App vs Web App

Luciën Martijn, Stijn Meijerink, Daniël Hana, Terry Bommels, Dion David Haneveld

23 October 2020

```r
# Include libraries used in lectures
library(tidyverse)
library(car)
library(bestNormalize)
library(ggplot2)
library(effsize)
library(xtable)
library(EnvStats)
```

First we read recursively all data collected to construct one big dataframe.

```r
setwd('..')

# Get all joule_results.*.csv
csv_paths <- list.files(path = './dataset/',
                        recursive = TRUE,
                        pattern = "^Joule.*\\.csv",
                        full.names = TRUE)
# read all csv
app_data <- csv_paths %>%
  lapply(read_csv) %>%
  bind_rows

# Add paths to data frame for reading out the correct names later
app_data['path'] = csv_paths

# Based on the paths get the device tier and add it to the dataframe
app_data['device'] <- csv_paths %>%
  strsplit('-', fixed = TRUE) %>%
  rapply(nth, n=1) %>%
  strsplit('/', fixed = TRUE) %>%
  rapply(nth, n=6) %>%
  factor

extract_app_type <- function(splitString){
  temp <- strsplit(x = splitString, split = '-', fixed = TRUE)
  result <- unlist(temp)[3]
  if(str_detect(result, 'www')){
    result <- "web"
  }
  if(str_detect(result, 'runner')){
    result <- "native"
  }
  return(result)
}
# Extract app type based on path
app_type <- app_data$path %>%
  lapply(extract_app_type)
app_data['app_type']<- unlist(app_type)

extract_app_name <- function(path)  {
  temp <- strsplit(x = path, split = '/', fixed = TRUE)
  result <- unlist(temp)[7]
  if(str_detect(result, '(.*https-www.*)')){
    result <- unlist(strsplit(x = result, split = '-'))[3]
  }
  if(str_detect(result, '(.*android-runner-experiments.*)')){
    result <- unlist(strsplit(x = result, split = '-'))[6]
  }
  return(result)
}

# Extract app names based on path
app_names <- app_data$path %>%
  lapply(extract_app_name)
app_data['app'] <- unlist(app_names)

updateMisalignedAppNames <- function(app){
  if(app == 'alibaba'){
    app = 'aliexpress'
  } else if(app == 'inditex'){
    app = 'zara'
  } else if(app == 'ninegag'){
    app = '9gag'
  } else if(app == 'google'){
    app = 'youtube'
  }else {
    app = app
  }
}

# The native app name for aliexpress is Alibaba so we update it to aliexpress
updatedAppNames <- app_data$app %>%
  lapply(updateMisalignedAppNames)

# Add the updated names to the dataframe again
app_data['app'] <- unlist(updatedAppNames)

app_data$app_type <- as.factor(app_data$app_type)
app_data$app <- as.factor(app_data$app)
```

```r
# Show a part of the data frame
tail(app_data)
```

```
## # A tibble: 6 x 5
##   Joule_calculated path                                            device app_type app
##              <dbl> <chr>                                           <fct>  <fct>    <fct>
## 1              661. ./dataset//low.end.web/data/low-end/ht~ low    web      zara
## 2              626. ./dataset//low.end.web/data/low-end/ht~ low    web      zara
## 3              640. ./dataset//low.end.web/data/low-end/ht~ low    web      zara
## 4              626. ./dataset//low.end.web/data/low-end/ht~ low    web      zara
## 5              635. ./dataset//low.end.web/data/low-end/ht~ low    web      zara
## 6              618. ./dataset//low.end.web/data/low-end/ht~ low    web      zara
```

Let us also see if all treatments are present. Indeed for all combination we have $8 * 15 = 120$ observations.

```
table(app_data$device, app_data$app_type)
```

```
##
##         native web
##   high     120 120
##   low      120 120
```

And write the full results to a file, to distribute for the replication package.

```
write_csv(path = './full_results.csv', x = app_data)
# Create plots dir
dir.create("./plots")
```

1. Discriptive statistics

```
# Define a summarize data function
summarize_data <- function(data){
  data %>%
  group_by(app) %>%
  summarize(n = n(),
            mean = mean(Joule_calculated),
            median = median(Joule_calculated),
            sd = sd(Joule_calculated),
            IQR = IQR(Joule_calculated),
            min = min(Joule_calculated),
            max = max(Joule_calculated))
}

total_summarize <- summarize_data(app_data)
total_summarize
```

```
## # A tibble: 8 x 8
##   app              n  mean median    sd   IQR   min   max
##   <fct>        <int> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 9gag            60  334.   295. 189.  283.   96.4  778.
## 2 aliexpress      60  242.   213. 101.  107.  107.   483.
## 3 deliveroo       60  181.   169.  61.1  73.0 103.   287.
## 4 reddit          60  224.   169.  99.8 152.  110.   625.
## 5 tripadvisor     60  240.   185. 145.  179.   90.5  506.
## 6 weather         60  261.   180. 167.  295.  102.   665.
## 7 youtube         60  219.   199.  71.8 130.  141.   486.
## 8 zara            60  264.   164. 217.  261.  102.   661.
```

```
print(xtable(total_summarize, type = "latex"))
```

```
## % latex table generated in R 3.6.2 by xtable 1.8-4 package
## % Sun Oct 25 22:07:06 2020
## \begin{table}[ht]
## \centering
## \begin{tabular}{rlrrrrrrr}
##   \hline
##  & app & n & mean & median & sd & IQR & min & max \\
##   \hline
## 1 & 9gag &  60 & 333.90 & 294.82 & 188.70 & 283.04 & 96.45 & 777.64 \\
##   2 & aliexpress &  60 & 242.01 & 212.94 & 100.89 & 107.06 & 106.90 & 483.22 \\
##   3 & deliveroo &  60 & 180.88 & 169.42 & 61.13 & 72.97 & 102.53 & 286.58 \\
##   4 & reddit &  60 & 224.39 & 168.96 & 99.85 & 152.08 & 110.33 & 624.61 \\
##   5 & tripadvisor &  60 & 240.12 & 185.36 & 144.63 & 179.00 & 90.51 & 505.83 \\
##   6 & weather &  60 & 260.77 & 179.50 & 166.86 & 295.19 & 102.22 & 664.56 \\
##   7 & youtube &  60 & 219.14 & 198.92 & 71.79 & 129.65 & 141.08 & 486.44 \\
##   8 & zara &  60 & 264.45 & 163.71 & 216.65 & 260.76 & 102.38 & 661.46 \\
##    \hline
## \end{tabular}
## \end{table}
```

The full data frame is summarised. We see quite some big standard deviations. If we now group by device we see a clear pattern arrise.

```
app_data %>%
  group_by(app_type, device) %>%
  summarise(mean = mean(Joule_calculated),
            sd = sd(Joule_calculated))
```

```
## # A tibble: 4 x 4
## # Groups:   app_type [2]
##   app_type device  mean    sd
##   <fct>    <fct>  <dbl> <dbl>
## 1 native   high    131.  25.6
## 2 native   low     261. 117.
## 3 web      high    175.  29.7
## 4 web      low     416. 152.
```

This shows us that the mean of high-end data is lower. Furthermore we observe that the standard deviation is much less at the high-end deivce tier. The last observation we can make is that in general app_type web consumes more energy.

Visualizing the full data frame with a boxplot results in the following plot.

```
jpeg(file="./plots/boxplot_full.jpeg")
boxplot(app_data$Joule_calculated, main = "Joules Distribution of entire dataset",  ylab = "Joules", co
dev.off()
```

```
## pdf
##   2
```

We see a positively skewed dataset with quite some outliers.

Let us split this up on device type and we see two more normal distributed boxplots. However for a low-end device the distribution still has a heavy right tail.

```
jpeg(file="./plots/boxplot_device.jpeg")
boxplot(Joule_calculated ~ device,
  data = app_data,
  main = "Joules Distribution by device tier",
  xlab = "Device tier",
  ylab = "Joules",
  col = "steelblue",
  border = "black",
  las = 2 #make x-axis labels perpendicular
)
dev.off()
```

```
## pdf
##   2
```

```
jpeg(file="./plots/boxplot_device_app_type.jpeg")
boxplot(Joule_calculated ~ device : app_type,
  data = app_data,
  main = "Joules Distribution by app type by device",
  xlab = "device:app type",
  ylab = "Joules",
  col = "steelblue",
  border = "black",
  las = 1 #make x-axis labels perpendicular
)
dev.off()
```

```
## pdf
##   2
```

Heavy right tails still on the low-end data. Some outlier on the high web.

This lets us conclude that we should take the device tier as blocking factor. For the rest of the analysis we therefore will use this as blocking factor. Therefore a two way ANOVA is not possible anymore, because we are missing the interaction factor in the model. This however also splits RQ1 in 2 sub-questions.

```
# To account for both treatments on one subject (in this case app) we aggregate the data as follows:
mean_device_app_type <- app_data %>%
  group_by(app_type, device, app) %>%
  summarise(mean(Joule_calculated)) %>%
  arrange(app)

names(mean_device_app_type)[names(mean_device_app_type)=="mean(Joule_calculated)"] <- "Joule"

# This leaves us with 32 rows = 2 (app_type) * 2 (device tiers) * 8 (apps)
```

2. Hypothesis Testing

The back-up statisical test that we get to know is the one way ANOVA, because we are left with one response variable and one explainatiry factor (fixed on two levels). However this is equivalent to a paired t-test. Therefore we will conduct a paired t test with a split on device tier, to account for the blocking factor.

Assumption 1

The subjects should be selected randomly from the population. This has been accounted for in the experiment setup.

Assumption 2

The differences between the pairs should be approximately normally distributed

```r
check_normality <- function(data){
  plot(density(data))
  car::qqPlot(data)
  shapiro.test(data)
}
```

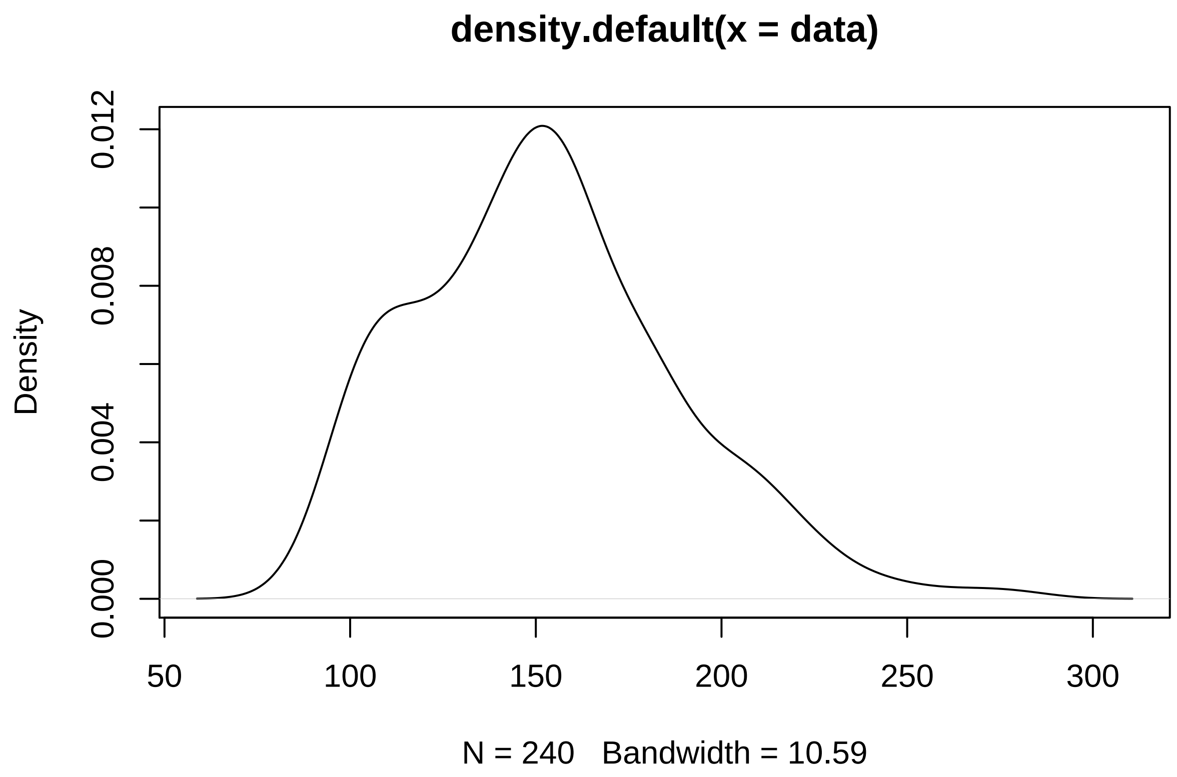
This is aggregated over 15 datapoints per subject. Here for both low-end and high-end devices differences are taken. This is not good, as device type is blocking factor.

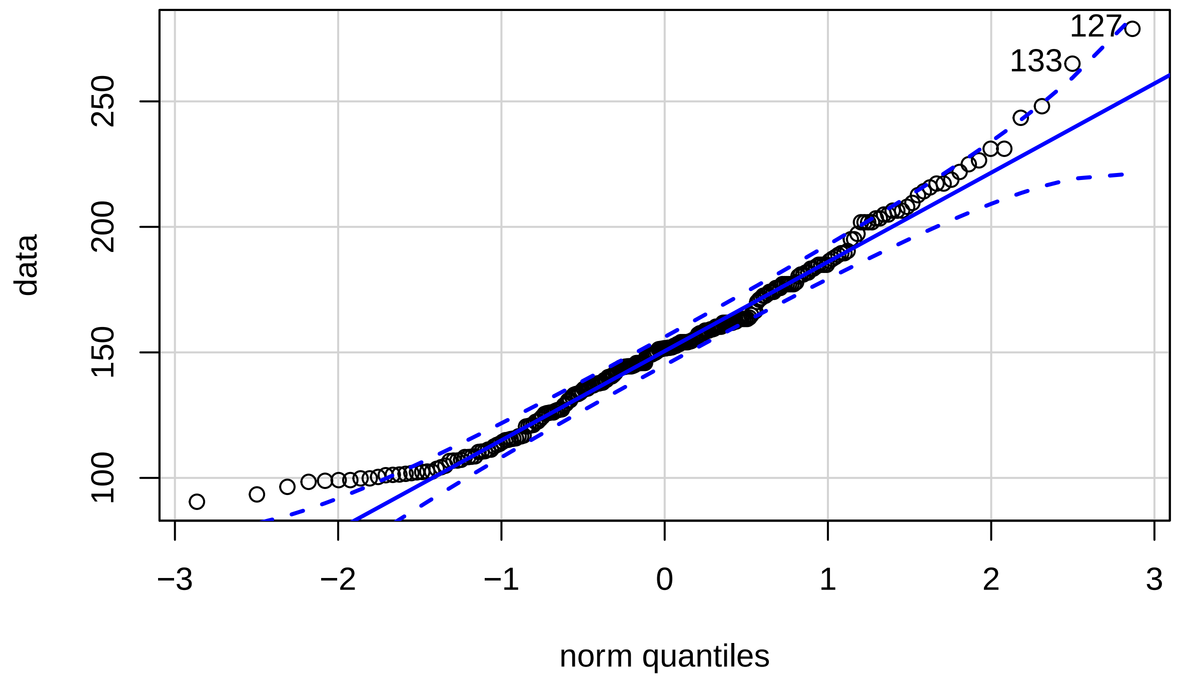
```r
mean_device_app_type_high <- app_data %>%
  filter(device == 'high') %>%
  arrange(app_type, app)

mean_device_app_type_low <- app_data %>%
  filter(device == 'low')%>%
  arrange(app_type, app)

# differences <- with(mean_device_app_type, Joule[app_type == "web"] - Joule[app_type == "native"])
```

Split dataset in high-tier & low-tier due to blocking factor device type (each group of 210 points)

```r
mean_device_app_type_high <- app_data %>%
  filter(device == 'high') %>%
  arrange(app_type, app)

mean_device_app_type_low <- app_data %>%
  filter(device == 'low') %>%
  arrange(app_type, app)
```

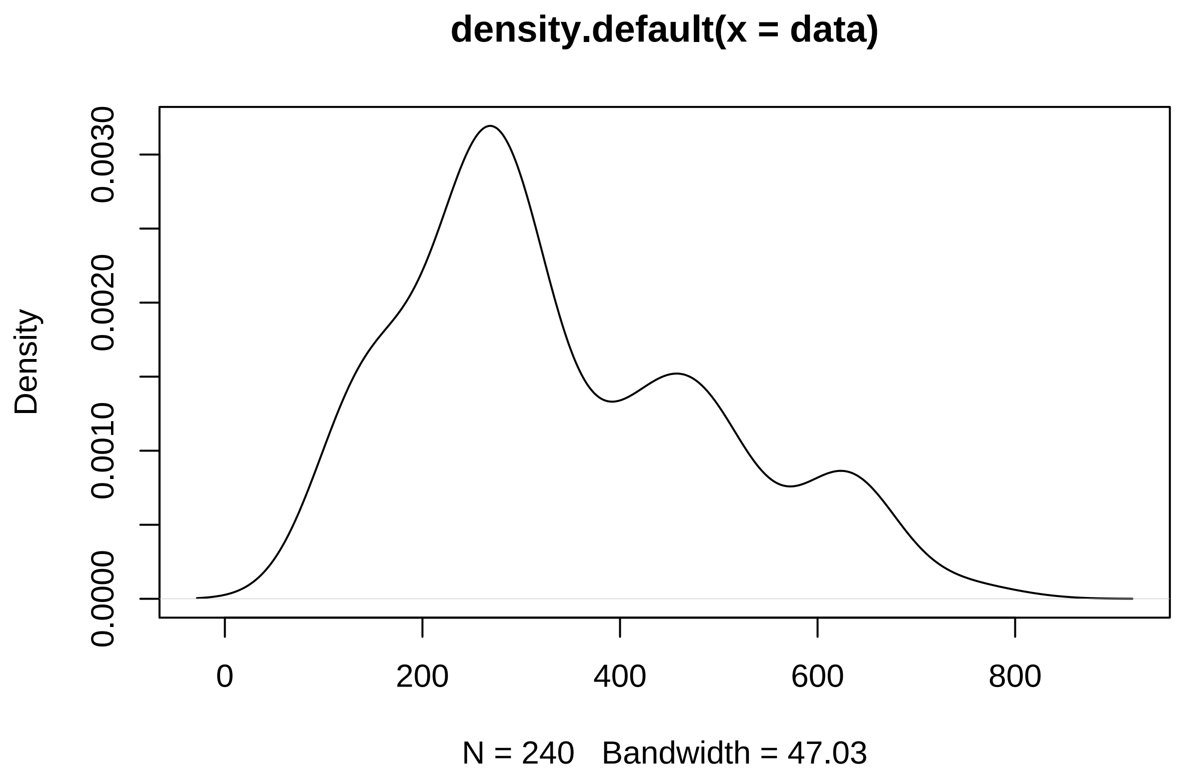
```r
check_normality(mean_device_app_type_high$Joule_calculated)
```

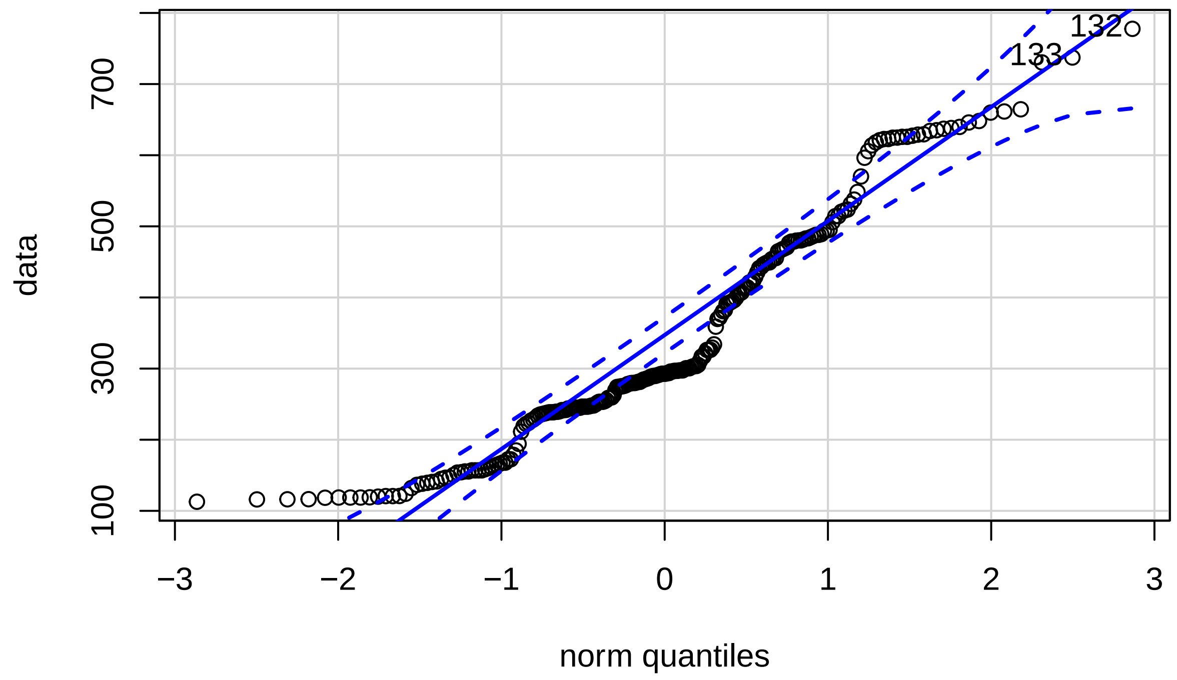

density.default(x = data)
Density
0.000
0.004
0.008
0.012
50
100
150
200
250
300
N = 240   Bandwidth = 10.59



```
##
##  Shapiro-Wilk normality test
##
## data:  data
## W = 0.97051, p-value = 6.948e-05
```

```
check_normality(mean_device_app_type_low$Joule_calculated)
```

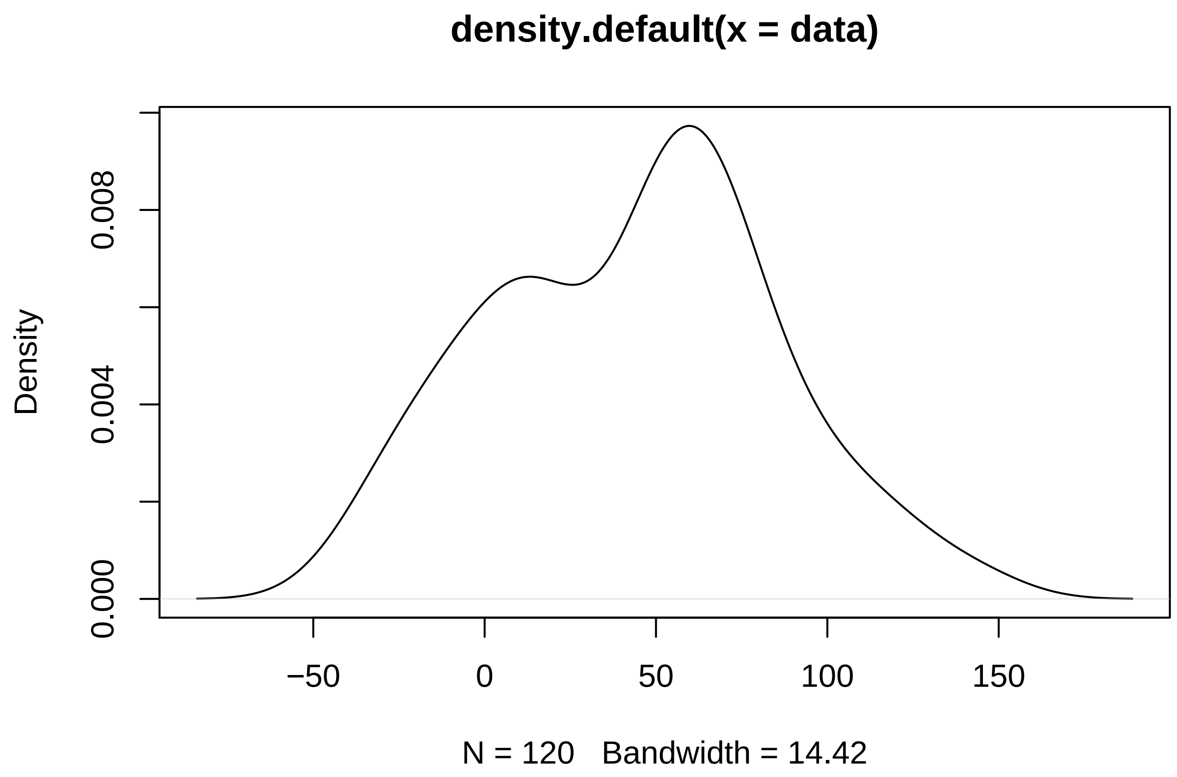

density.default(x = data)
Density
0.0000
0.0010
0.0020
0.0030
0
200
400
600
800
N = 240 Bandwidth = 47.03

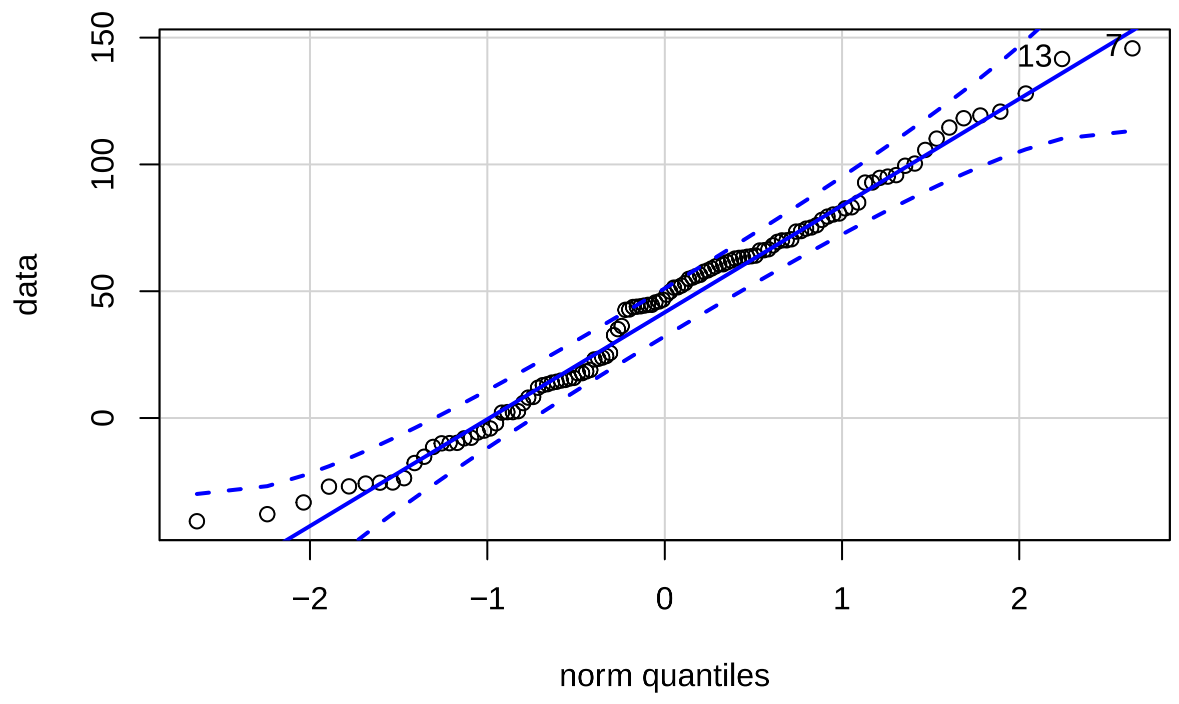


```
##
##  Shapiro-Wilk normality test
##
## data:  data
## W = 0.93443, p-value = 7.33e-09
```

Check normality of differences for high end device tier:

```
differences_high <- with(mean_device_app_type_high, Joule_calculated[app_type == "web"] - Joule_calcula
normality_high <- check_normality(differences_high)
```


density.default(x = data)
Density
0.000
0.004
0.008
−50
0
50
100
150
N = 120   Bandwidth = 14.42



```
normality_high
```

```
##
##  Shapiro-Wilk normality test
##
## data:  data
## W = 0.98358, p-value = 0.1527
```

```
# Likely to stem from normal distribution
```

```
# Write image
jpeg("./plots/qq_high.jpeg")
car::qqPlot(differences_high, main="QQ-plot of differences in joule for high-end device")
```

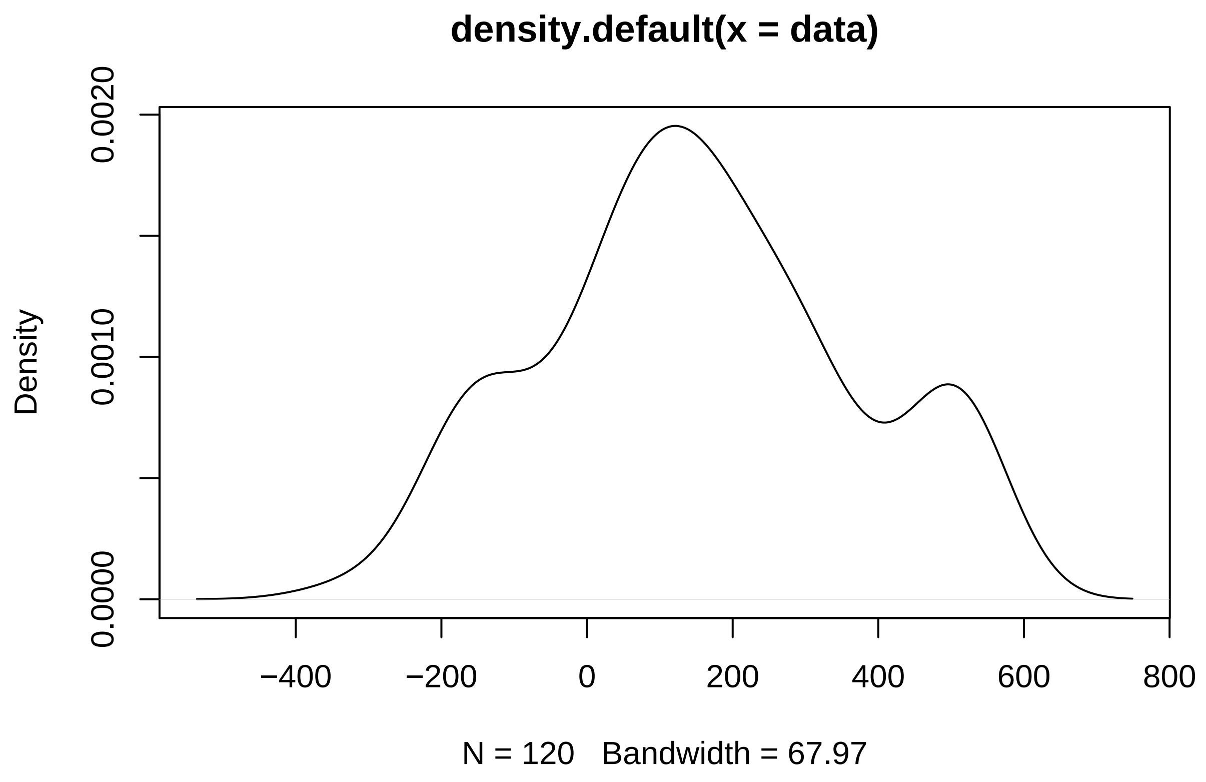
```
## [1]  7 13
```

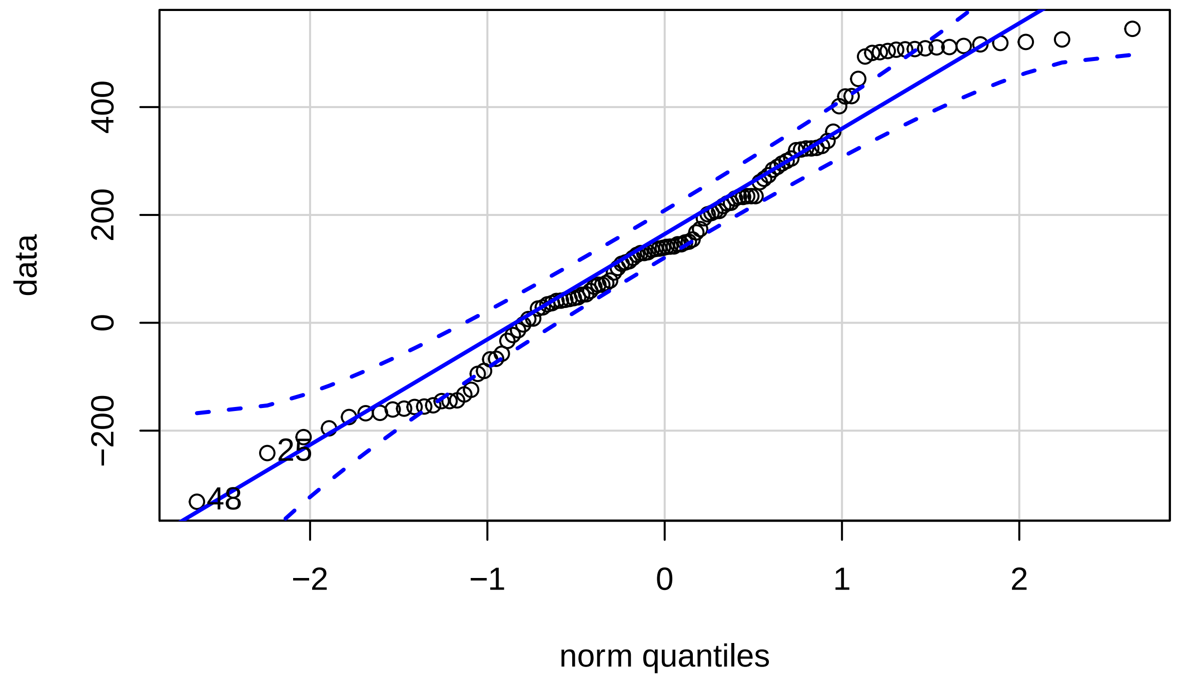
```
dev.off()
```

```
## pdf
##   2
```

Check normality of differences for low end device tier:

```
differences_low <- with(mean_device_app_type_low, Joule_calculated[app_type == "web"] - Joule_calculatec
check_normality(differences_low)
```


density.default(x = data)
Density
0.0000
0.0010
0.0020
−400
−200
0
200
400
600
800
N = 120 Bandwidth = 67.97



```
##
##  Shapiro-Wilk normality test
##
## data:  data
## W = 0.96794, p-value = 0.005807
```

```
# Does not stem from normal distribution
```

Because it doesn't fit the low device data does not fit the normal distribution we try to add some transformations.

```
# Write image
jpeg("./plots/qq_low.jpeg")
car::qqPlot(differences_low, main="QQ-plot of differences in joule for low-end device")
```

```
## [1] 48 25
```

```
dev.off()
```

```
## pdf
##   2
```

Add transformations

```
add_transform_data <- function(device_app_data){
  return(device_app_data %>%
    mutate(joule_log = log(Joule_calculated),
```

```r
           joule_sqrt = sqrt(Joule_calculated),
           joule_reciprocal = 1/Joule_calculated))
}
mean_device_app_type_high <- add_transform_data(mean_device_app_type_high)
mean_device_app_type_low <- add_transform_data(mean_device_app_type_low)
```

```r
normality_diff_log <- shapiro.test(with(mean_device_app_type_low, joule_log[app_type == "web"] - joule_l
normality_diff_sqrt <- shapiro.test(with(mean_device_app_type_low, joule_sqrt[app_type == "web"] - joule
normality_diff_recip <- shapiro.test(with(mean_device_app_type_low, joule_reciprocal[app_type == "web"]

normality_diff_log$p.value
```

```
## [1] 0.001504485
```

```r
normality_diff_sqrt$p.value
```

```
## [1] 0.005062697
```

```r
normality_diff_recip$p.value
```

```
## [1] 0.000152925
```

All are under alpha treshold. So normality can't be assumed. So normality can't be assumed. Therefore for the low data we perform a non parametric alternative.

Assumption 3 (for paired t-test)

There should be no extreme outliers in the differences.

```r
par(mfrow=c(1,2))
jpeg("./plots/combined_box_plots")
boxplot(differences_high)
# boxplot(Joule_calculated~app_type, data= mean_device_app_type_high, main = "Distribution for a high e
# boxplot(Joule_calculated~app_type, data= mean_device_app_type_low, main = "Distribution for a low end
dev.off()
```

```
## pdf
##   2
```

Differences of high end data has no outliers.

RQ1 (high-end)

RQ1 (high-end) will be answered with the paired t-test.

```r
# Perform paired t test
high_end_web <- mean_device_app_type_high %>%
  filter(app_type == 'web')
high_end_native <- mean_device_app_type_high %>%
  filter(app_type == 'native')
highRes <- t.test(x = high_end_web$Joule_calculated, y = high_end_native$Joule_calculated, paired = T, a

highRes$p.value
```

```
## [1] 7.033588e-21
```

The obtained p-value for high-end device is $7.0335876 \times 10^{-21}$ which is significant.

```r
low_end_web <- mean_device_app_type_low %>%
  filter(app_type == 'web')
low_end_native <- mean_device_app_type_low %>%
  filter(app_type == 'native')
lowRes <- wilcox.test(x = low_end_web$Joule_calculated, y = low_end_native$Joule_calculated, paired = T

lowRes$p.value
```

```
## [1] 1.270736e-10
```

The obtained p-value for low-end device is $7.0335876 \times 10^{-21}$ which is significant.

However because we perform a t test multiple times we have to adjust the p values.

```r
# adjust p values
adjustedPVals <- p.adjust(c(highRes$p.value, lowRes$p.value), method = 'BH')
adjustedPVals
```

```
## [1] 1.406718e-20 1.270736e-10
```

The adjusted p-values for high and low end device are $1.4067175 \times 10^{-20}$ and $1.2707357 \times 10^{-10}$ respectively.

This leads us to reject the null-hypothesis of equal means for the high-end device. For the low end device the statistical signifiance is also significant enough to reject the null-hypothesis.

This leads us to the following conclusion for RQ1: there is a significant difference in energy consumption.

3. Effect size

```r
# Effect size
cohenResHigh <- cohen.d(high_end_web$Joule_calculated, high_end_native$Joule_calculated, paired = T, po
cliffResLow <- cliff.delta(low_end_web$Joule_calculated, low_end_native$Joule_calculated, paired = T, p

cohenResHigh$estimate #Effect size is large for high
```

```
## [1] 1.573352
```

```r
cliffResLow$estimate #Effect size is large for low
```

```
## [1] 0.5852083
```

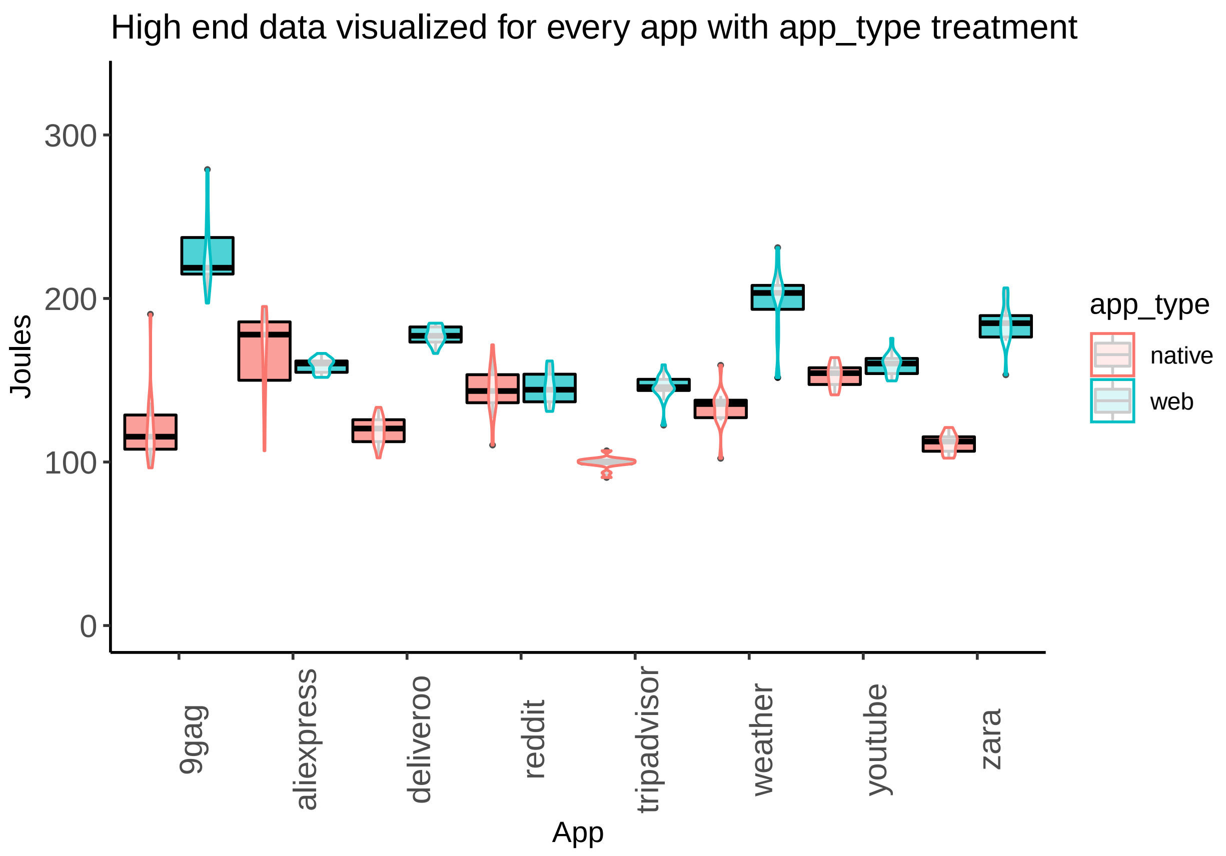
Effect size for both device tiers is large.

Extra visualizations

```r
font_size <- 12

visualize_both <- function(data, title){
  result <- ggplot(data, aes(x = app, y = Joule_calculated)) +
  theme_classic() +
  xlab("App") + ylab("Joules") +
  ylim(c(0, max(data$Joule_calculated) + 50)) +
  geom_boxplot(mapping = aes(label = app_type, fill = app_type), width = 1, color = 'black', outlier.si
  geom_violin(mapping = aes(label = app_type, color = app_type), trim = TRUE, alpha = .8, width = 1) +
  # stat_summary(mapping = aes(color = app_type), fun = mean, geom = 'point', shape = 5, size = 2, show
  theme(
    strip.text.x = element_text(size = font_size),
    strip.text.y = element_text(size = font_size),
    axis.text.x = element_text(size = font_size, angle = 90),
    axis.text.y = element_text(size = font_size),
    )  +
    labs(title = title,subtitle = NULL,caption = NULL)

  return(result)
}
```

```
visualize_both(mean_device_app_type_high, "High end data visualized for every app with app_type treatmen
```

```
## Warning: Ignoring unknown aesthetics: label

## Warning: Ignoring unknown aesthetics: label
```



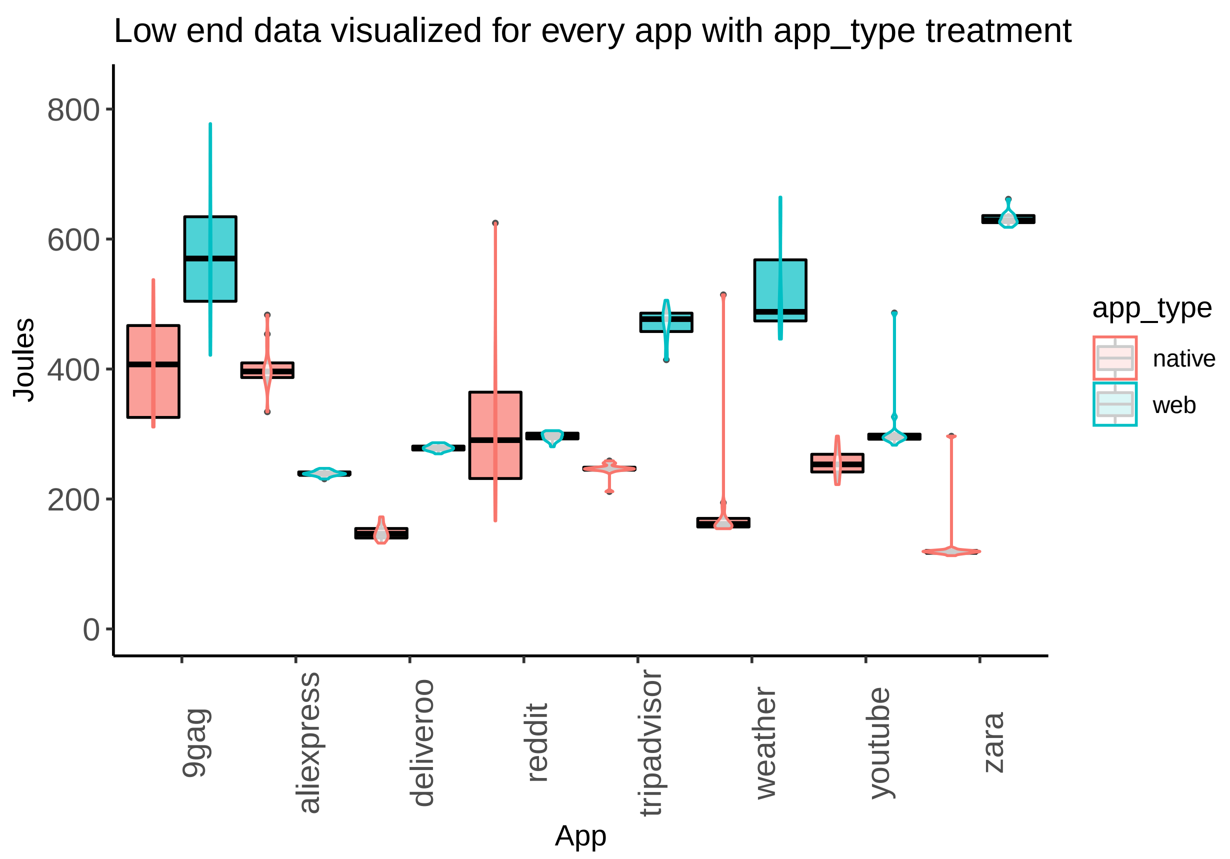
```
ggsave("./plots/gg_plot_high_end.jpeg")
```

```
## Saving 6.5 x 4.5 in image
```

Except for aliexpress every web app has a higher energy consumption. Aliexpress however also shows a high standard deviation for the native app, which could possibly indicate a measurement error.

```
visualize_both(mean_device_app_type_low, "Low end data visualized for every app with app_type treatment
```

```
## Warning: Ignoring unknown aesthetics: label

## Warning: Ignoring unknown aesthetics: label
```



```r
ggsave("./plots/gg_plot_low_end.jpeg")
```

```
## Saving 6.5 x 4.5 in image
```

The low end device's data has more deviation in the data with some outliers for example at weather_native and youtube_web.

```r
app_data %>% summarise(
  min = min(Joule_calculated),
  max = max(Joule_calculated),
  median = median(Joule_calculated),
  mean = mean(Joule_calculated),
  sd = sd(Joule_calculated),
  cv = cv(Joule_calculated)
)
```

```
## # A tibble: 1 x 6
##     min   max median  mean    sd    cv
##   <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1  90.5  778.   184.  246.  147. 0.596
```

Energy consumption summarized for both devices.

```r
app_data %>%
group_by(device) %>% summarise(
  min = min(Joule_calculated),
  max = max(Joule_calculated),
  median = median(Joule_calculated),
  mean = mean(Joule_calculated),
  sd = sd(Joule_calculated),
  cv = cv(Joule_calculated)
)
```

```
## # A tibble: 2 x 7
##   device   min   max median  mean    sd    cv
##   <fct>  <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 high    90.5  279.   152.  153.  35.2 0.231
## 2 low    113.   778.   293.  339. 156.  0.462
```

Energy consumption summarized per device.

```
app_data %>%
group_by(device, app_type) %>% summarise(
  min = min(Joule_calculated),
  max = max(Joule_calculated),
  median = median(Joule_calculated),
  mean = mean(Joule_calculated),
  sd = sd(Joule_calculated),
  cv = cv(Joule_calculated)
)
```

```
## # A tibble: 4 x 8
## # Groups:   device [2]
##   device app_type   min   max median  mean    sd    cv
##   <fct>  <fct>    <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1 high   native    90.5  195.   127.  131.  25.6 0.195
## 2 high   web      122.   279.   164.  175.  29.7 0.170
## 3 low    native   113.   625.   247.  261. 117.  0.450
## 4 low    web      231.   778.   418.  416. 152.  0.366
```

Energy consumption per device and app_type