133370  1.51133370
[25] -0.51954029 -0.51954029  1.50251862  1.50251862 -0.37287335 -0.37287335
[31] -0.06542098 -0.06542098
```

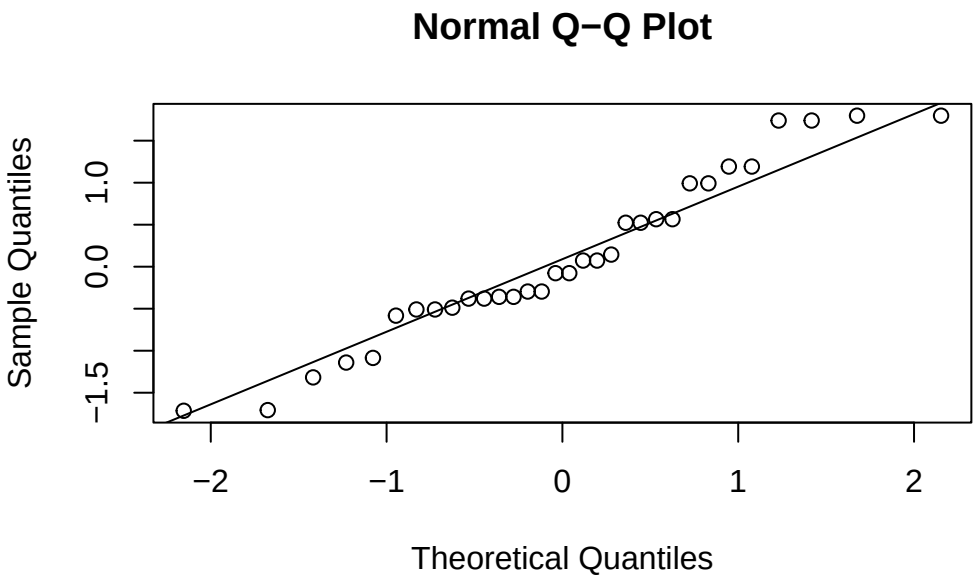





```
$breaks
[1] 7.0 7.2 7.4 7.6 7.8 8.0 8.2 8.4 8.6
```

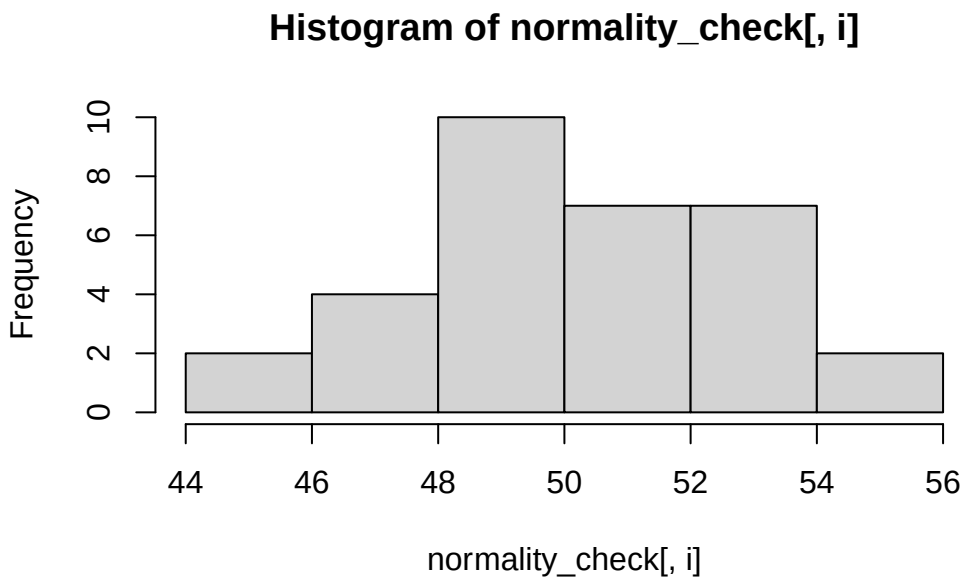
```
$counts
[1] 2 3 4 8 7 2 2 4

$density
[1] 0.31250 0.46875 0.62500 1.25000 1.09375 0.31250 0.31250 0.62500

$mids
[1] 7.1 7.3 7.5 7.7 7.9 8.1 8.3 8.5

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -2.15387469  0.27769044 -0.62609901 -1.67593972 -1.41779714 -0.94678176
 [7] -1.22985876 -1.07751557  1.67593972  2.15387469  0.94678176  1.07751557
[13]  1.22985876  1.41779714  0.72451438  0.83051088 -0.83051088 -0.72451438
[19] -0.53340971 -0.44509652  0.36012989  0.44509652 -0.19709908 -0.11776987
[25]  0.11776987  0.19709908  0.53340971  0.62609901 -0.36012989 -0.27769044
[31] -0.03917609  0.03917609

$y
 [1] 7.127 7.833 7.593 7.130 7.278 7.557 7.345 7.366 8.461 8.461 8.231 8.231
[13] 8.439 8.439 8.155 8.155 7.585 7.585 7.634 7.634 7.977 7.977 7.666 7.666
[25] 7.806 7.806 7.993 7.993 7.642 7.642 7.749 7.749
```

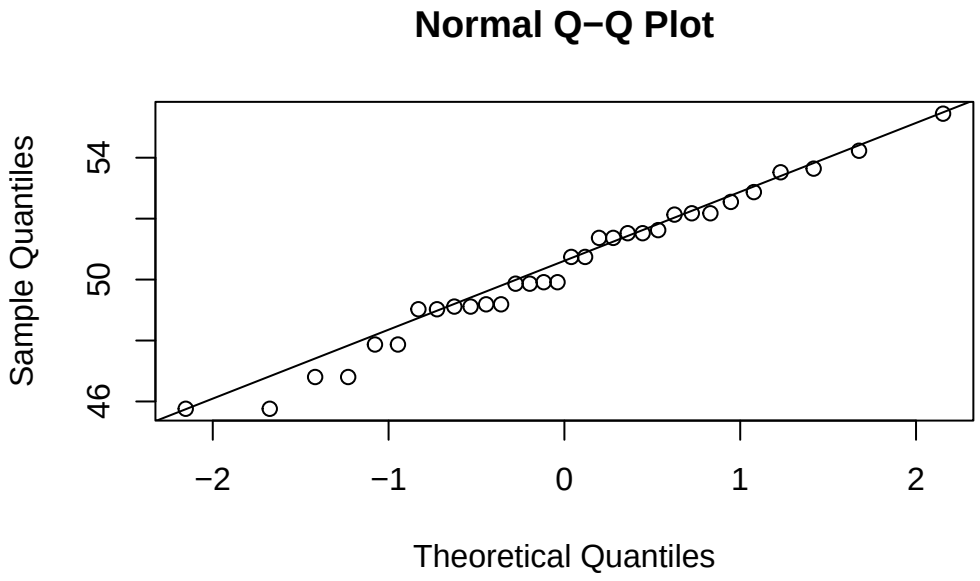





```
$breaks
[1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0
```

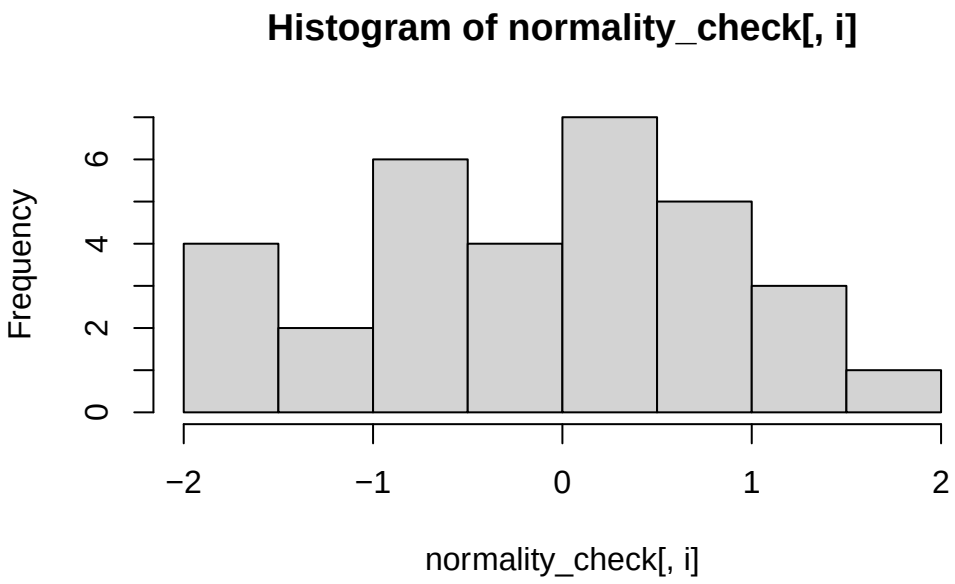
```
$counts
[1] 2 3 3 9 3 6 2 4

$density
[1] 0.1250 0.1875 0.1875 0.5625 0.1875 0.3750 0.1250 0.2500

$mids
[1] -1.75 -1.25 -0.75 -0.25  0.25  0.75  1.25  1.75

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -2.15387469  0.27769044 -0.62609901 -1.67593972 -1.41779714 -0.94678176
 [7] -1.22985876 -1.07751557  1.67593972  2.15387469  0.94678176  1.07751557
[13]  1.22985876  1.41779714  0.72451438  0.83051088 -0.83051088 -0.72451438
[19] -0.53340971 -0.44509652  0.36012989  0.44509652 -0.19709908 -0.11776987
[25]  0.11776987  0.19709908  0.53340971  0.62609901 -0.36012989 -0.27769044
[31] -0.03917609  0.03917609

$y
 [1] -1.71497799  0.14401197 -0.48793929 -1.70707860 -1.31737532 -0.58273198
 [7] -1.14095560 -1.08565986  1.79761777  1.79761777  1.19199781  1.19199781
[13]  1.73968891  1.73968891  0.99187991  0.99187991 -0.50900433 -0.50900433
[19] -0.37998095 -0.37998095  0.52318273  0.52318273 -0.29572078 -0.29572078
[25]  0.07291745  0.07291745  0.56531281  0.56531281 -0.35891591 -0.35891591
[31] -0.07717097 -0.07717097
```

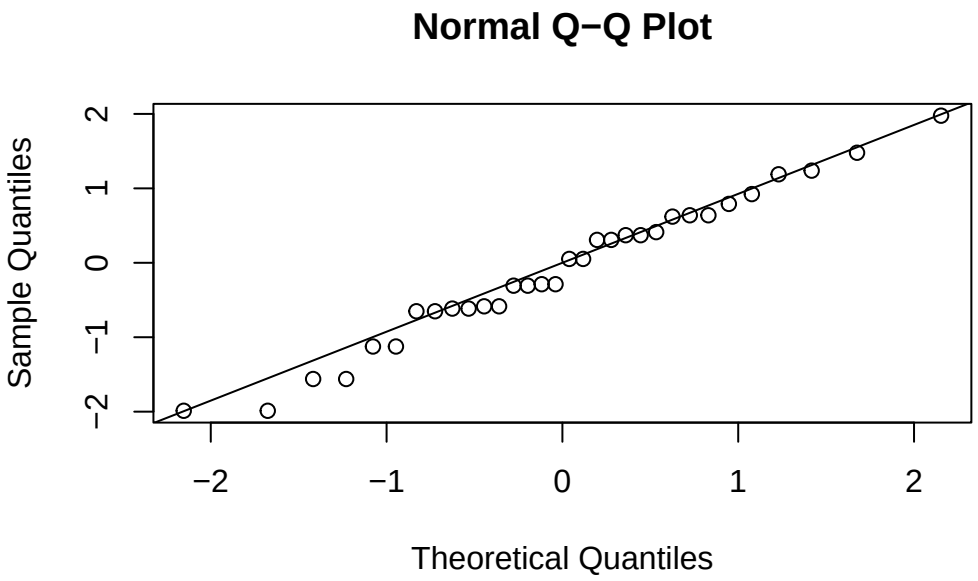





```
$breaks
[1] 44 46 48 50 52 54 56
```

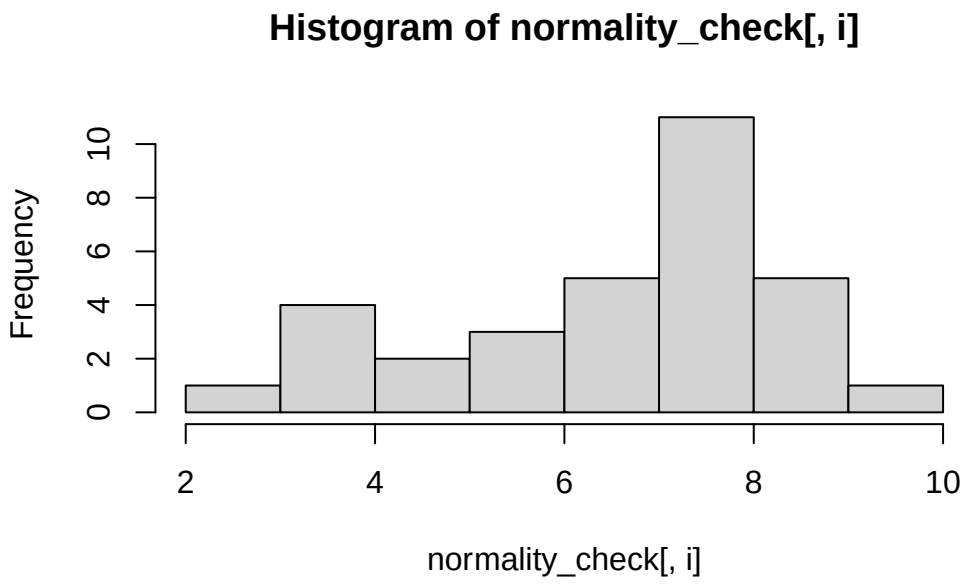
```
$counts
[1]  2  4 10  7  7  2

$density
[1] 0.031250 0.062500 0.156250 0.109375 0.109375 0.031250

$mids
[1] 45 47 49 51 53 55

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1]  0.53340971  1.07751557  2.15387469  1.67593972  0.94678176  1.41779714
 [7]  0.62609901  1.22985876  0.72451438  0.83051088  0.36012989  0.44509652
[13] -0.44509652 -0.36012989  0.03917609  0.11776987 -0.83051088 -0.72451438
[19] -0.11776987 -0.03917609 -2.15387469 -1.67593972  0.19709908  0.27769044
[25] -0.62609901 -0.53340971 -0.27769044 -0.19709908 -1.07751557 -0.94678176
[31] -1.41779714 -1.22985876

$y
 [1] 51.623 52.871 55.445 54.228 52.549 53.641 52.132 53.521 52.176 52.176
[11] 51.522 51.522 49.188 49.188 50.743 50.743 49.026 49.026 49.914 49.914
[21] 45.760 45.760 51.368 51.368 49.114 49.114 49.866 49.866 47.869 47.869
[31] 46.803 46.803
```

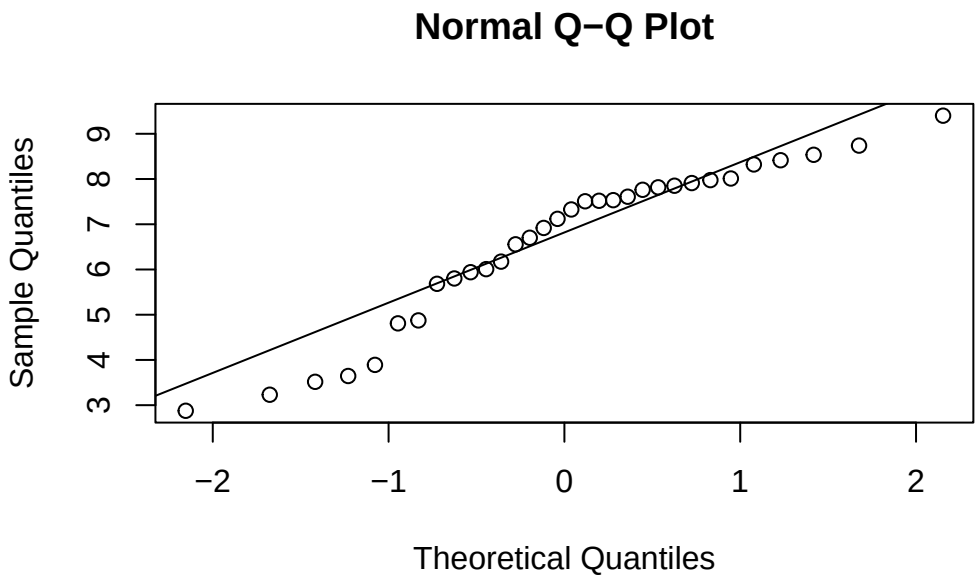





```
$breaks
[1] -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0
```

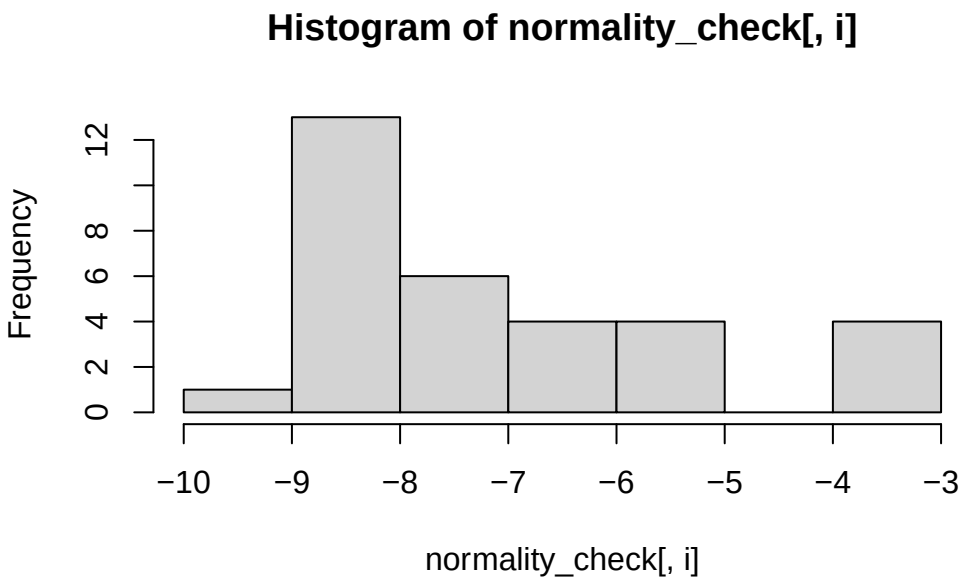
```
$counts
[1] 4 2 6 4 7 5 3 1

$density
[1] 0.2500 0.1250 0.3750 0.2500 0.4375 0.3125 0.1875 0.0625

$mids
[1] -1.75 -1.25 -0.75 -0.25  0.25  0.75  1.25  1.75

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1]  0.53340971  1.07751557  2.15387469  1.67593972  0.94678176  1.41779714
 [7]  0.62609901  1.22985876  0.72451438  0.83051088  0.36012989  0.44509652
[13] -0.44509652 -0.36012989  0.03917609  0.11776987 -0.83051088 -0.72451438
[19] -0.11776987 -0.03917609 -2.15387469 -1.67593972  0.19709908  0.27769044
[25] -0.62609901 -0.53340971 -0.27769044 -0.19709908 -1.07751557 -0.94678176
[31] -1.41779714 -1.22985876

$y
 [1]  0.41196395  0.92286254  1.97659088  1.47838288  0.79104415  1.23808042
 [7]  0.62033525  1.18895555  0.63834770  0.63834770  0.37061719  0.37061719
[13] -0.58486143 -0.58486143  0.05171494  0.05171494 -0.65117999 -0.65117999
[19] -0.28765600 -0.28765600 -1.98819506 -1.98819506  0.30757361  0.30757361
[25] -0.61515509 -0.61515509 -0.30730594 -0.30730594 -1.12482556 -1.12482556
[31] -1.56121811 -1.56121811
```

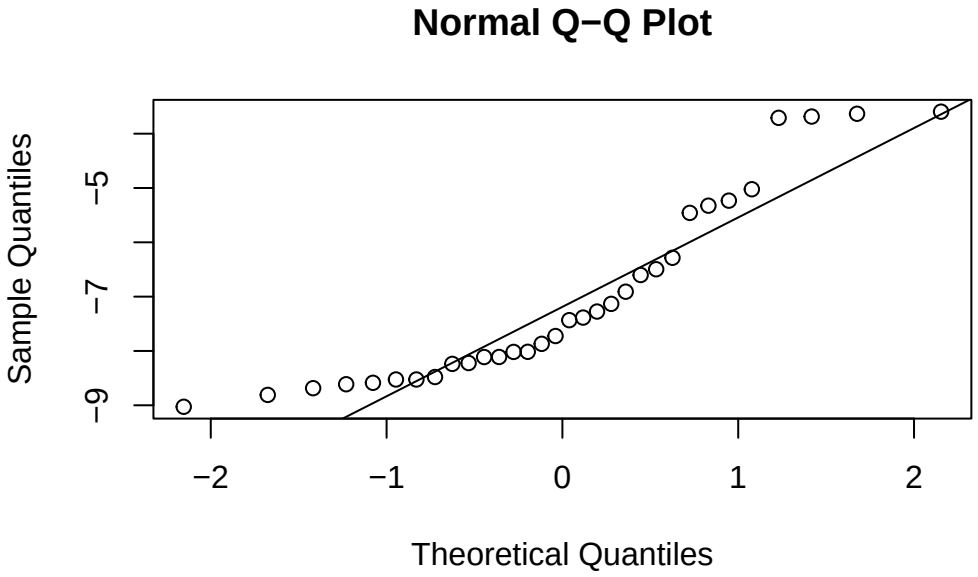





```
$breaks
[1]  2  3  4  5  6  7  8  9 10
```

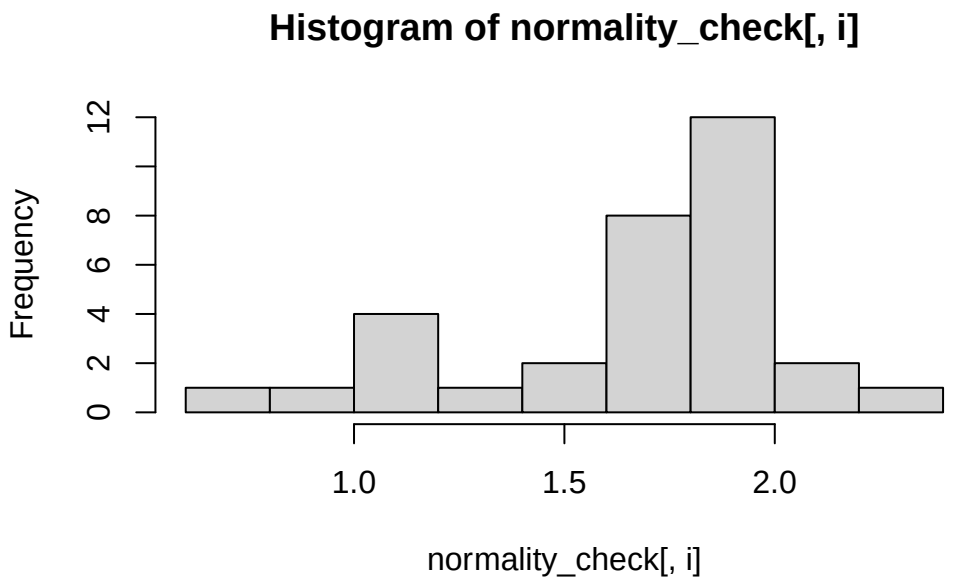
```
$counts
[1]  1  4  2  3  5 11  5  1

$density
[1] 0.03125 0.12500 0.06250 0.09375 0.15625 0.34375 0.15625 0.03125

$mids
[1] 2.5 3.5 4.5 5.5 6.5 7.5 8.5 9.5

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -1.07751557 -1.67593972 -0.94678176 -1.22985876 -2.15387469 -1.41779714
 [7] -0.72451438 -0.83051088  0.94678176  0.72451438  0.11776987  1.67593972
[13] -0.36012989  0.19709908  0.03917609  0.36012989  0.62609901 -0.11776987
[19] -0.53340971 -0.44509652 -0.19709908 -0.62609901  1.07751557  0.27769044
[25] -0.03917609  0.53340971  1.41779714  2.15387469  0.44509652  1.22985876
[31] -0.27769044  0.83051088

$y
 [1] 3.890672 3.231285 4.808738 3.645144 2.875872 3.516435 5.683158 4.874083
 [9] 8.011522 7.910500 7.507671 8.738960 6.173588 7.518970 7.326716 7.608495
[17] 7.851499 6.918311 5.939729 6.007211 6.701496 5.801163 8.321902 7.532586
[25] 7.119299 7.816501 8.535760 9.400748 7.761859 8.415416 6.556105 7.975860
```

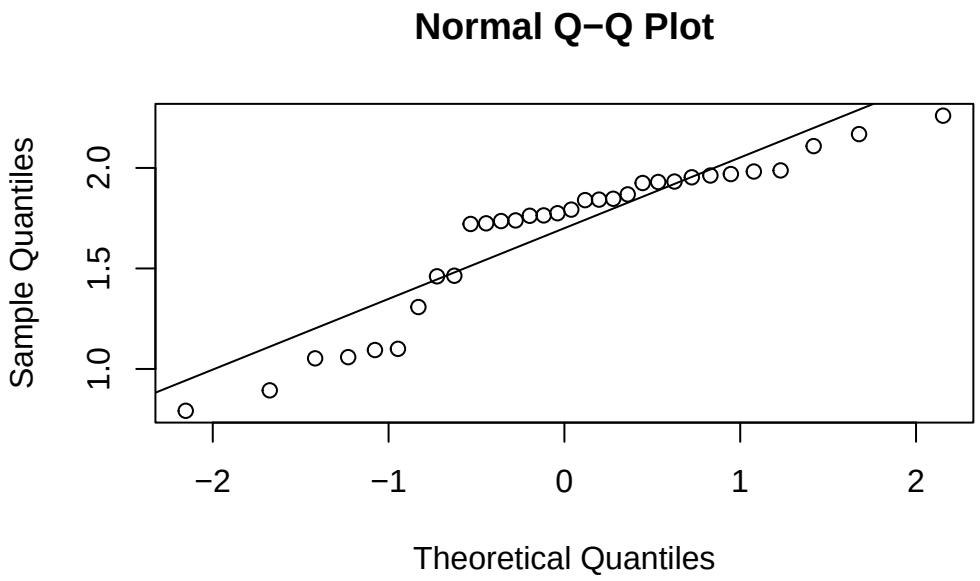





```
$breaks
[1] -10  -9  -8  -7  -6  -5  -4  -3
```

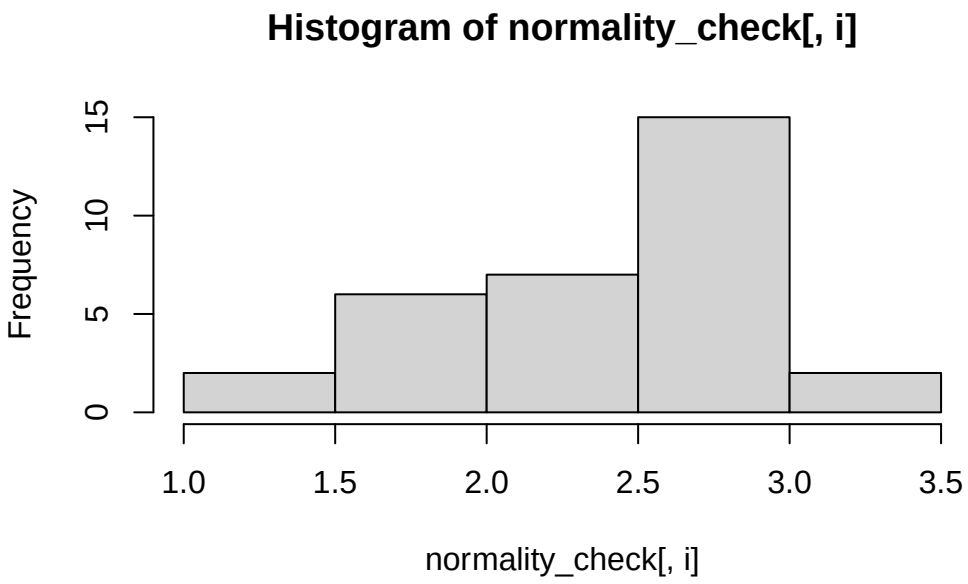
```
$counts
[1]  1 13  6  4  4  0  4

$density
[1] 0.03125 0.40625 0.18750 0.12500 0.12500 0.00000 0.12500

$mids
[1] -9.5 -8.5 -7.5 -6.5 -5.5 -4.5 -3.5

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1]  1.07751557  1.67593972  0.94678176  1.41779714  1.22985876  2.15387469
 [7]  0.83051088  0.72451438 -0.83051088 -0.62609901  0.11776987 -0.72451438
[13]  0.36012989 -0.44509652 -0.27769044 -0.03917609 -0.19709908  0.19709908
[19]  0.62609901  0.53340971  0.27769044  0.44509652 -1.67593972 -0.11776987
[25] -0.53340971 -1.22985876 -1.41779714 -2.15387469 -0.36012989 -1.07751557
[31]  0.03917609 -0.94678176

$y
 [1] -5.023197 -3.633288 -5.232813 -3.684881 -3.708440 -3.594767 -5.325482
 [8] -5.457998 -8.525116 -8.238200 -7.385437 -8.479650 -6.909063 -8.111643
[15] -8.017739 -7.725070 -8.015539 -7.274615 -6.284718 -6.495897 -7.132975
[22] -6.601818 -8.808847 -7.868638 -8.222534 -8.612609 -8.685142 -9.027618
[29] -8.110864 -8.586173 -7.432712 -8.525387
```

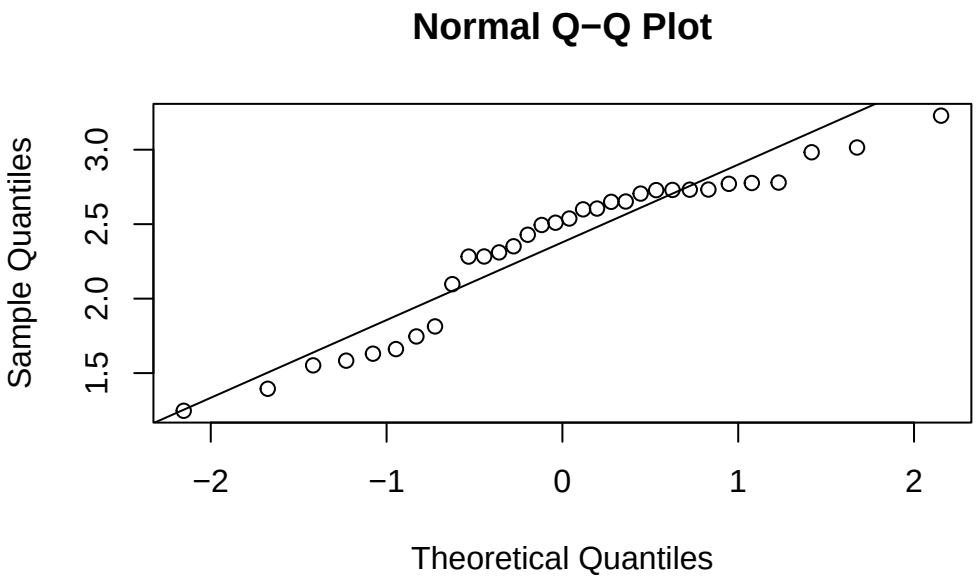





```
$breaks
 [1] 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0 2.2 2.4
```

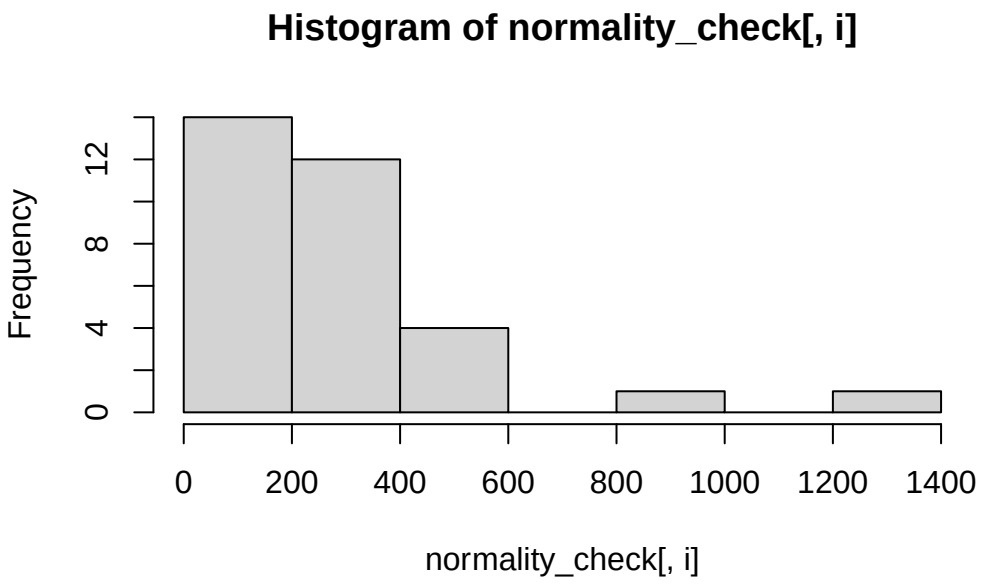
```
$counts
[1]  1  1  4  1  2  8 12  2  1

$density
[1] 0.15625 0.15625 0.62500 0.15625 0.31250 1.25000 1.87500 0.31250 0.15625

$mids
[1] 0.7 0.9 1.1 1.3 1.5 1.7 1.9 2.1 2.3

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -0.83051088 -1.67593972 -1.41779714 -2.15387469 -0.94678176 -1.22985876
 [7] -0.62609901 -1.07751557  0.83051088  1.22985876  0.03917609  0.36012989
[13] -0.53340971 -0.19709908 -0.36012989  0.11776987  1.07751557  0.19709908
[19] -0.44509652 -0.27769044  0.53340971 -0.11776987  0.72451438  1.41779714
[25] -0.72451438  0.94678176  1.67593972  2.15387469  0.27769044  0.62609901
[31] -0.03917609  0.44509652

$y
 [1] 1.3081290 0.8936953 1.0528009 0.7920078 1.1001789 1.0588378 1.4638829
 [8] 1.0942594 1.9624332 1.9877641 1.7927165 1.8685049 1.7213091 1.7620721
[15] 1.7358641 1.8399743 1.9822478 1.8425181 1.7245821 1.7389694 1.9302044
[22] 1.7639578 1.9534537 2.1086957 1.4609616 1.9696228 2.1684372 2.2601229
[29] 1.8467551 1.9323678 1.7748309 1.9251616
```

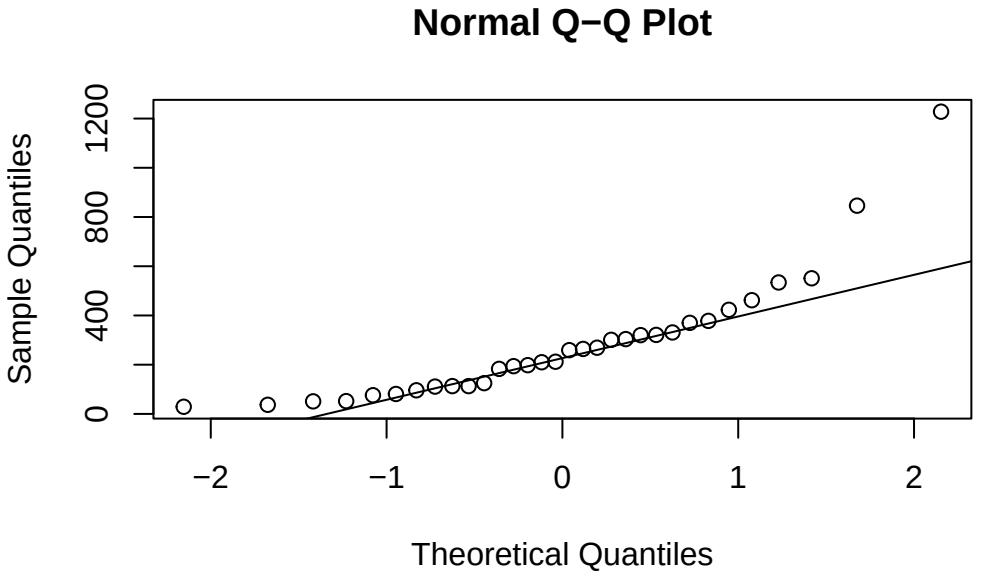





```
$breaks
[1] 1.0 1.5 2.0 2.5 3.0 3.5
```

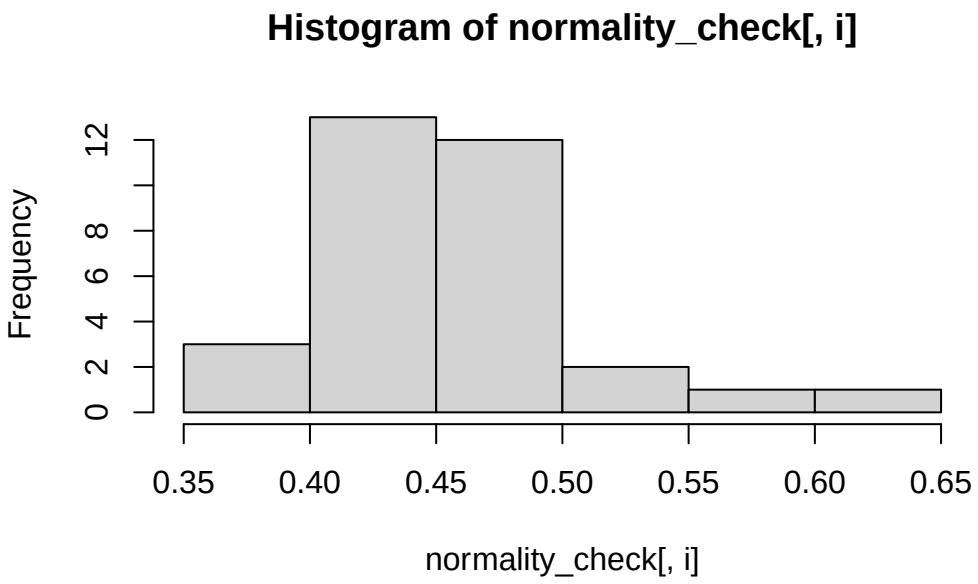
```
$counts
[1]  2  6  7 15  2

$density
[1] 0.1250 0.3750 0.4375 0.9375 0.1250

$mids
[1] 1.25 1.75 2.25 2.75 3.25

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -1.07751557 -1.67593972 -1.22985876 -0.94678176 -2.15387469 -1.41779714
 [7] -0.72451438 -0.83051088  1.67593972  0.53340971 -0.44509652  0.72451438
[13]  0.94678176  0.83051088  2.15387469  0.11776987 -0.19709908 -0.36012989
[19] -0.53340971 -0.62609901  0.27769044 -0.03917609  0.44509652  0.36012989
[25]  0.03917609  1.07751557  1.22985876  1.41779714 -0.27769044  0.62609901
[31] -0.11776987  0.19709908

$y
 [1] 1.630470 1.394872 1.583581 1.661268 1.247033 1.551994 1.813380 1.746114
 [9] 3.014889 2.728564 2.283000 2.731522 2.770766 2.732418 3.228468 2.599488
[17] 2.428925 2.310493 2.282409 2.097170 2.650000 2.509762 2.705257 2.652035
[25] 2.537788 2.776236 2.779197 2.982842 2.351355 2.729855 2.494523 2.604846
```

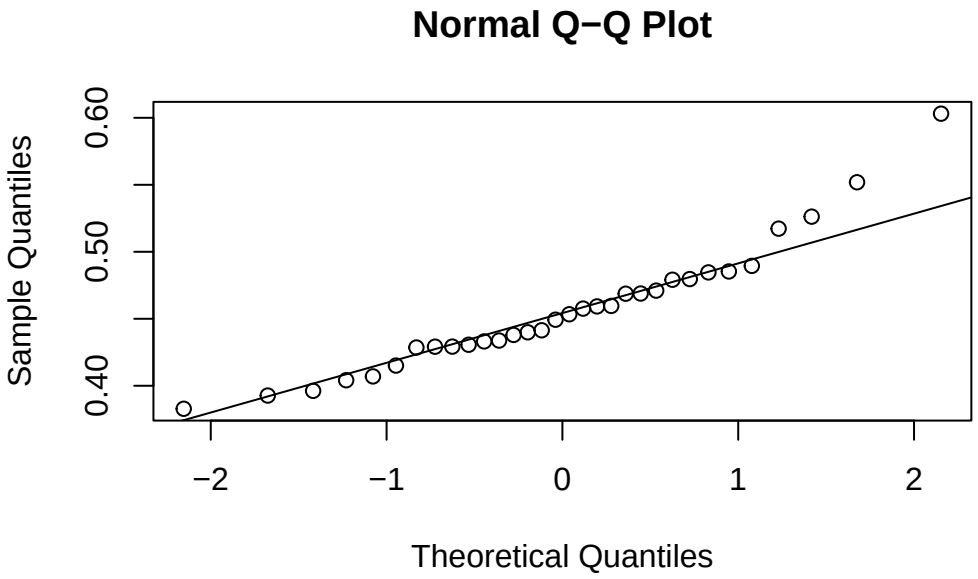





```
$breaks
[1]    0  200  400  600  800 1000 1200 1400
```

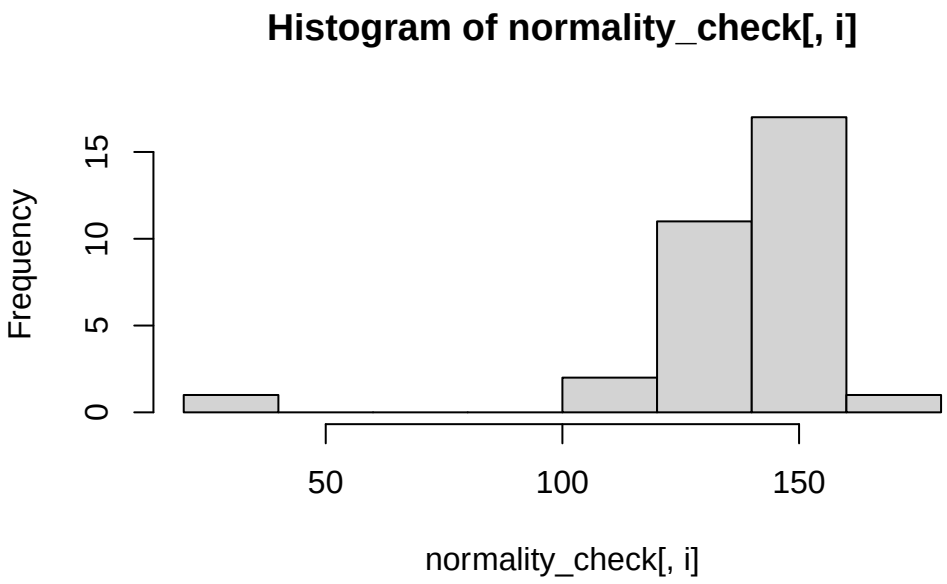
```
$counts
[1] 14 12  4  0  1  0  1

$density
[1] 0.00218750 0.00187500 0.00062500 0.00000000 0.00015625 0.00000000 0.00015625

$mids
[1]  100  300  500  700  900 1100 1300

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -0.62609901 -1.67593972  0.72451438 -1.22985876 -1.07751557 -0.36012989
 [7] -0.53340971  0.27769044  1.07751557  1.67593972  0.44509652 -0.94678176
[13] -0.19709908  0.36012989 -0.83051088  2.15387469 -0.27769044  0.83051088
[19]  0.62609901  0.94678176  0.11776987  0.19709908 -0.03917609  0.53340971
[25] -0.11776987  1.22985876 -0.44509652 -1.41779714  1.41779714  0.03917609
[31] -0.72451438 -2.15387469

$y
 [1]  113   37  370   52   76  183  113  301  462  846  320   81  198  304   96
[16] 1228  194  378  331  423  264  269  212  321  210  534  125   51  551  259
[31]  111   29
```

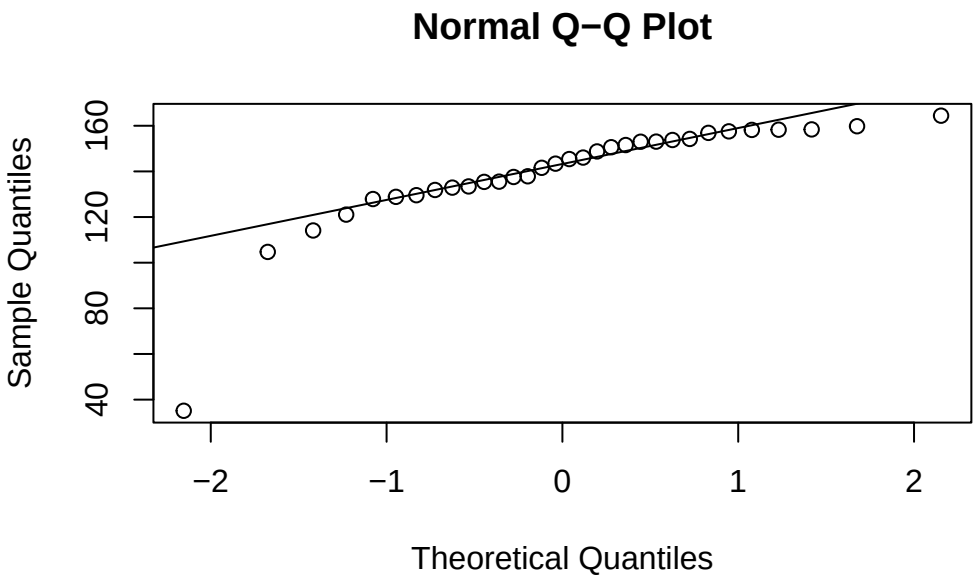





```
$breaks
[1] 0.35 0.40 0.45 0.50 0.55 0.60 0.65
```

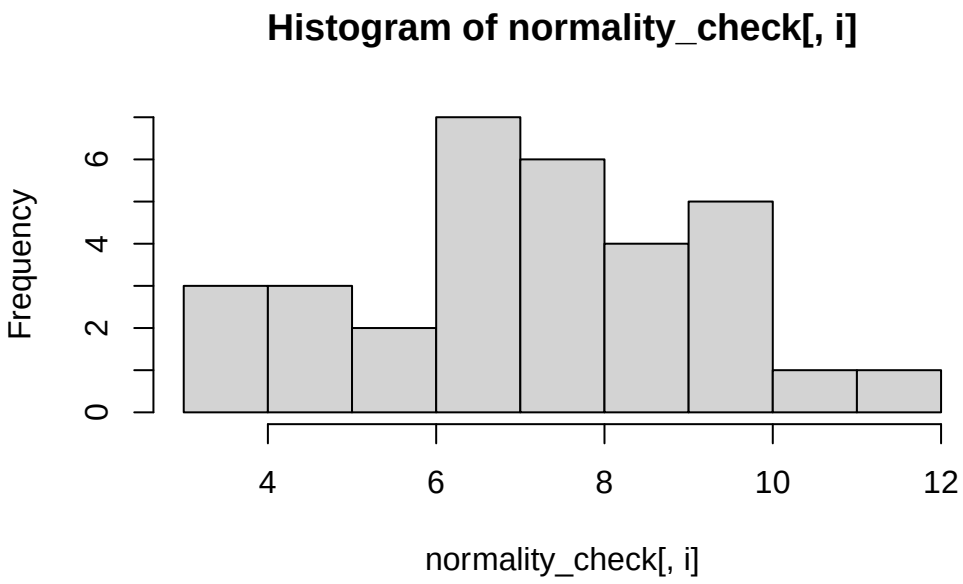
```
$counts
[1]  3 13 12  2  1  1

$density
[1] 1.875 8.125 7.500 1.250 0.625 0.625

$mids
[1] 0.375 0.425 0.475 0.525 0.575 0.625

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1]  0.53340971  1.67593972  1.07751557  0.83051088  2.15387469  1.22985876
 [7]  1.41779714  0.72451438 -1.67593972 -2.15387469 -1.41779714 -1.07751557
[13]  0.44509652 -0.11776987 -0.94678176 -0.36012989 -0.44509652  0.11776987
[19] -0.03917609  0.94678176 -0.53340971  0.36012989 -1.22985876 -0.19709908
[25]  0.03917609 -0.62609901 -0.72451438  0.19709908  0.27769044 -0.83051088
[31]  0.62609901 -0.27769044

$y
 [1] 0.4710177 0.5518919 0.4894865 0.4846154 0.6030921 0.5172678 0.5262281
 [8] 0.4796346 0.3926840 0.3828073 0.3961838 0.4069753 0.4688384 0.4413980
[15] 0.4150521 0.4337541 0.4330928 0.4575794 0.4493051 0.4853073 0.4306061
[22] 0.4686989 0.4041038 0.4399068 0.4533571 0.4292135 0.4291600 0.4592157
[29] 0.4595833 0.4285441 0.4791441 0.4379310
```

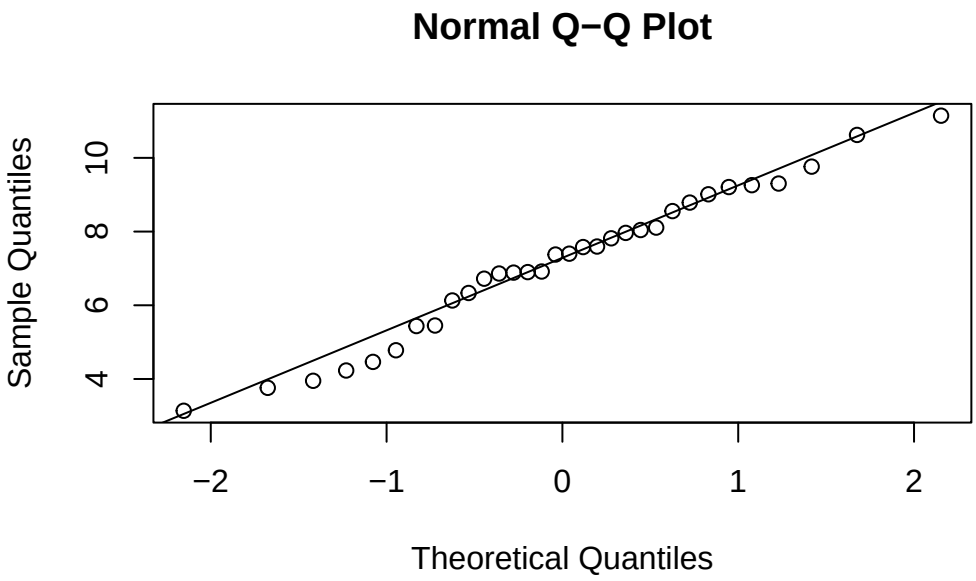





```
$breaks
[1]  20  40  60  80 100 120 140 160 180
```

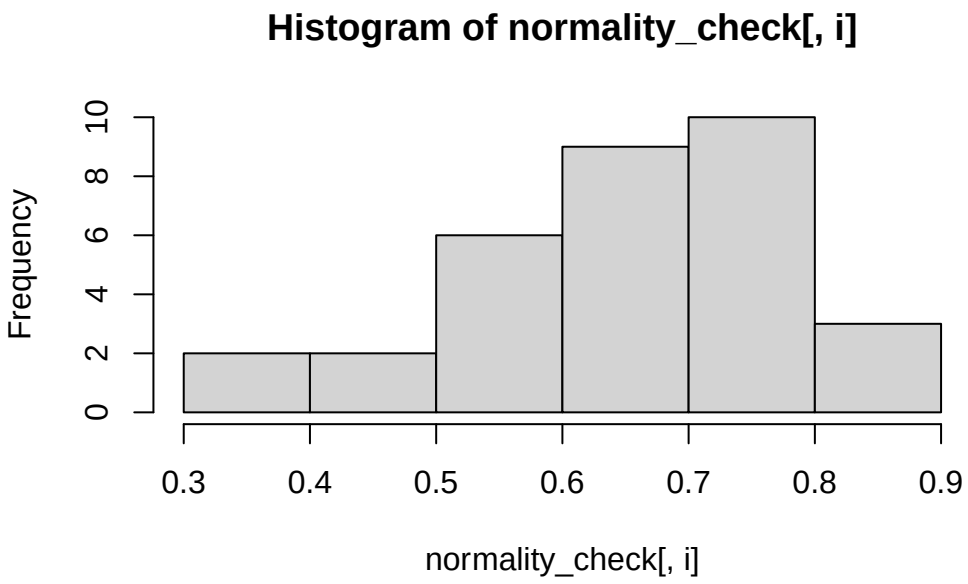
```
$counts
[1]  1  0  0  0  2 11 17  1

$density
[1] 0.0015625 0.0000000 0.0000000 0.0000000 0.0031250 0.0171875 0.0265625
[8] 0.0015625

$mids
[1]  30  50  70  90 110 130 150 170

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -2.15387469 -0.83051088  1.41779714 -1.07751557 -1.22985876 -0.94678176
 [7] -1.41779714 -1.67593972  1.22985876 -0.44509652  0.27769044  0.72451438
[13] -0.11776987  0.83051088  2.15387469  1.07751557  0.11776987 -0.03917609
[19] -0.62609901 -0.72451438 -0.36012989  0.53340971 -0.27769044  0.03917609
[25] -0.53340971  1.67593972 -0.19709908  0.19709908  0.62609901  0.94678176
[31]  0.36012989  0.44509652

$y
 [1]  35.10664 129.57679 158.39823 127.89120 121.06144 128.85237 114.10998
 [8] 104.71457 158.27781 135.40465 150.59965 154.22330 141.57012 156.88531
[15] 164.42228 158.17472 146.06139 143.39655 132.88958 131.86684 135.50588
[22] 153.04967 137.55892 145.40832 133.41204 159.79031 137.85989 148.74437
[29] 153.76176 157.54786 151.55575 152.99654
```

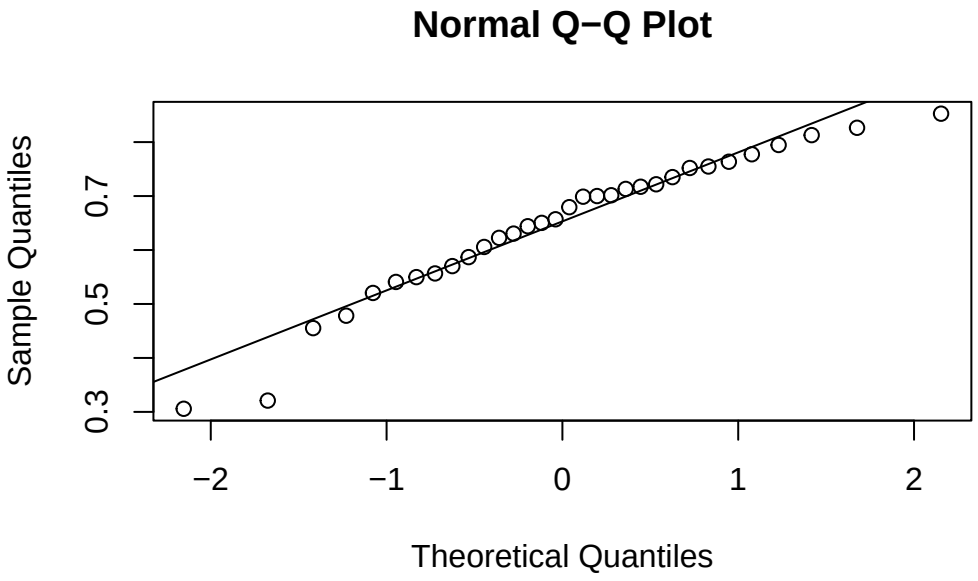





```
$breaks
 [1]  3  4  5  6  7  8  9 10 11 12
```

```
$counts
[1] 3 3 2 7 6 4 5 1 1

$density
[1] 0.09375 0.09375 0.06250 0.21875 0.18750 0.12500 0.15625 0.03125 0.03125

$mids
[1]  3.5  4.5  5.5  6.5  7.5  8.5  9.5 10.5 11.5

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1] -2.15387469 -1.07751557 -0.83051088 -0.72451438 -1.67593972 -1.41779714
 [7] -1.22985876 -0.94678176  1.41779714  0.94678176 -0.19709908 -0.11776987
[13]  0.44509652 -0.27769044 -0.62609901  0.53340971 -0.44509652 -0.53340971
[19]  0.11776987  0.03917609  0.19709908 -0.36012989  1.07751557  0.83051088
[25] -0.03917609  0.27769044  1.22985876  1.67593972  0.72451438  2.15387469
[31]  0.36012989  0.62609901

$y
 [1]  3.136821  4.463386  5.436229  5.450587  3.759559  3.950278  4.230113
 [8]  4.779340  9.762163  9.208000  6.900293  6.915988  8.041911  6.886189
[15]  6.130149  8.107901  6.721441  6.333255  7.579317  7.400137  7.593697
[22]  6.861476  9.258847  9.011321  7.377045  7.816760  9.303589 10.622760
[29]  8.786355 11.145158  7.963540  8.555708
```






```
$breaks
[1] 0.3 0.4 0.5 0.6 0.7 0.8 0.9
```

```
$counts
[1]  2  2  6  9 10  3

$density
[1] 0.6250 0.6250 1.8750 2.8125 3.1250 0.9375

$mids
[1] 0.35 0.45 0.55 0.65 0.75 0.85

$xname
[1] "normality_check[, i]"

$equidist
[1] TRUE

attr(,"class")
[1] "histogram"
$x
 [1]  1.41779714 -2.15387469 -0.83051088 -1.67593972 -0.19709908 -1.41779714
 [7] -0.03917609  2.15387469 -0.72451438 -0.36012989  0.03917609  1.67593972
[13] -1.22985876  0.11776987  0.53340971  0.72451438  0.36012989  0.44509652
[19] -0.11776987  0.27769044 -0.62609901  1.07751557 -0.94678176 -0.44509652
[25]  0.94678176  1.22985876 -0.53340971 -1.07751557  0.83051088  0.19709908
[31]  0.62609901 -0.27769044

$y
 [1] 0.8129496 0.3057851 0.5497771 0.3209877 0.6440678 0.4552239 0.6569767
 [8] 0.8526912 0.5566265 0.6225166 0.6794055 0.8265306 0.4782609 0.6988506
[15] 0.7218045 0.7519902 0.7132353 0.7172676 0.6502947 0.7014925 0.5701944
[22] 0.7774566 0.5408163 0.6056604 0.7636364 0.7946429 0.5868545 0.5204082
[29] 0.7547945 0.7000000 0.7350993 0.6304348
```

For each of the above distributions, I will group based on if the data is normal or otherwise distributed. Then using the mean and variance from the summary (mean and variance) data frame I will be able to establish fake data for this data set upon which to build a model.

The list of variables, in order, are below. This order corresponds with the order of plots from the above loop.

1. mass
2. massSTD

3. w.lifted
4. w.liftedSTD
5. wing.ar
6. wing.arSTD
7. wing.length
8. wing.lengthSTD
9. mean.hor_acel
10. mean.hor_decel
11. mean.arc_avghvel
12. mean.total_vel
13. ss.pitch_roll
14. mean.pr_time
15. mean.pr_degrees
16. mean.arc_force
17. PRTpercent

Proposed Distributions

Normal 3. w.lifted 4. w.liftedSTD 5. wing.ar - appears to increase at end, hard to tell but most close to normal 6. wing.arSTD 7. wing.length 8. wing.lengthSTD 16. mean.arc_force

Beta 1. mass - skewed right 2. massSTD 9. mean.hor_acel - left-skewed data 10. mean.hor_decel - left skewed data

11. mean.arc_avghvel - left skewed data

12. mean.total_vel - left skewed data

13.ss.pitch_roll - right skewed

14. mean.pr_time - right skwewed

15. mean.pr_degrees - left skewed data

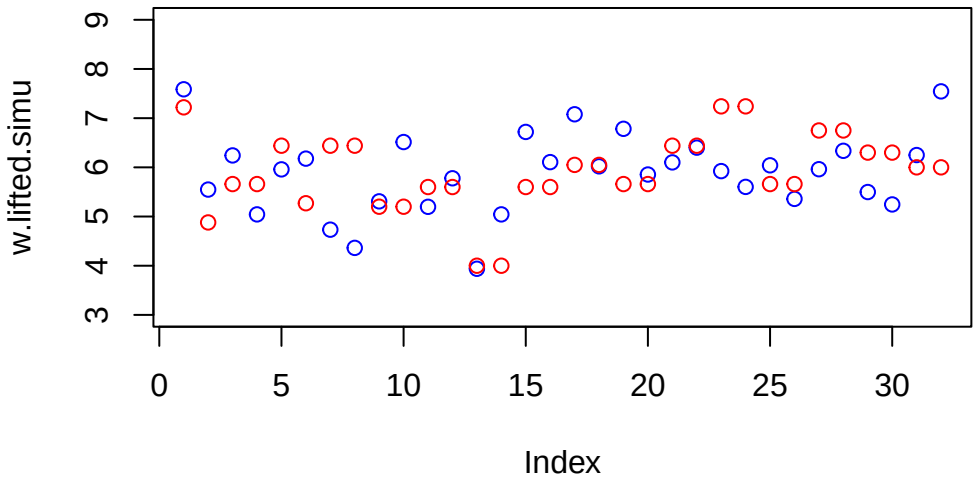
17. PRTpercent - left skewed

Generating Fake Data

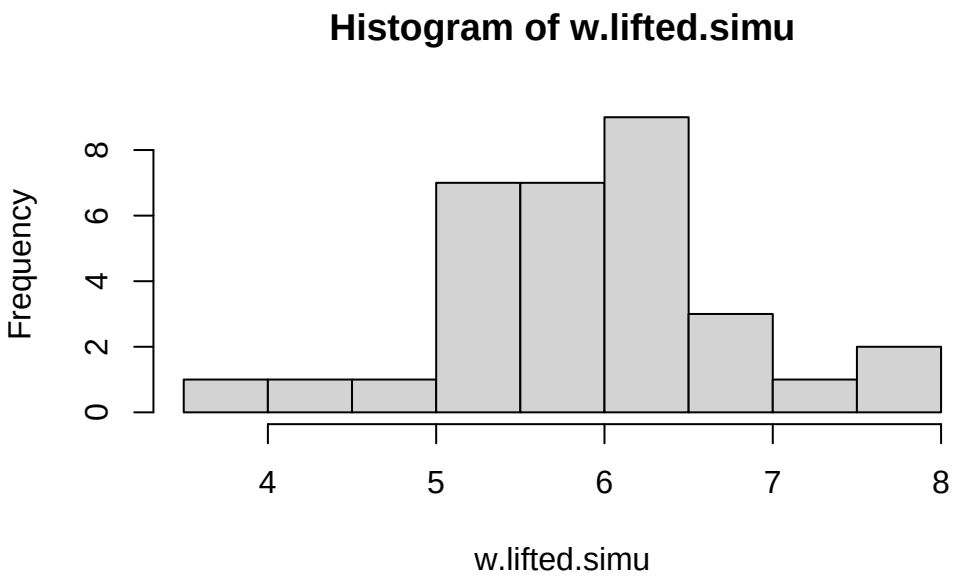
For each variable, I will create a fake data set that has the same mean and variance, with a sample size of 32 individuals (same as the Segre et al., 2015 paper) and a distribution that is consistent with the original data.

Weight Lifted

```
w.lifted.simu <- rnorm(n =32, mean = annas_summ[3,2], sd= sqrt(annas_summ[3,3]))

plot(w.lifted.simu, col = "blue",ylim = c(3,9))
points(annas_clean$w.lifted, col = "red")
```



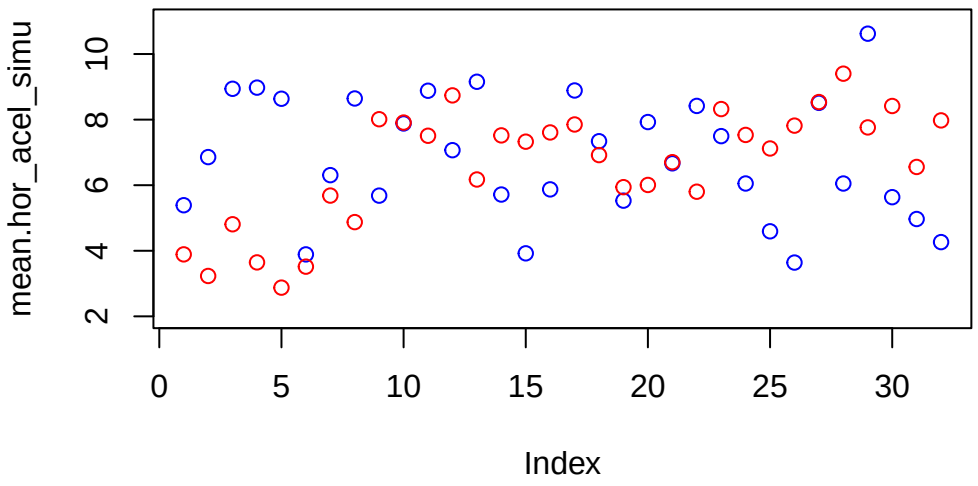

```
hist(w.lifted.simu)
```



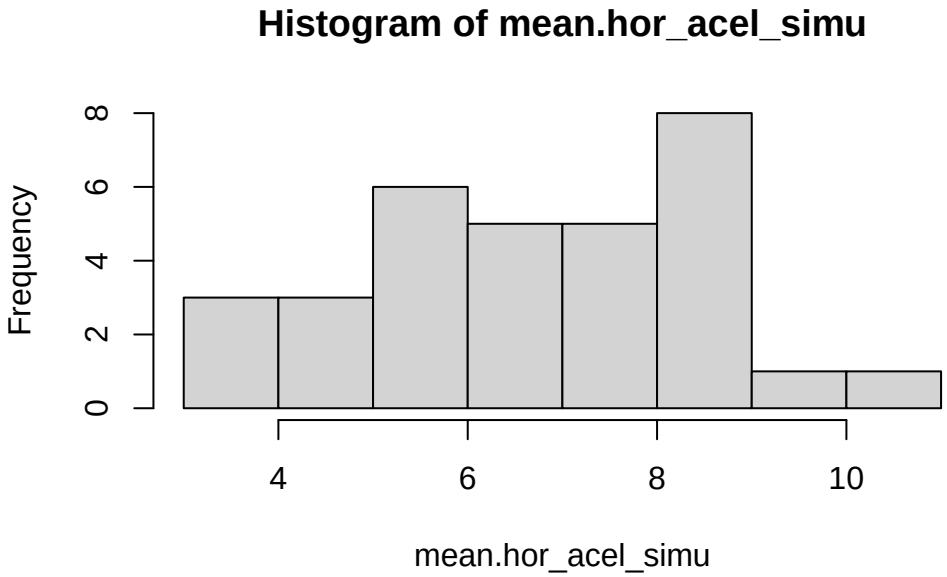
Horizontal Acceleration

I am having difficulty creating fake data sets with Beta distributions. For now, I am making a normally distributed fake data set for horizontal acceleration (variable 9) so that I can work on building a model around it, but need to come back to this distribution issue.

```
mean.hor_acel_simu <- rnorm(n =32, mean = annas_summ[9,2], sd= sqrt(annas_summ[9,3]))

plot(mean.hor_acel_simu, col = "blue", ylim = c(2,11))
points(annas_clean$mean.hor_acel, col = "red")
```




```
hist(mean.hor_acel_simu)
```



Running a Phenotype-Performance Model

Creating a Stan Model

To begin, I have started with a simple stan model, connecting a linear regression of weight lifted as a phenotype and mean horizontal acceleration as a performance variable. Below is the commented Stan code.

```
# data{
#   int<lower=0> n; //number of samples
#   vector [n] wlifted; //independent variable in data, weight lifted
#   vector [n] mhoracel; //dependent varible, mean horizontal acceleration
# }
#
# parameters{
#   real a; //real number that is intercept of linear regression
#   real b; //real number that is slope of linear regression
#   real <lower=0> sigma; //real number that is variance
# }
#
# model{
#   vector[n] yhat;
#   yhat = a + b*wlifted;
#   //setting priors for data
#   mhoracel ~ normal(yhat, sigma);
#   a ~ normal(0,5);
#   b ~ normal(0,10);
#   sigma ~ normal(0,10);
# }
```

Running Stan Model

```
annas_data_simu <- list(
      n = length(w.lifted.simu),
      mhoracel = as.numeric(mean.hor_acel_simu),
      wlifted = as.numeric(w.lifted.simu)
)

annas_fit <- stan(file = "annas_performance_model.stan", data = annas_data_simu, iter = 10
```

```
SAMPLING FOR MODEL 'annas_performance_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 4.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.42 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.121164 seconds (Warm-up)
Chain 1:                0.096962 seconds (Sampling)
Chain 1:                0.218126 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'annas_performance_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.2e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
```

```
Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.107284 seconds (Warm-up)
Chain 2:                0.116184 seconds (Sampling)
Chain 2:                0.223468 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'annas_performance_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 1.2e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.12 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.103648 seconds (Warm-up)
Chain 3:                0.09853 seconds (Sampling)
Chain 3:                0.202178 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'annas_performance_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 1.1e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.11 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
```

```
Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.081975 seconds (Warm-up)
Chain 4:                0.073389 seconds (Sampling)
Chain 4:                0.155364 seconds (Total)
Chain 4:
```

```
print(annas_fit)
```

```
Inference for Stan model: annas_performance_model.
4 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=2000.

        mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
a       8.94    0.09 2.17   4.47   7.50   9.13  10.40  13.04   571 1.00
b      -0.37    0.02 0.37  -1.04  -0.62  -0.39  -0.13   0.41   581 1.00
sigma   1.86    0.01 0.27   1.44   1.69   1.83   2.00   2.48   687 1.01
lp__  -36.29    0.06 1.31 -39.60 -36.88 -35.94 -35.33 -34.86   465 1.01

Samples were drawn using NUTS(diag_e) at Thu Mar 16 11:32:48 2023.
For each parameter, n_eff is a crude measure of effective sample size,
and Rhat is the potential scale reduction factor on split chains (at
convergence, Rhat=1).
```

References

Arnold. 1983. Morphology, Performance and Fitness. *American Zoology* 23:347-361.

Bergmann and McElroy. 2014. Many-to-Many Mapping of Phenotype to Performance: An Extension of the F-Matrix for Studying Functional Complexity. *Evolutionary Biology* 41:546–560. DOI 10.1007/s11692-014-9288-1

Márquez-Luna et al. 2022. Genetic relatedness and morphology as drivers of interspecific dominance hierarchy in hummingbirds. *PeerJ.* DOI 10.7717/peerj.13331