# Exploratory Data Analysis

Introduction to EDA

*Pramod Duvvuri*

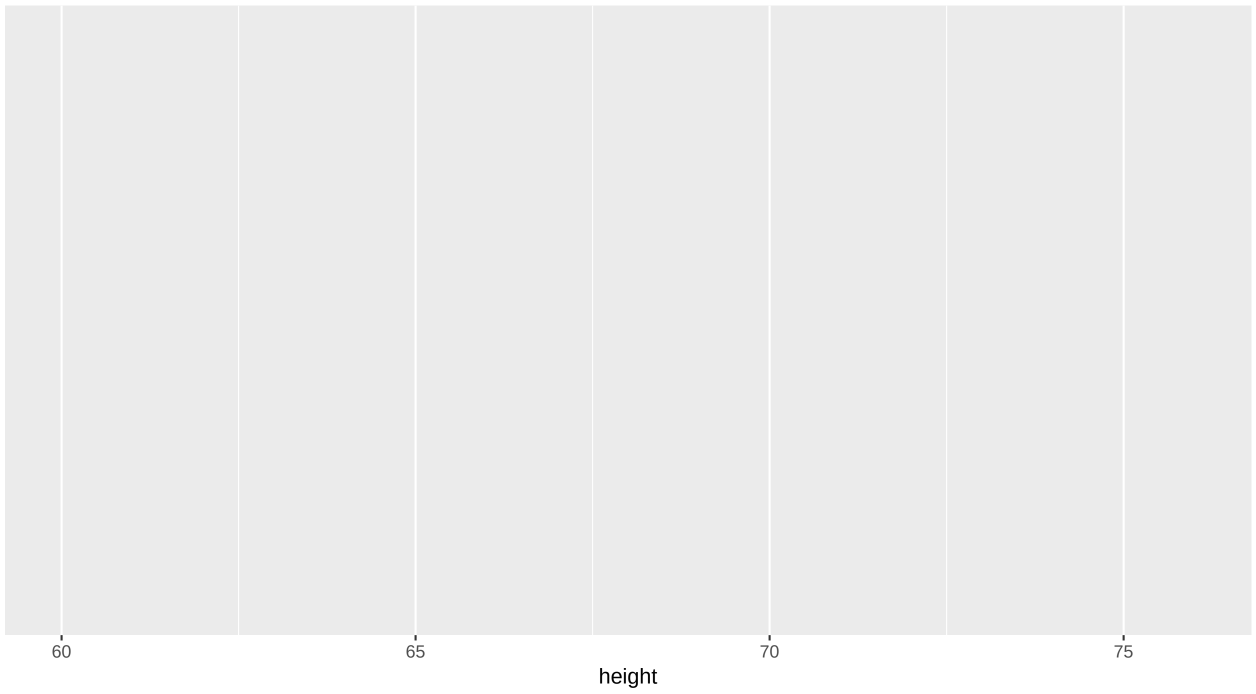
*3/11/2019*

## Univariate Data

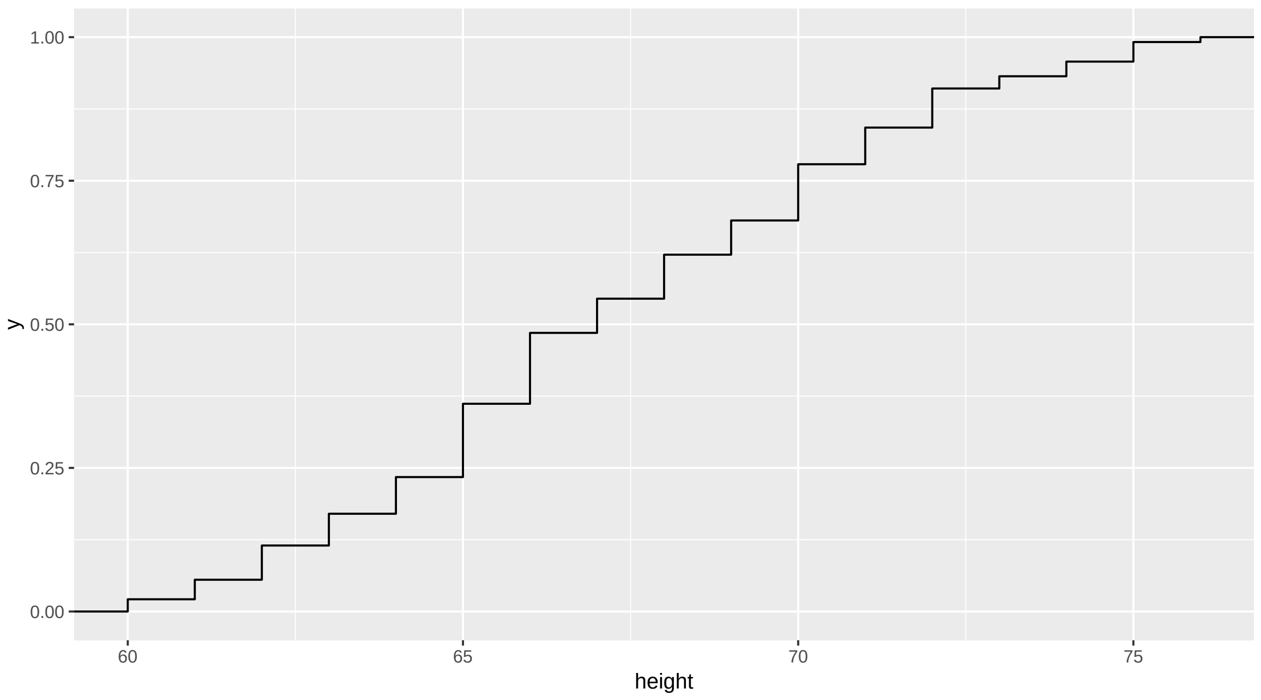
### Learning Grammar of Graphics (ggplot2)

```r
# Load installed packages
library('lattice')
library('ggplot2')
```

```r
ggplot(singer, aes(x = height))
```

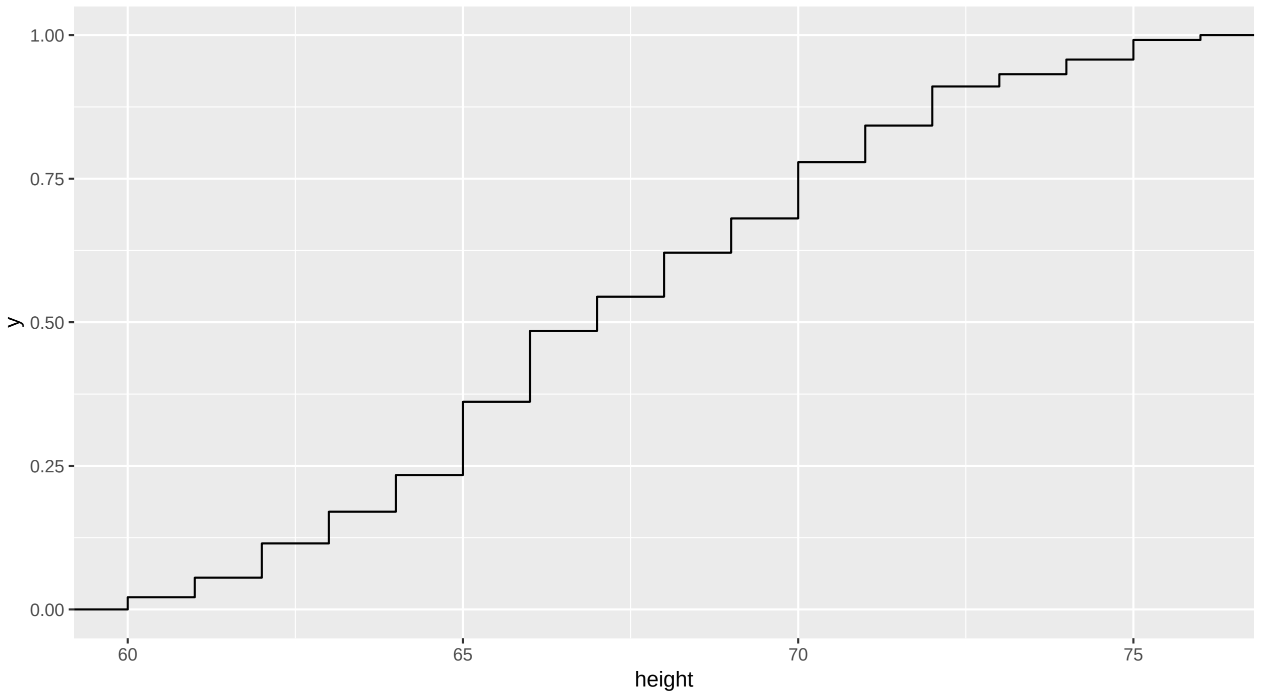


```r
# ECDF in ggplot2
ggplot(singer, aes(x = height)) + stat_ecdf()
```

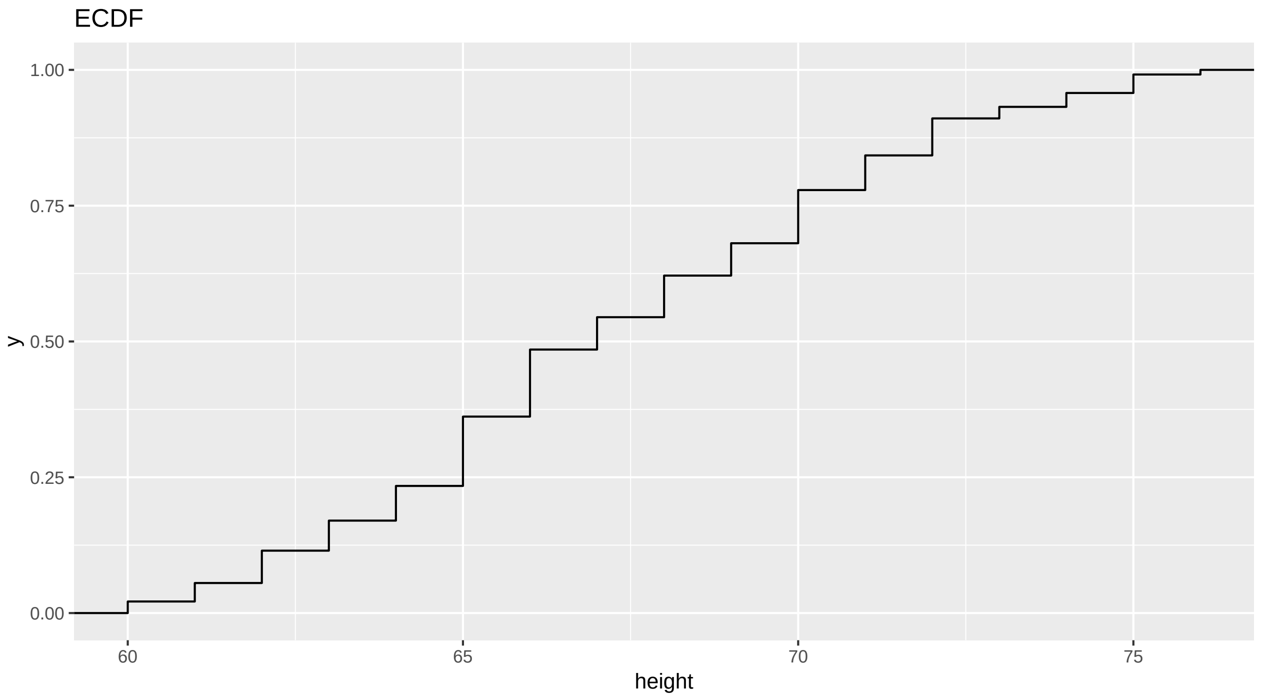


### Basic Plots

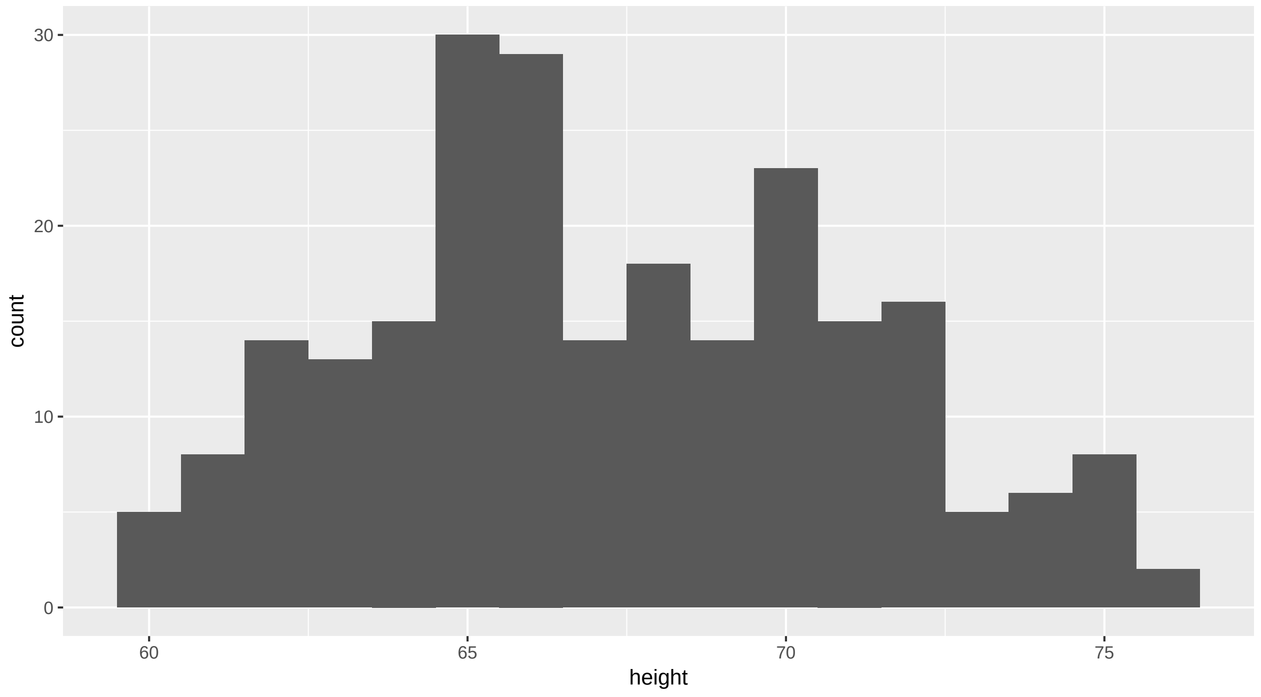
```r
# Alternate way to get ECDF
singer_gg <- ggplot(singer, aes(x = height))
singer_gg + stat_ecdf()
```



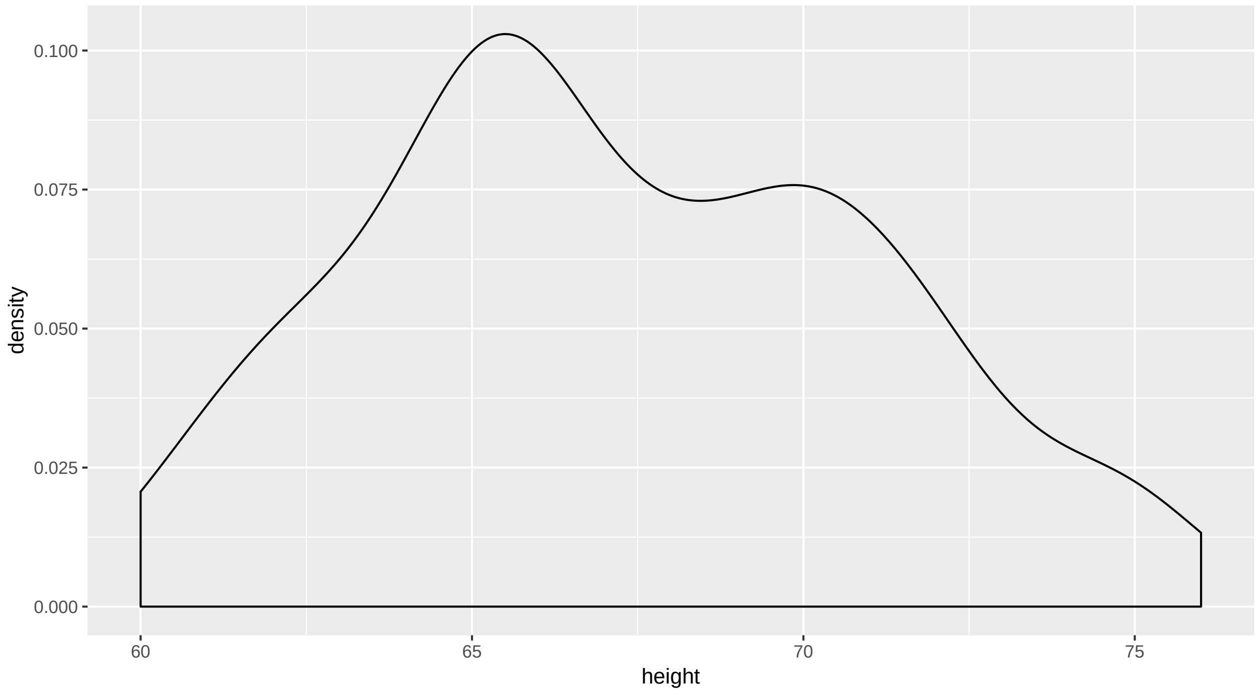
```r
# Add Labels
singer_gg + stat_ecdf() + ggtitle('ECDF')
```



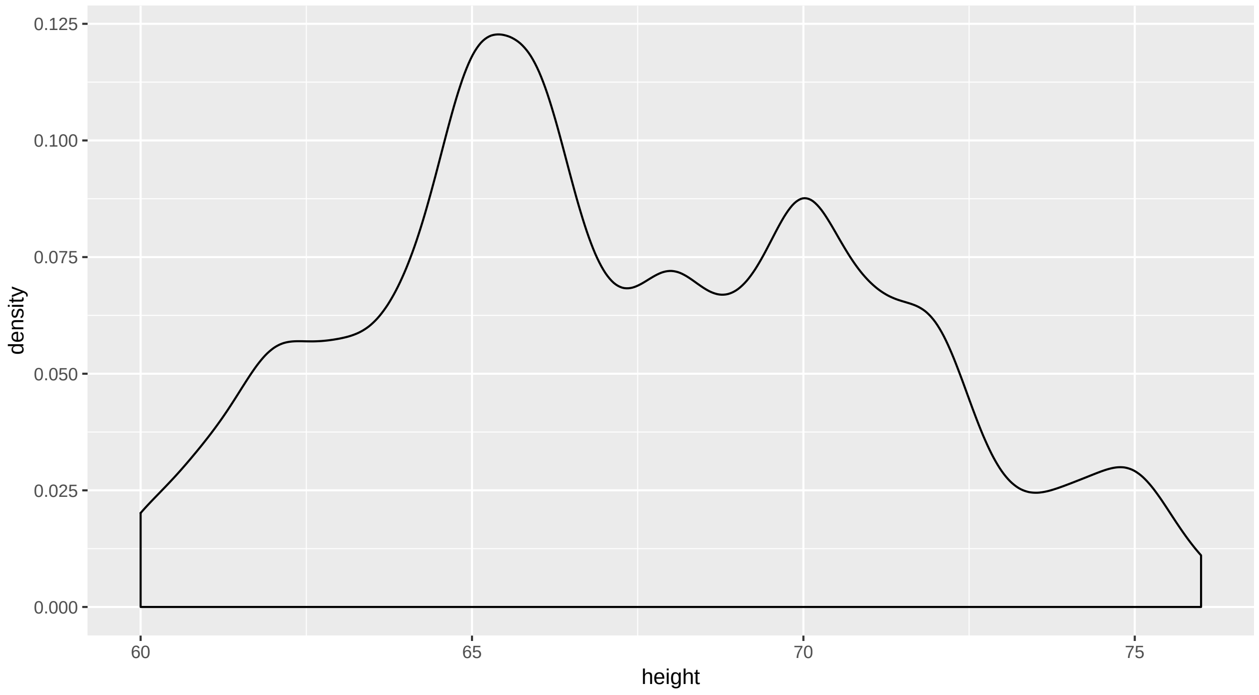
```r
# Histogram
#ggplot(singer, aes(x= height)) + geom_histogram()
ggplot(singer, aes(x= height)) + geom_histogram(binwidth = 1)
```



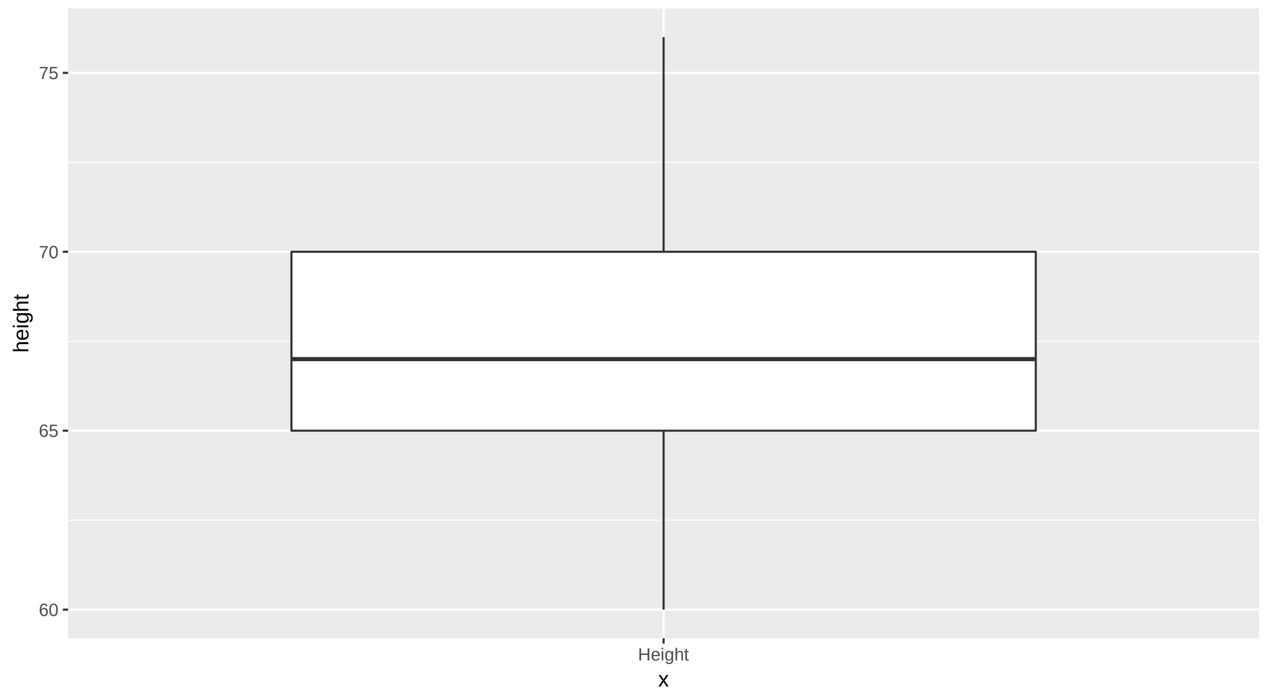
```r
# Density Plot
ggplot(singer, aes(x= height)) +
  geom_density()
```



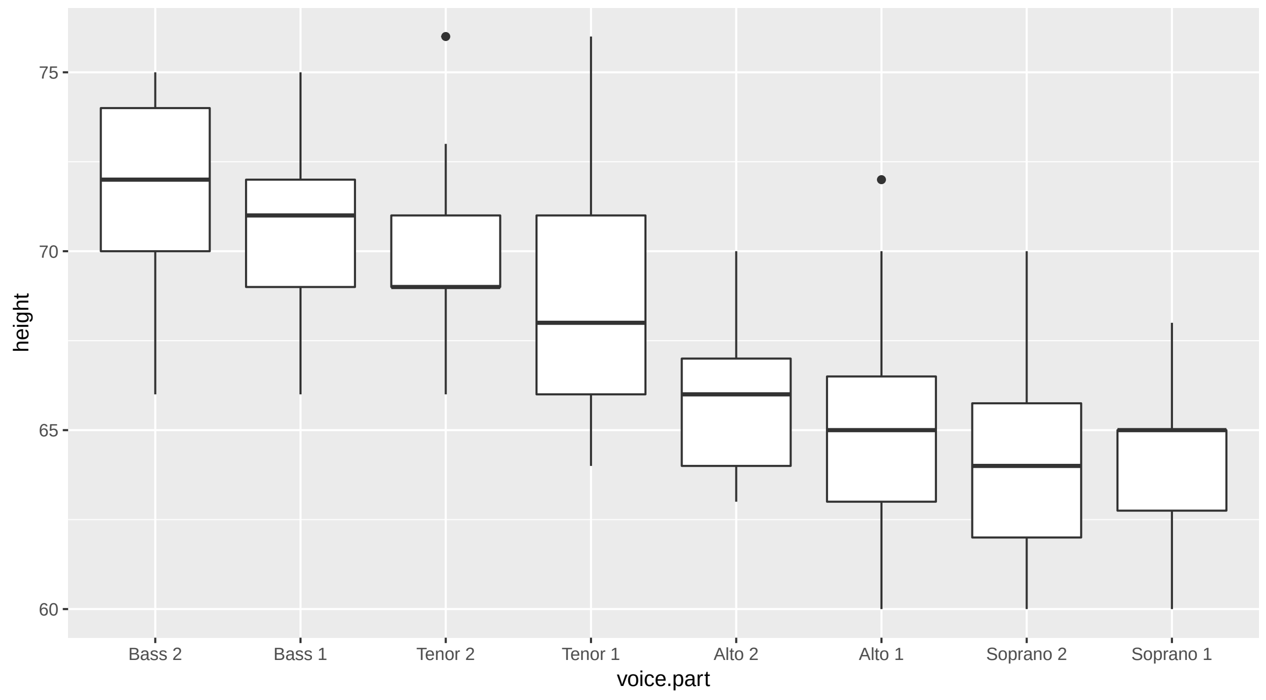
```r
ggplot(singer, aes(x= height)) +
  geom_density(adjust = 0.5)
```



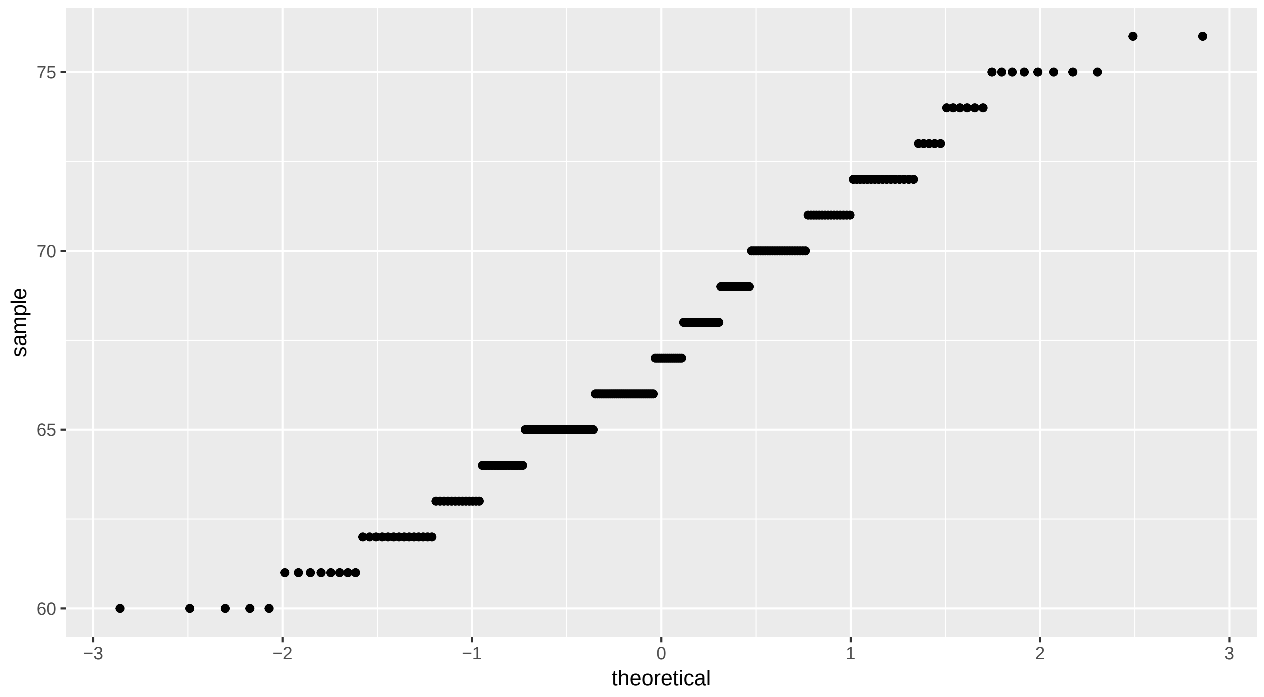
```r
# Boxplot
ggplot(singer, aes(x= "Height", y = height)) + geom_boxplot()
```



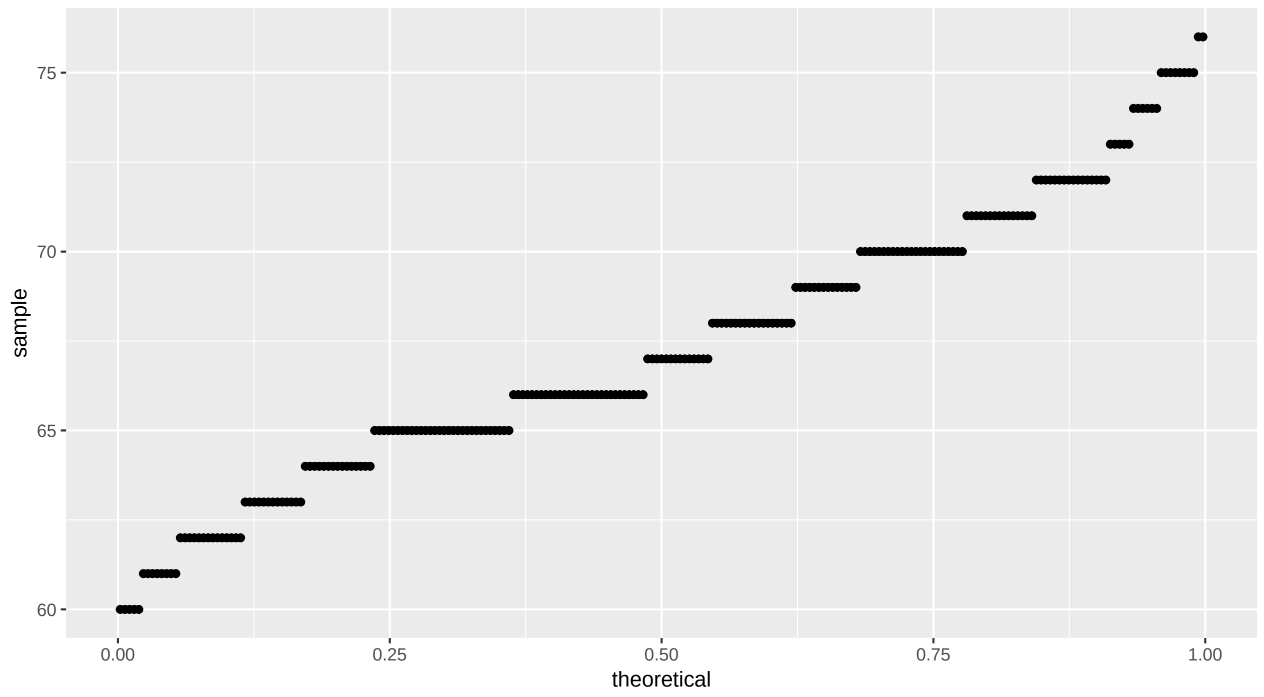
```
# Multiple Boxplots
ggplot(singer, aes(x = voice.part, y = height)) + geom_boxplot()
```



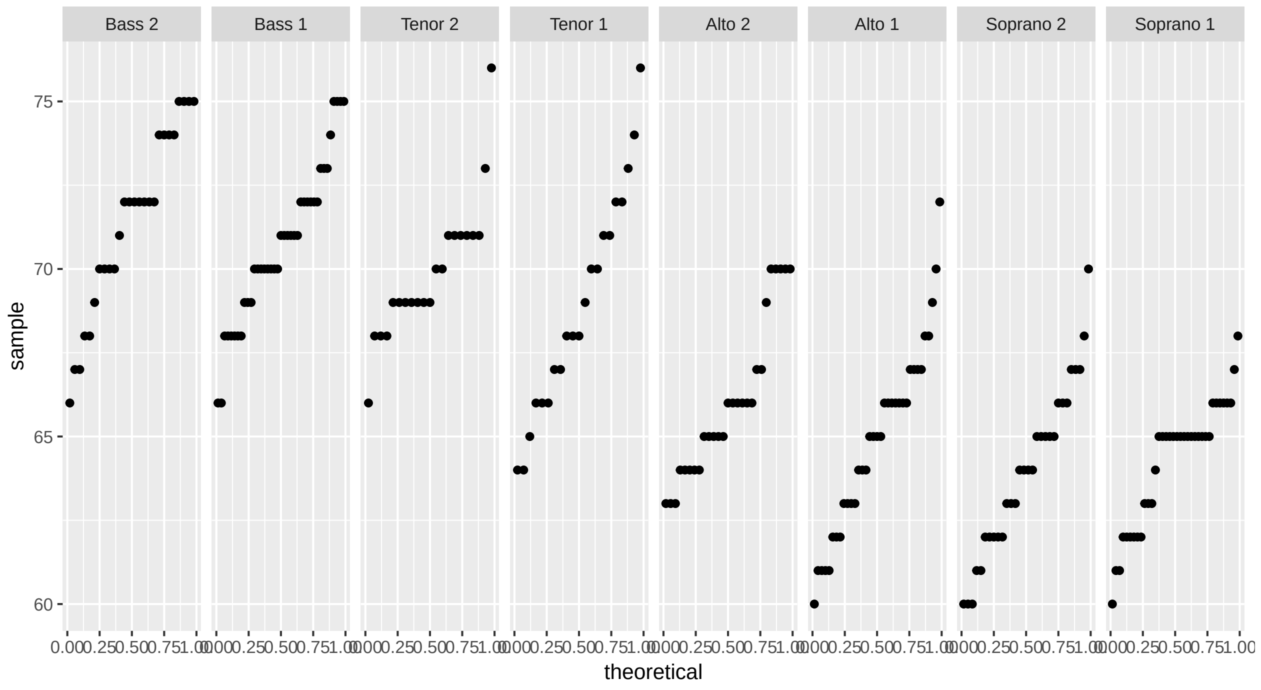
```
# QQ-Plot using a Normal Distribution
ggplot(singer, aes(sample = height)) + stat_qq()
```



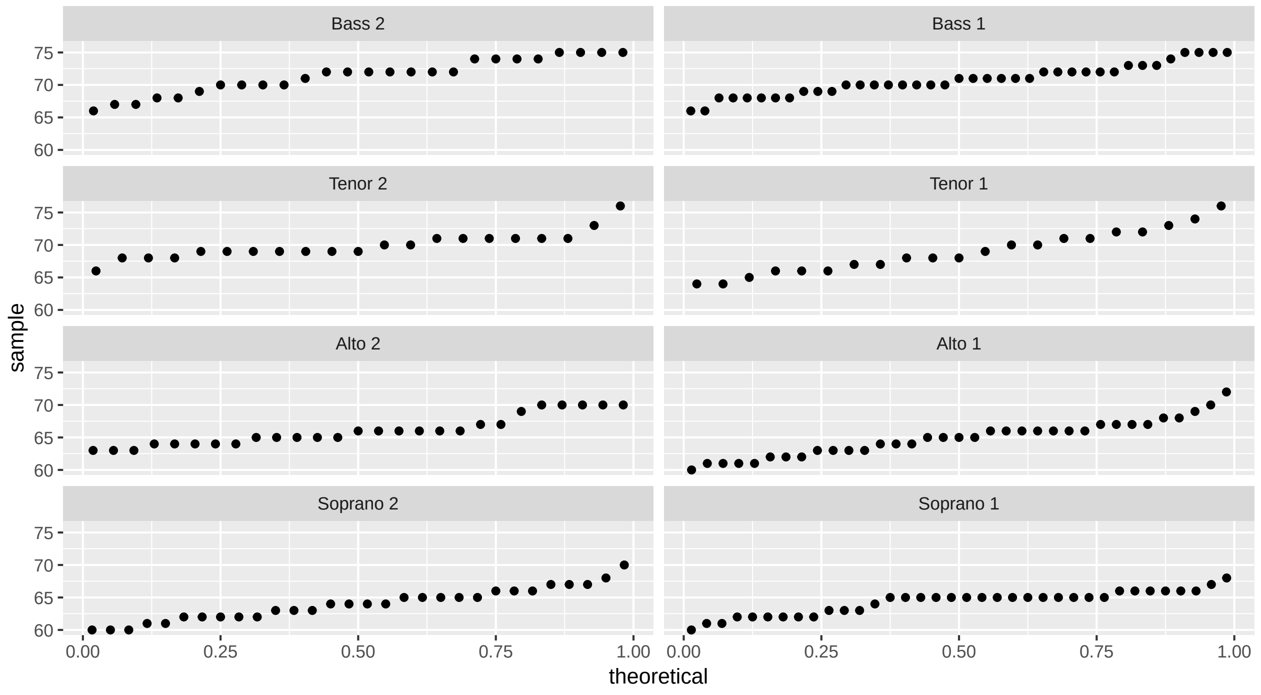
```r
# QQ-Plot using a Uniform Distribution
ggplot(singer, aes(sample = height)) +
  stat_qq(distribution = qunif)
```



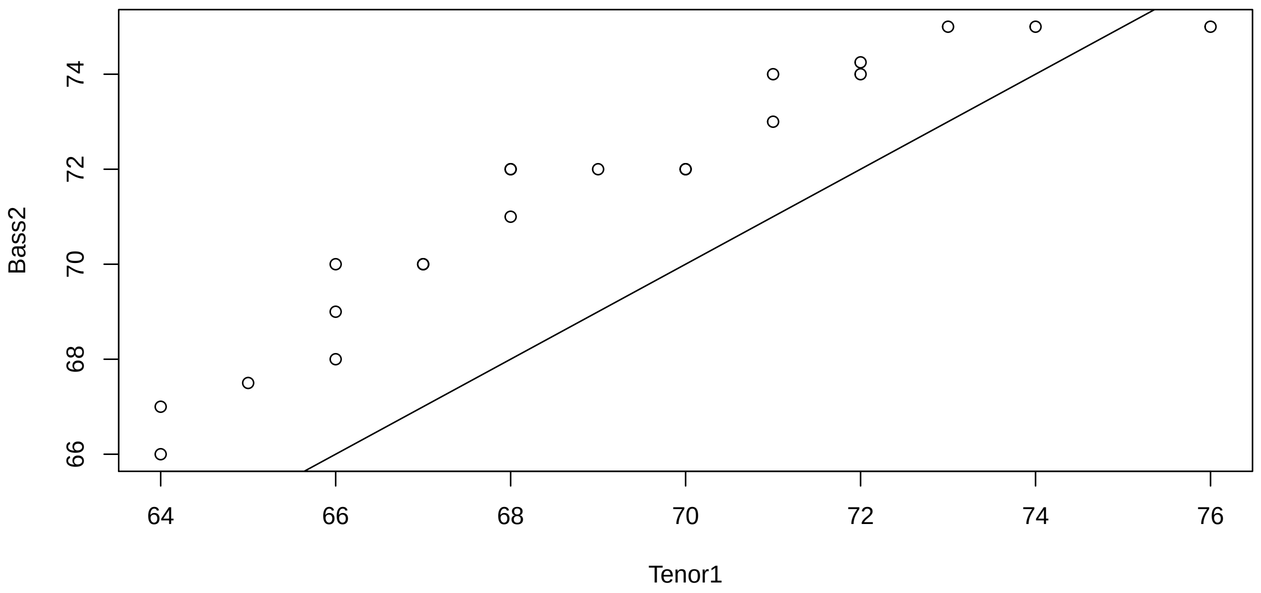
```r
# Facet
ggplot(singer,aes(sample = height)) +
  stat_qq(distribution = qunif) +
  facet_grid(~voice.part)
```



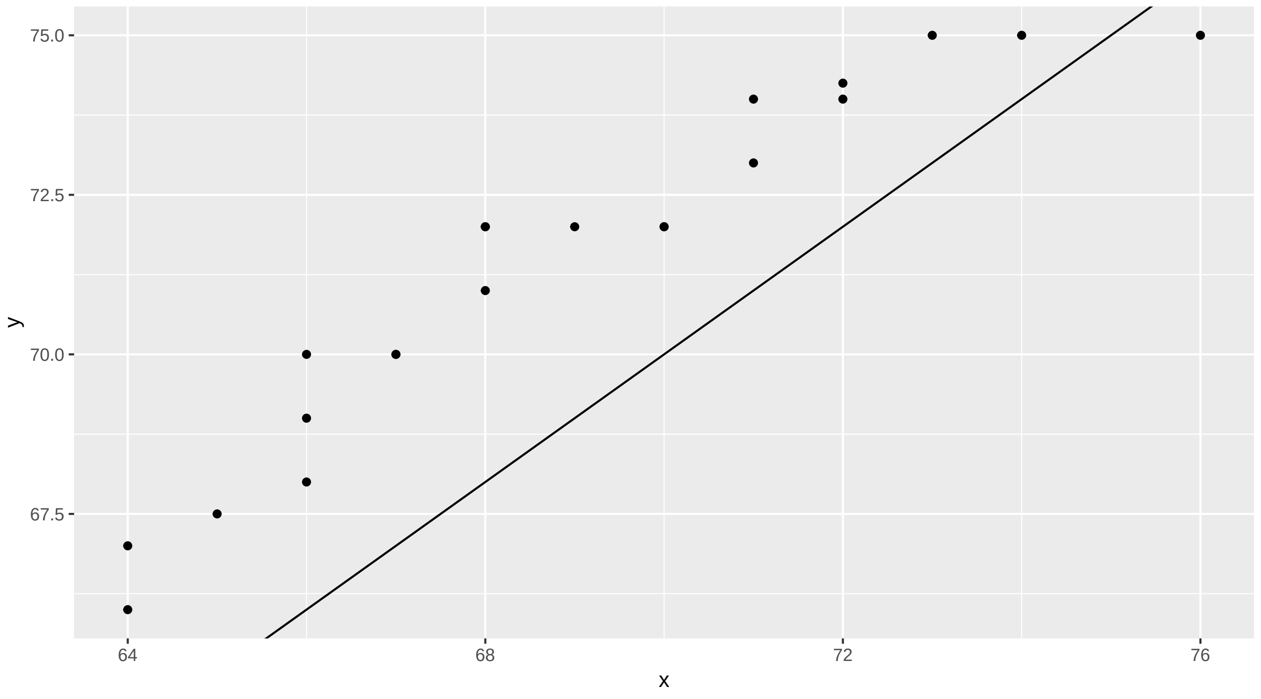
```r
# The plot above looks cramped, so let us try a grid with columns
ggplot(singer, aes(sample = height)) +
  stat_qq(distribution = qunif) +
  facet_wrap(~voice.part, ncol = 2)
```



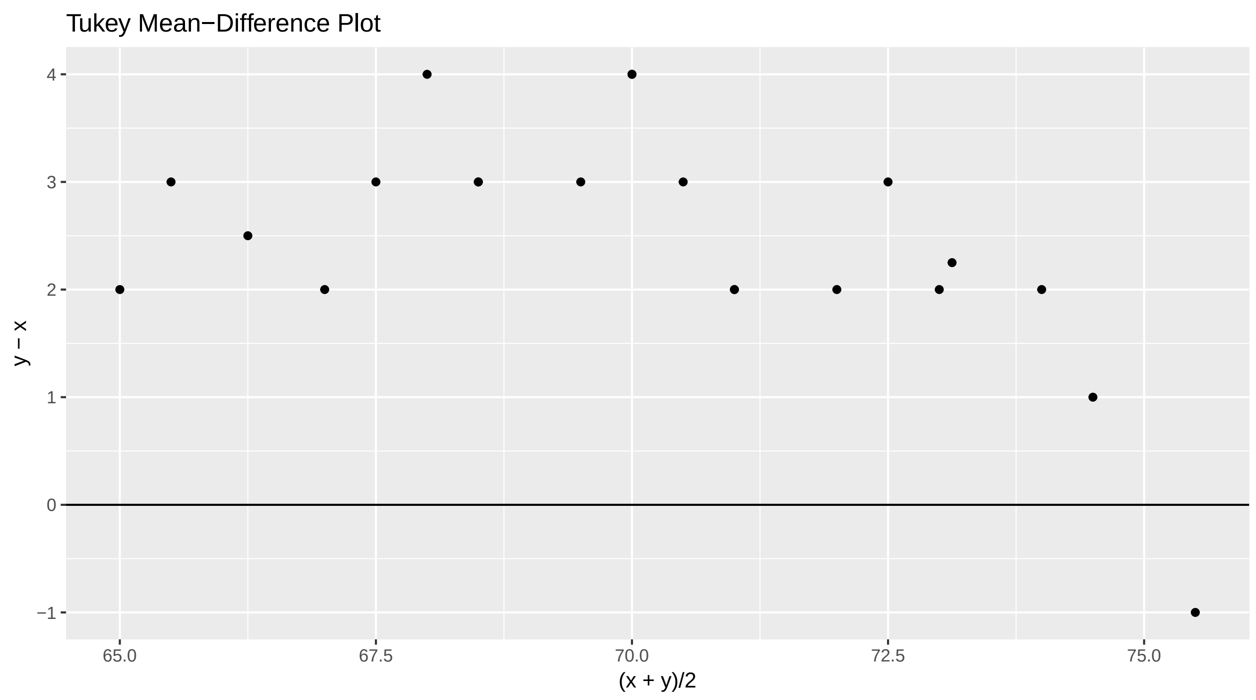
```r
# QQ-Plot using Base R function qqplot()
Tenor1 = singer$height[singer$voice.part == "Tenor 1"]
Bass2 = singer$height[singer$voice.part == "Bass 2"]
qqplot(Tenor1, Bass2)
abline(0, 1)
```



```
# Using ggplot
#library(tibble)
#qq_df <- as.tibble(qqplot(Tenor1, Bass2, plot.it = FALSE))
qq_df <- as.data.frame(qqplot(Tenor1, Bass2, plot.it = FALSE))
ggplot(data = qq_df, mapping =
         aes(x = x, y = y)) +
  geom_point() +
  geom_abline()
```

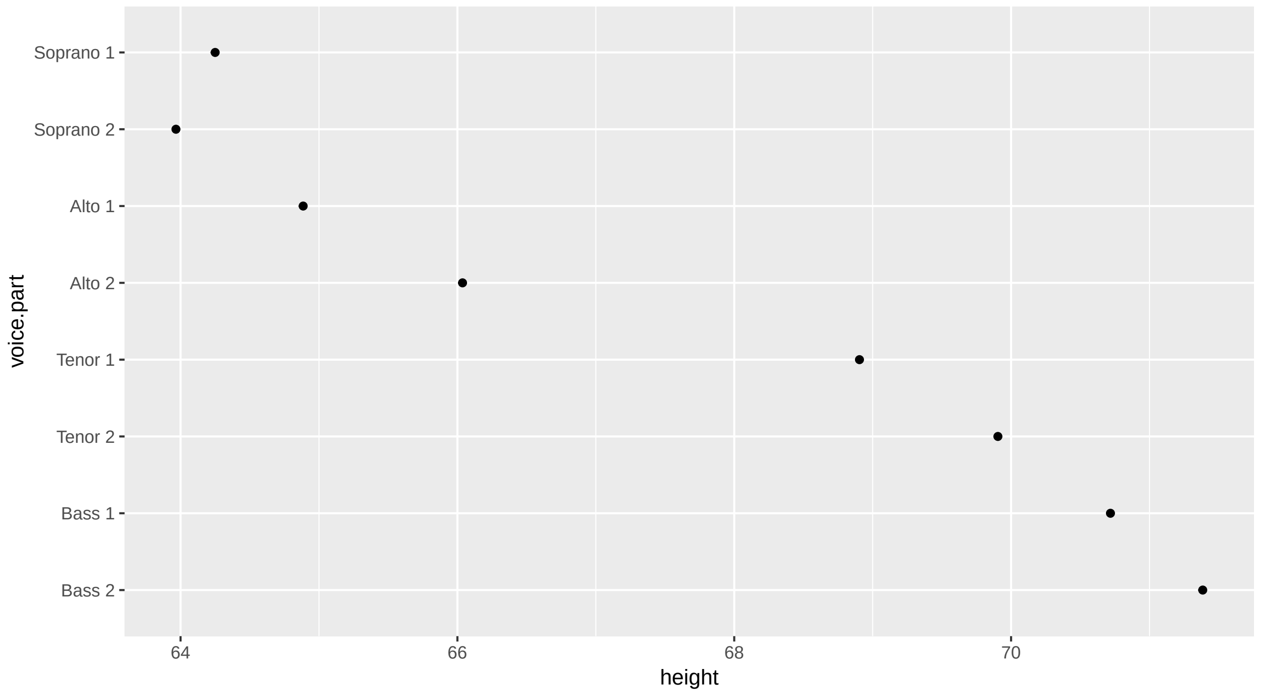


```
## Tukey-Mean difference Plot
ggplot(data = qq_df, mapping = aes(x = (x + y)/2, y = y - x)) +
  geom_point() +
  geom_abline(slope = 0) + ggtitle("Tukey Mean-Difference Plot")
```



```r
singer_means <- aggregate(height ~ voice.part, FUN = mean, data = singer)
```

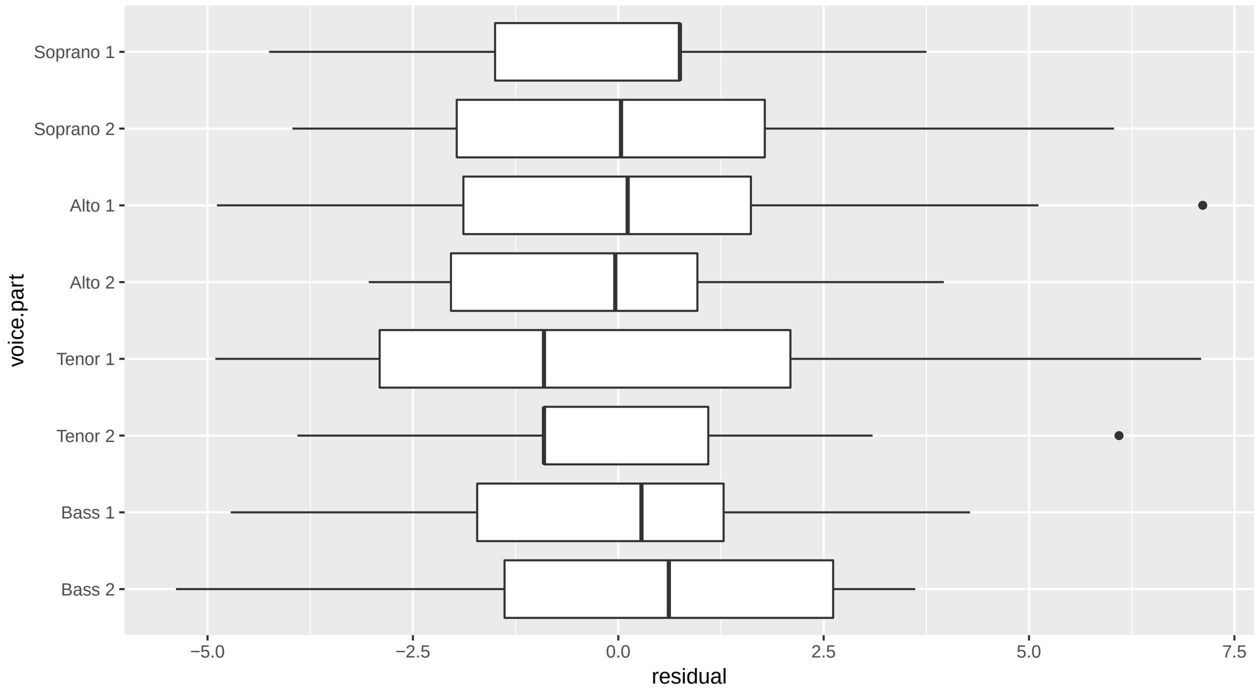
```r
ggplot(singer_means, aes(x = voice.part, y = height)) + geom_point() + coord_flip()
```



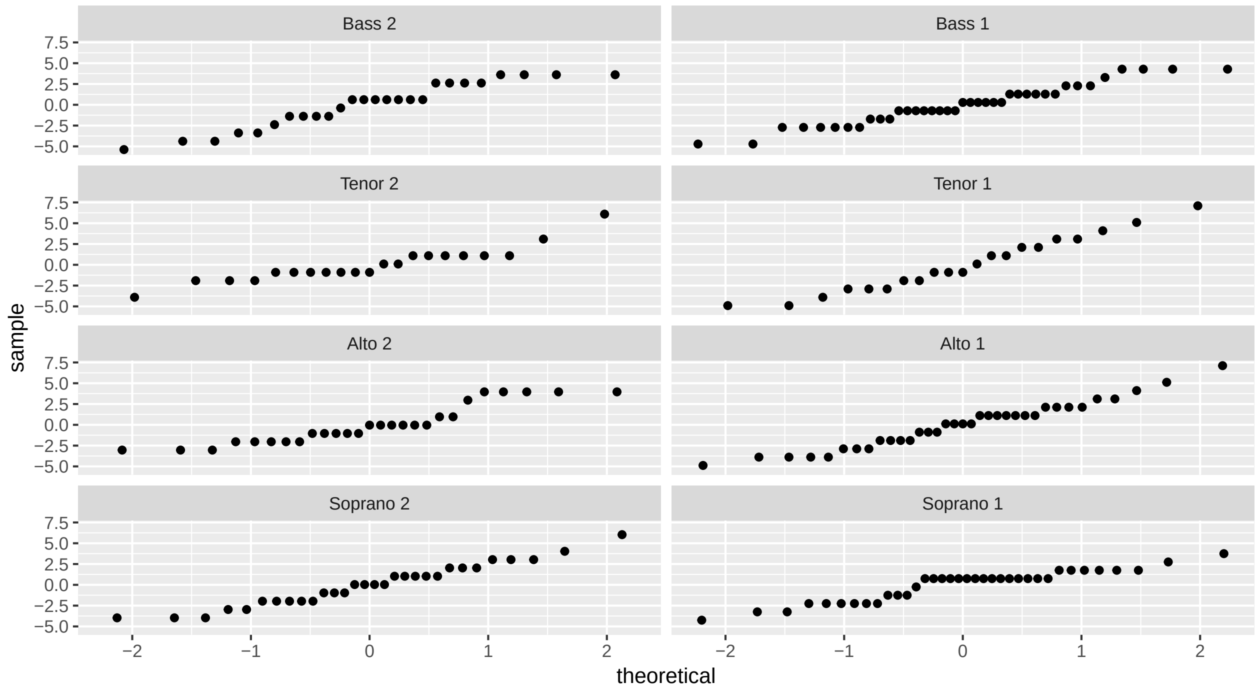
```r
# Fitting a linear model
singer.lm <- lm(height ~ voice.part, data = singer)
```

```r
# Extracting residual values
singer.res <- data.frame(voice.part = singer$voice.part, residual = residuals(singer.lm))
```

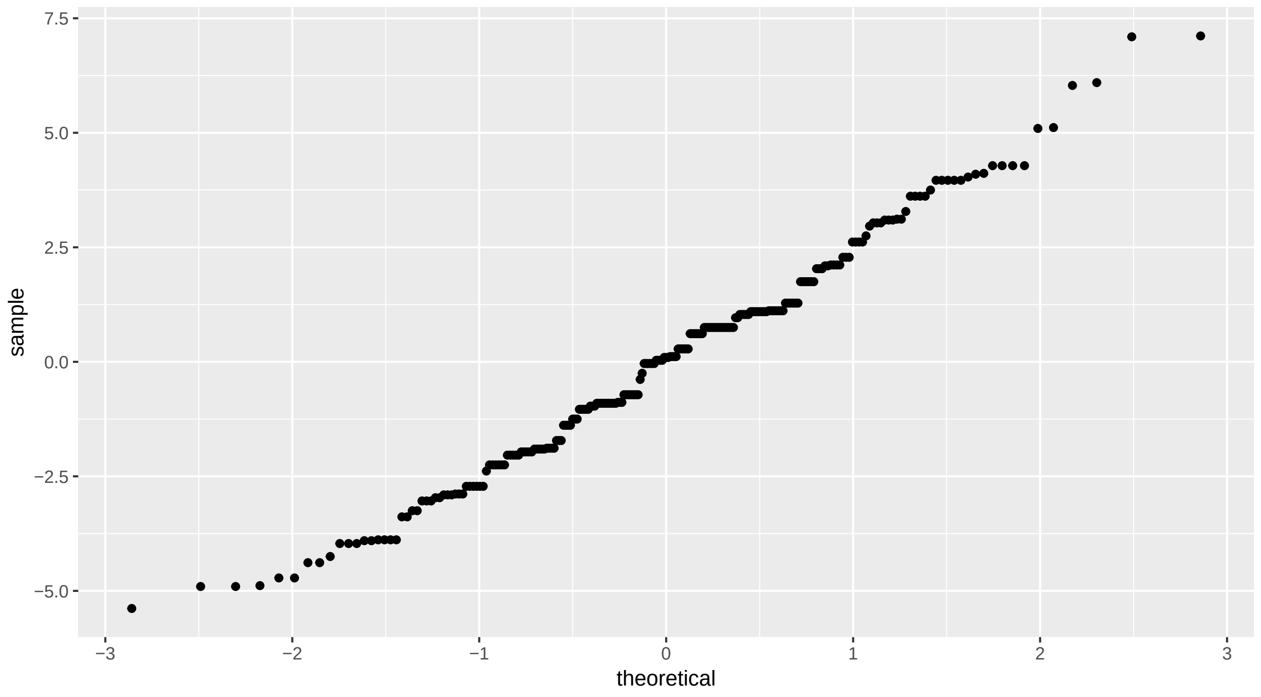
```
# Observing Residuals using boxplots
ggplot(singer.res, aes(x = voice.part, y = residual)) + geom_boxplot() + coord_flip()
```



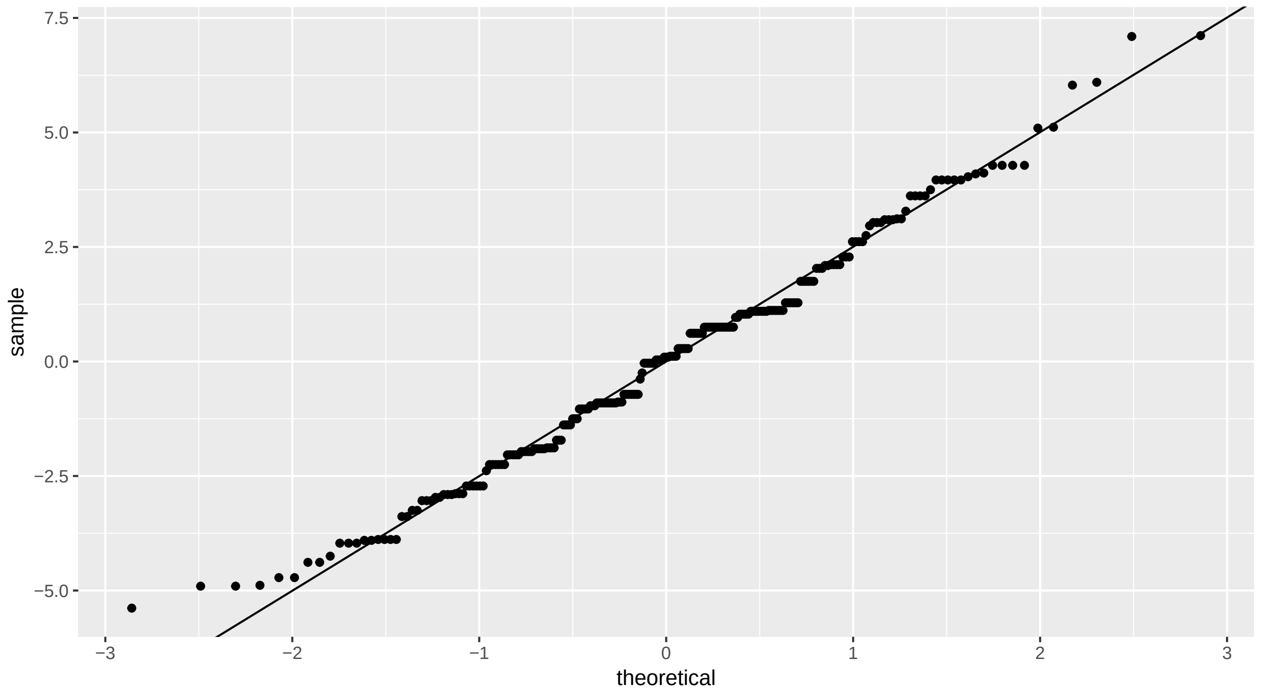
```
# Checking normality of residuals
ggplot(singer.res, aes(sample = residual)) +
  stat_qq() +
  facet_wrap(~voice.part, ncol = 2)
```



```
# QQ Plot for Normality
ggplot(singer.res, aes(sample = residual)) +
  stat_qq()
```



```
# QQ Plot for Normality and adding slope and intercept
ggplot(singer.res, aes(sample = residual)) +
  stat_qq() + geom_abline(intercept = 0,
slope = summary(singer.lm)$sigma)
```

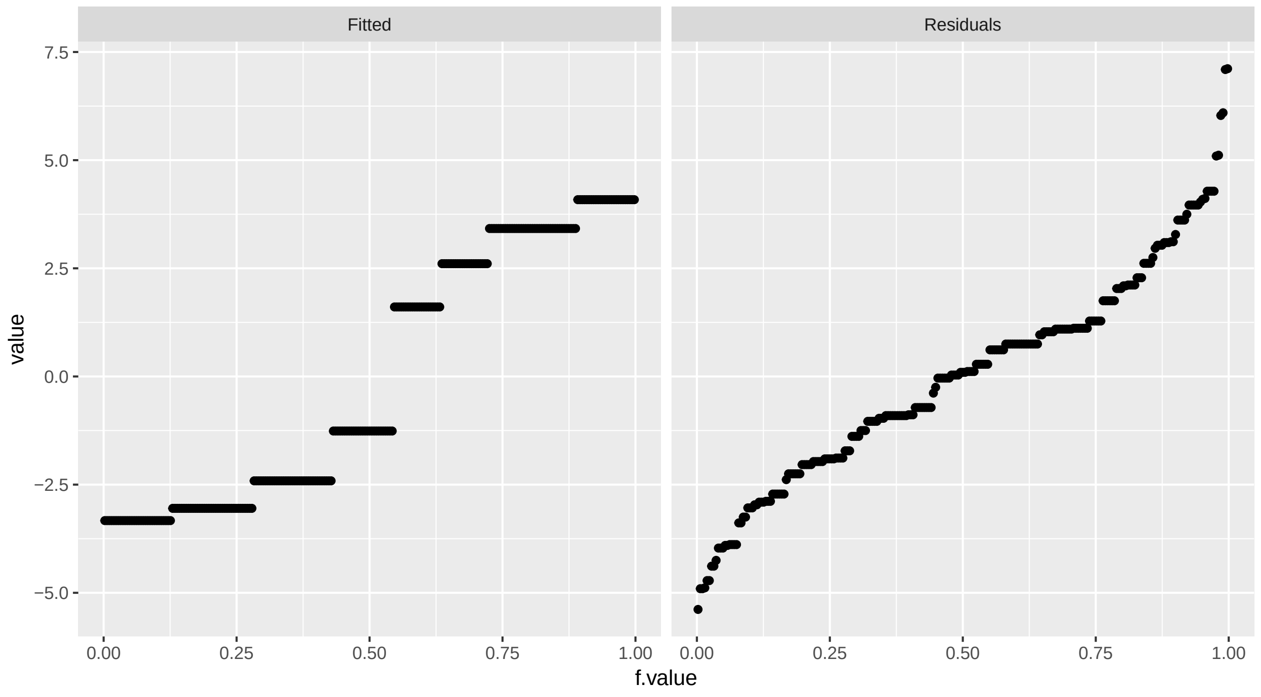


```
# Prepare Data
singer.fitted <- sort(fitted.values(singer.lm)) - mean(fitted.values(singer.lm))
singer.residuals <- sort(residuals(singer.lm))
```

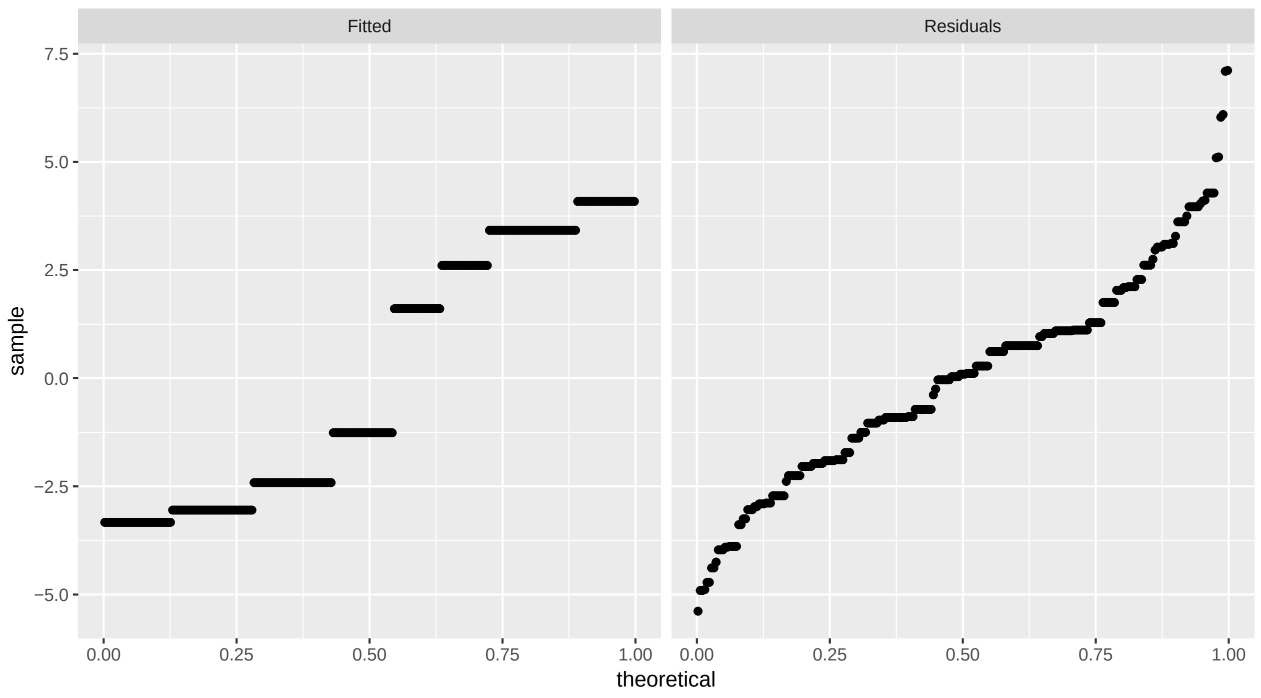
```r
n = length(singer.residuals)
f.value = (0.5:(n - 0.5))/n
singer.fit = data.frame(f.value, Fitted = singer.fitted, Residuals = singer.residuals)
```

```r
library(tidyr)
singer.fit.long = singer.fit %>% gather(type, value, Fitted:Residuals)
```

```r
# Residual-Fit spread plot
ggplot(singer.fit.long, aes(x = f.value, y = value)) +
  geom_point() +
  facet_wrap(~type)
```



```r
# Using stat_qq() to calculate f-values
ggplot(singer.fit.long, aes(sample = value)) +
  stat_qq(distribution = "qunif") +
  facet_grid(~type)
```



```r
# Using R-Squared
round(var(singer.fitted)/var(singer$height),3)
```

```
## [1] 0.585
```

```r
url <- "https://github.com/hadley/tidy-data/raw/master/data/billboard.csv"
billboard.raw <- read.csv(url, stringsAsFactors = FALSE)
```

```r
# Creating a Tibble
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
billboard <- tbl_df(billboard.raw)
billboard
```

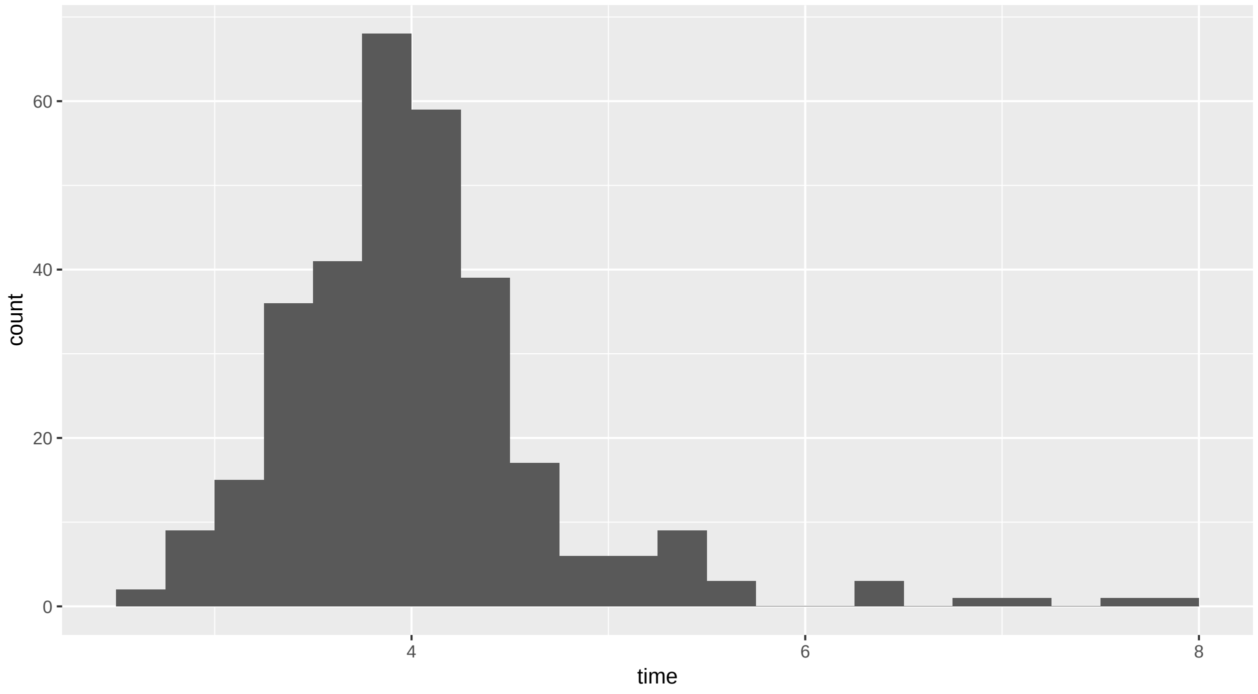
```
## # A tibble: 317 x 83
##     year artist.inverted track time  genre date.entered date.peaked
##    <int> <chr>           <chr> <chr> <chr> <chr>        <chr>
##  1  2000 Destiny's Child Inde~ 3:38  Rock  2000-09-23   2000-11-18
##  2  2000 Santana         Mari~ 4:18  Rock  2000-02-12   2000-04-08
##  3  2000 Savage Garden   I Kn~ 4:07  Rock  1999-10-23   2000-01-29
##  4  2000 Madonna         Music 3:45  Rock  2000-08-12   2000-09-16
##  5  2000 Aguilera, Chri~ Come~ 3:38  Rock  2000-08-05   2000-10-14
##  6  2000 Janet           Does~ 4:17  Rock  2000-06-17   2000-08-26
##  7  2000 Destiny's Child Say ~ 4:31  Rock  1999-12-25   2000-03-18
##  8  2000 Iglesias, Enri~ Be W~ 3:36  Latin 2000-04-01   2000-06-24
##  9  2000 Sisqo           Inco~ 3:52  Rock  2000-06-24   2000-08-12
## 10  2000 Lonestar        Amaz~ 4:25  Coun~ 1999-06-05   2000-03-04
## # ... with 307 more rows, and 76 more variables: x1st.week <int>,
## #   x2nd.week <int>, x3rd.week <int>, x4th.week <int>, x5th.week <int>,
## #   x6th.week <int>, x7th.week <int>, x8th.week <int>, x9th.week <int>,
## #   x10th.week <int>, x11th.week <int>, x12th.week <int>,
## #   x13th.week <int>, x14th.week <int>, x15th.week <int>,
## #   x16th.week <int>, x17th.week <int>, x18th.week <int>,
## #   x19th.week <int>, x20th.week <int>, x21st.week <int>,
## #   x22nd.week <int>, x23rd.week <int>, x24th.week <int>,
## #   x25th.week <int>, x26th.week <int>, x27th.week <int>,
## #   x28th.week <int>, x29th.week <int>, x30th.week <int>,
## #   x31st.week <int>, x32nd.week <int>, x33rd.week <int>,
## #   x34th.week <int>, x35th.week <int>, x36th.week <int>,
## #   x37th.week <int>, x38th.week <int>, x39th.week <int>,
## #   x40th.week <int>, x41st.week <int>, x42nd.week <int>,
## #   x43rd.week <int>, x44th.week <int>, x45th.week <int>,
## #   x46th.week <int>, x47th.week <int>, x48th.week <int>,
## #   x49th.week <int>, x50th.week <int>, x51st.week <int>,
## #   x52nd.week <int>, x53rd.week <int>, x54th.week <int>,
## #   x55th.week <int>, x56th.week <int>, x57th.week <int>,
## #   x58th.week <int>, x59th.week <int>, x60th.week <int>,
## #   x61st.week <int>, x62nd.week <int>, x63rd.week <int>,
## #   x64th.week <int>, x65th.week <int>, x66th.week <lgl>,
## #   x67th.week <lgl>, x68th.week <lgl>, x69th.week <lgl>,
## #   x70th.week <lgl>, x71st.week <lgl>, x72nd.week <lgl>,
## #   x73rd.week <lgl>, x74th.week <lgl>, x75th.week <lgl>, x76th.week <lgl>
```

```r
# Calculating time-length of each song
billboard.time <- strsplit(billboard$time, ":")
billboard.time <- matrix(unlist(billboard.time), byrow = T, ncol = 2)
billboard.mins <- as.numeric(billboard.time[, 1])
billboard.secs <- as.numeric(billboard.time[, 2])
billboard.time <- billboard.mins * 60 + billboard.secs
billboard$time <- billboard.time/60
```

```r
billboard.long <- billboard %>%
  gather(week, rank, x1st.week:x76th.week, na.rm = TRUE)
billboard.long
```

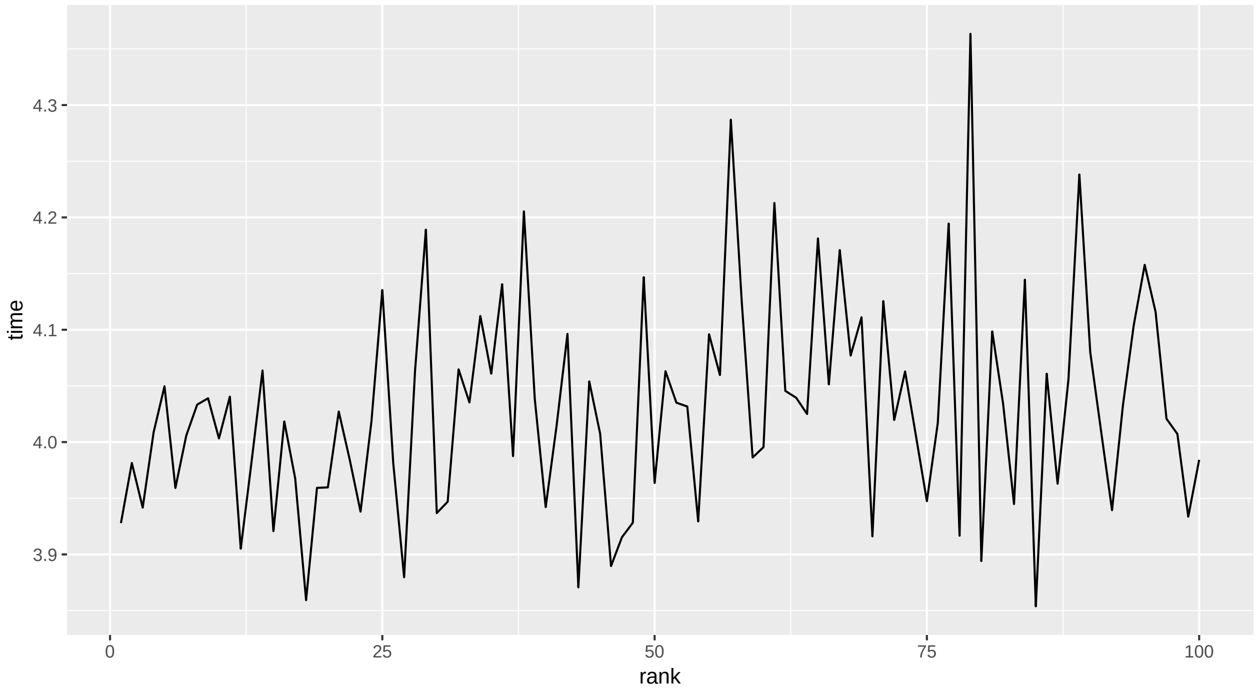
```
## # A tibble: 5,307 x 9
##     year artist.inverted track  time genre date.entered date.peaked week
##    <int> <chr>           <chr> <dbl> <chr> <chr>        <chr>       <chr>
##  1  2000 Destiny's Child Inde~  3.63 Rock  2000-09-23   2000-11-18  x1st~
##  2  2000 Santana         Mari~  4.3  Rock  2000-02-12   2000-04-08  x1st~
##  3  2000 Savage Garden   I Kn~  4.12 Rock  1999-10-23   2000-01-29  x1st~
##  4  2000 Madonna         Music  3.75 Rock  2000-08-12   2000-09-16  x1st~
##  5  2000 Aguilera, Chri~ Come~  3.63 Rock  2000-08-05   2000-10-14  x1st~
##  6  2000 Janet           Does~  4.28 Rock  2000-06-17   2000-08-26  x1st~
##  7  2000 Destiny's Child Say ~  4.52 Rock  1999-12-25   2000-03-18  x1st~
##  8  2000 Iglesias, Enri~ Be W~  3.6  Latin 2000-04-01   2000-06-24  x1st~
##  9  2000 Sisqo           Inco~  3.87 Rock  2000-06-24   2000-08-12  x1st~
## 10  2000 Lonestar        Amaz~  4.42 Coun~ 1999-06-05   2000-03-04  x1st~
## # ... with 5,297 more rows, and 1 more variable: rank <int>
```

```r
ggplot(billboard, aes(x = time)) +
  geom_histogram(breaks = seq(2.5, 8, 0.25))
```



```r
# Finding mean song length
time.means = aggregate(time ~ rank, FUN = mean, data = billboard.long)
```

```r
# Plotting mean song lenghts
ggplot(time.means, aes(x = rank, y = time)) + geom_line()
```



### Transformations

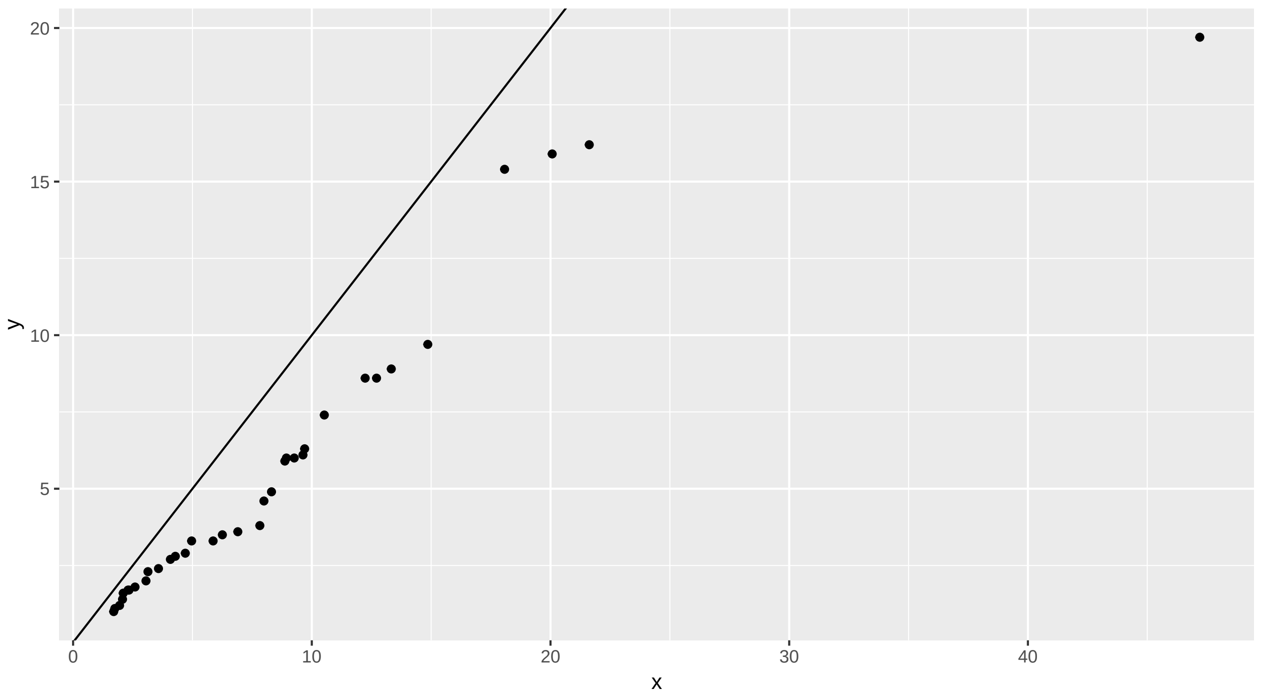
### Log Transformation

```r
# Load data from workspace
load("./data/lattice.RData")
```

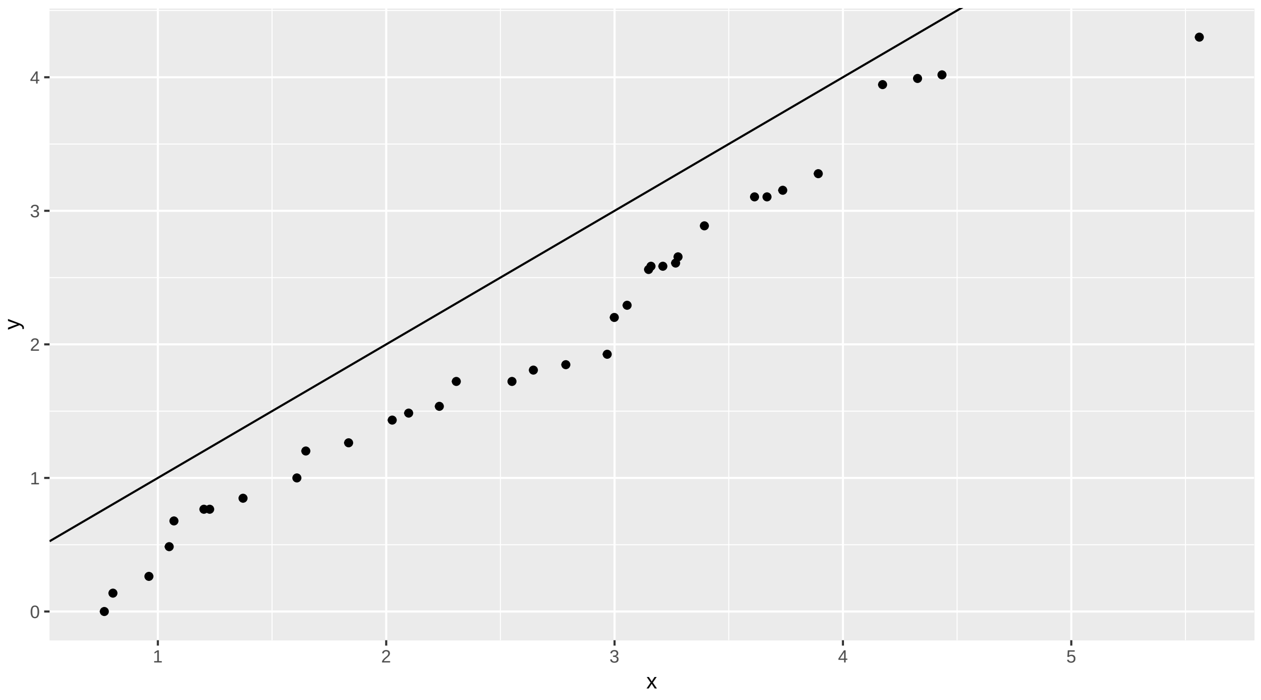
```r
summary(fusion.time)
```

```
##       time         nv.vv
##  Min.   : 1.000   NV:43
##  1st Qu.: 2.475   VV:35
##  Median : 5.250
##  Mean   : 7.210
##  3rd Qu.: 9.050
##  Max.   :47.200
```

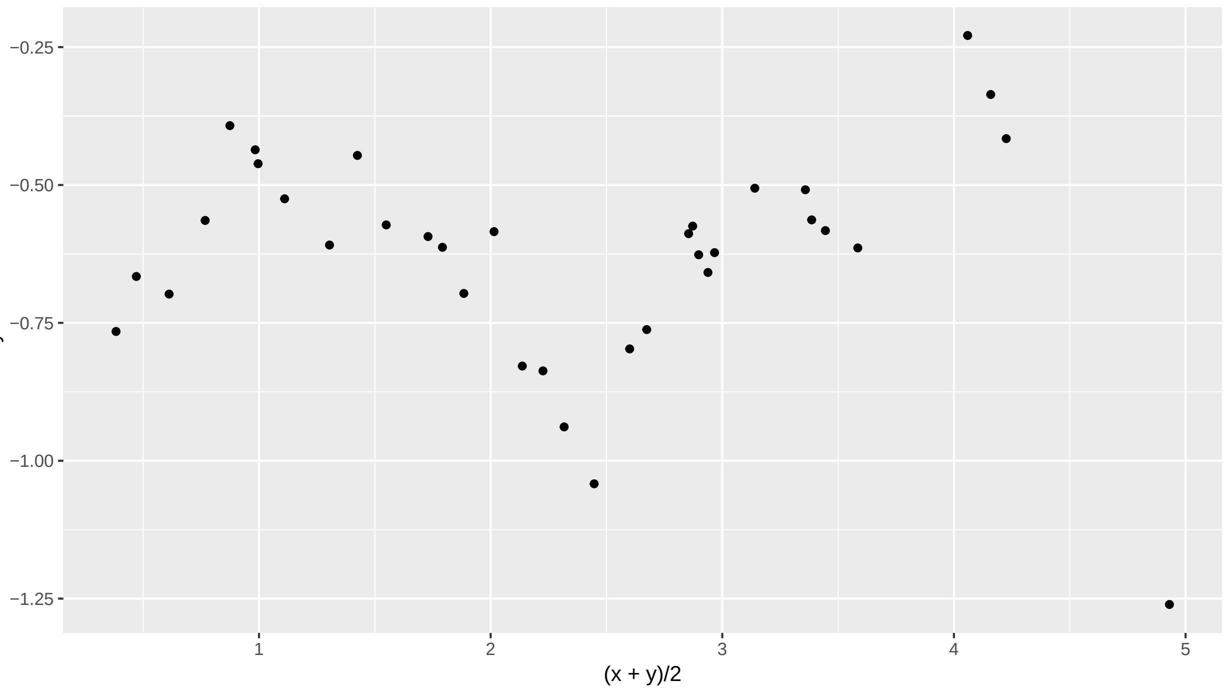
```r
# Plot without data transformation
time = fusion.time$time
nv.vv = fusion.time$nv.vv
NV.times = sort(time[nv.vv == "NV"])
VV.times = sort(time[nv.vv == "VV"])
NV.VV.qq = as.data.frame(qqplot(NV.times, VV.times, plot.it = FALSE))

ggplot(NV.VV.qq, aes(x, y)) + geom_point() + geom_abline()
```



```r
# Using a log transformation
NV.times.log = sort(log2(time[nv.vv == "NV"]))
VV.times.log = sort(log2(time[nv.vv == "VV"]))
NV.VV.qq.log = as.data.frame(qqplot(NV.times.log, VV.times.log, plot.it = FALSE))

ggplot(NV.VV.qq.log, aes(x, y)) + geom_point() + geom_abline()
```

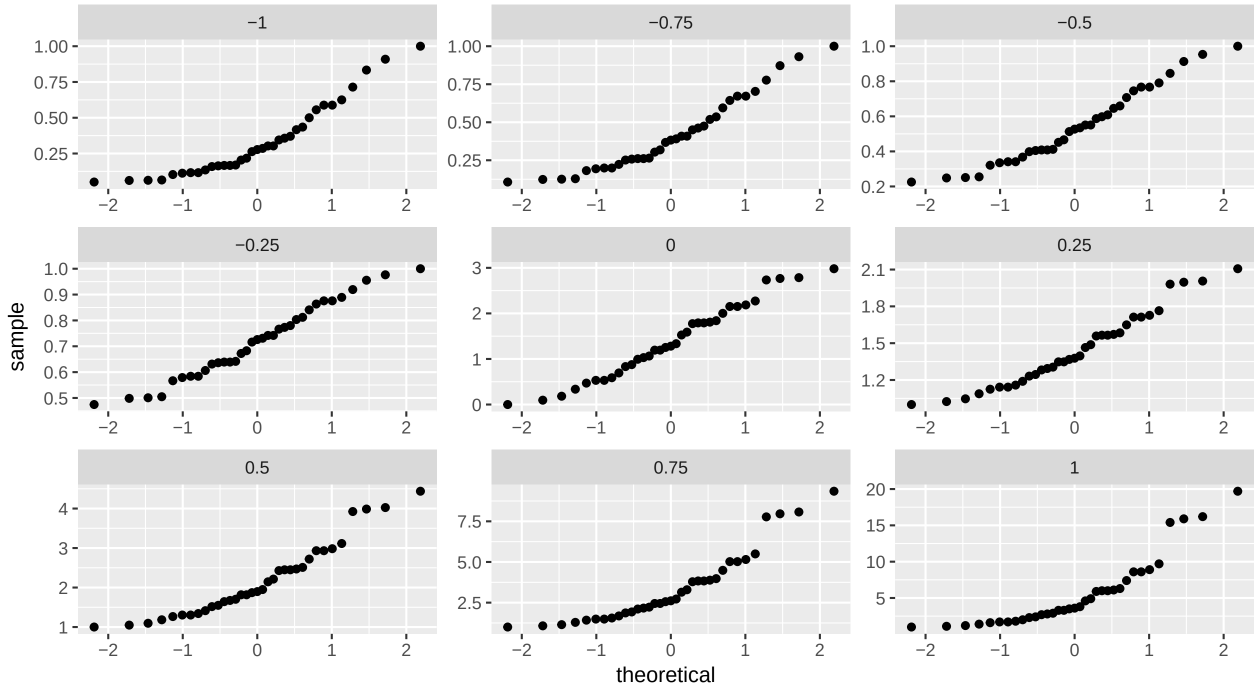


```r
# Tukey Mean-Difference Plot
ggplot(NV.VV.qq.log, aes((x + y)/2, y - x)) + geom_point()
```



### Power Transformation

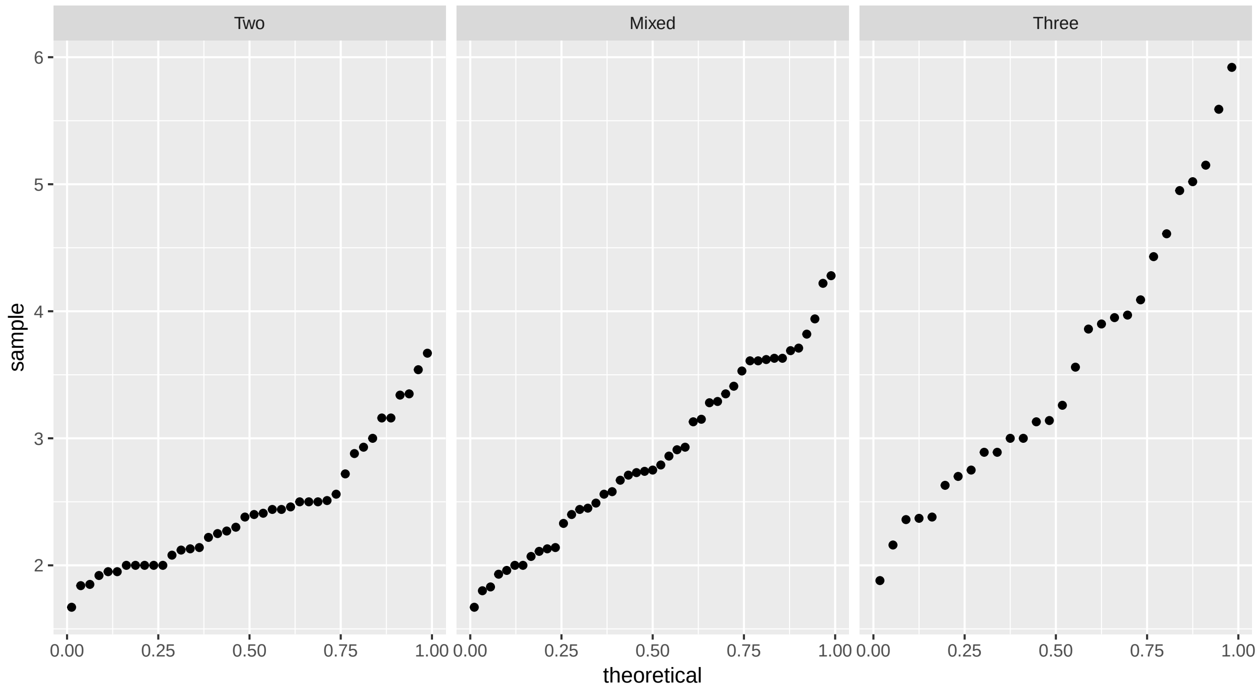
```r
n.VV = length(VV.times)
power = rep(seq(-1, 1, 0.25), each = n.VV)
VV.time = c(VV.times^-1, VV.times^-0.75,
            VV.times^-0.5, VV.times^-0.25,
            log(VV.times), VV.times^0.25,
            VV.times^0.5, VV.times^0.75, VV.times)

ggplot(data.frame(power, VV.time), aes(sample = VV.time)) +
  stat_qq() + facet_wrap(~power, scales = "free")
```



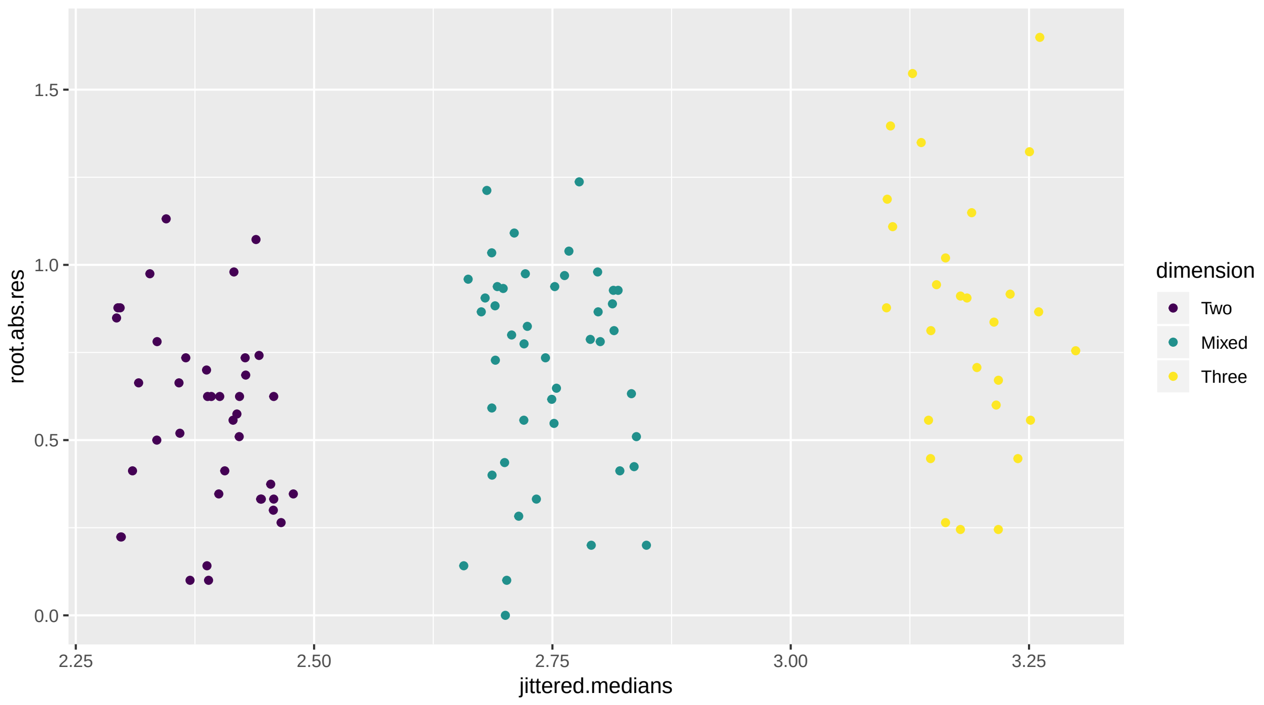
```
summary(food.web)
```

```
##   mean.length    dimension
##  Min.   :1.670   Two  :40
##  1st Qu.:2.220   Mixed:45
##  Median :2.710   Three:28
##  Mean   :2.885
##  3rd Qu.:3.350
##  Max.   :5.920
```

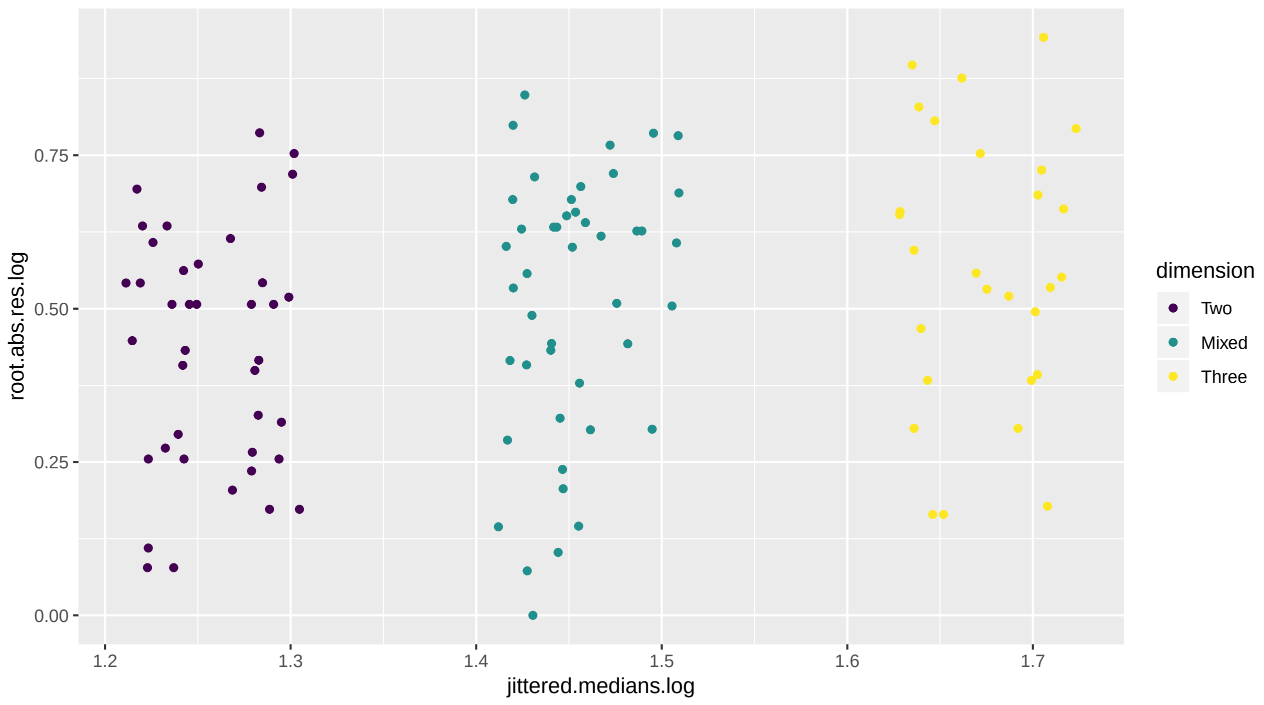
```
ggplot(food.web, aes(sample = mean.length)) +
  stat_qq(distribution = qunif) + facet_grid(~dimension)
```



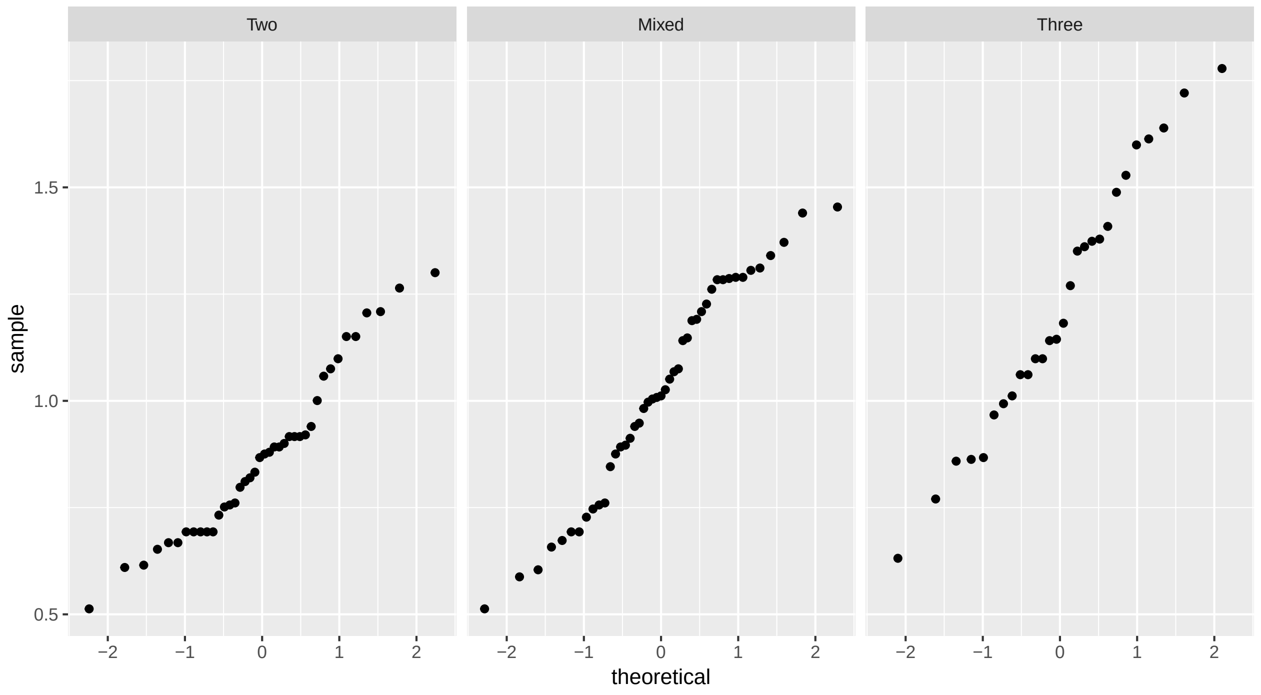
```
web.length = food.web$mean.length
dimension = food.web$dimension
n = nrow(food.web)
median.3 = median(web.length[dimension == "Three"])
median.2 = median(web.length[dimension == "Two"])
median.mixed = median(web.length[dimension == "Mixed"])
group.median = rep(NA, n)
group.median[dimension == "Three"] = median.3
group.median[dimension == "Two"] = median.2
group.median[dimension == "Mixed"] = median.mixed
jittered.medians = group.median + runif(n, -0.1, 0.1)
root.abs.res = sqrt(abs(web.length - group.median))
food.web.sl = data.frame(jittered.medians, root.abs.res, dimension)
ggplot(food.web.sl, aes(jittered.medians, root.abs.res, col = dimension)) +
geom_point()
```



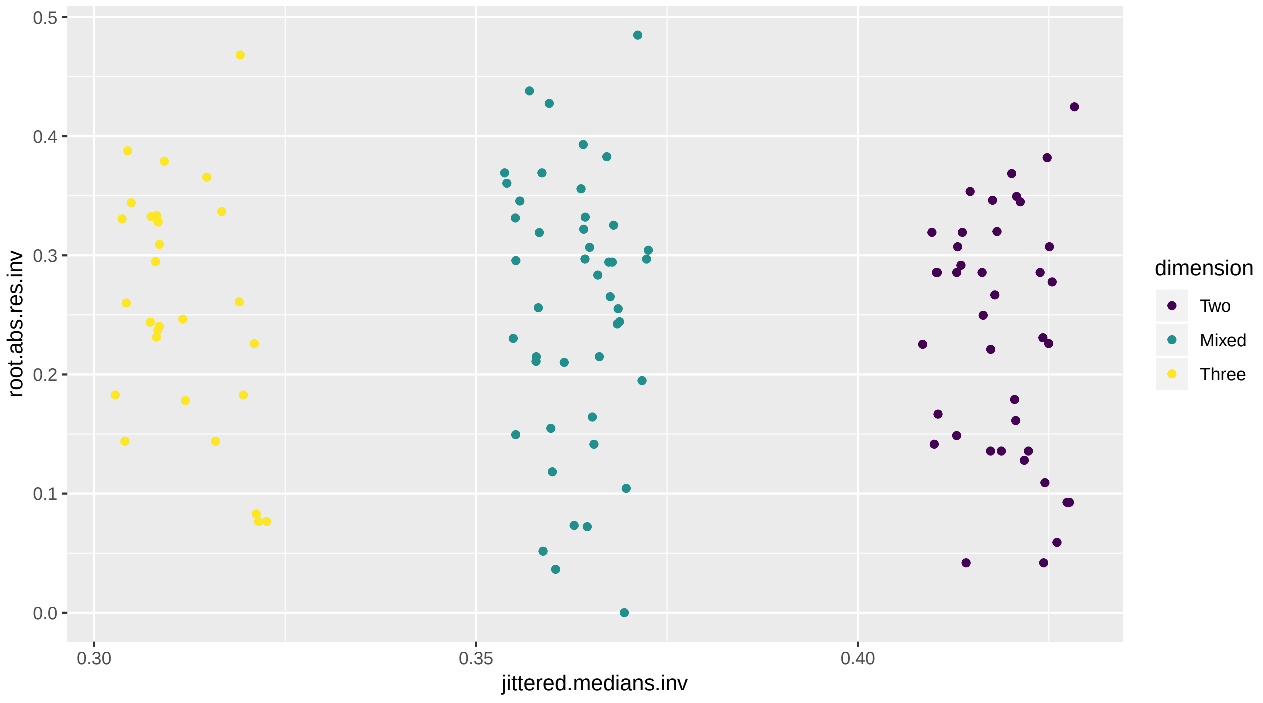
```r
# Using log transformation

log.web.length = log2(food.web$mean.length)
median.3.log = median(log.web.length[dimension == "Three"])
median.2.log = median(log.web.length[dimension == "Two"])
median.mixed.log = median(log.web.length[dimension == "Mixed"])
group.median.log = rep(NA, n)
group.median.log[dimension == "Three"] = median.3.log
group.median.log[dimension == "Two"] = median.2.log
group.median.log[dimension == "Mixed"] = median.mixed.log
jittered.medians.log = group.median.log + runif(n, -0.05, 0.05)
root.abs.res.log = sqrt(abs(log.web.length - group.median.log))
food.web.log.sl = data.frame(jittered.medians.log, root.abs.res.log, dimension)
ggplot(food.web.log.sl, aes(jittered.medians.log, root.abs.res.log, col = dimension)) +
geom_point()
```



```
ggplot(food.web, aes(sample = log(mean.length))) +
  stat_qq() +
  facet_wrap(~dimension)
```



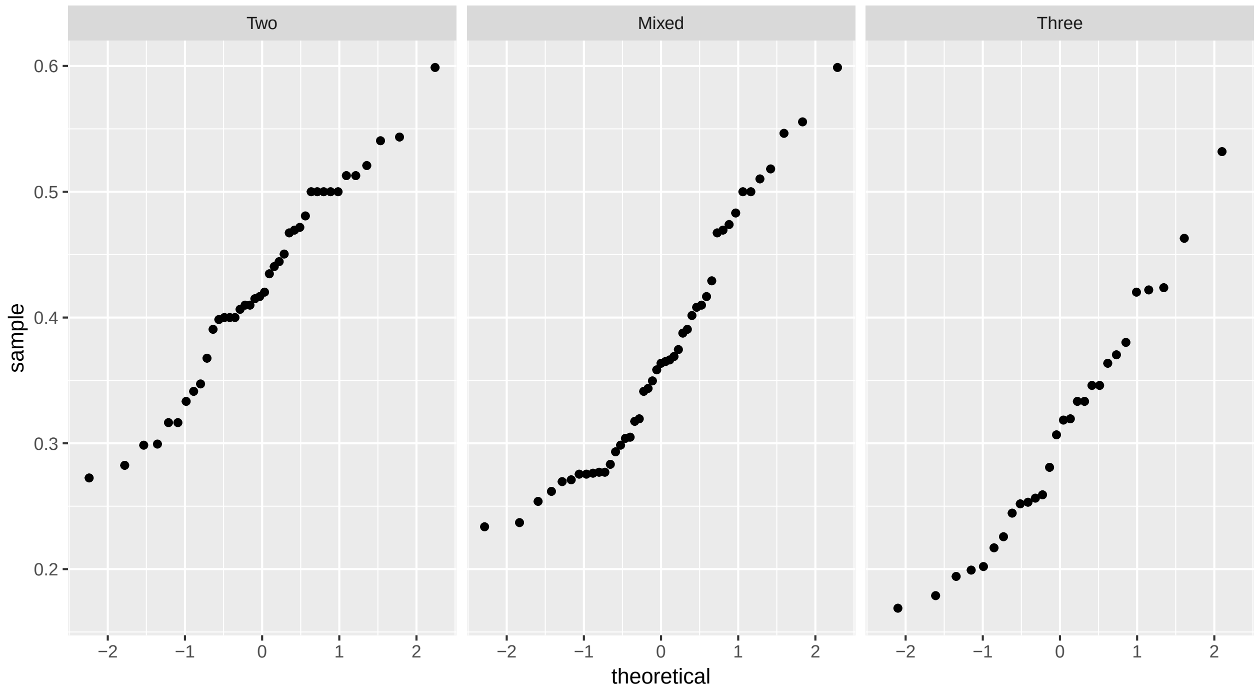
```
# Using Inverse Transformation
inv.web.length = 1/food.web$mean.length
median.3.inv = median(inv.web.length[dimension == "Three"])
median.2.inv = median(inv.web.length[dimension == "Two"])
median.mixed.inv = median(inv.web.length[dimension == "Mixed"])
group.median.inv = rep(NA, n)
group.median.inv[dimension == "Three"] = median.3.inv
group.median.inv[dimension == "Two"] = median.2.inv
group.median.inv[dimension == "Mixed"] = median.mixed.inv
jittered.medians.inv = group.median.inv + runif(n, -0.01, 0.01)
root.abs.res.inv = sqrt(abs(inv.web.length - group.median.inv))
food.web.inv.sl = data.frame(jittered.medians.inv, root.abs.res.inv, dimension)
ggplot(food.web.inv.sl, aes(jittered.medians.inv, root.abs.res.inv, col = dimension)) +
geom_point()
```



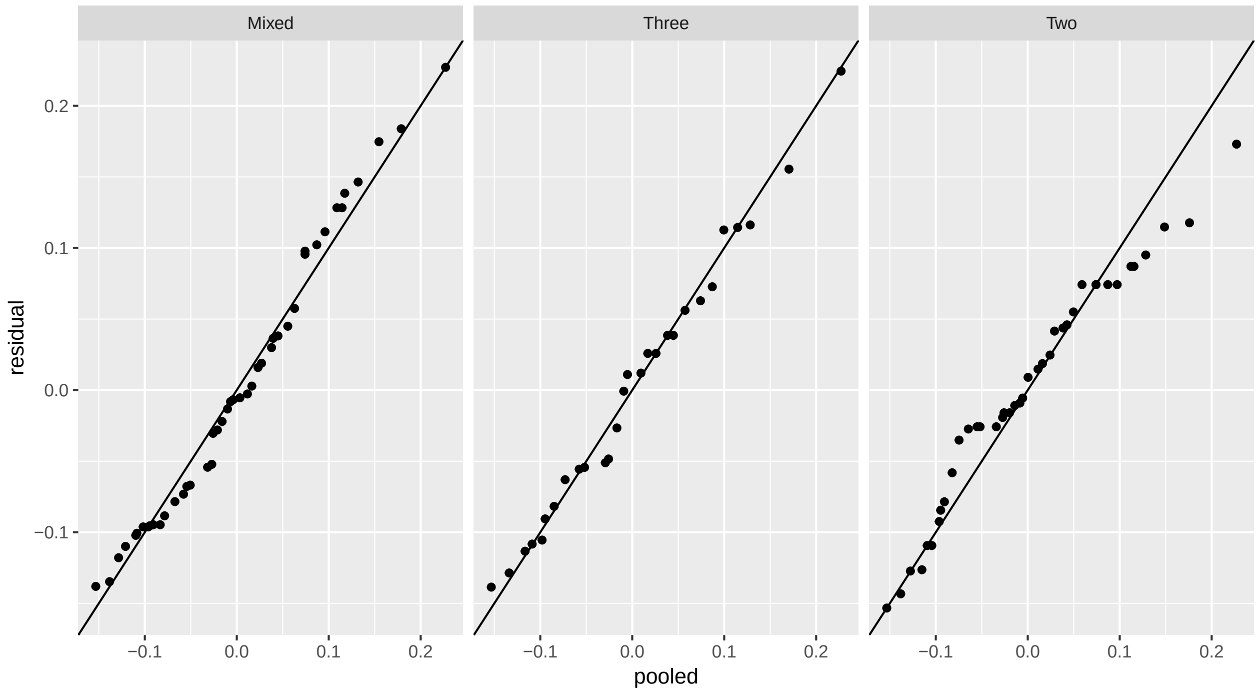
```
# Calculate Mean
aggregate(root.abs.res.inv ~ dimension, FUN = mean)
```

```
##   dimension root.abs.res.inv
## 1       Two        0.2332382
## 2     Mixed        0.2563367
## 3     Three        0.2579944
```

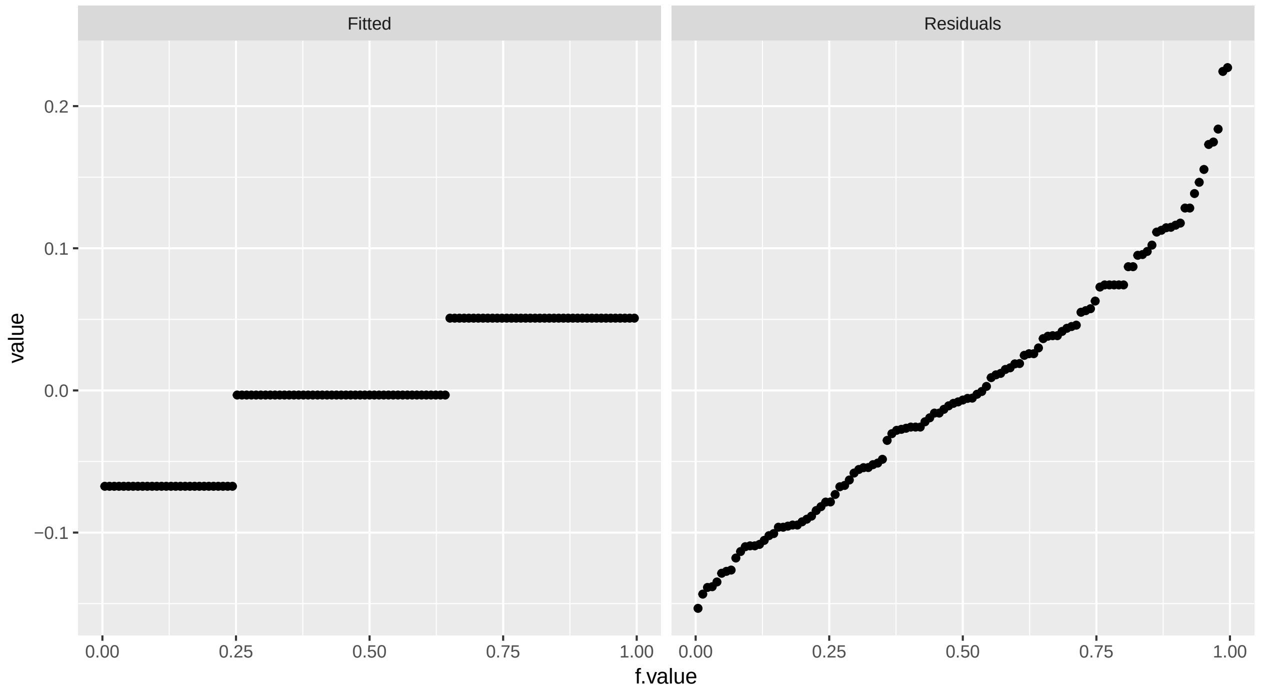
```
# Normality Check
ggplot(food.web, aes(sample = 1/mean.length)) + stat_qq() + facet_wrap(~dimension)
```



```
food.web.lm = lm(inv.web.length ~ dimension)
food.web.res = residuals(food.web.lm)
res.qq.3 = qqplot(food.web.res, food.web.res[dimension == "Three"], plot.it = FALSE)
res.qq.2 = qqplot(food.web.res, food.web.res[dimension == "Two"], plot.it = FALSE)
res.qq.mixed = qqplot(food.web.res, food.web.res[dimension == "Mixed"], plot.it = FALSE)
food.web.res.qq = data.frame(pooled = c(res.qq.3$x, res.qq.2$x, res.qq.mixed$x),
residual = c(res.qq.3$y, res.qq.2$y, res.qq.mixed$y), dimension = c(rep("Three",
length(res.qq.3$x)), rep("Two", length(res.qq.2$x)), rep("Mixed", length(res.qq.mixed$x))))
ggplot(food.web.res.qq, aes(pooled, residual)) + geom_point() + geom_abline() +
facet_wrap(~dimension)
```



```
# Residual Fitted Spread Plot
food.web.fitted = sort(fitted.values(food.web.lm)) - mean(fitted.values(food.web.lm))
n = length(inv.web.length)
f.value = (0.5:(n - 0.5))/n
food.web.fit = data.frame(f.value, Fitted = food.web.fitted, Residuals = sort(food.web.res))
library(tidyr)
food.web.fit.long = food.web.fit %>% gather(type, value, Fitted:Residuals)
ggplot(food.web.fit.long, aes(x = f.value, y = value)) + geom_point() + facet_wrap(~type)
```


Fitted
Residuals
value
f.value