ES207- Lab2

Christiana Ade

Feburary 3, 2018

Debugging

1. How can you find out where an error occurred?

You can find where an error occured using several methods.

- Sometimes the error message will include a problematic line, or you can google the error to figure out what exactly is going wrong with your code. I tend to find this is more the approach you take when you are working with someone else functions, rather than debugging your own
- You can use print statements throughtout your script to make sure certain 'chunks' of your code are doing what they expect.
- R also has an interactive debugging browser that you can access in three different ways (learned in this lab). This includes the functions `debug(), setBreakPoint()`, and `browser()`

2. What does browser() do? List the 5 single-key commands that you can use in the browser() environment.

The function `browser()` allows you to interrupt an expression from being executed. When evaluating this placed in a particular function itself at the place where you want the code to stop running and then allows inspection using the debugging browser (at the line you placed the function).

The five single key commands you can use in browser are:

- `c` exit the browser and continue to the next statement
- `f` finish execution for the current loop or function
- `n` evaluate next statement. stepping over function calls
- `s` evaluate next statement. stepping into fucntion calls
- `r` invoke a "resume" restart

3. Debug the function below. Demonstrate all of your steps, provide the correction function with clear documentation, and provide confirmation it works as expected.

```
# Step 1: We have the function with issues
ciao <- function (x) {
  if (x = 0) salutation) <-  "Buongiorno!" else
        else
    (salutation) <- "Arrivederci!"
}
```

```
## Error: <text>:3:9: unexpected '='
## 2: ciao <- function (x) {
## 3:   if (x =
##            ^
```

```
# Step 2: R is giving us an x on the line starting with the if statement
# This is because there is a unmatched parentheses right after salutation
ciao <- function (x) {
  if (x = 0) salutation <-  "Buongiorno!" else
        else
```

```
    (salutation) <- "Arrivederci!"
}
```

```
## Error: <text>:4:9: unexpected '='
## 3: ciao <- function (x) {
## 4:   if (x =
##            ^
```

```
# Step 3: next we get an error that says unexpected '='. This is because this needs to be a ==
ciao <- function (x) {
  if (x == 0) salutation <-  "Buongiorno!" else
        else
    (salutation) <- "Arrivederci!"
}
```

```
## Error: <text>:4:9: unexpected 'else'
## 3:   if (x == 0) salutation <-  "Buongiorno!" else
## 4:         else
##            ^
```

```
# Step 4: next we get an error that says unexpected 'else' in this line
# Lets look at the synax if an if...else statement in R

# if (test_expression){
# statement1
# } else {
# statement2
# }
# https://www.datamentor.io/r-programming/if-else-statement
# QUESTION FOR ERIN: does '/ /' not work as way for commenting large blocks of code in Rstudio
# because it works in eclipse

# oh so there should be {} brackets included for the if and the else statements
# oh and while we are at it, there should only be one else because we only have two statements
ciao <- function (x) {
  if (x == 0){
    salutation <-  "Buongiorno!"
    } else {
    (salutation) <- "Arrivederci!"
    }
}
# lets test it
ciao(0)
# we get no errors, but nothing returns and there is no return statement
```

```
## Step 5: now we have an error stating that the object salutation is not found
# oh there are () around salutation
ciao <- function (x) {
  if (x == 0){
    salutation <-  "Buongiorno!"
    } else {
    (salutation) <- "Arrivederci!"
    }
  return(salutation)
}
```

```r
# Step 6: FINALLY!
ciao <- function (x) {
  if (x == 0){
    salutation <-  "Buongiorno!"
    } else {
    salutation <- "Arrivederci!"
    }
  return(salutation)
}


# lets test it on several numbers and practice using lapply
testList <- c(0,1,-1,55)
lapply(testList, ciao)
```

```
## [[1]]
## [1] "Buongiorno!"
##
## [[2]]
## [1] "Arrivederci!"
##
## [[3]]
## [1] "Arrivederci!"
##
## [[4]]
## [1] "Arrivederci!"
```

```r
# actually lets simply to a character vector
sapply(testList, ciao)
```

```
## [1] "Buongiorno!"  "Arrivederci!" "Arrivederci!" "Arrivederci!"
```

4. The following function "lags" a vector, returning a version of x that is n values behind the original. Improve the function so that it (1) returns a useful error message if n is not a vector, and (2) has reasonable behaviour when n is 0 or longer than x.

```r
# QUESTION ERIN: you were missing a ) and had one in the wrong place on the c(rep()) line
# was that intentional?
lag <-  function (x, n=1L) {
  xlen <- length(x)
  c(rep(NA, n), x[seq_len(xlen-n)])
}


# Fixing the function
lag <- function(x, n = 1L) {
  #if(!is.numeric(x)) {stop('x is not numeric')} # I guess we still want this to work
  # not ness. but perhaps someone is confused about how the function is supposed to work
  if(length(x)==1 |length(x)==0){stop('x is not an acceptable length')}
  if(!is.numeric(n)) {stop('n is not numeric')} # even a single number is a vector
  if(n>length(x)){stop('n is longer than x')}
  if(n==0){stop('n is 0, so this will not work')} #could combine this and the other line
  xlen <- length(x)
  c(rep(NA, n), x[seq_len(xlen - n)])
}
```

```
lag(8,4) # we need x to be multiple values
```

```
## Error in lag(8, 4): x is not an acceptable length
```

```
lag(c("R","is","super","cool"),1)
```

```
## [1] NA      "R"     "is"    "super"
```

```
lag(1:10, 'meow') # in case we try to input something of
```

```
## Error in lag(1:10, "meow"): n is not numeric
```

```
lag(1:10, 17)
```

```
## Error in lag(1:10, 17): n is longer than x
```

```
lag(1:10, 0)
```

```
## Error in lag(1:10, 0): n is 0, so this will not work
```

```
lag(1:20,3) # it works
```

```
##  [1] NA NA NA  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
```

Apply your knowledge to real data

```
# read in BayDeltaWQ
require(tidyverse)
require(moments)
library(dplyr)
setwd("C:\\Users\\cade\\Documents\\PhDMerced\\Spr18Courses\\EnvironmentalDataAnalysis\\Homework\\Lab2")
wQ <- read_csv("BayDeltaWQ.csv", col_names = TRUE, na = c("NA", "n/p", "n/a"))
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 1)
```

```
## Warning: 3133 parsing failures.
## row # A tibble: 5 x 5 col     row col       expected               actual file              expected
## ... ................. ... ..................................................................... ........
## See problems(...) for more details.
```

```
#names(wQ) <- gsub("\\..","_",names(wQ)) # I don't like the dots...but this is not needed
#need conductance data "Conductance..EC." [4] and Secchi depth = "Secchi.Depth" [12]
```

5. Write a function that calculates the mean, median, standard deviation, interquartile range, and skew. Apply that function to EC data and Secchi Disk depth. Discuss the differences between these measures and what conclusions you can draw about the data.

Diffrence between measures

- `mean` is an average of the numbers in the dataset. Its a calculated "central" value (might not actually be the center), but if we looked a plot of the distribution the mean would be that central value of a regular normal distripution.
- `median` is essentially ranking all the values and then the value in the middle is picked with equal amounts of numbers on both sides.
- `standard deviation` is used to quanitfy how variation in a dataset. So how spread are the values.

- `interquartile range` another measure of variability that is determined by spliting the data up into quarities and it is the difference between the upper and lower quarites (75th and 25th percentile) he IQR is a measure of variability, based on dividing a data set into quartiles
- `skewness` is measure of the asymmetry of the probability distribution.

There are some cases in which the median better represents the data rather than the mean. For example, distributions that are skewed either to the left of right. For conductance, the difference between the mean value and the median value is extremely large. The median value is much lower suggesting that the data is strongly right skewed.While this is less drastic in terms of the Secchi Depth this is also true and the data is right skewed. This is futher supported by the standard deviation and the skew. The standard deviation is pretty large for both categories indicating that the data are spread out a long a larger range. While the skew is positive indicating that the tail on the right side is longer than that on the left. The large IQR also shows that the data is skewed because there is a large difference in the upper and lower quartiles.

```
summary1 <- function(x) {
  funs <- c(mean, median, sd, skewness, IQR )
  lapply(funs, function(f) f(x, na.rm = T))
}

wQSummary <- wQ %>%
  select("Conductance (EC)","Secchi Depth") %>%
  sapply(summary1) %>%
  data.frame() %>%
  mutate(statistic = c("mean","median","sd","skewness","IQR"))
#if this was in the summary1 function that would better. I know I need to use vapply
# where is tha example though?
wQSummary
```

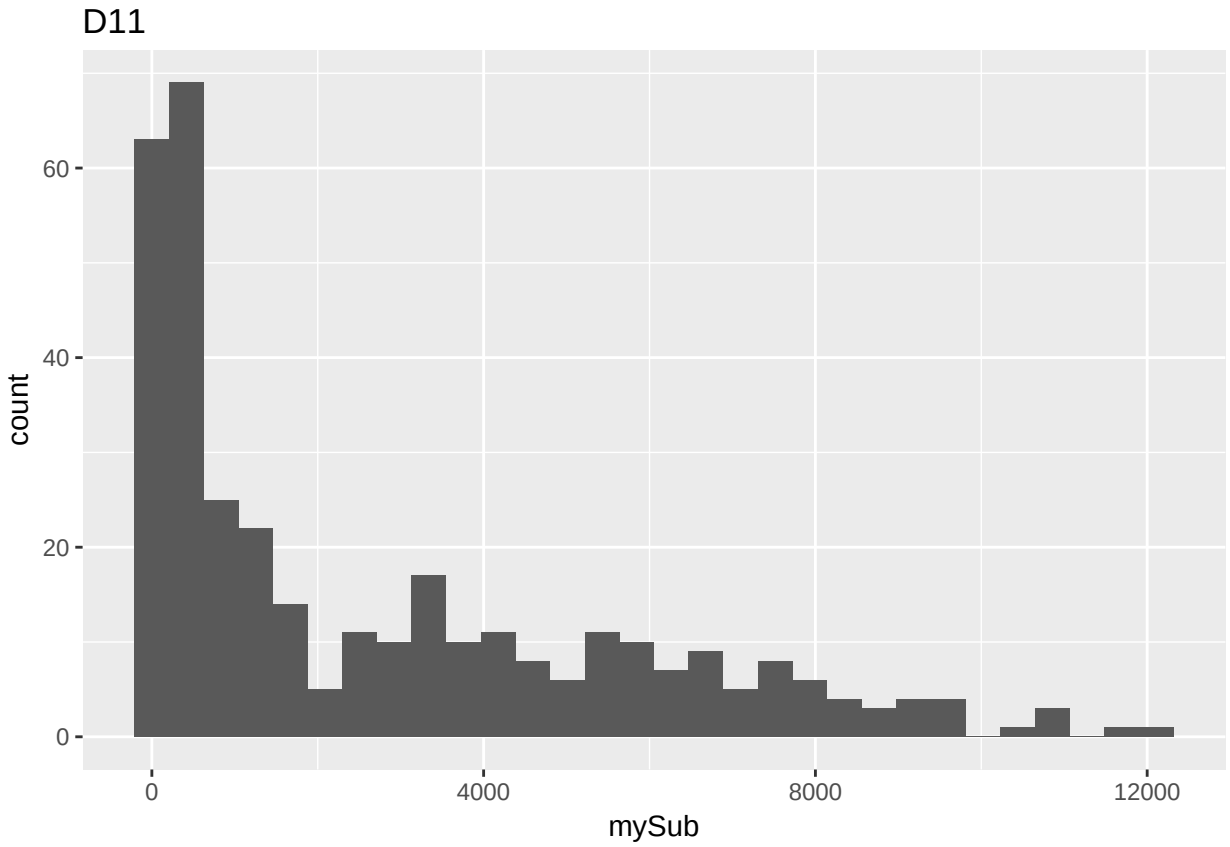
```
##   Conductance..EC. Secchi.Depth statistic
## 1         5715.291     55.97938      mean
## 2              722           48    median
## 3         9965.419     32.05139        sd
## 4         2.134902     2.032787  skewness
## 5             5866           32       IQR
```

6. Plot the histogram, boxplot, and cumulative density of EC data in the Bay Delta for EACH station.

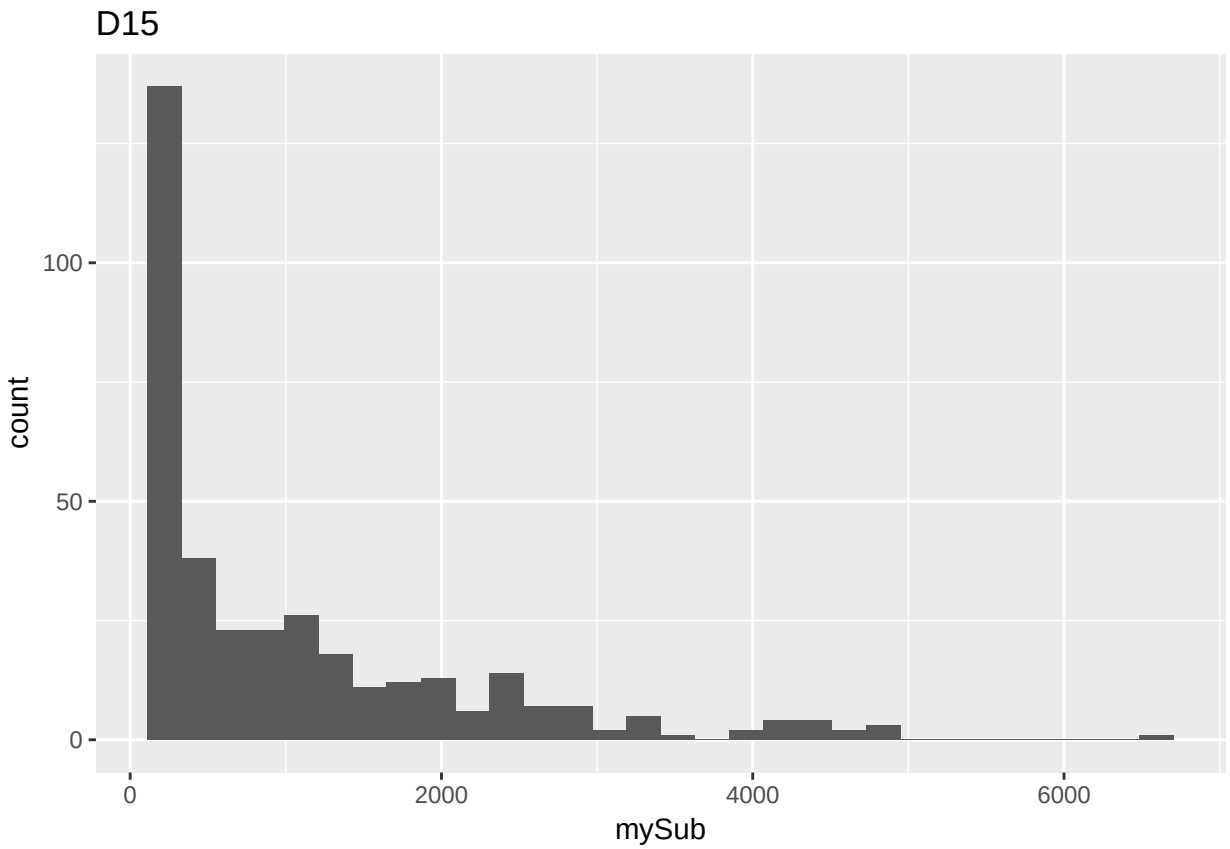
```
## Histograpm plotting
# solution using piping?
## not very effective plotting method with so much data
## how do you feed it into individual plots?
# nest and then map?
# wQ %>%
#   #group_by(StationCode) %>%
#   select(`Conductance (EC)`,StationCode) %>%
#   #nest(StationCode)  %>%
#   ggplot(aes(`Conductance (EC)`)) +    geom_histogram(na.rm =T ) +
#       facet_wrap(~StationCode)



### histogram...bad with loops
sites <- unique(wQ$StationCode)
try(for (i in 1:length(sites)){
   mySub <- subset(wQ,StationCode == sites[i] ,select = `Conductance (EC)` )
```

```
    myPlot <- ggplot(mapping = aes(mySub)) +
    geom_histogram(na.rm =T )
    print(myPlot + ggtitle(sites[i]))
})
```

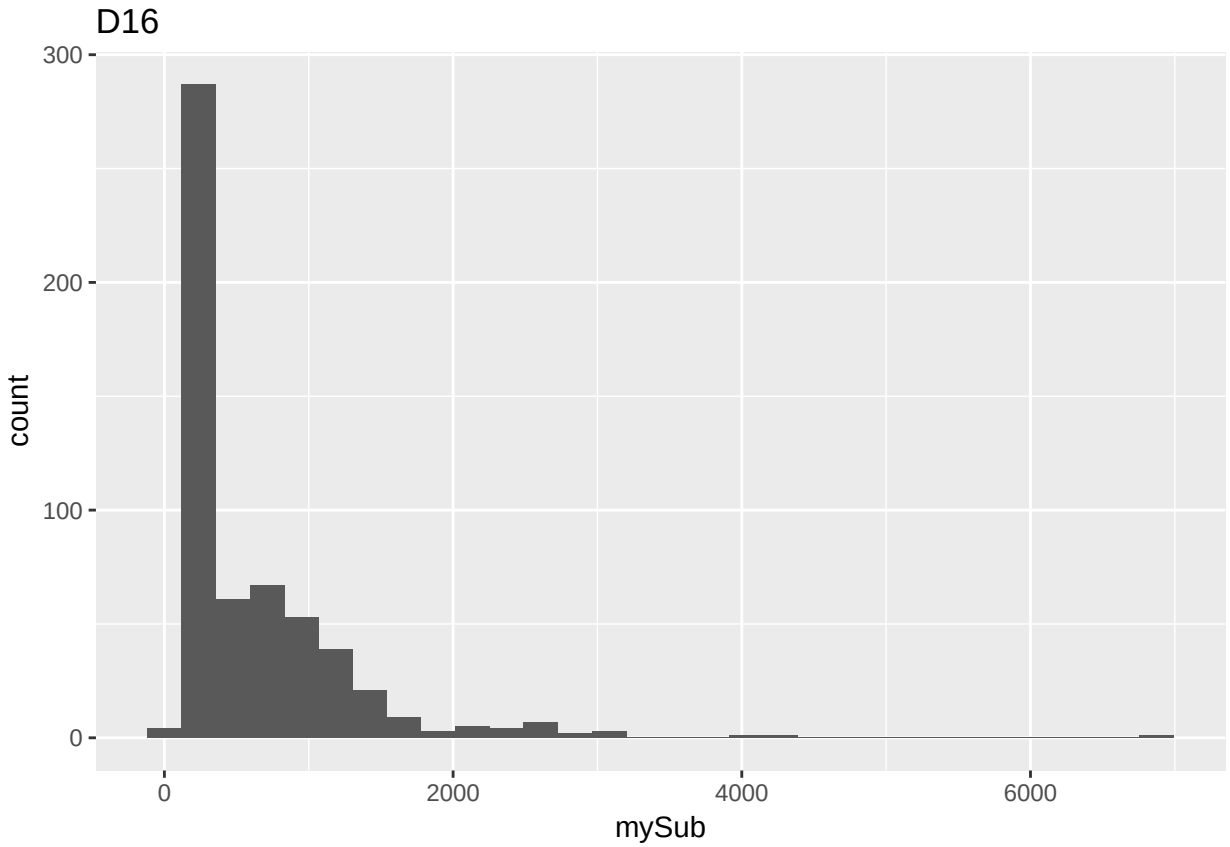
```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



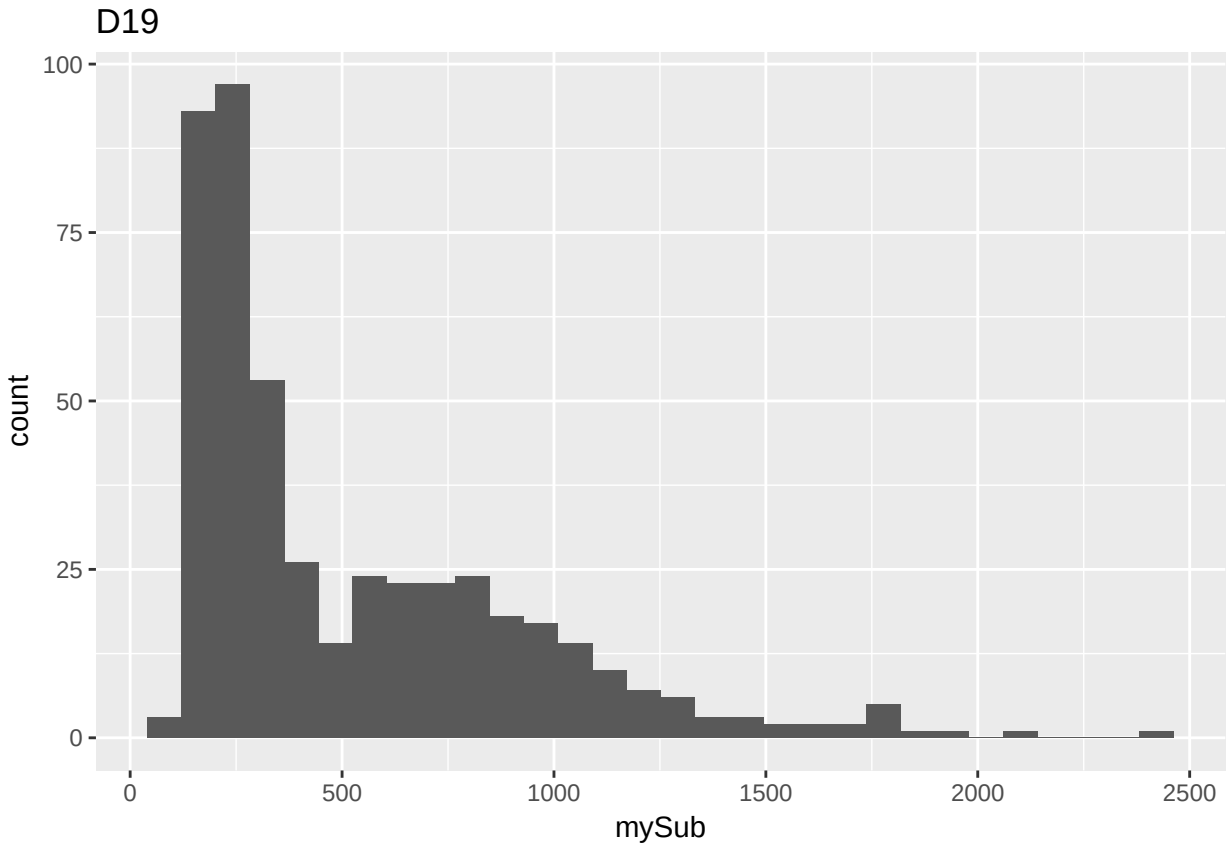

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



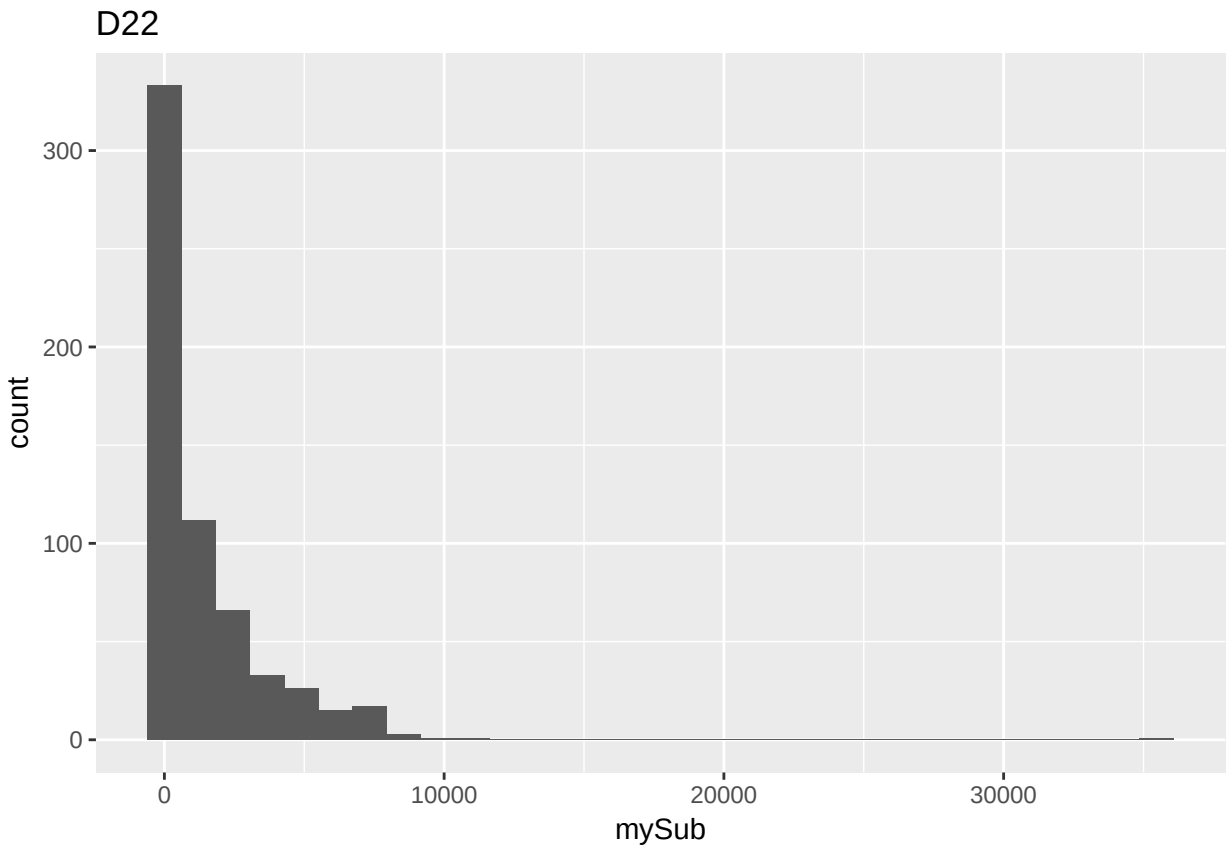

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



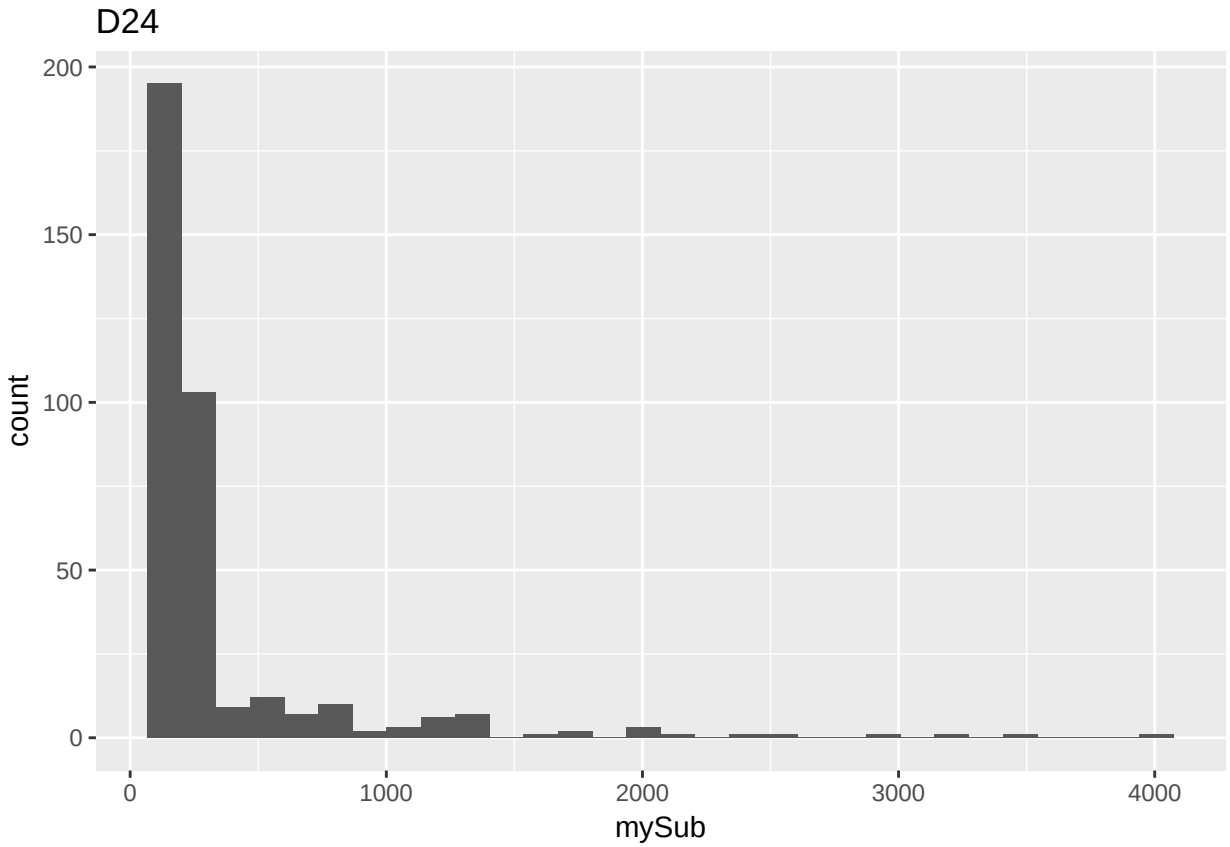

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



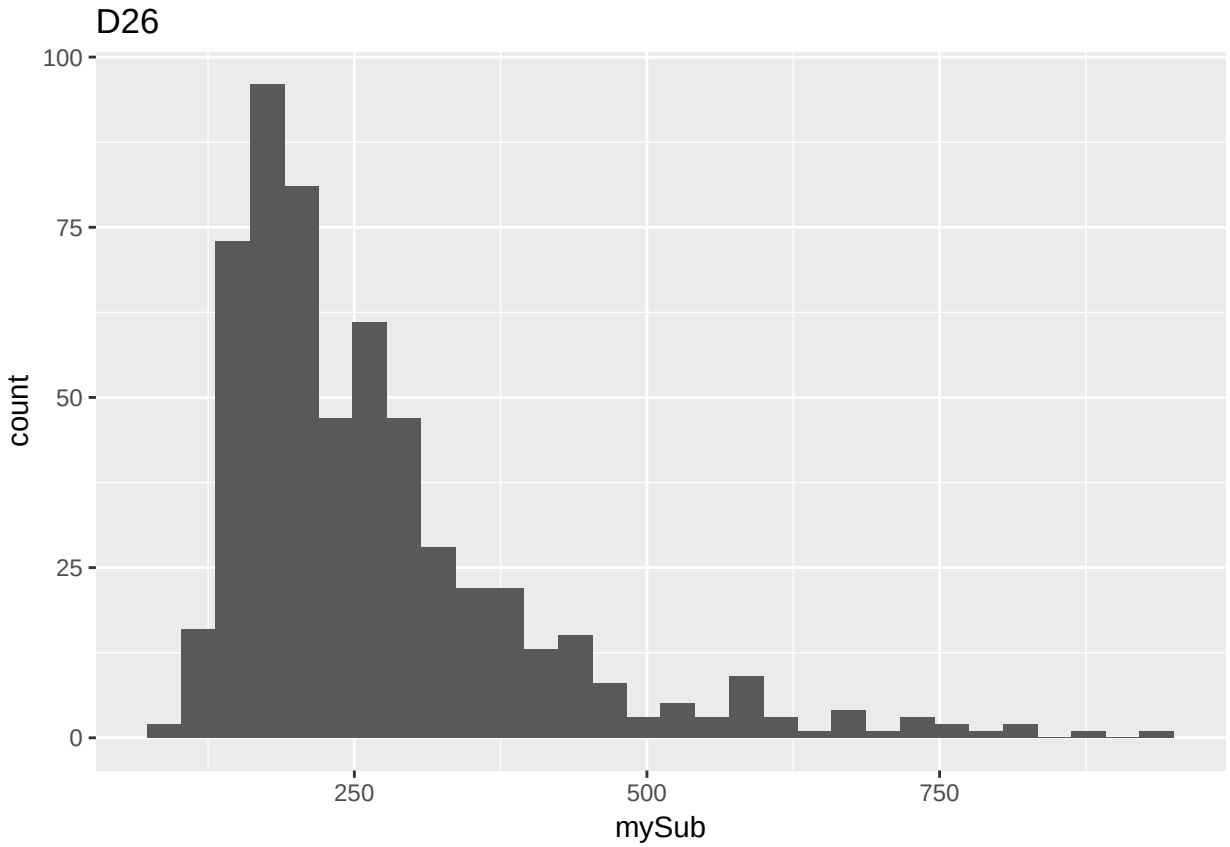

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



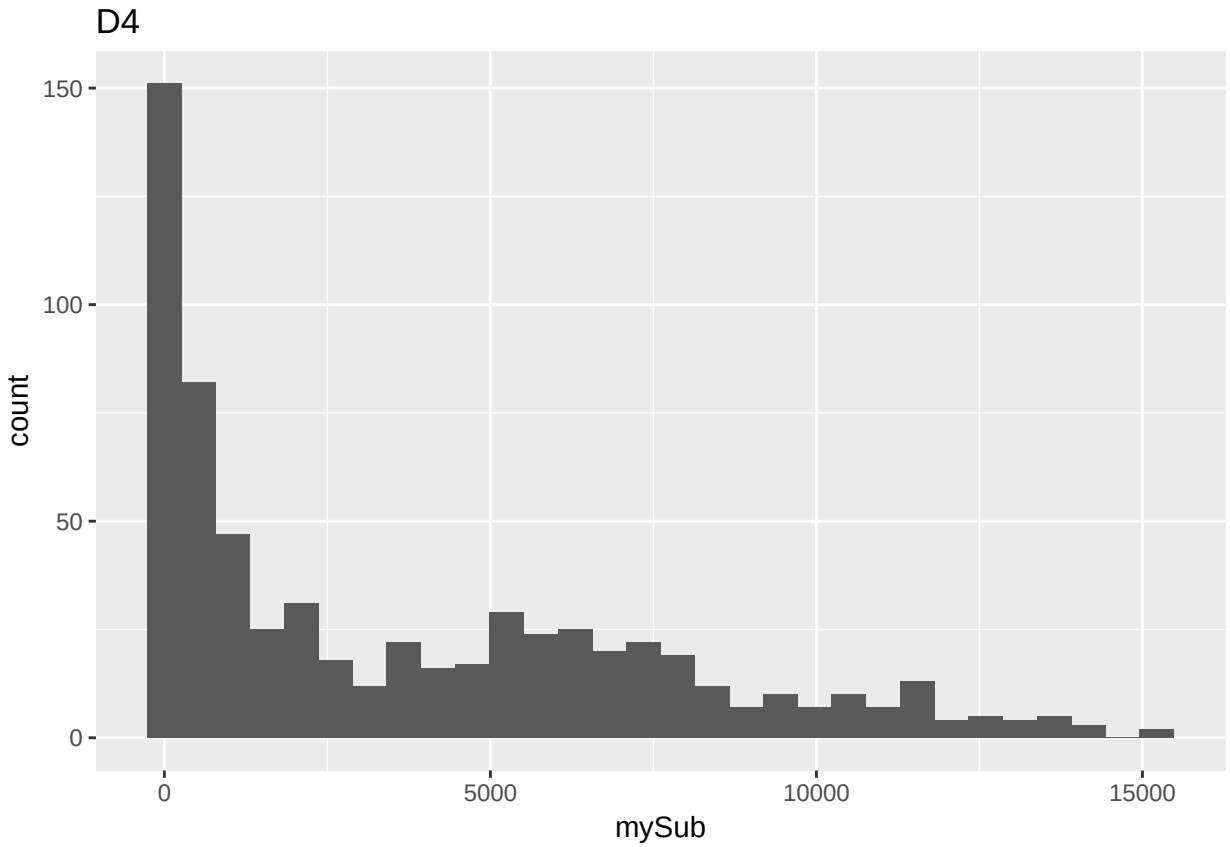

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



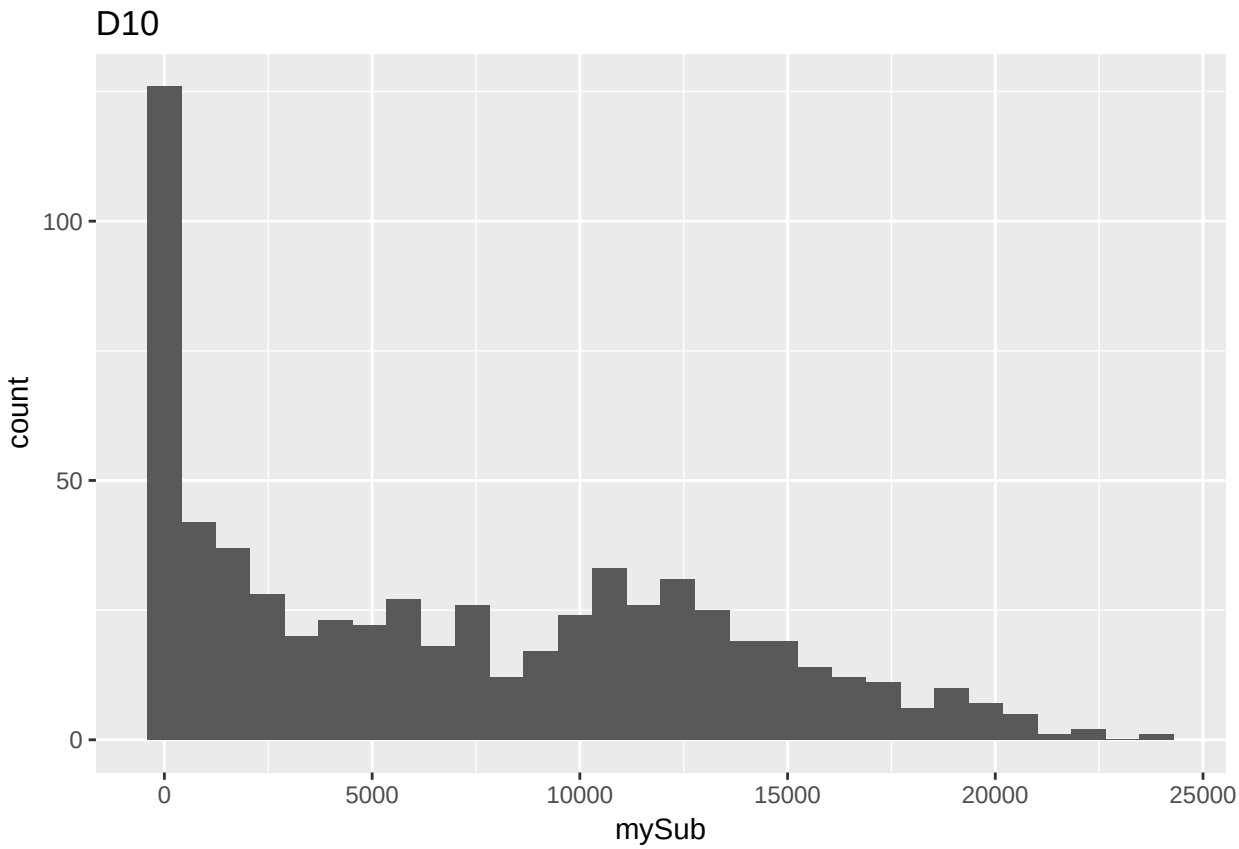

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



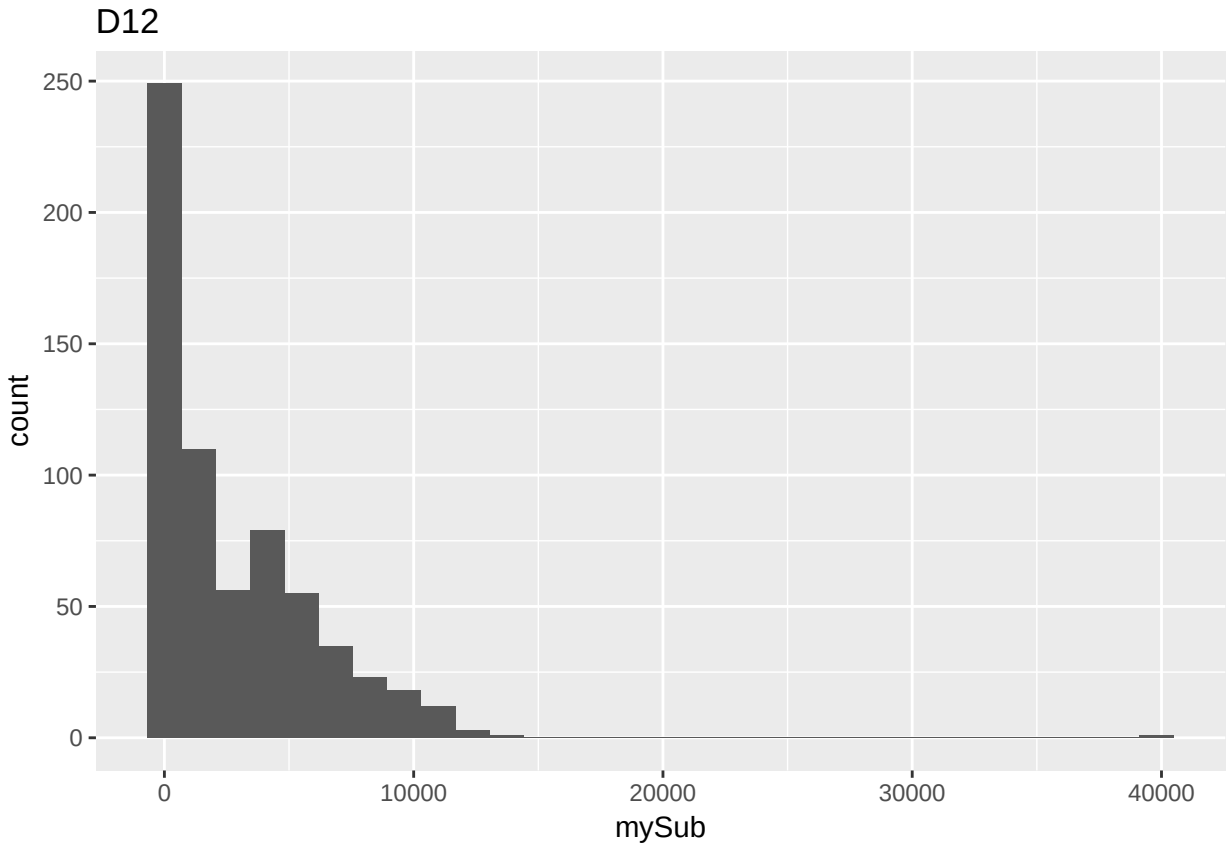

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



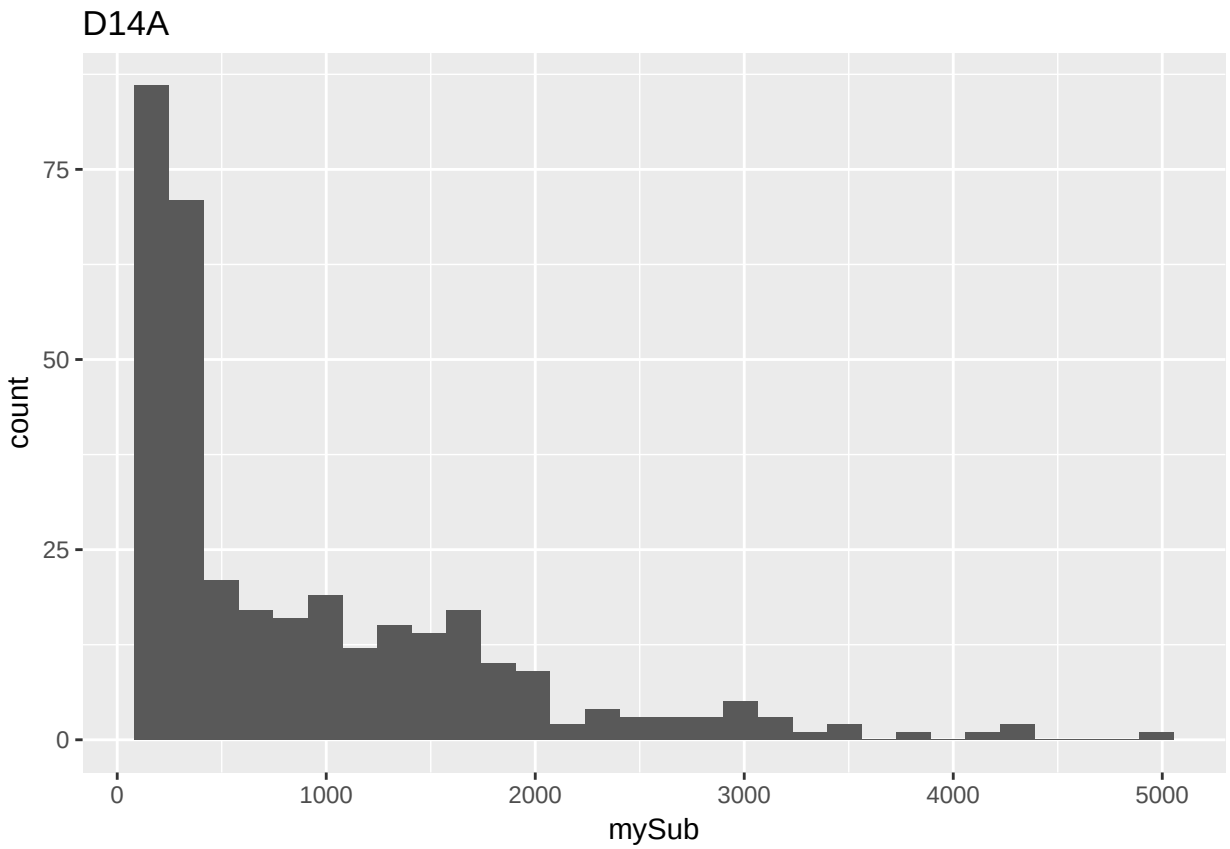

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



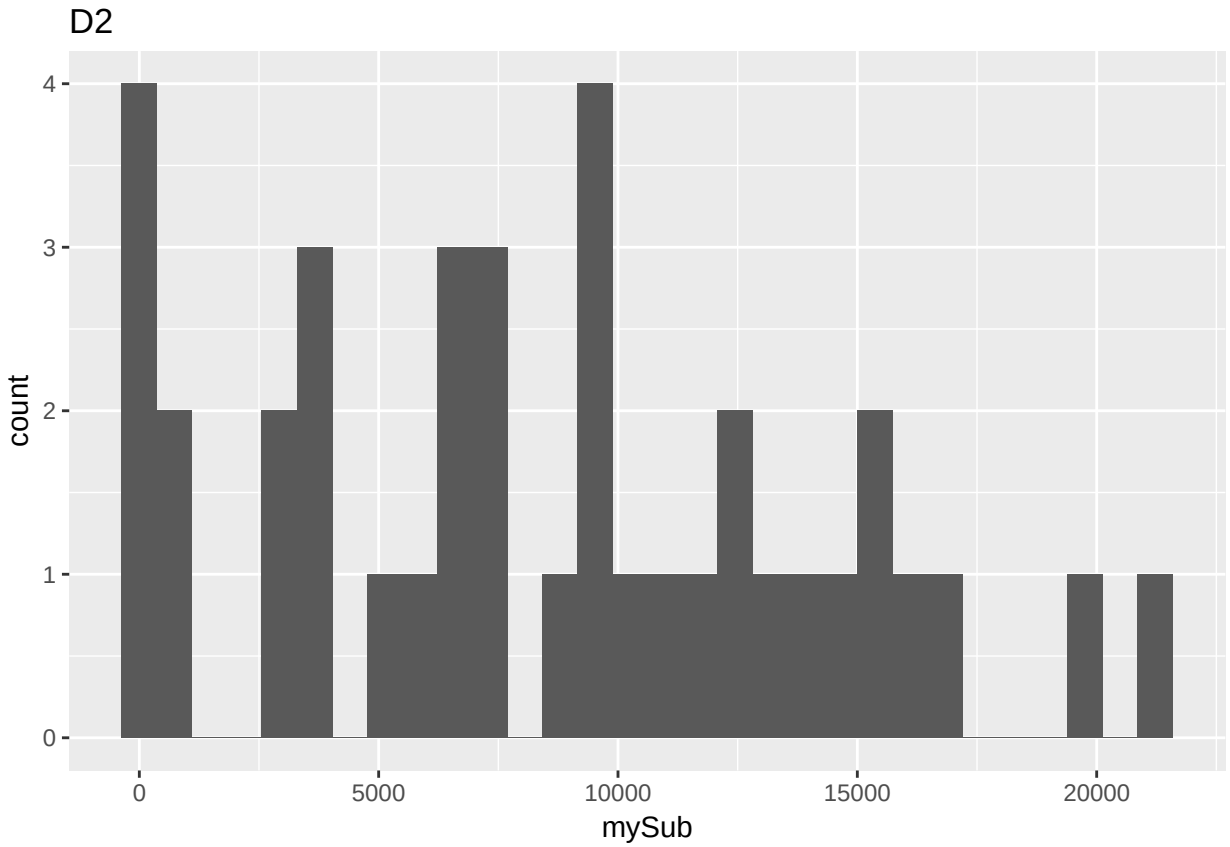

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



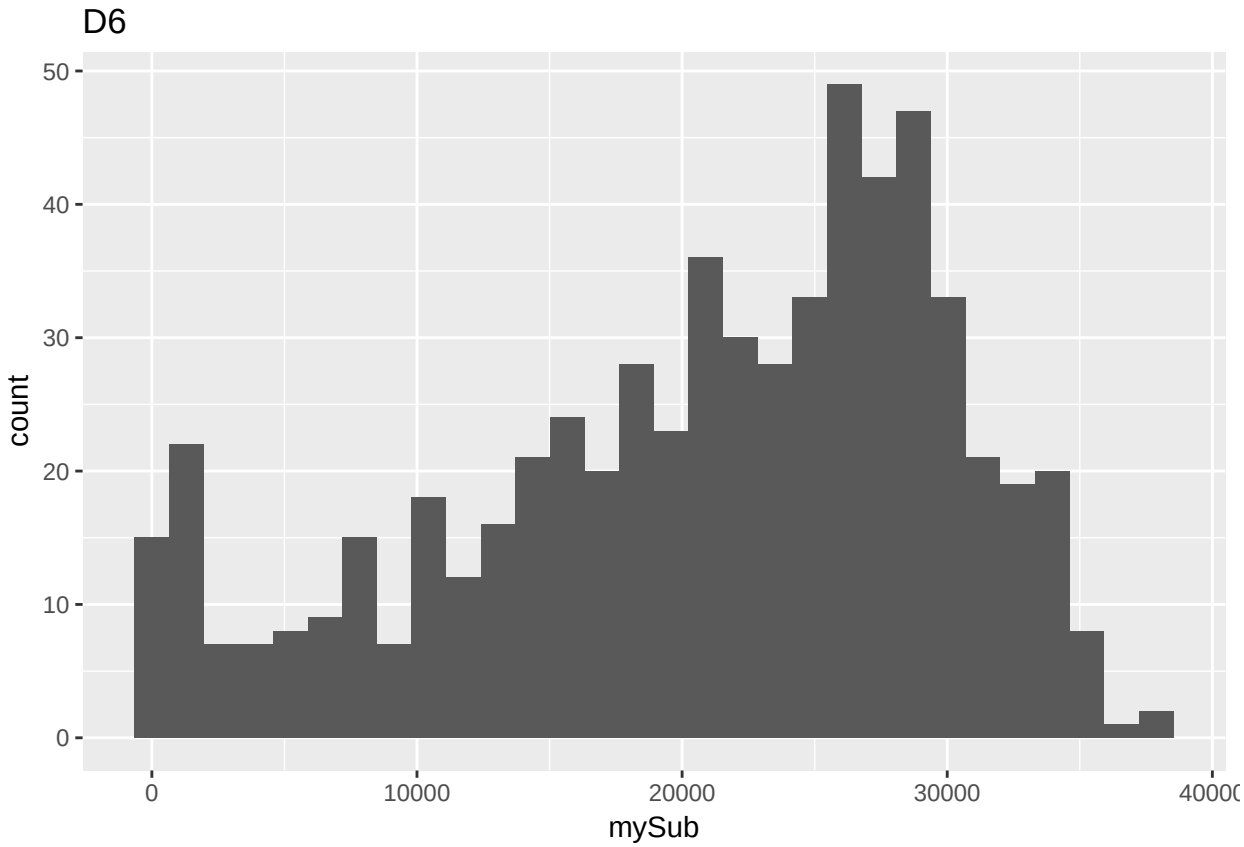

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



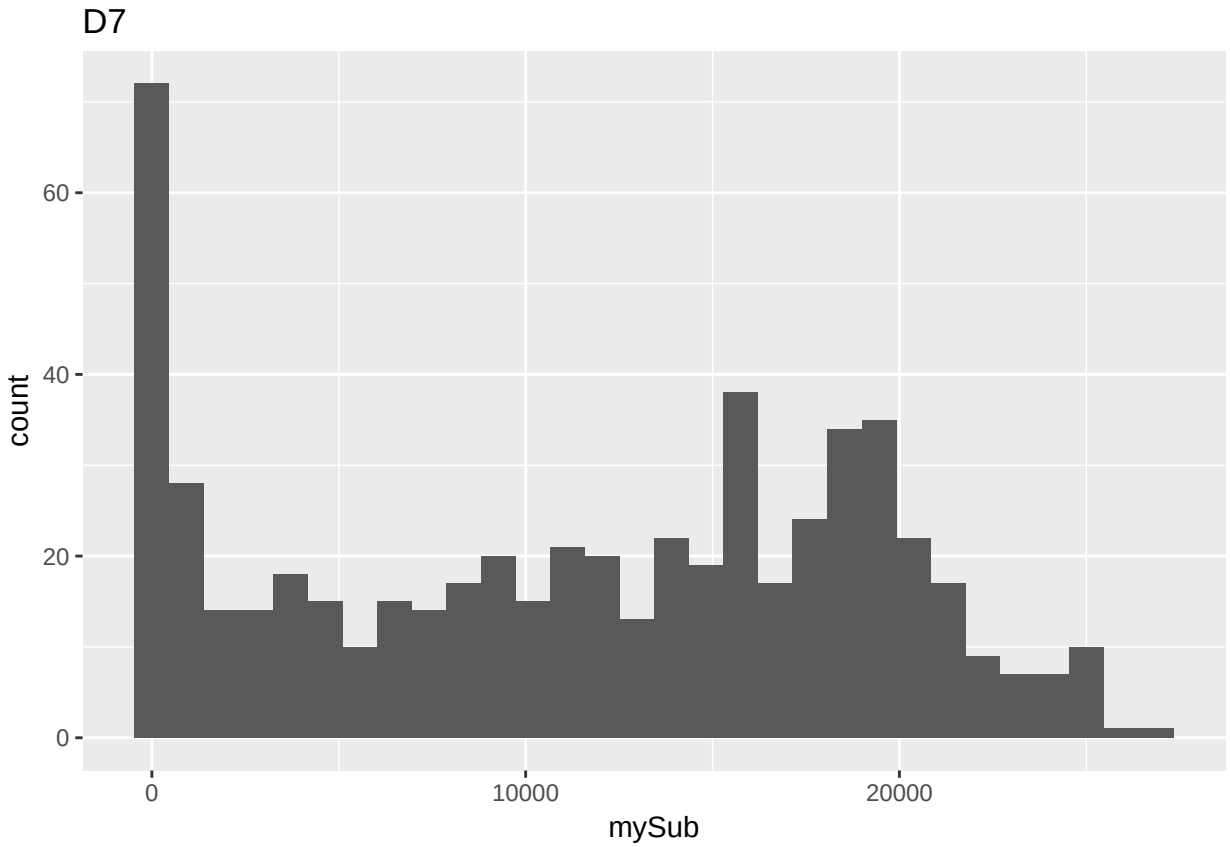

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



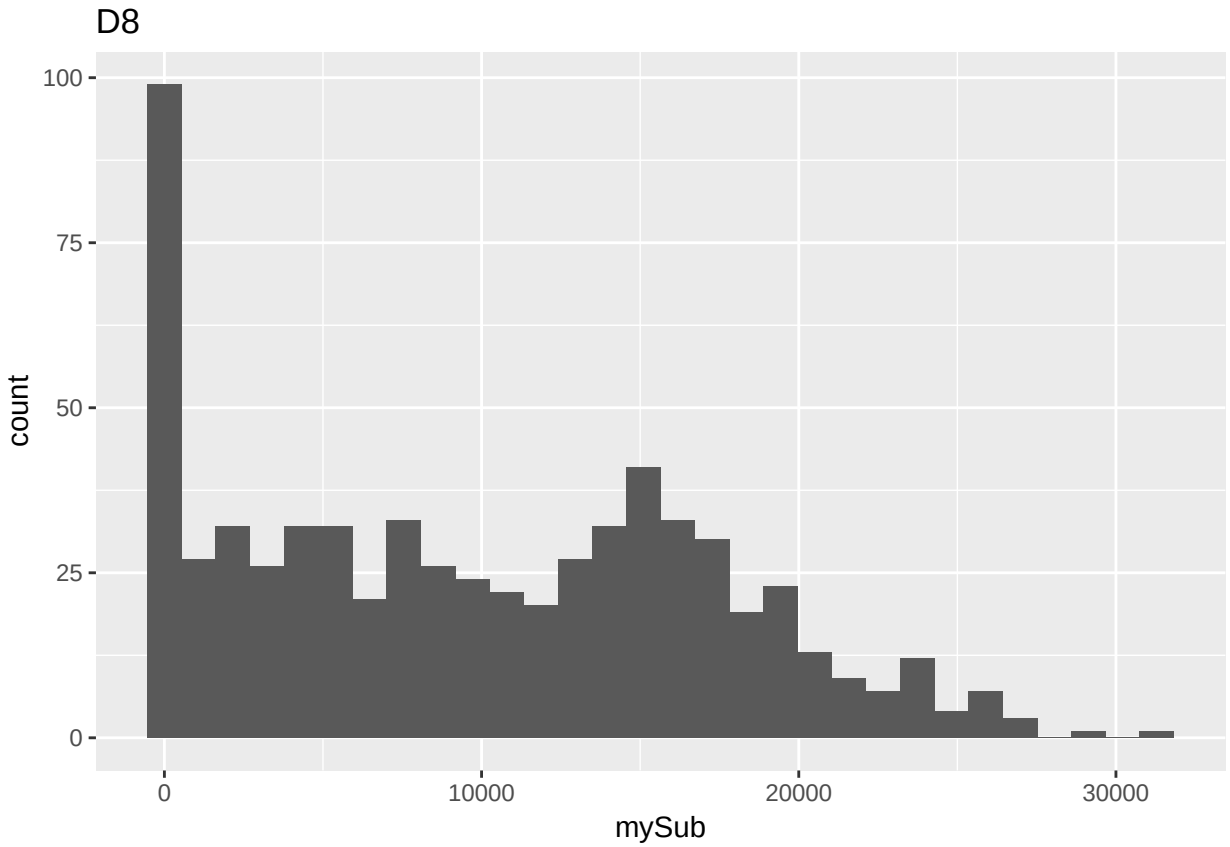

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



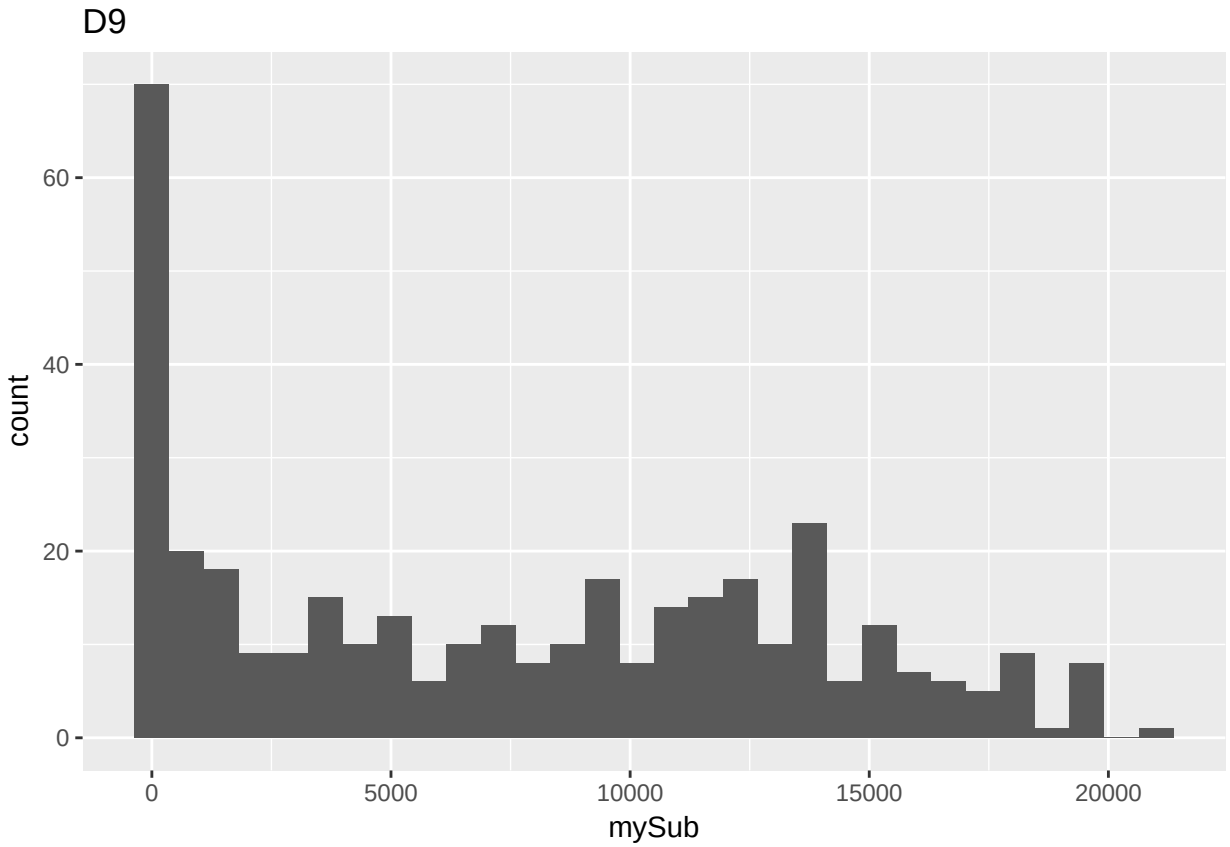

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



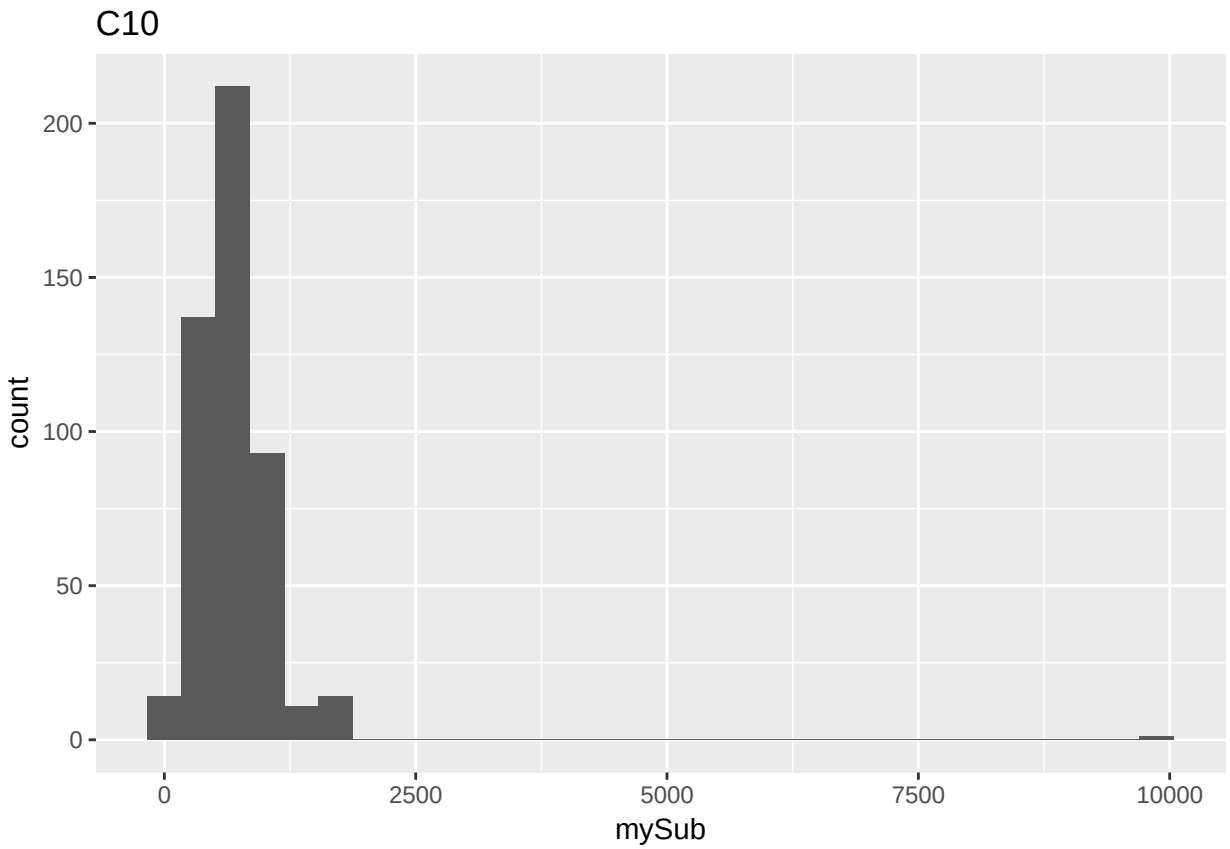

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



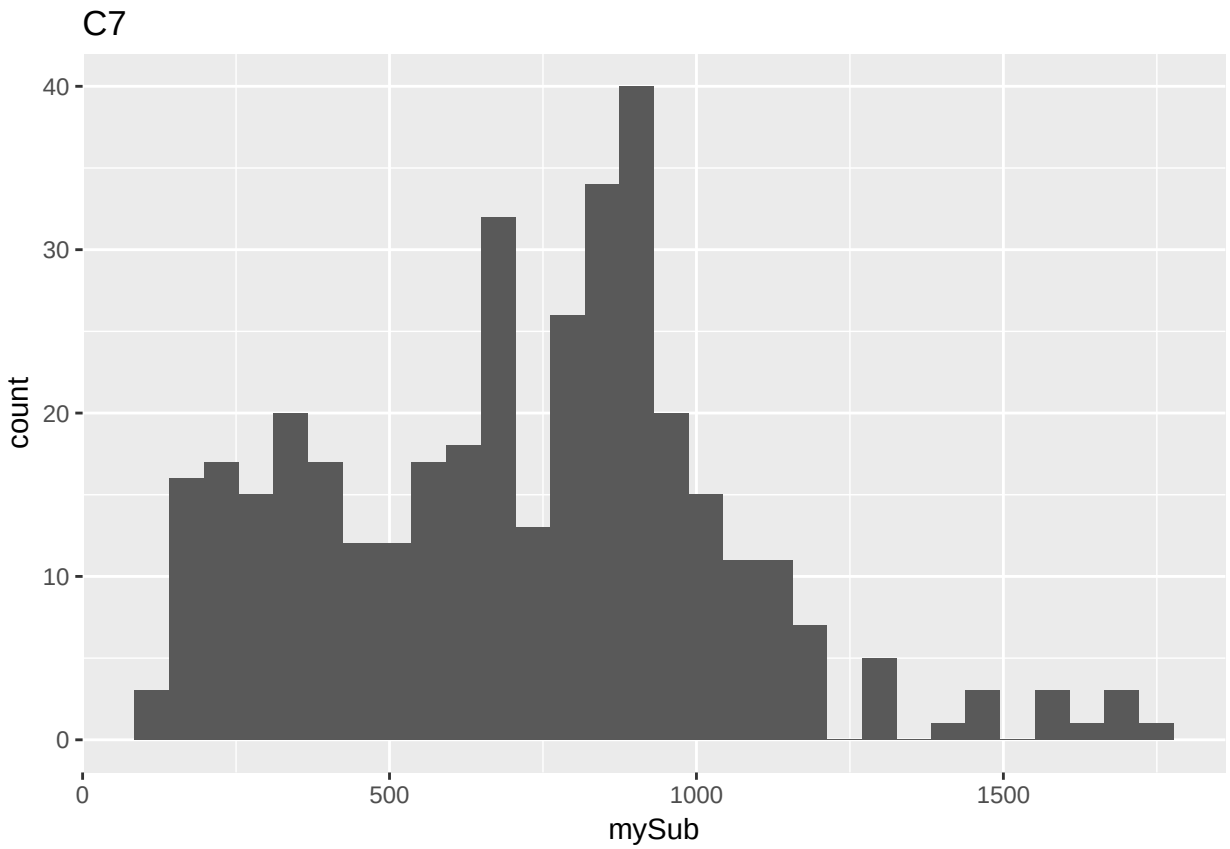

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



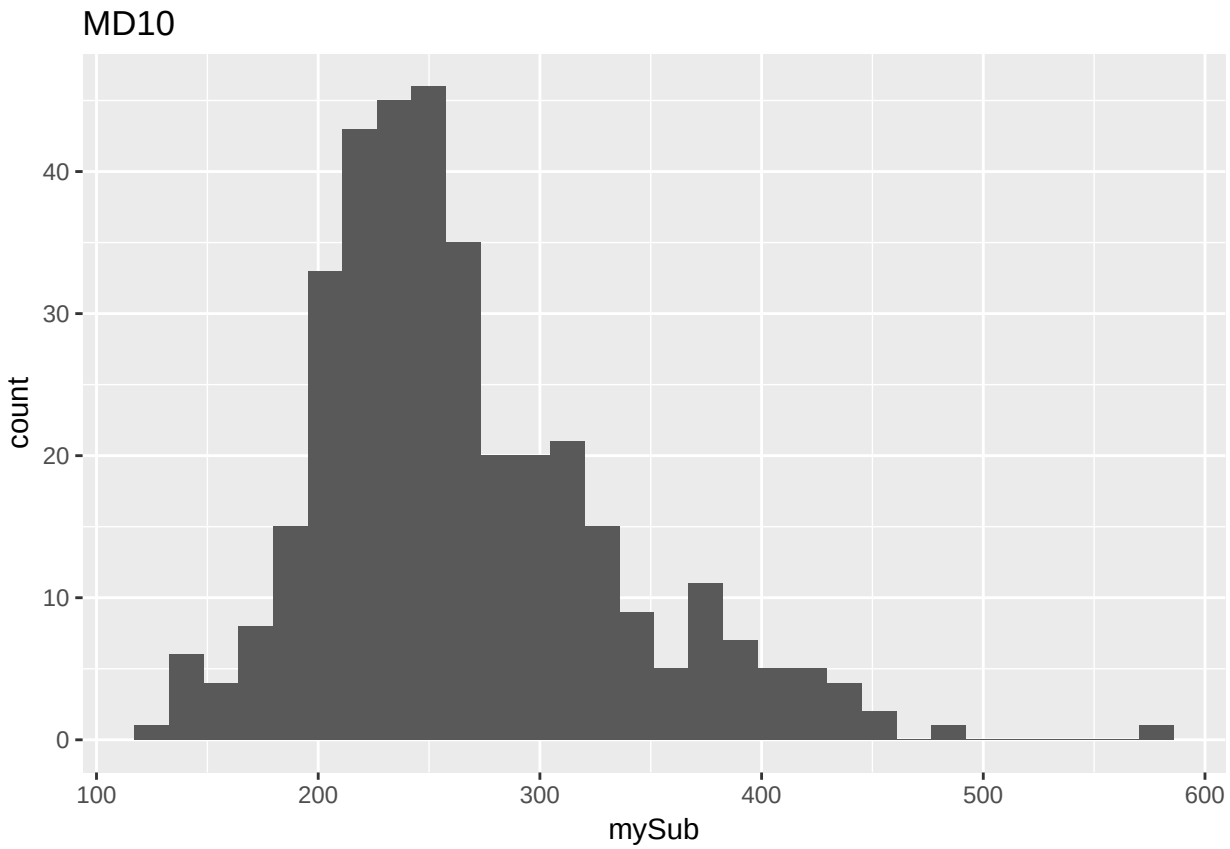

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



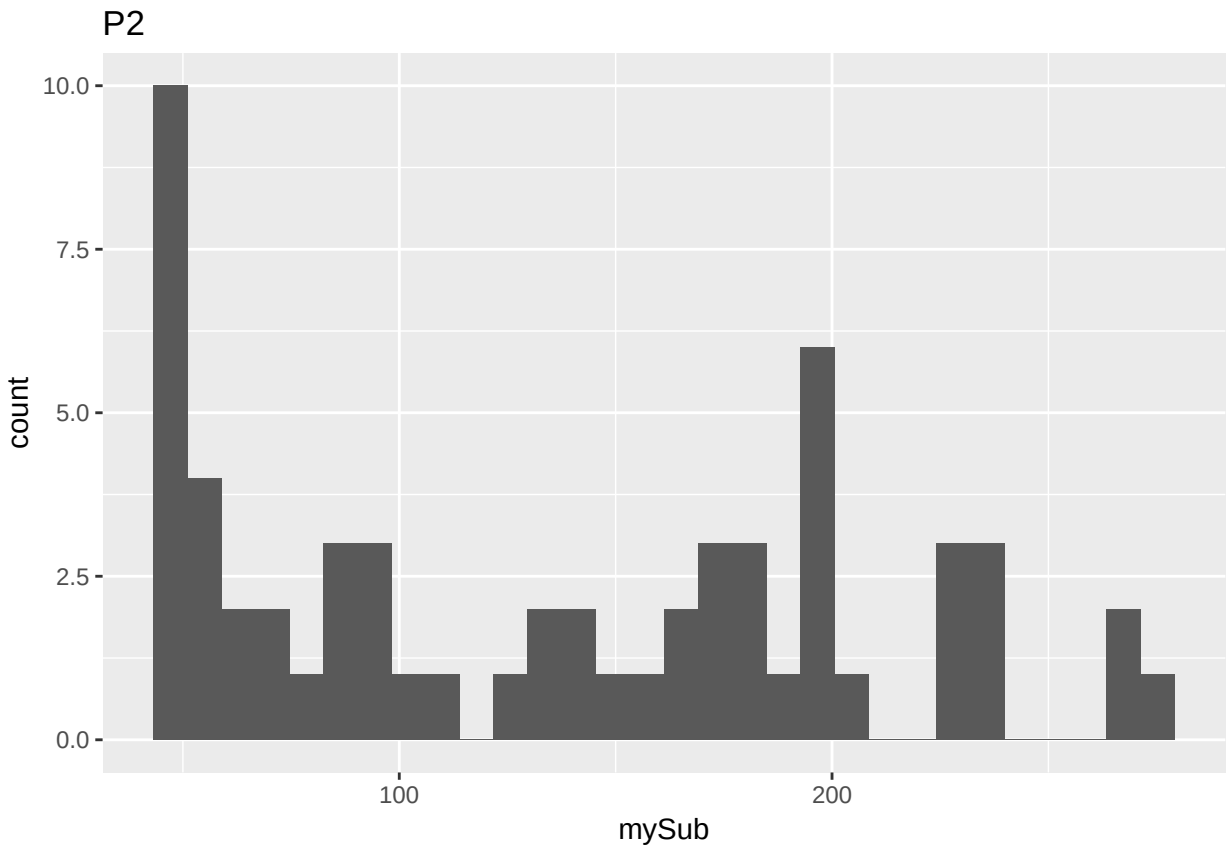

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



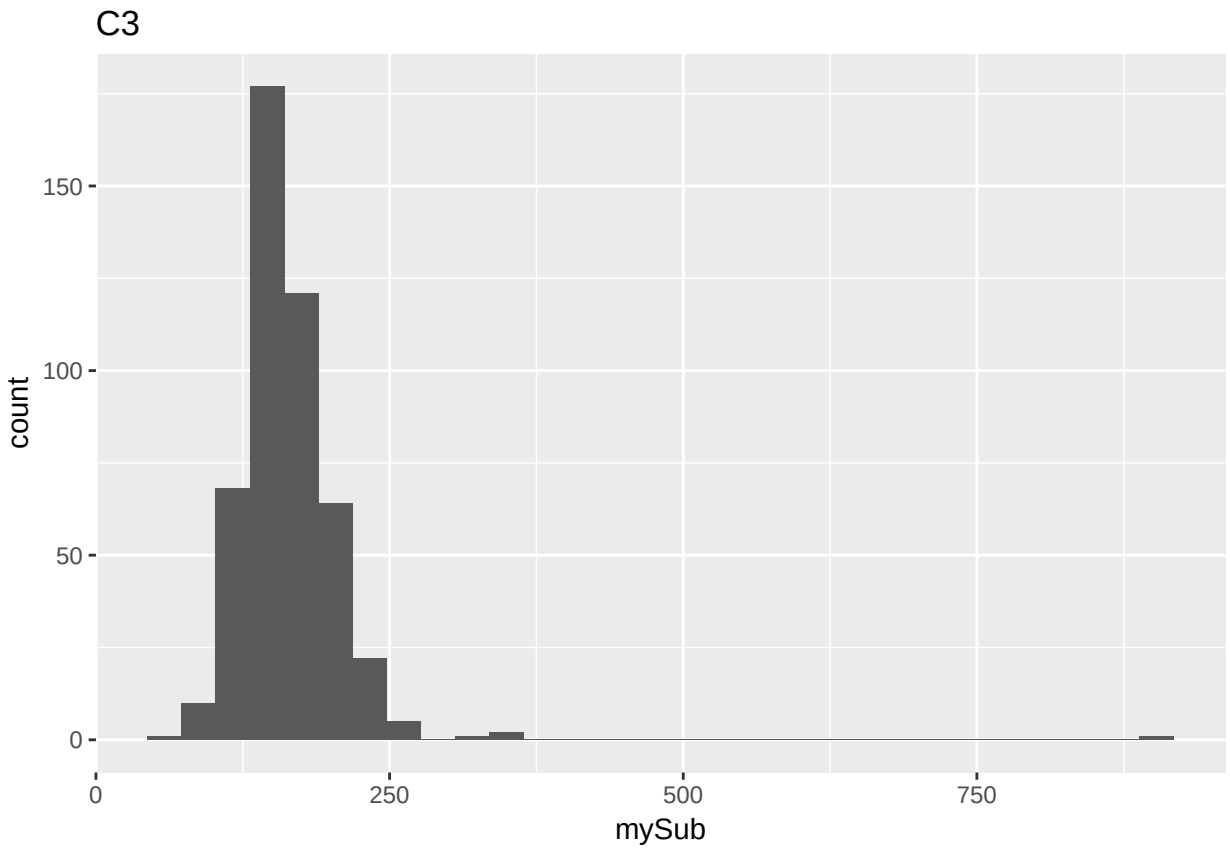

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



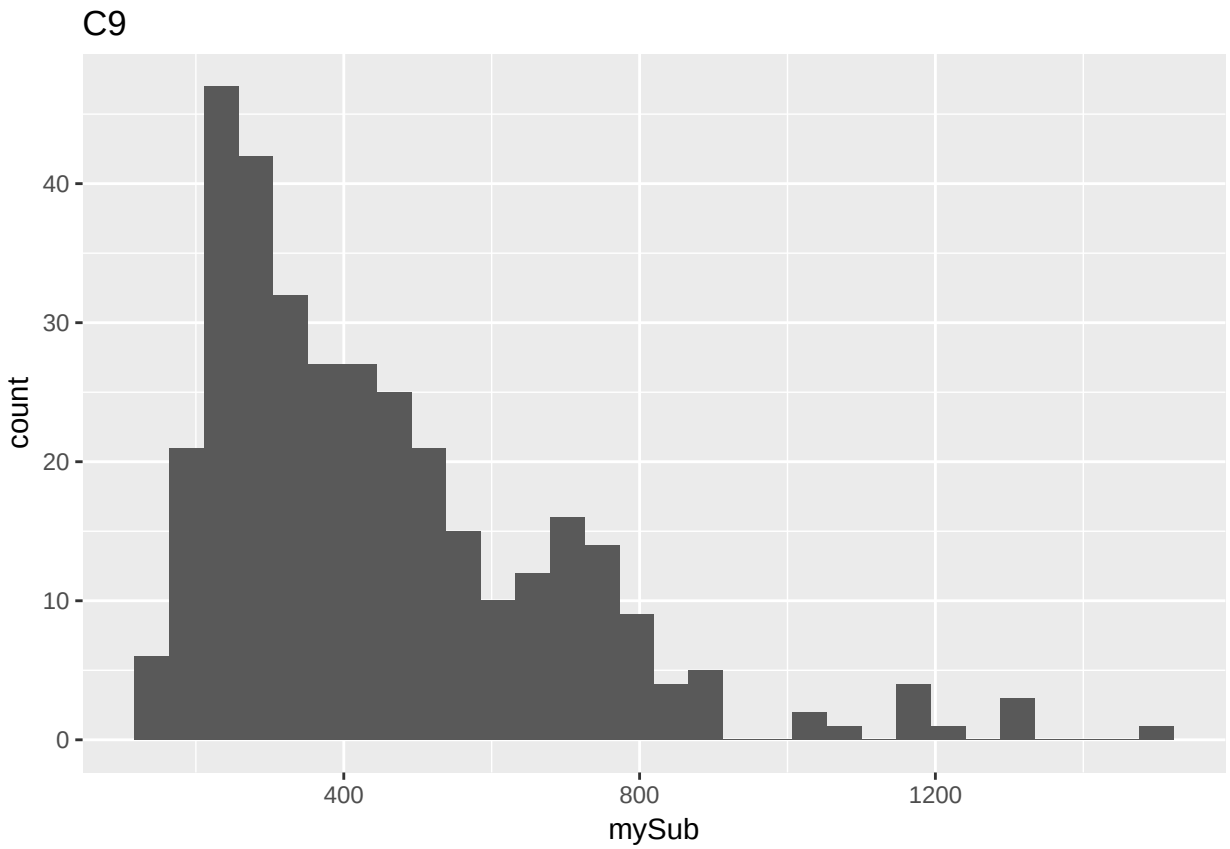

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



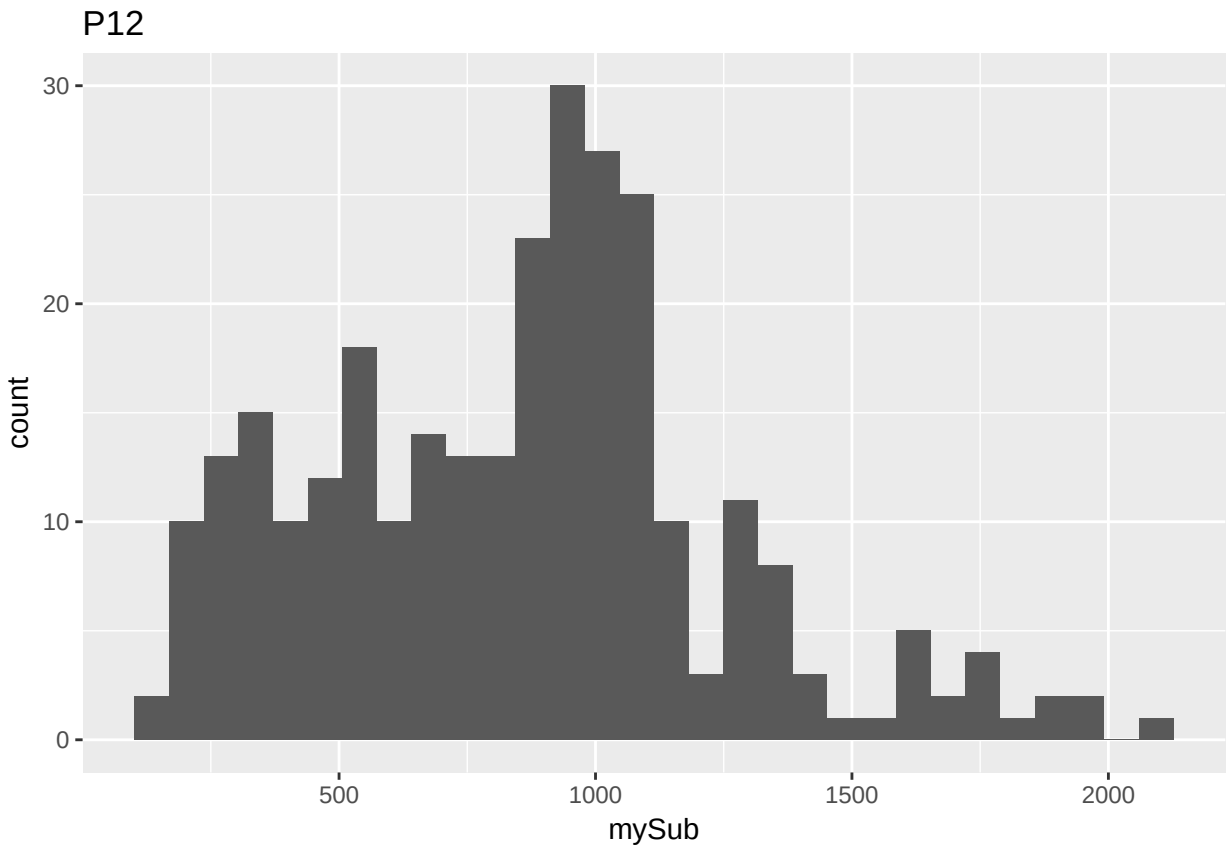

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



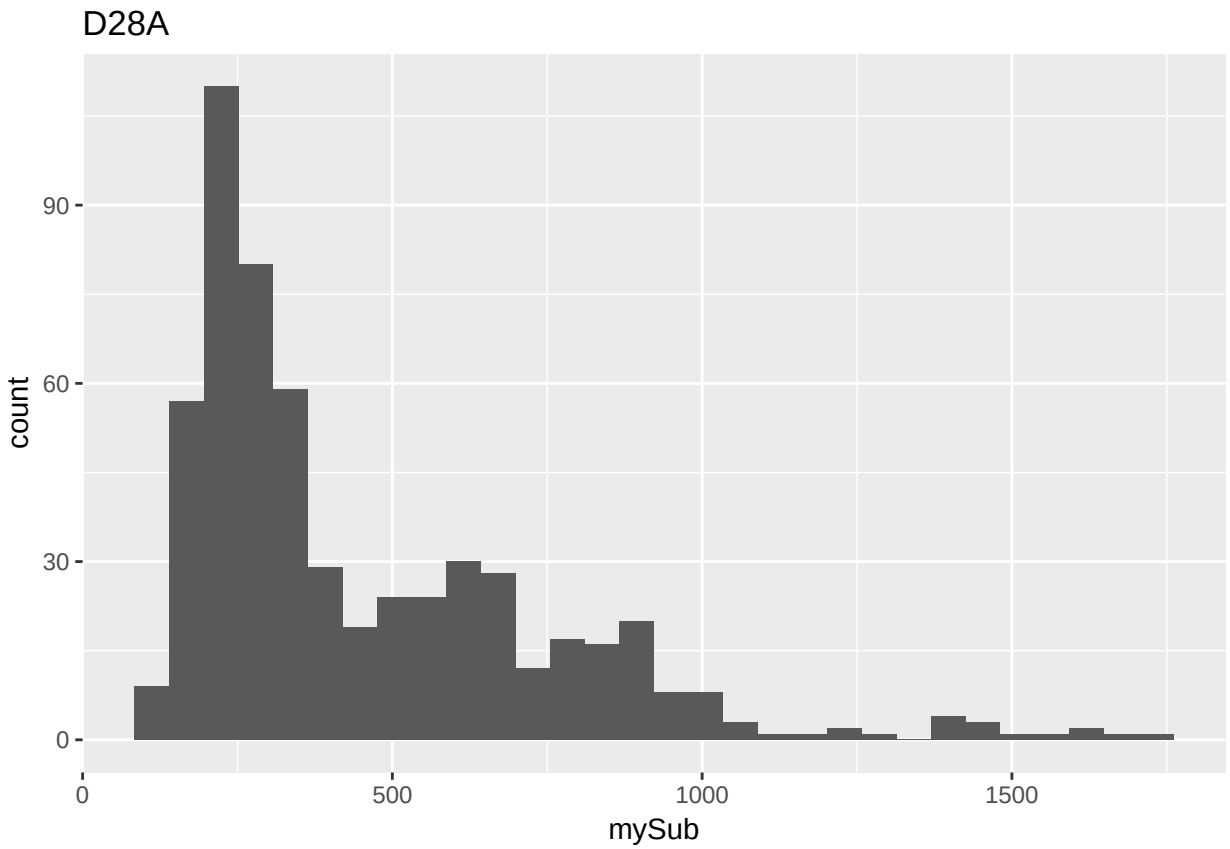

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



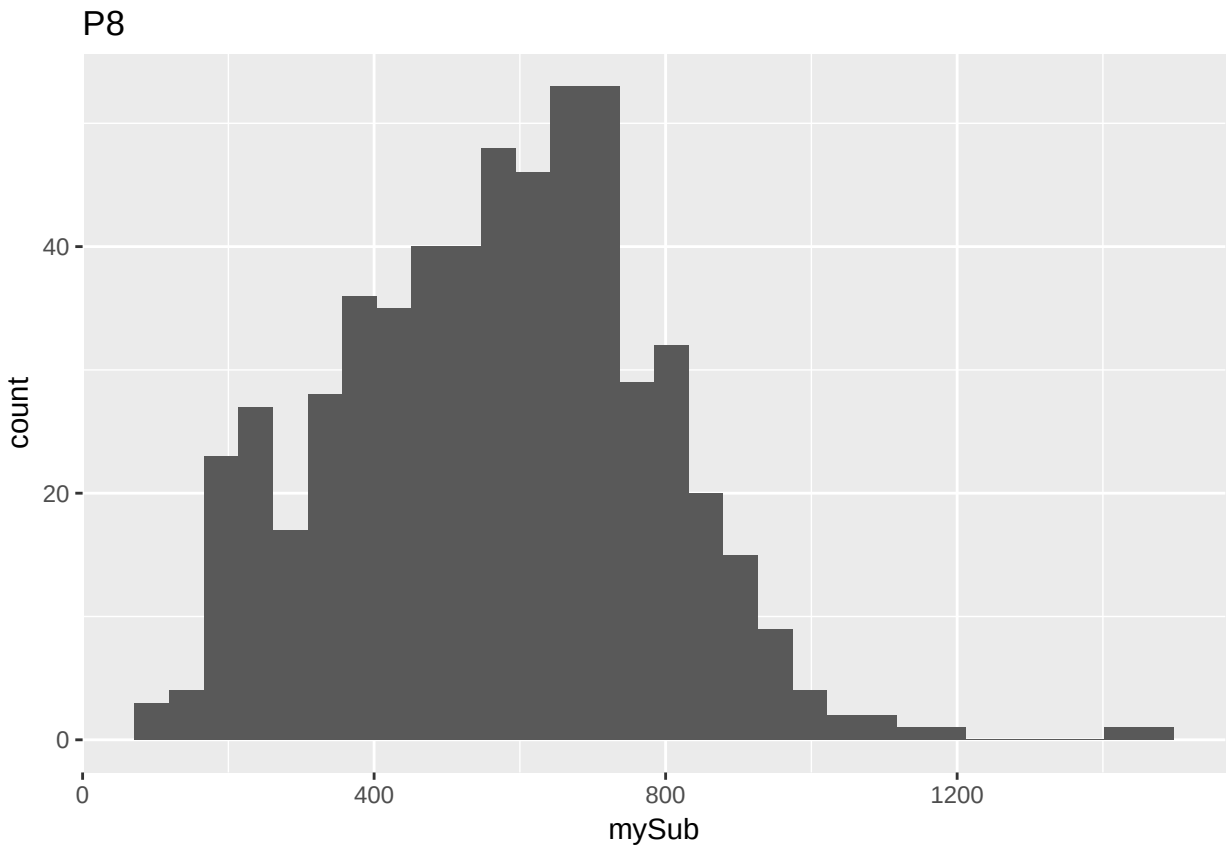

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



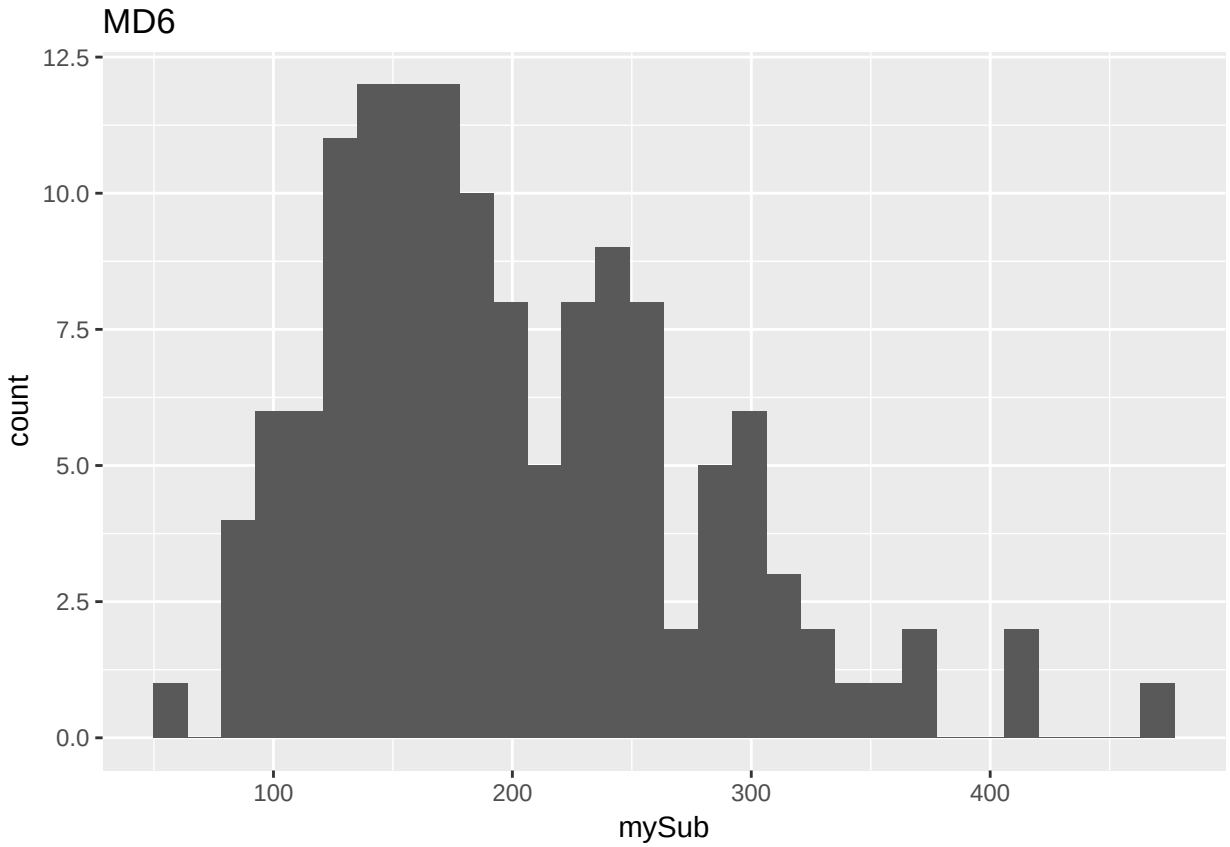

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



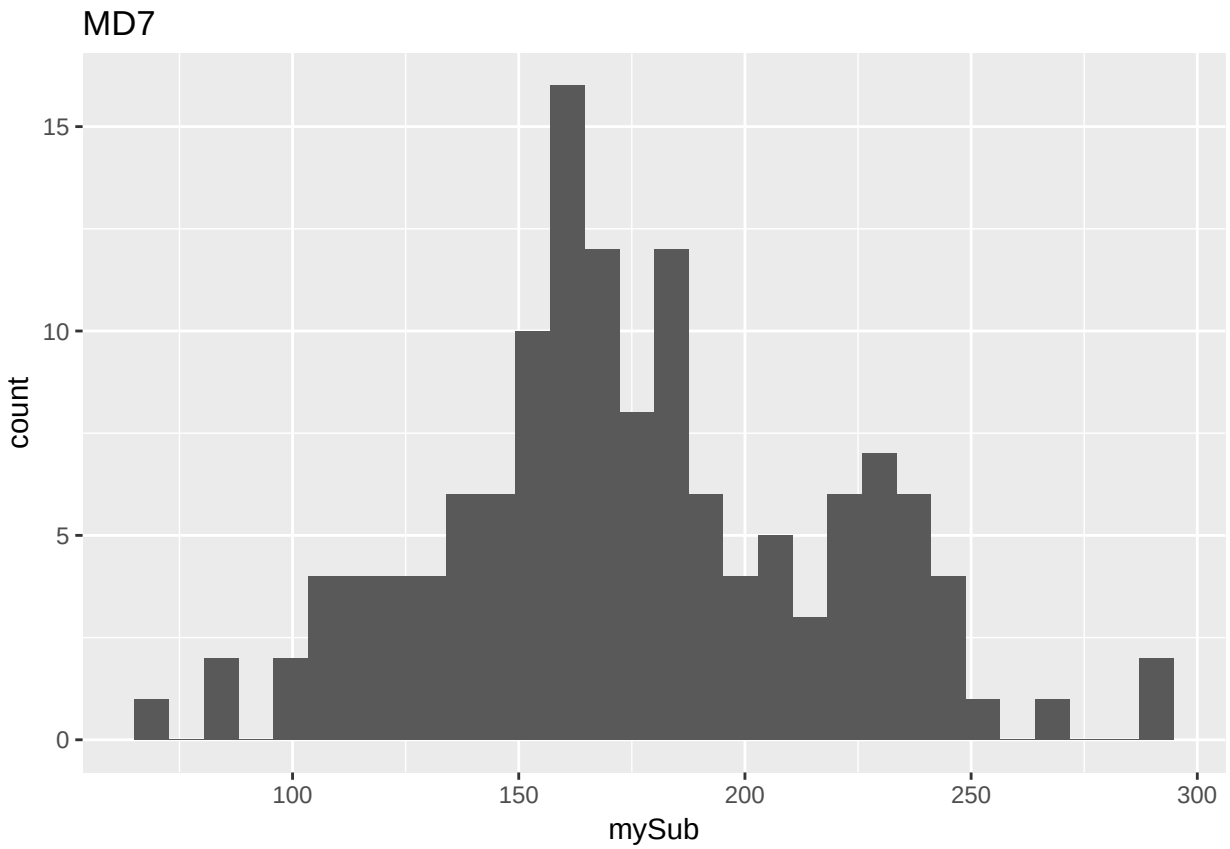

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



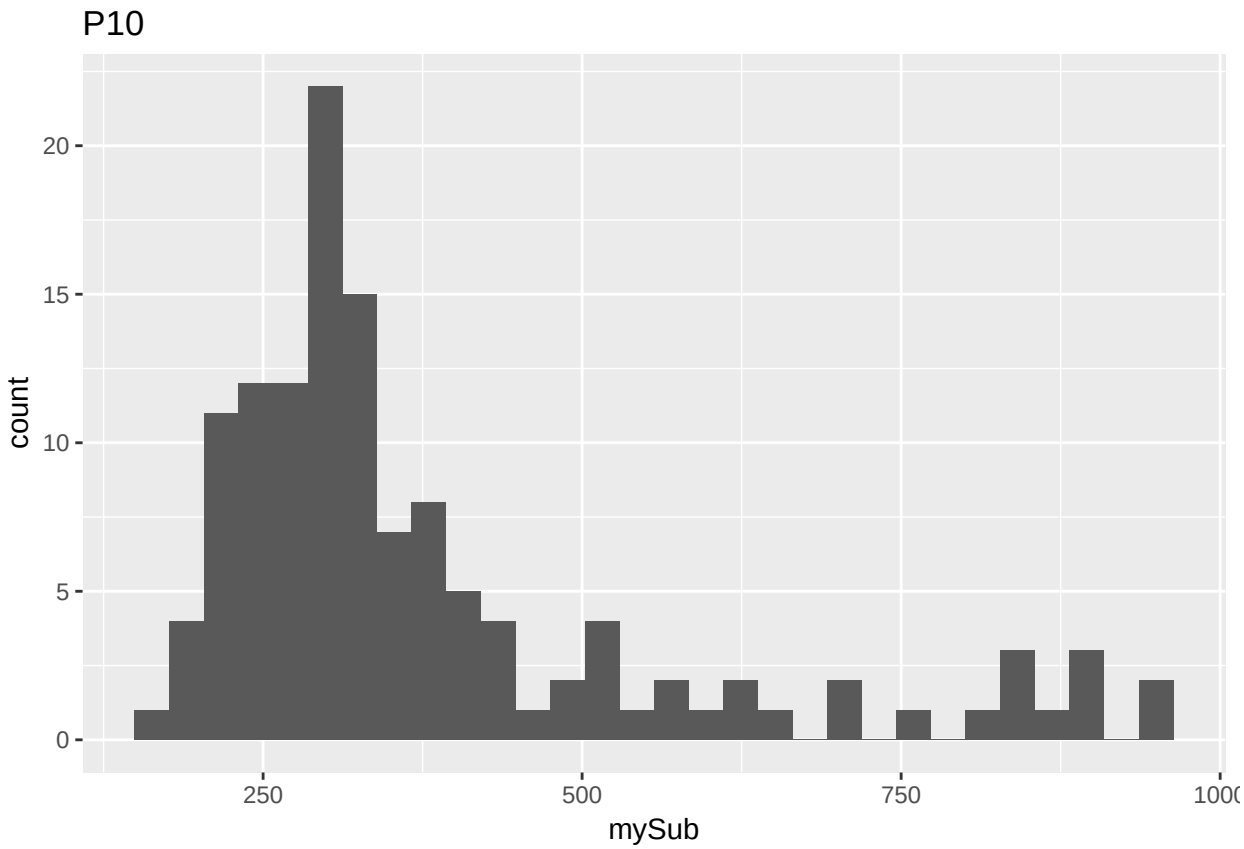

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



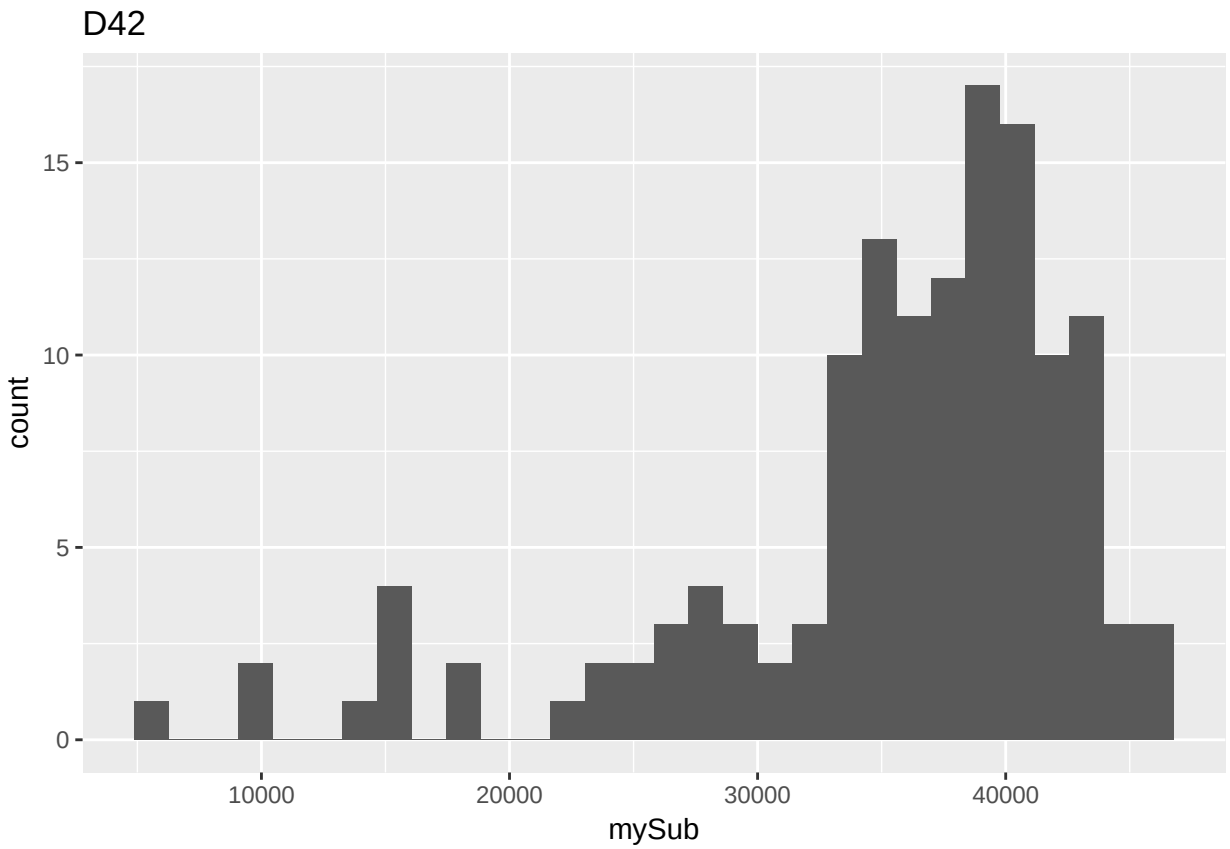

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



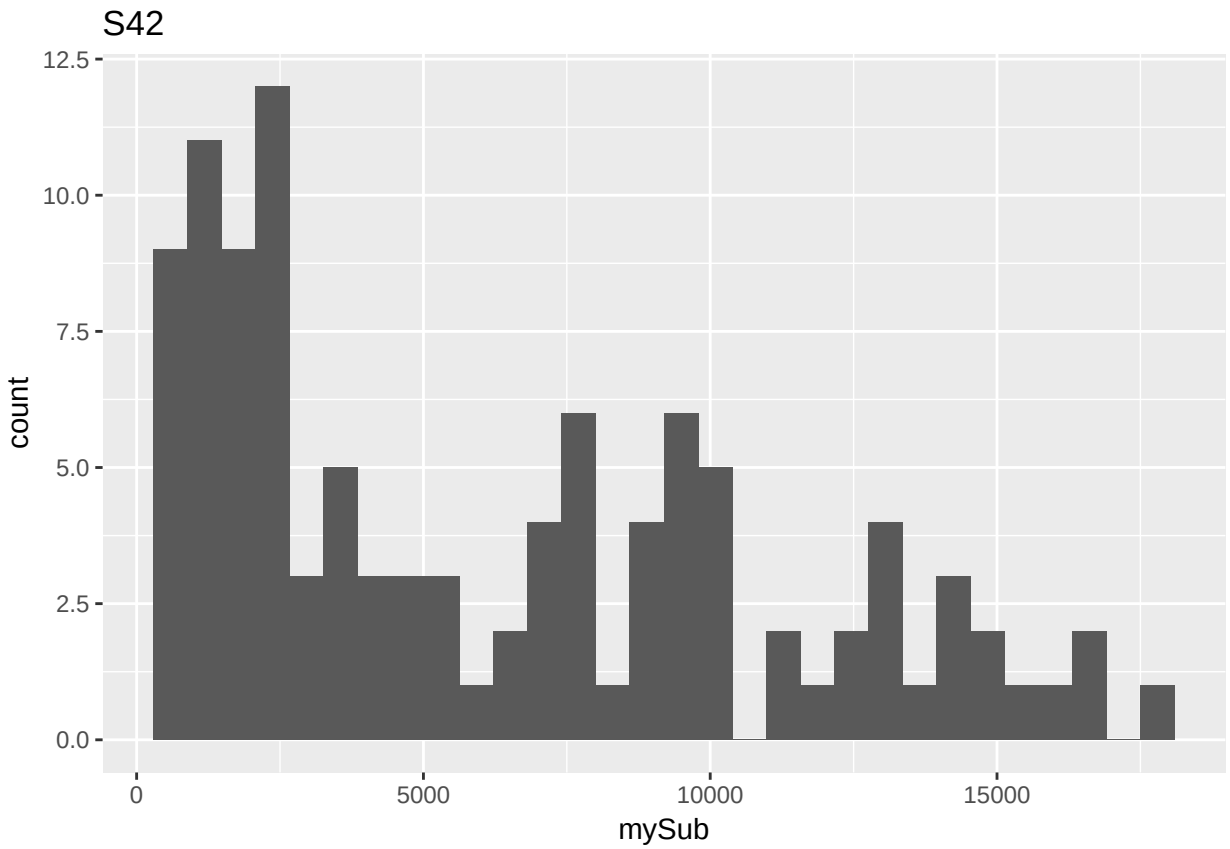

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



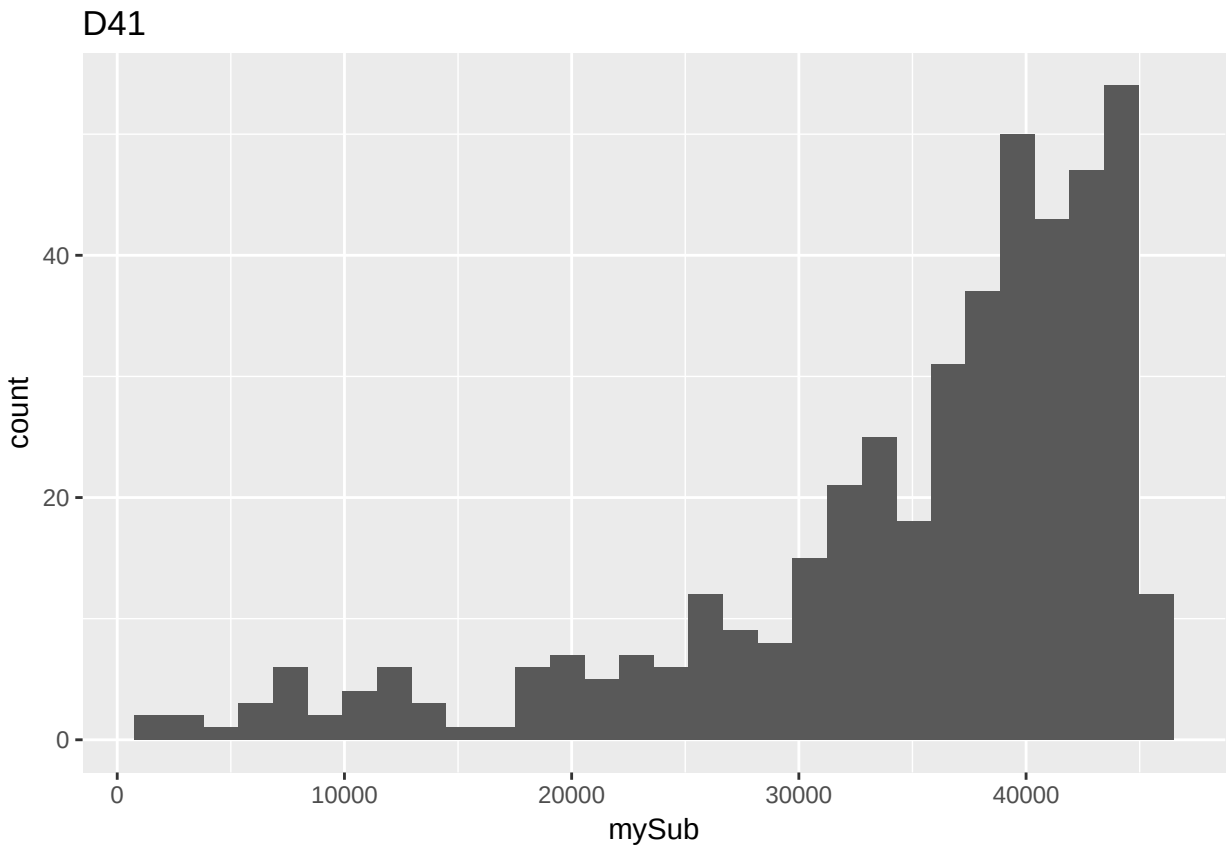

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



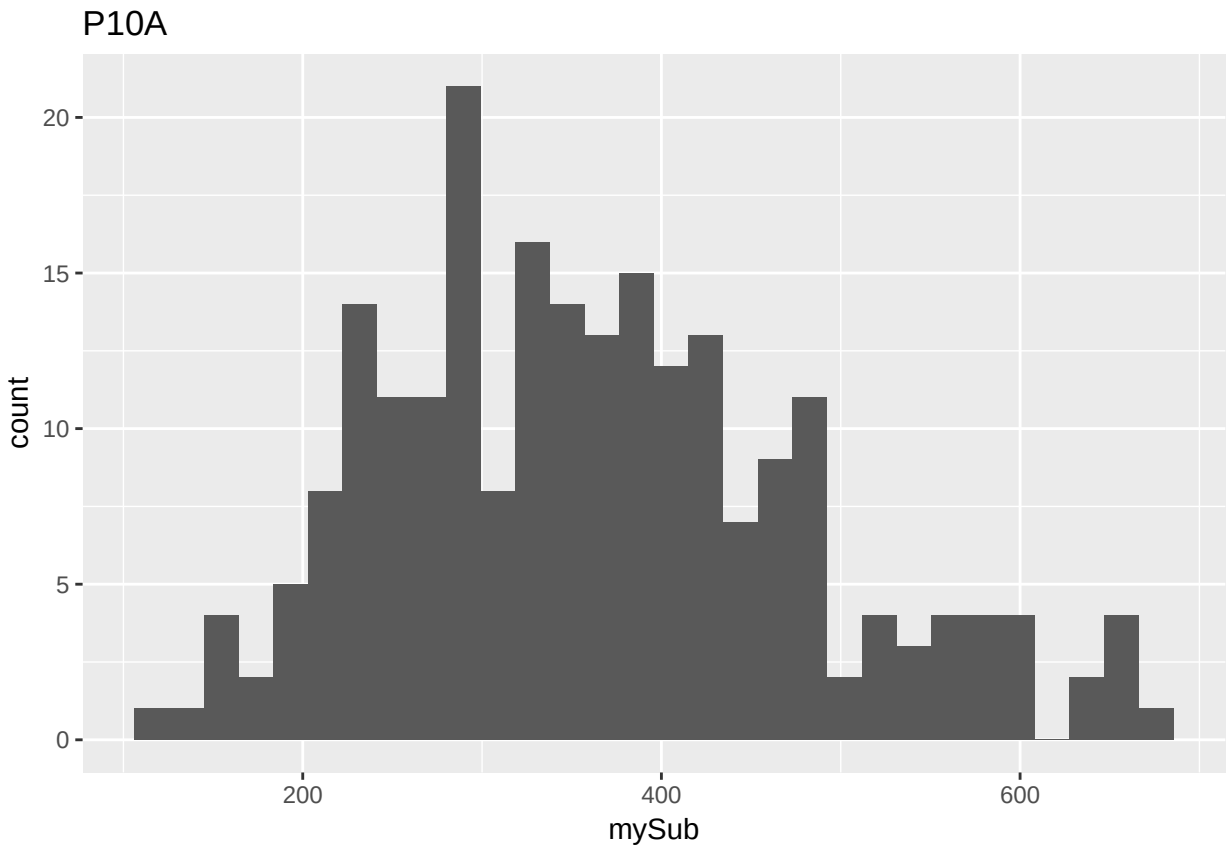

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



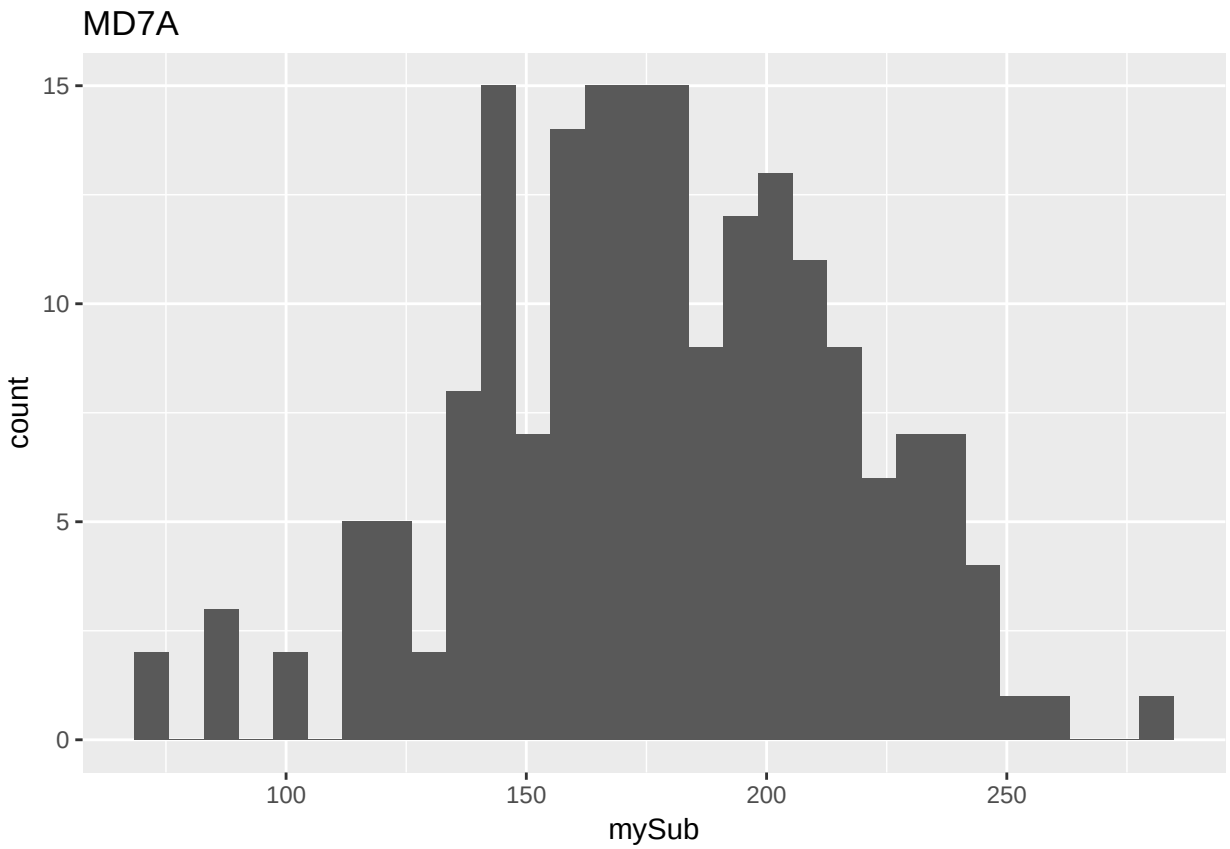

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



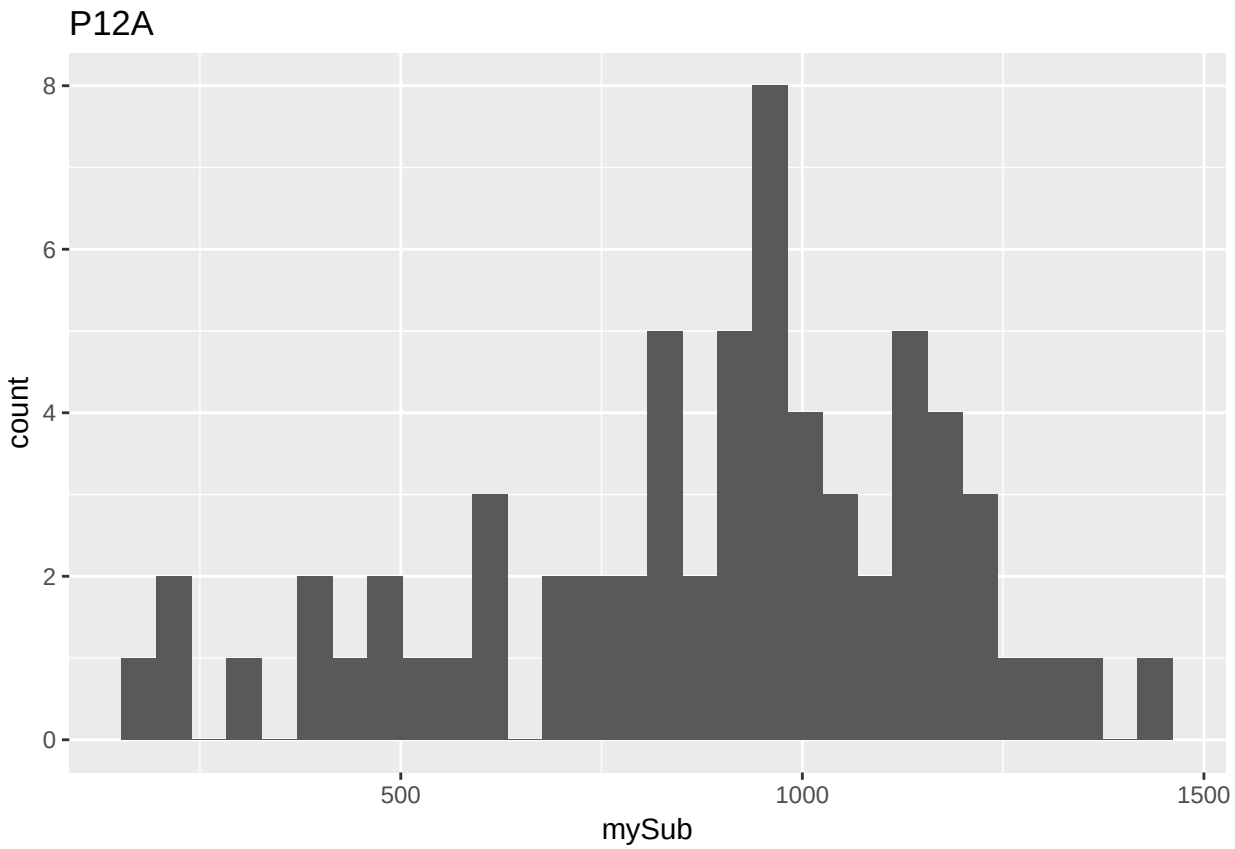

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



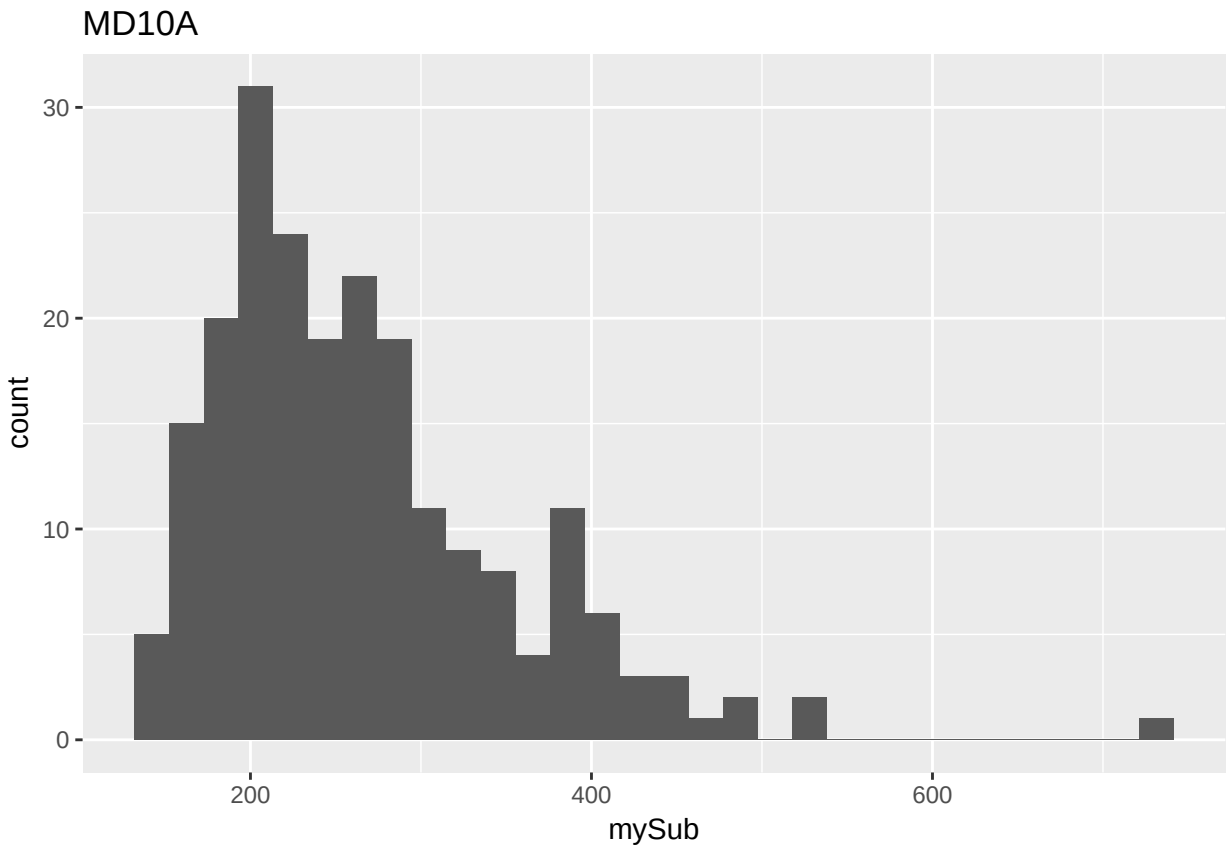

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



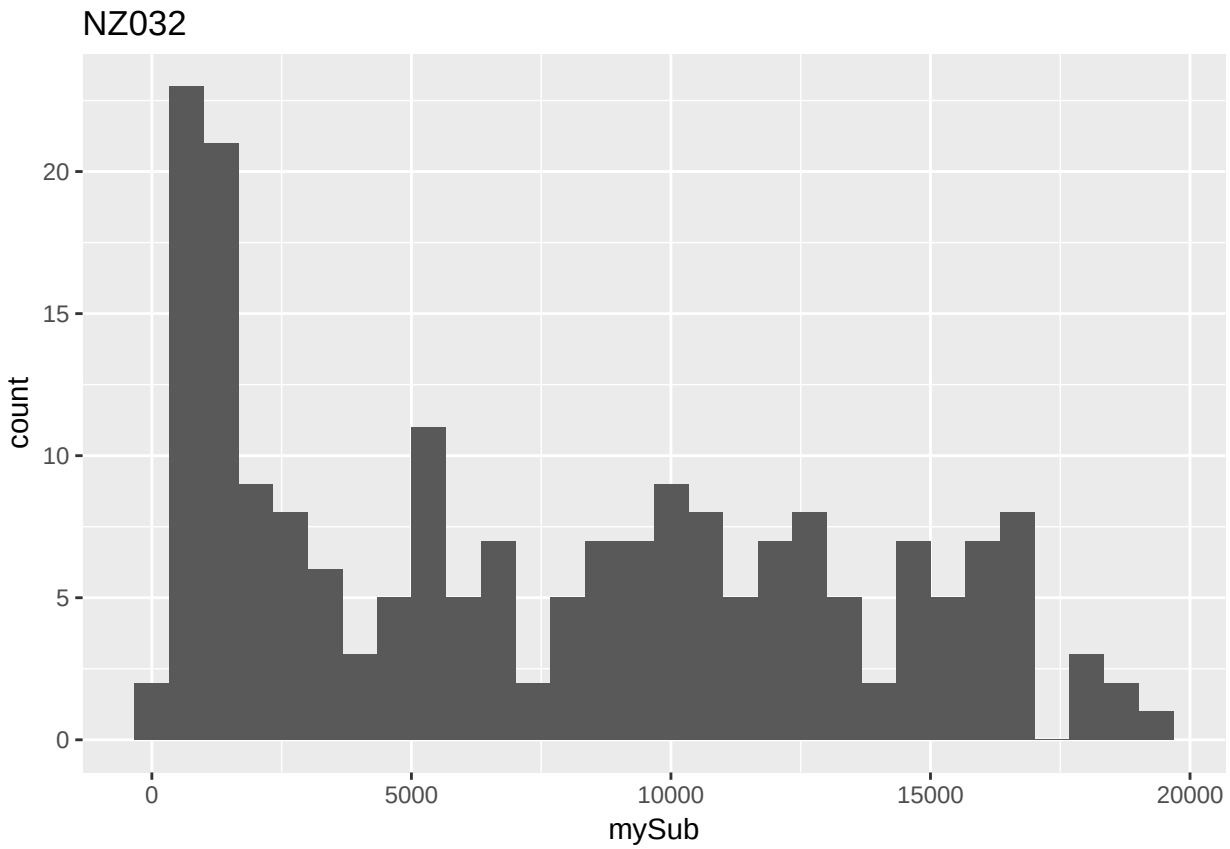

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



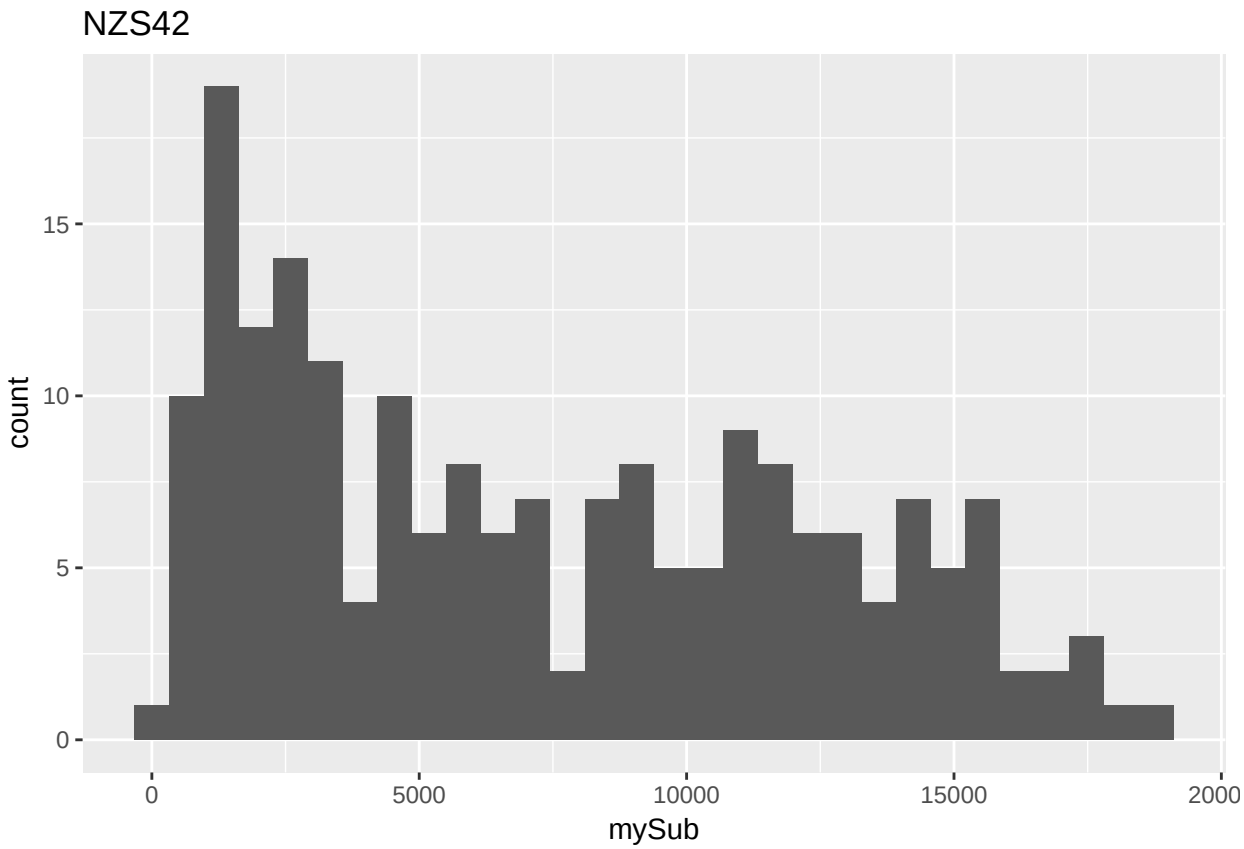

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



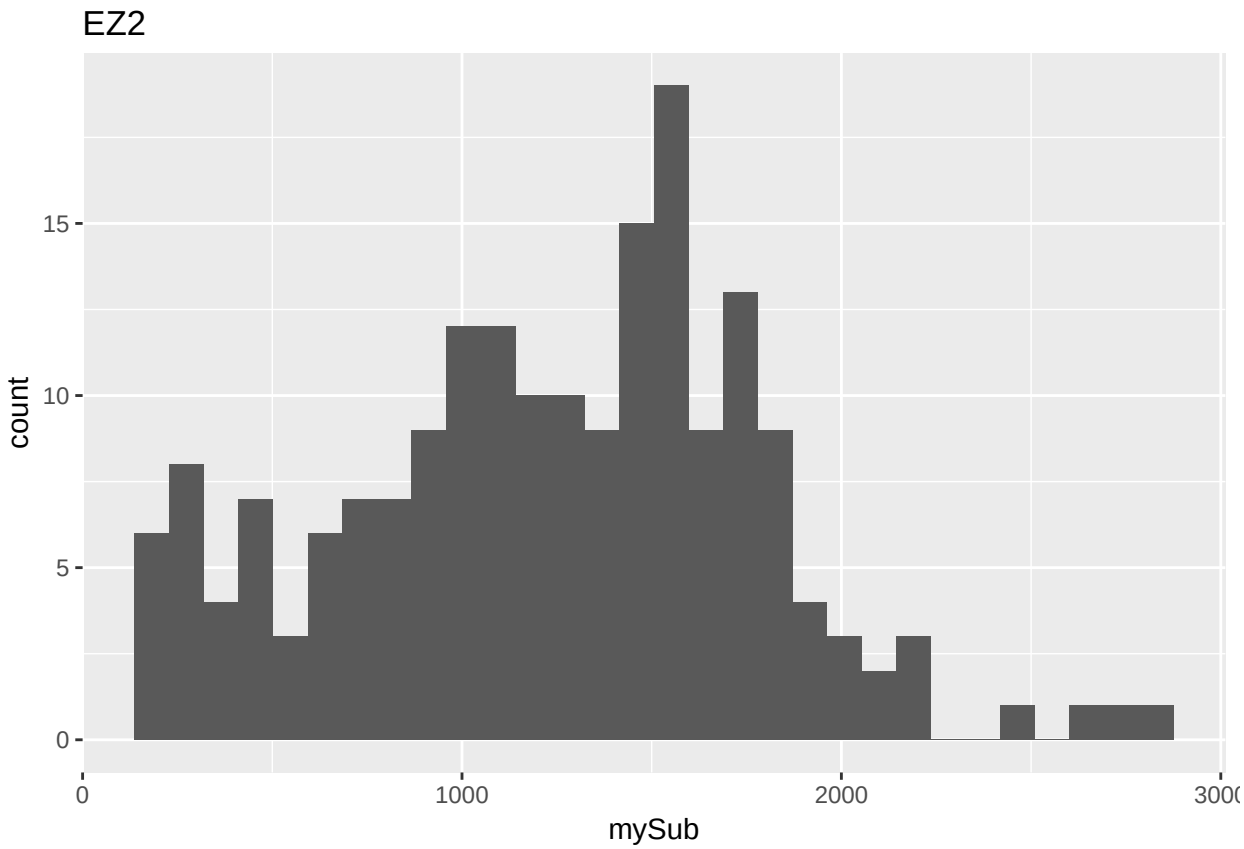

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



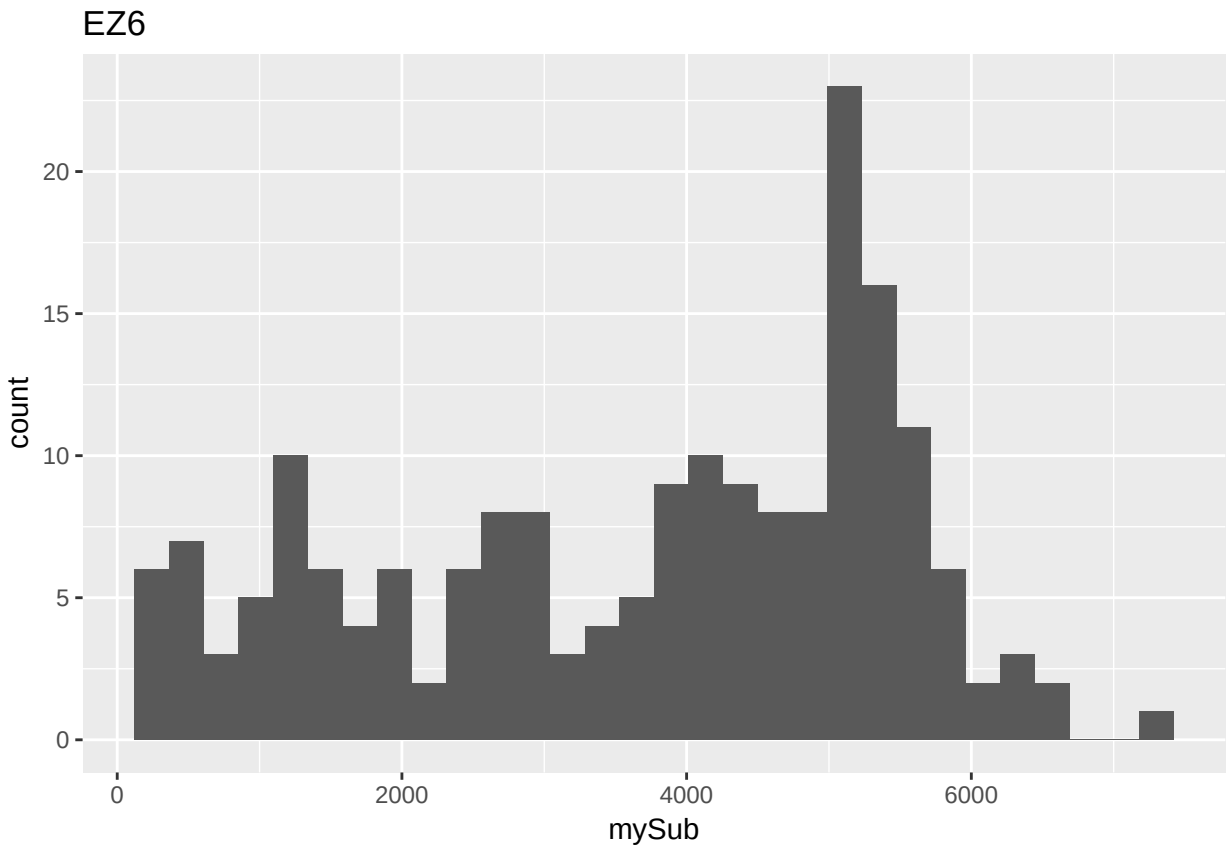

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



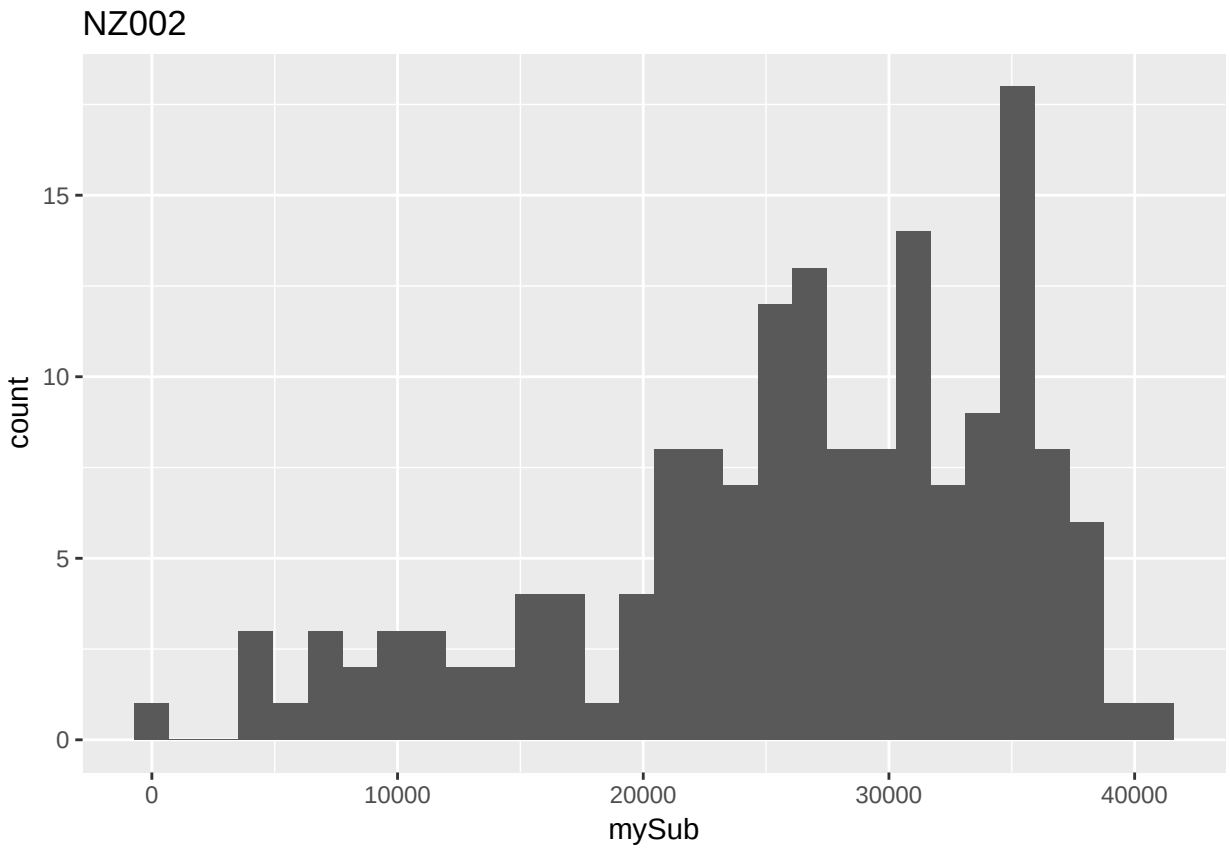

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



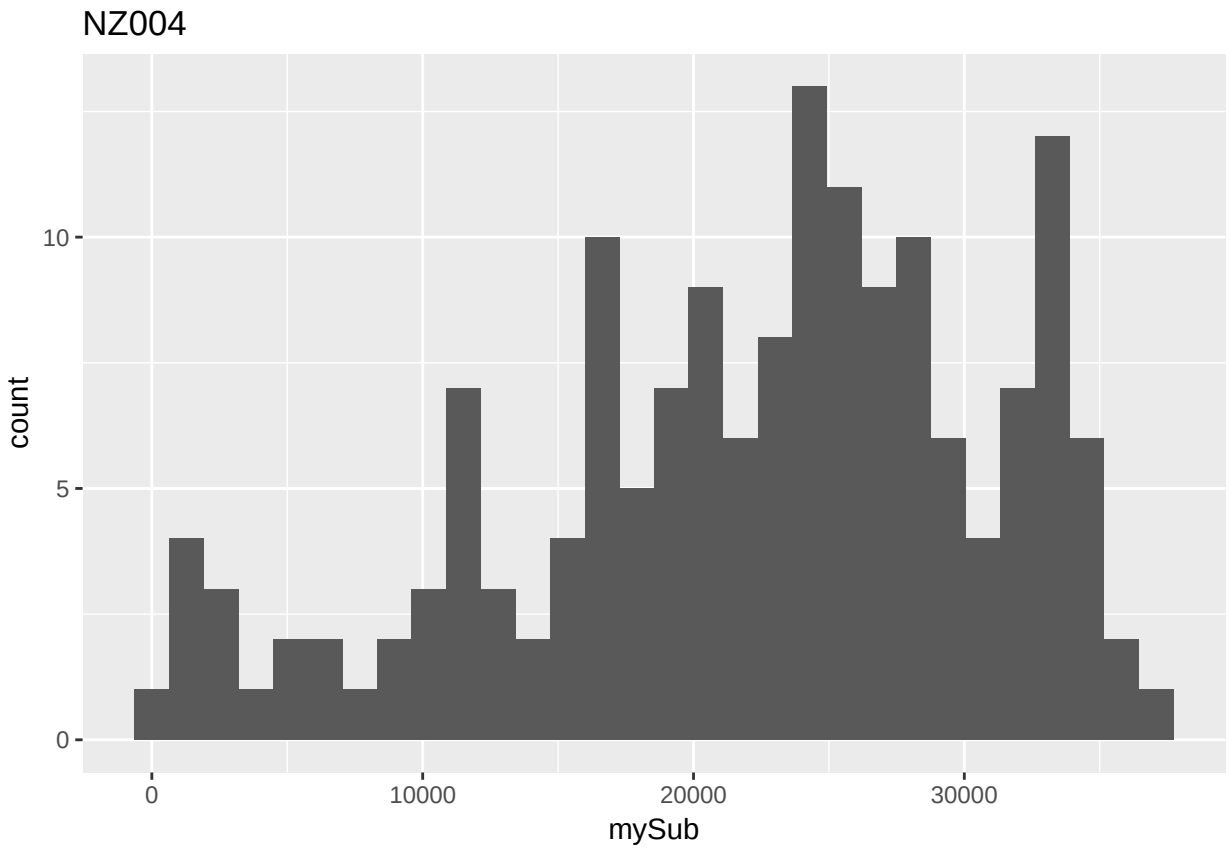

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



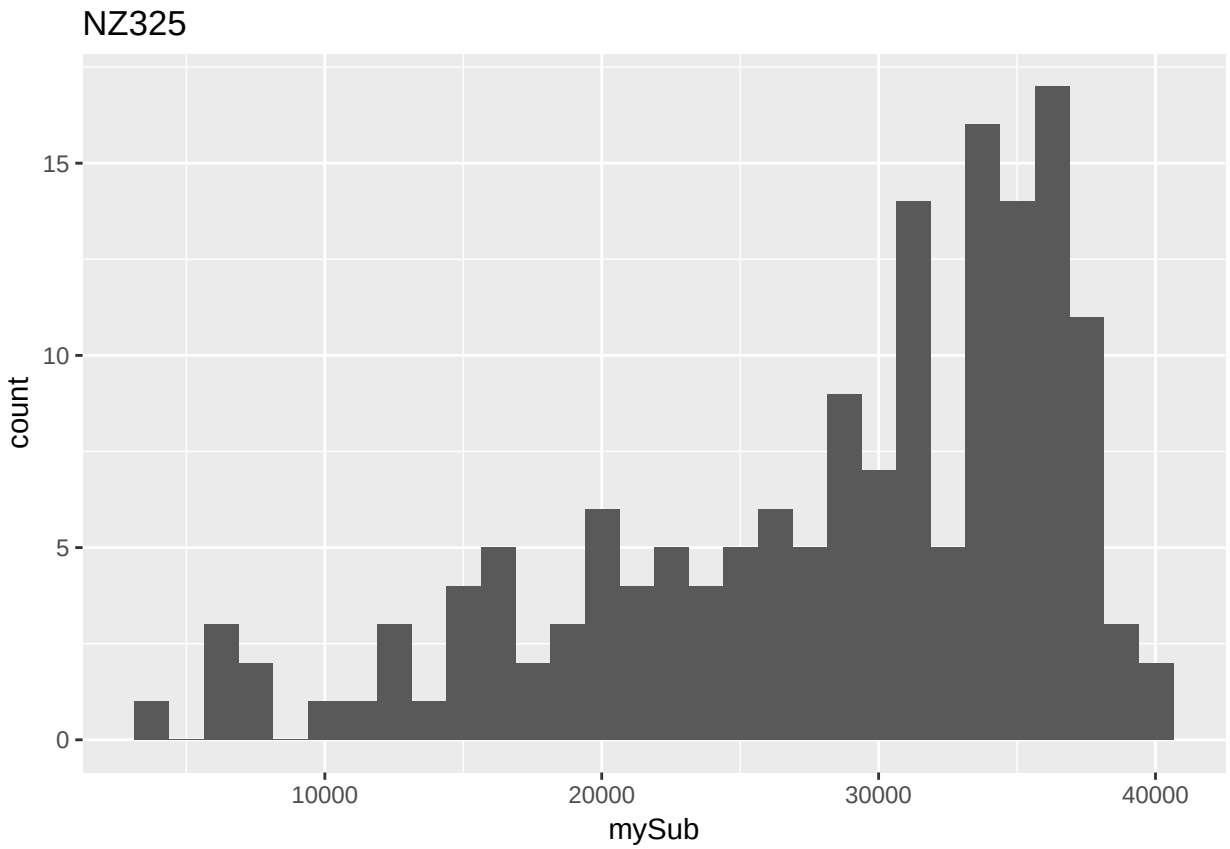

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



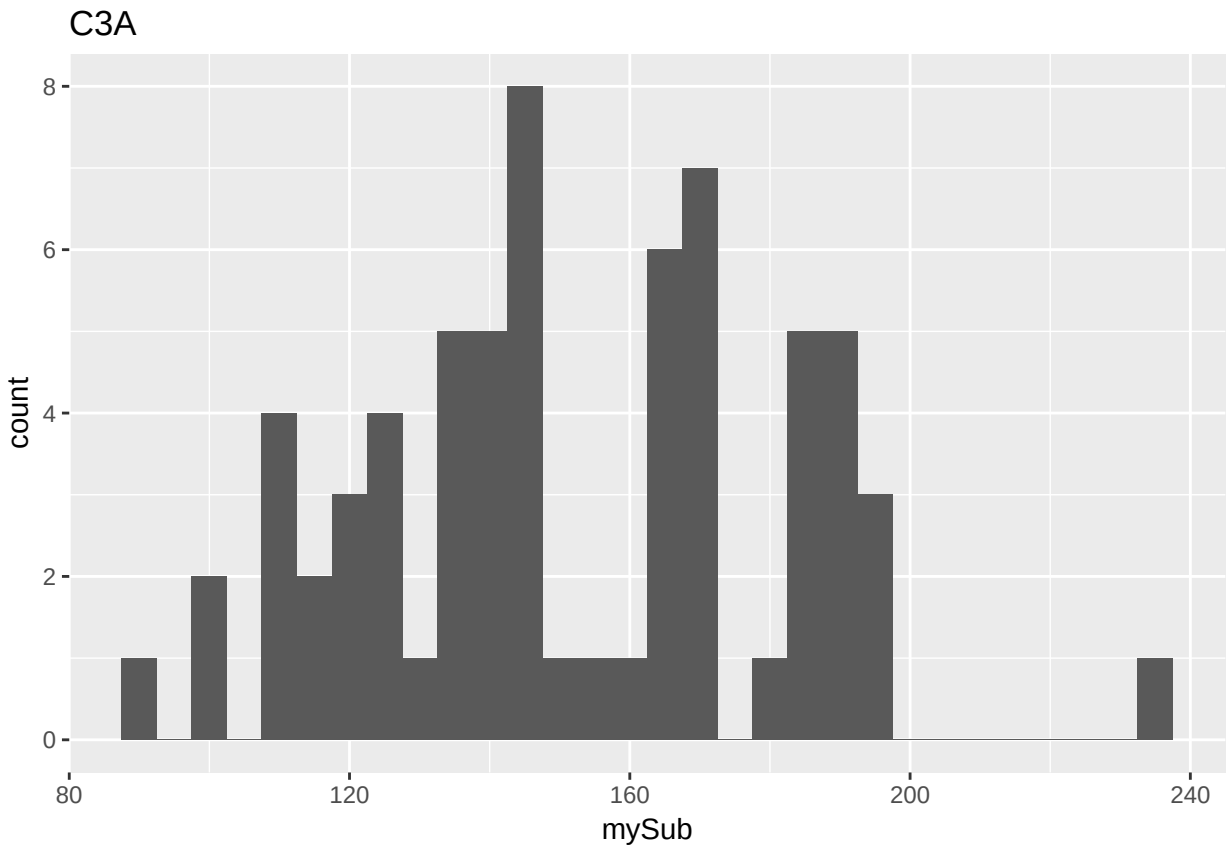

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



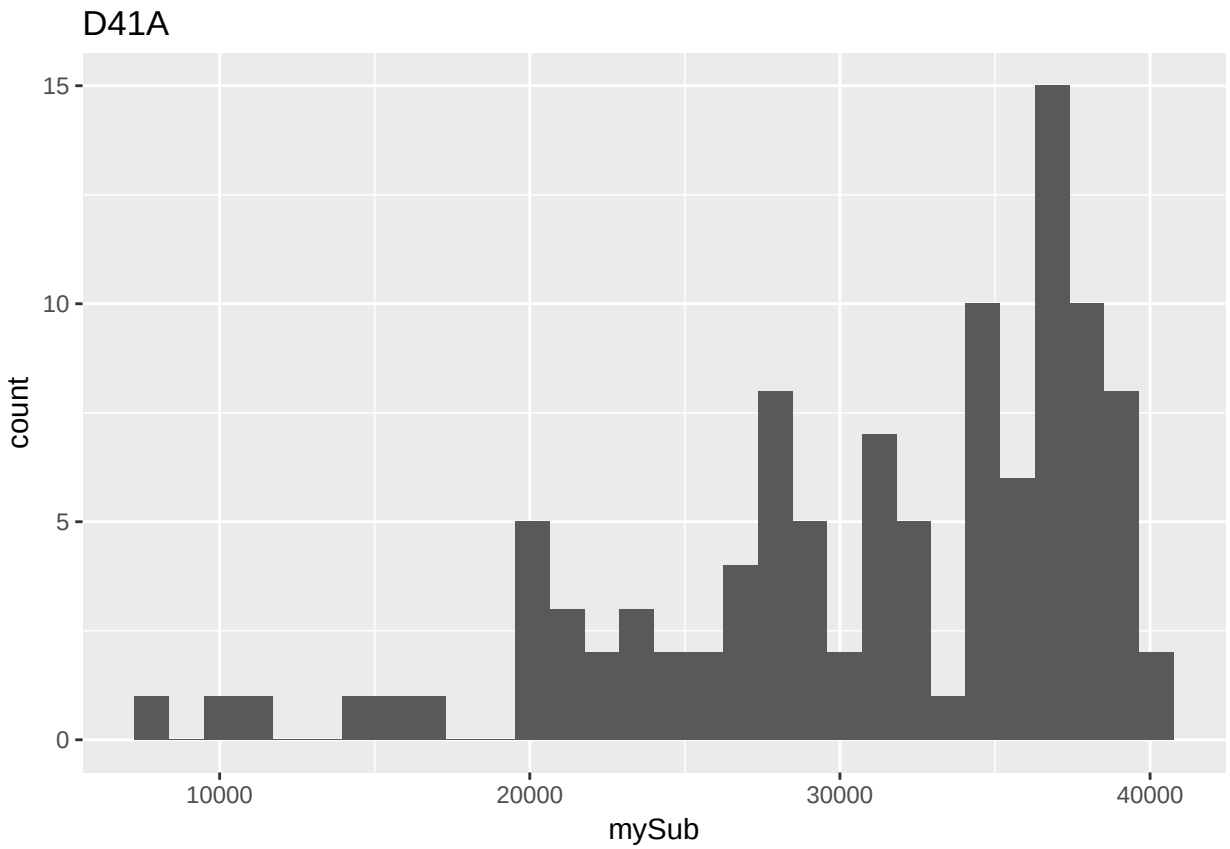

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



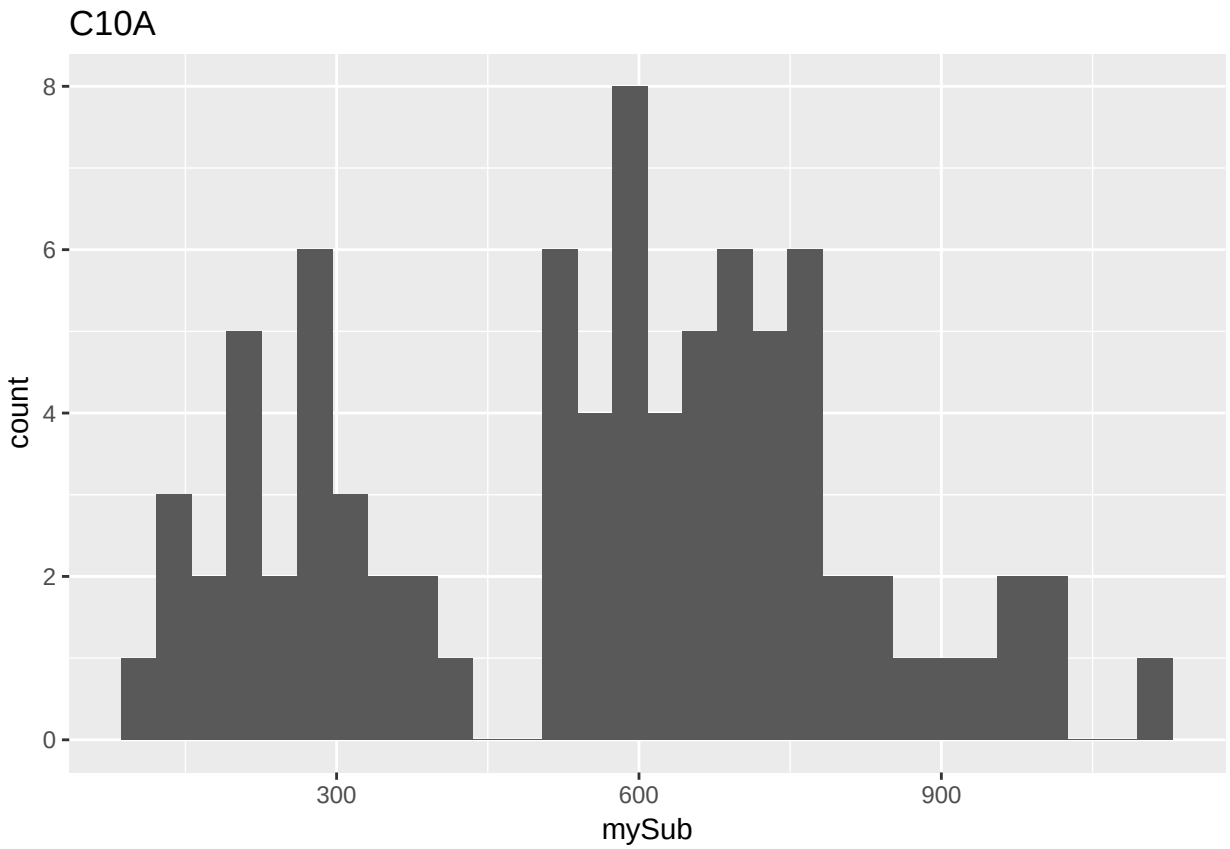

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



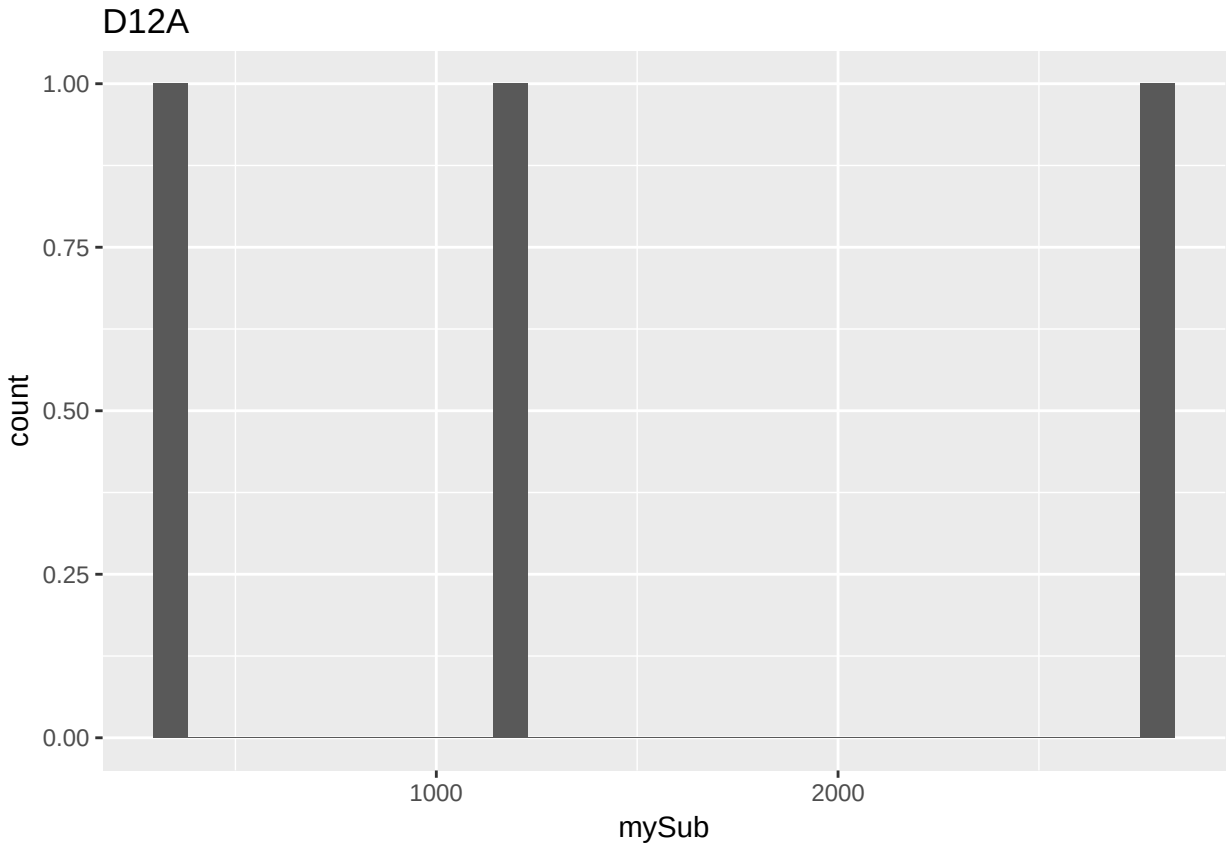

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



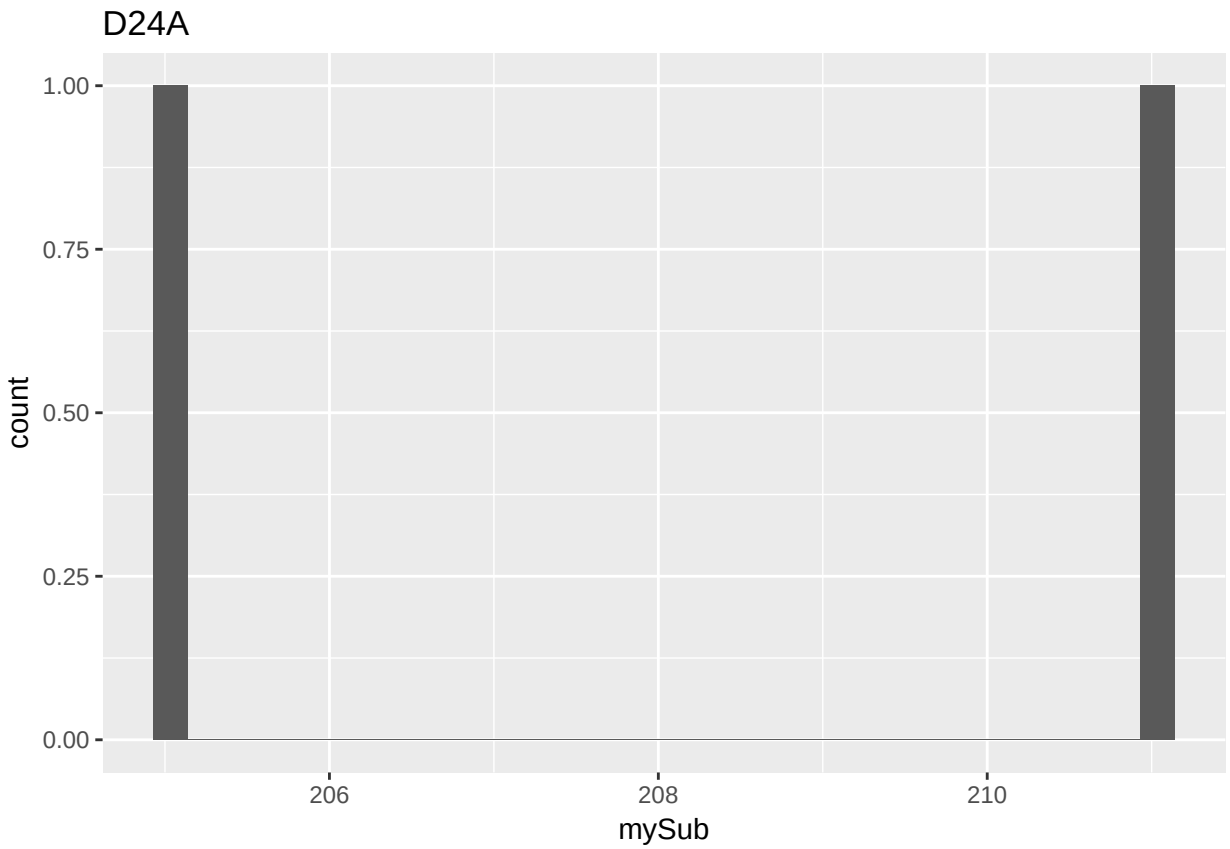

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



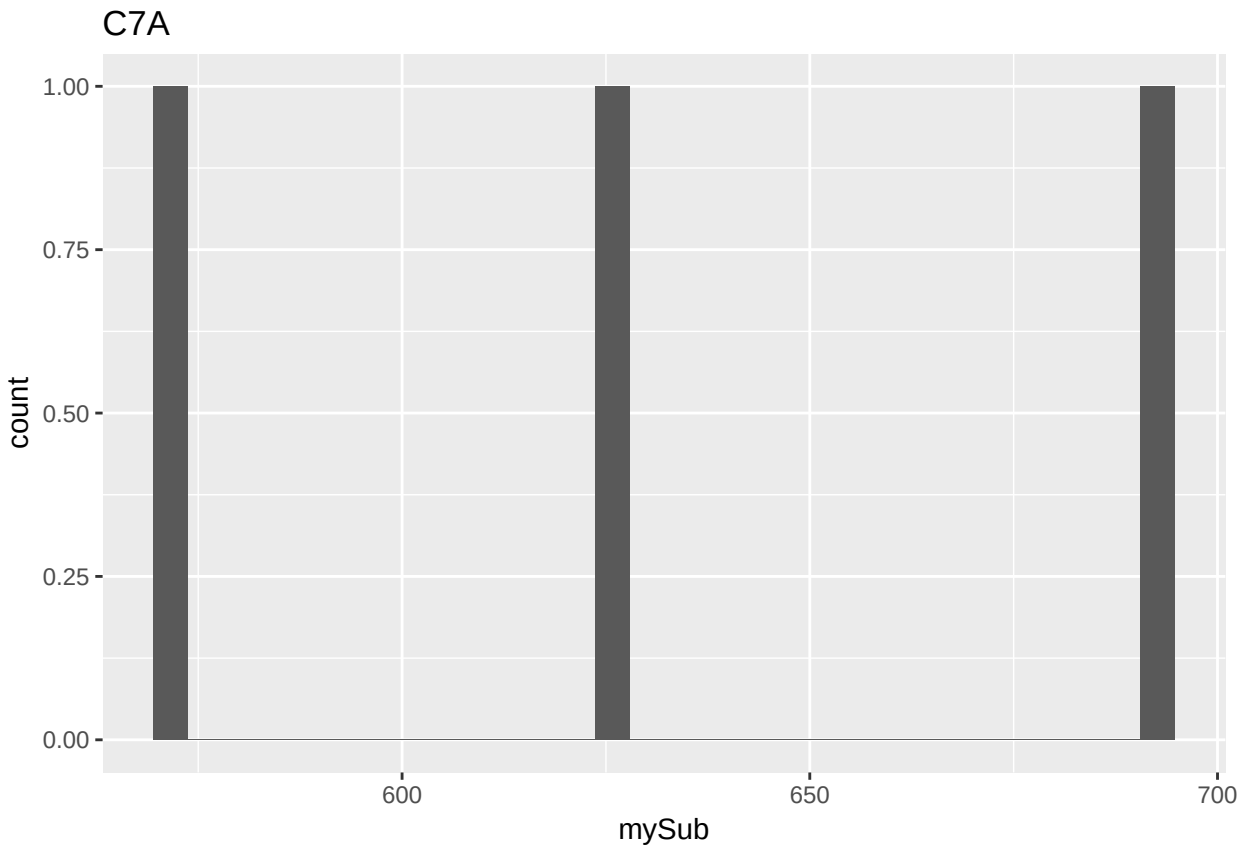

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



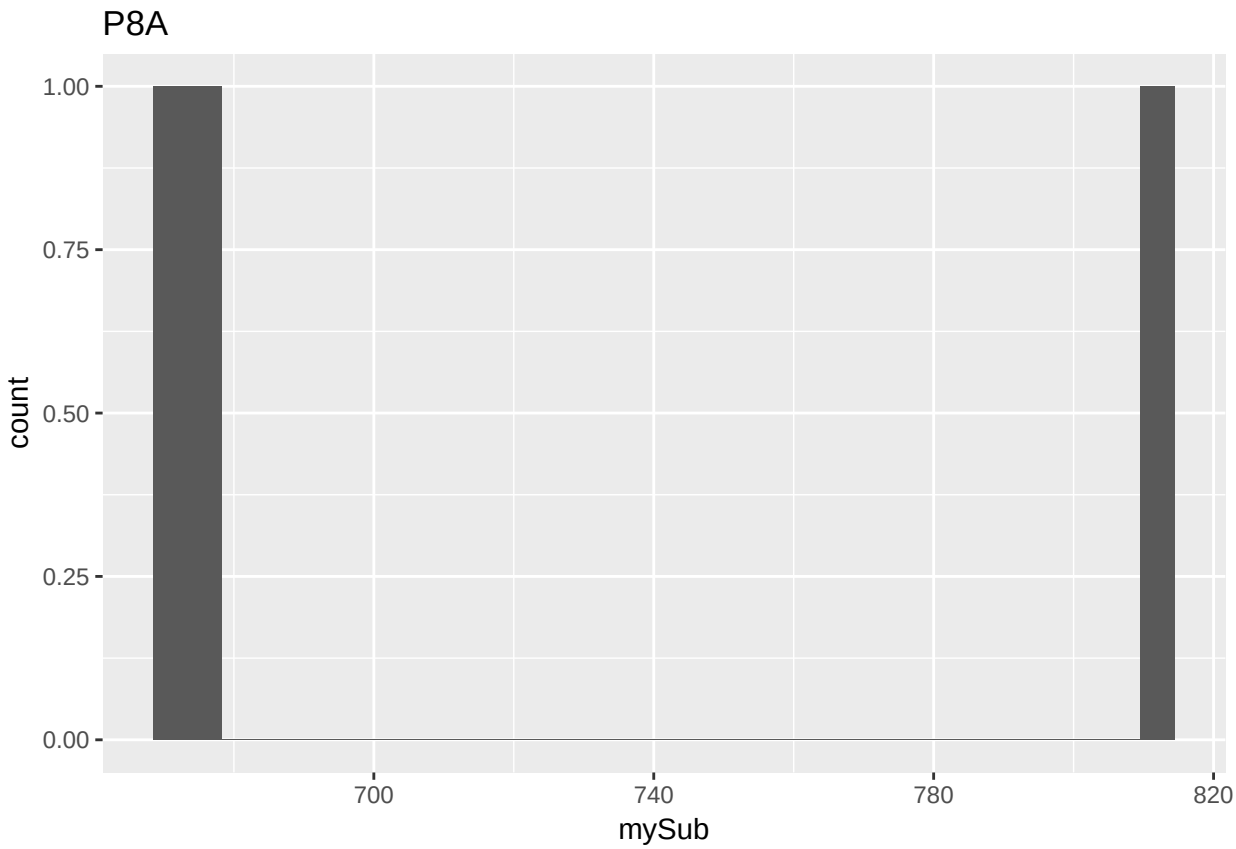

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



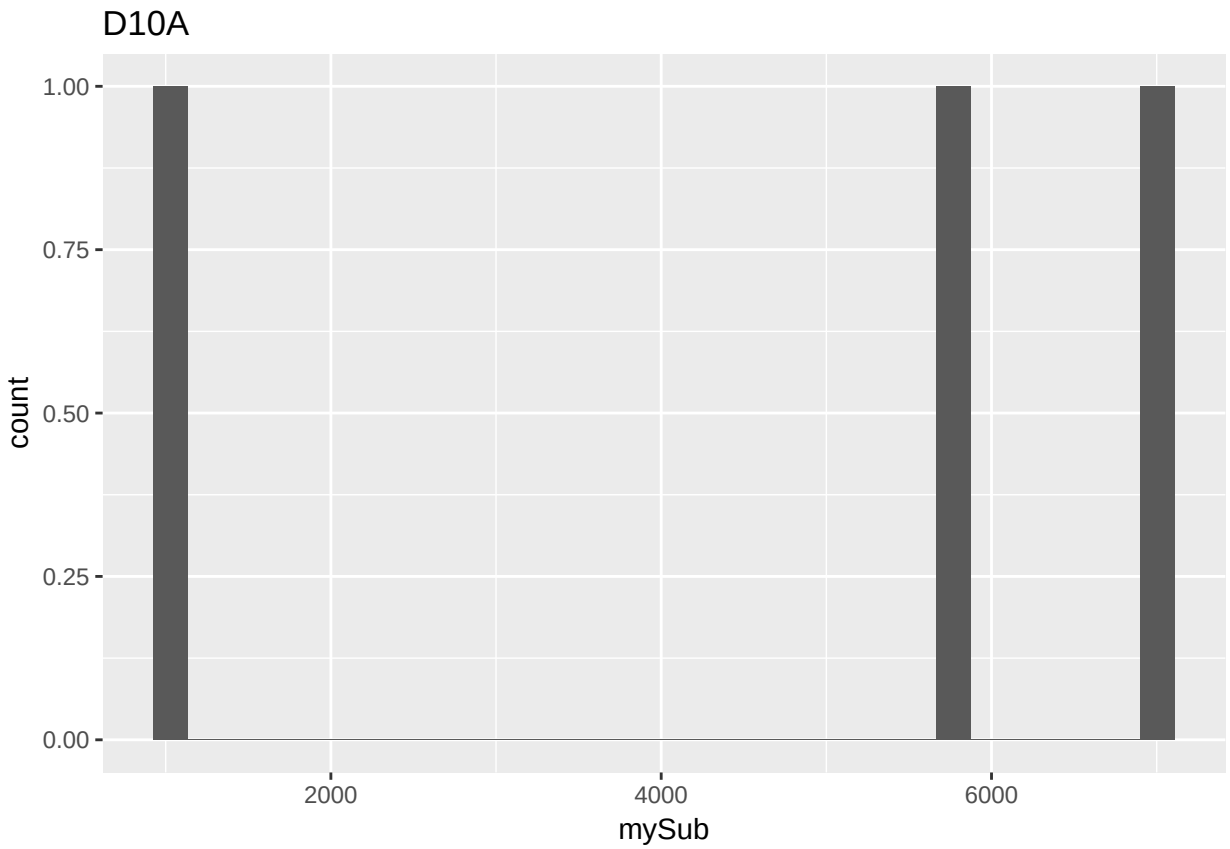

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



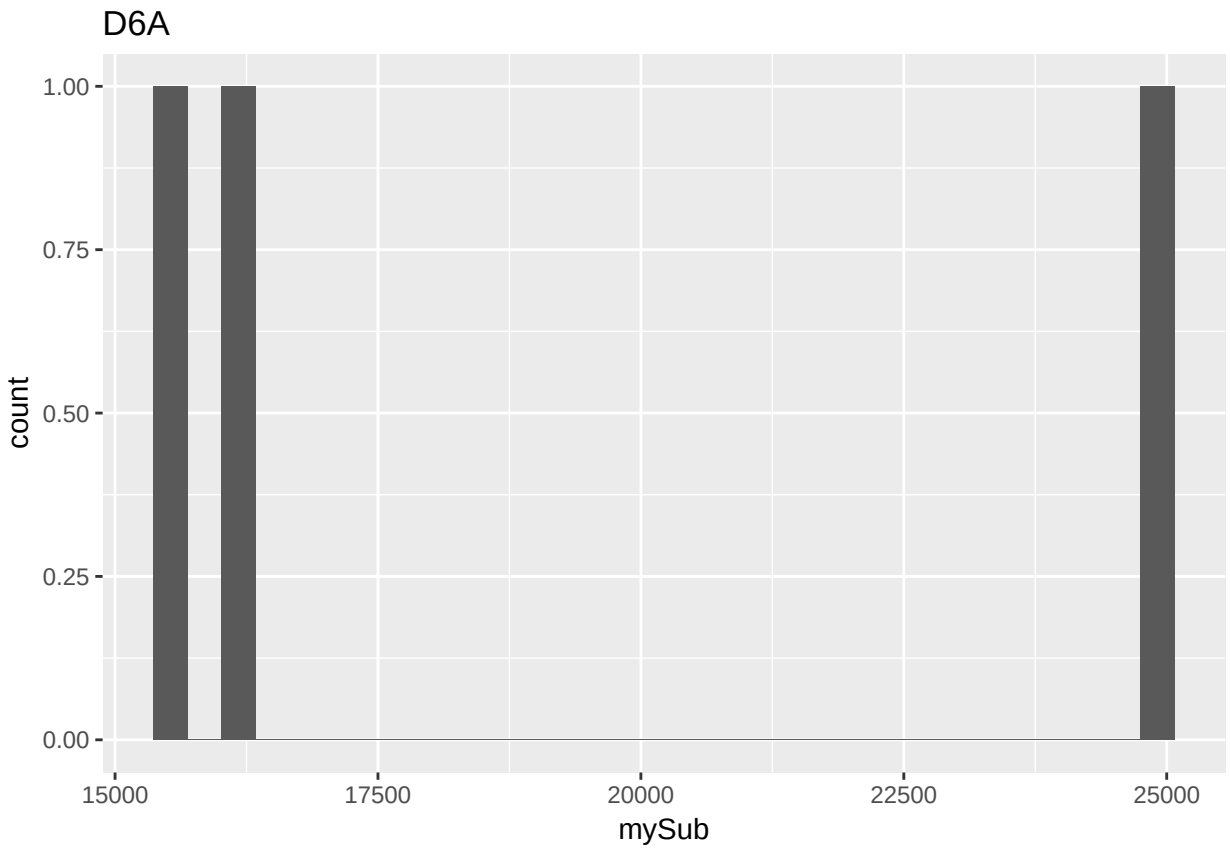

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



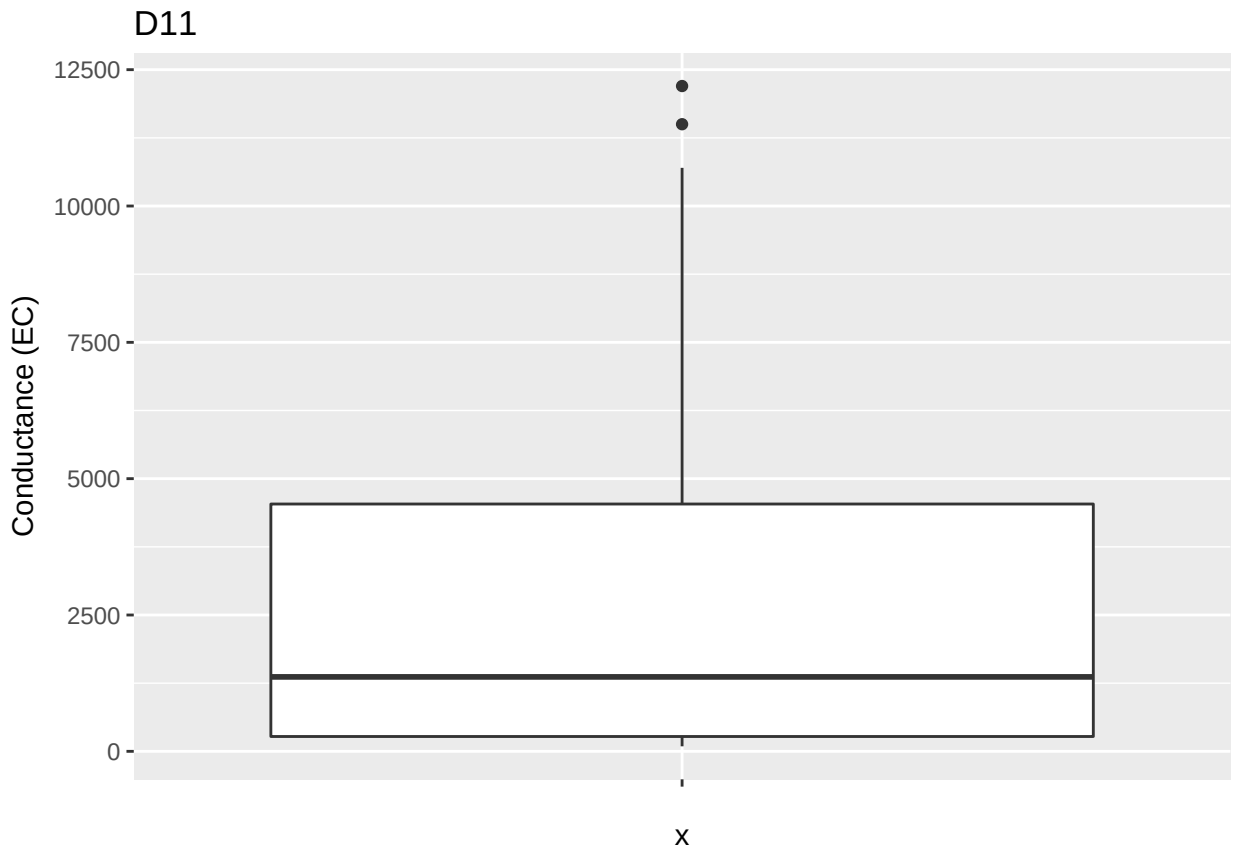

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

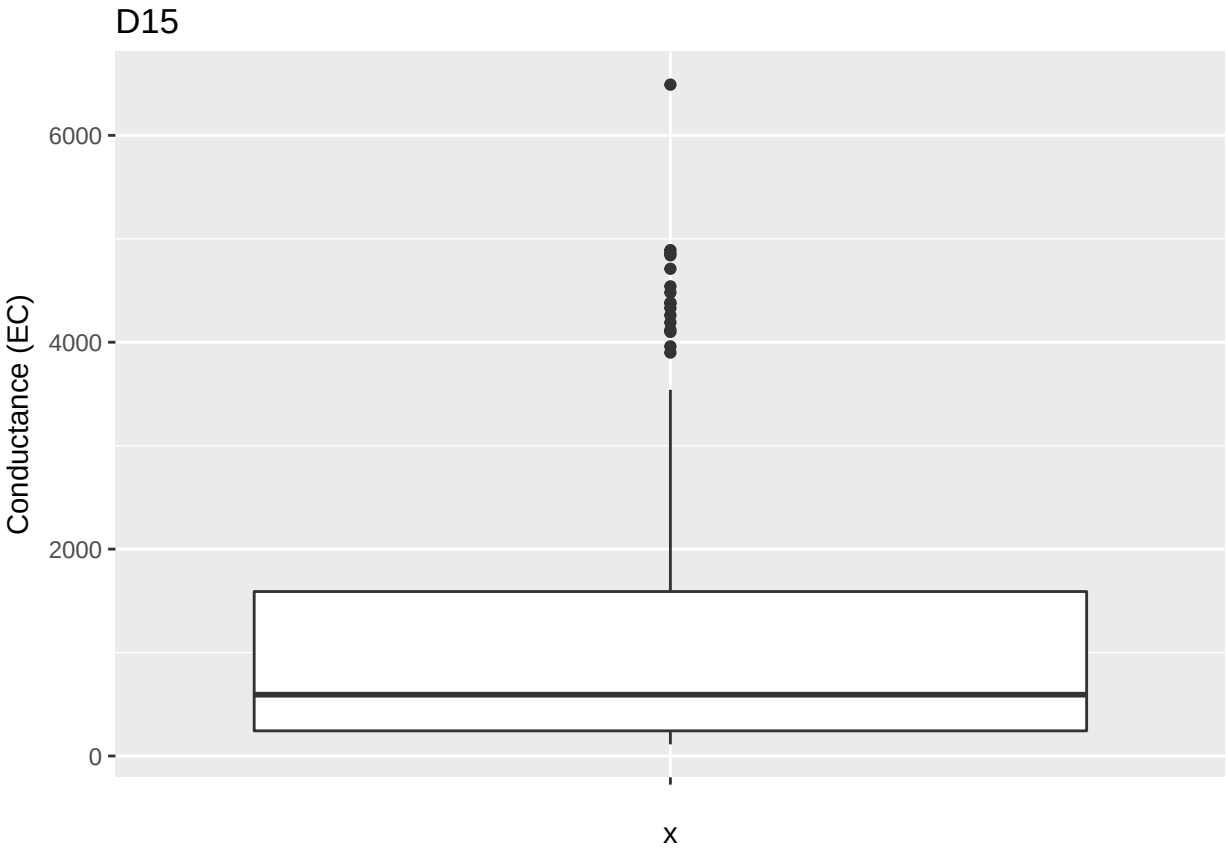


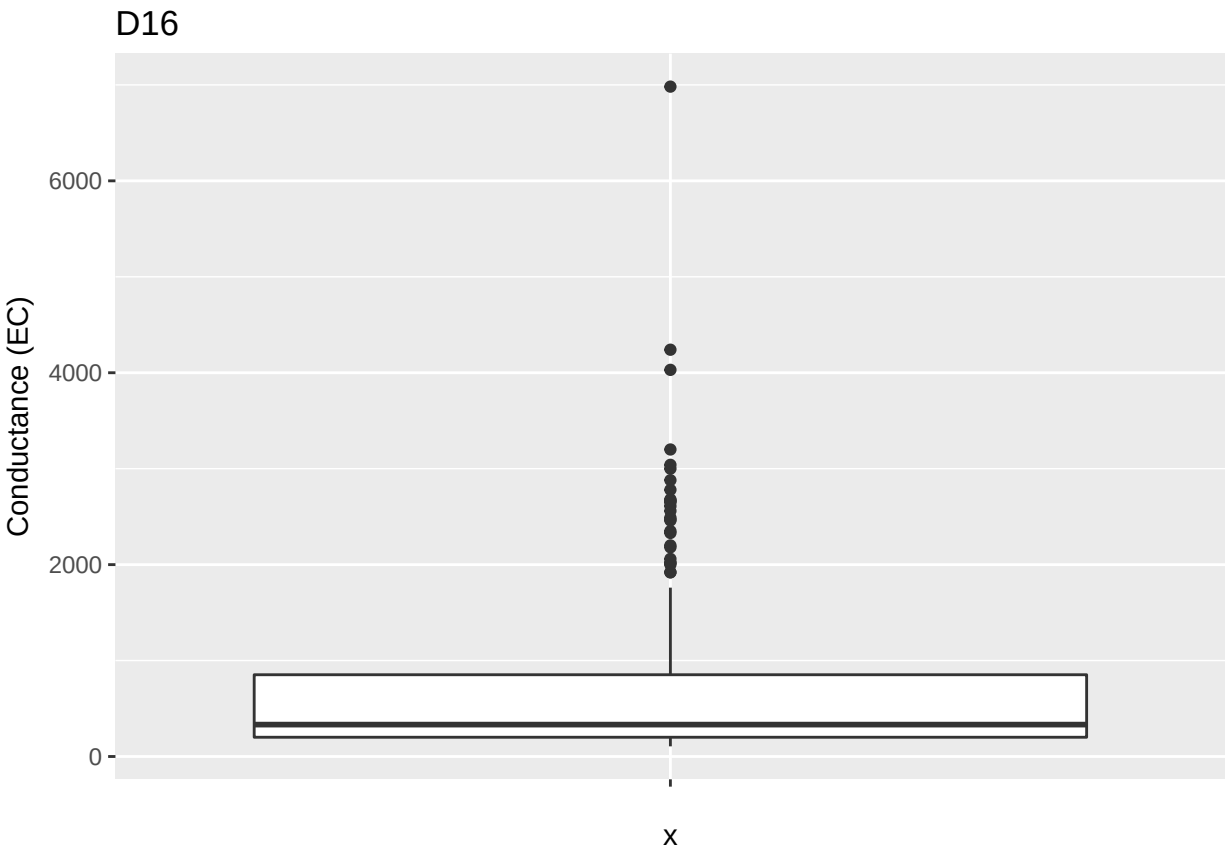

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

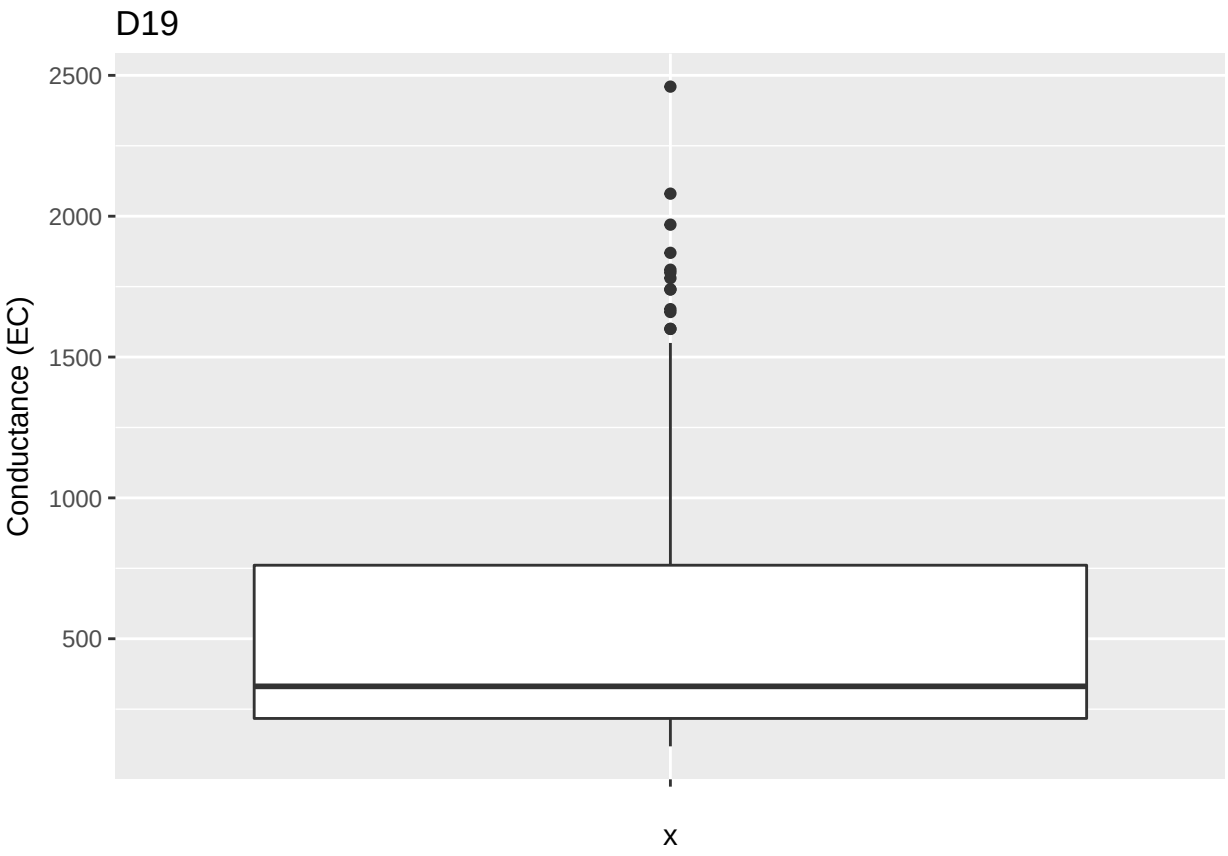


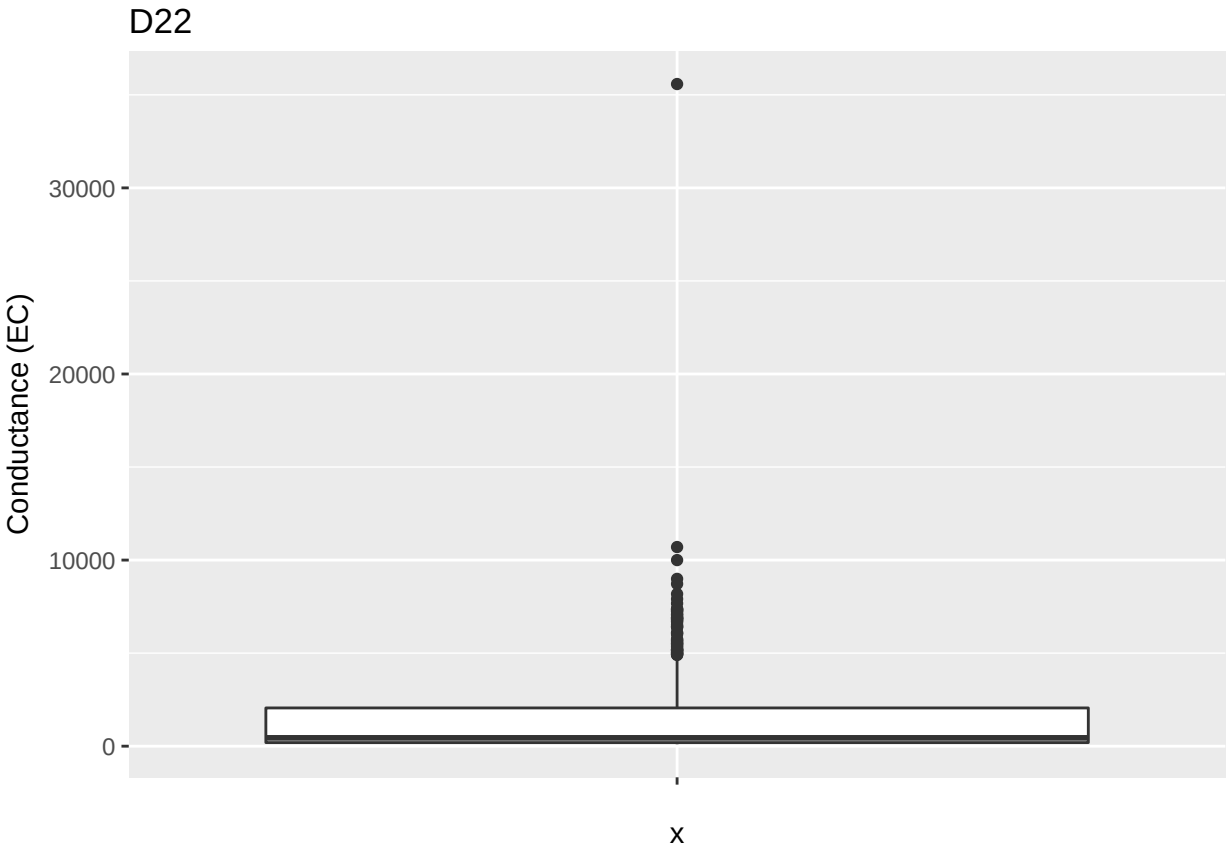

```
## boxplot

sites <- unique(wQ$StationCode)
try(for (i in 1:length(sites)){
   mySub <- subset(wQ,StationCode == sites[i] ,select = `Conductance (EC)`)
    myPlot <- ggplot(mySub, aes(x= "",y = `Conductance (EC)`)) +
    geom_boxplot(na.rm = T)
    print(myPlot + ggtitle(sites[i]))
})
```

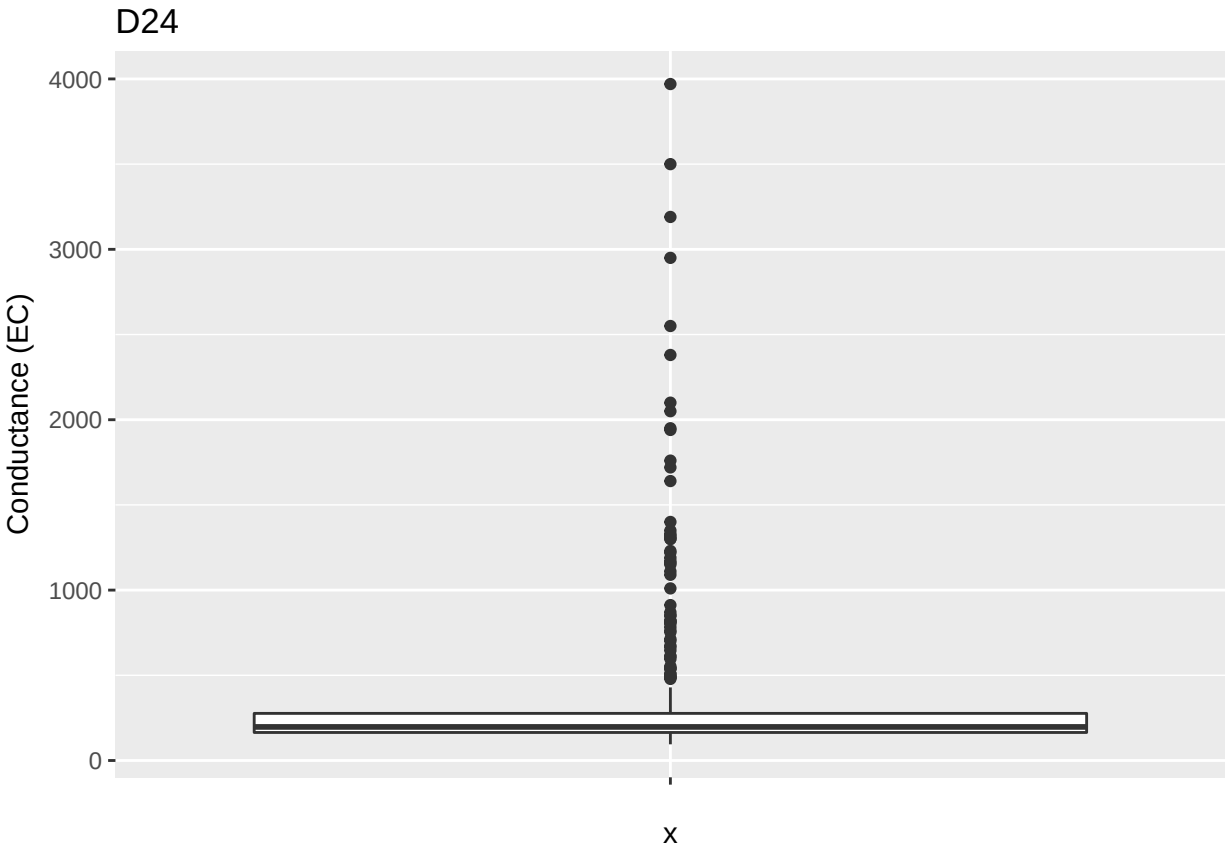

D11
12500
10000
7500
5000
2500
0
Conductance (EC)
x

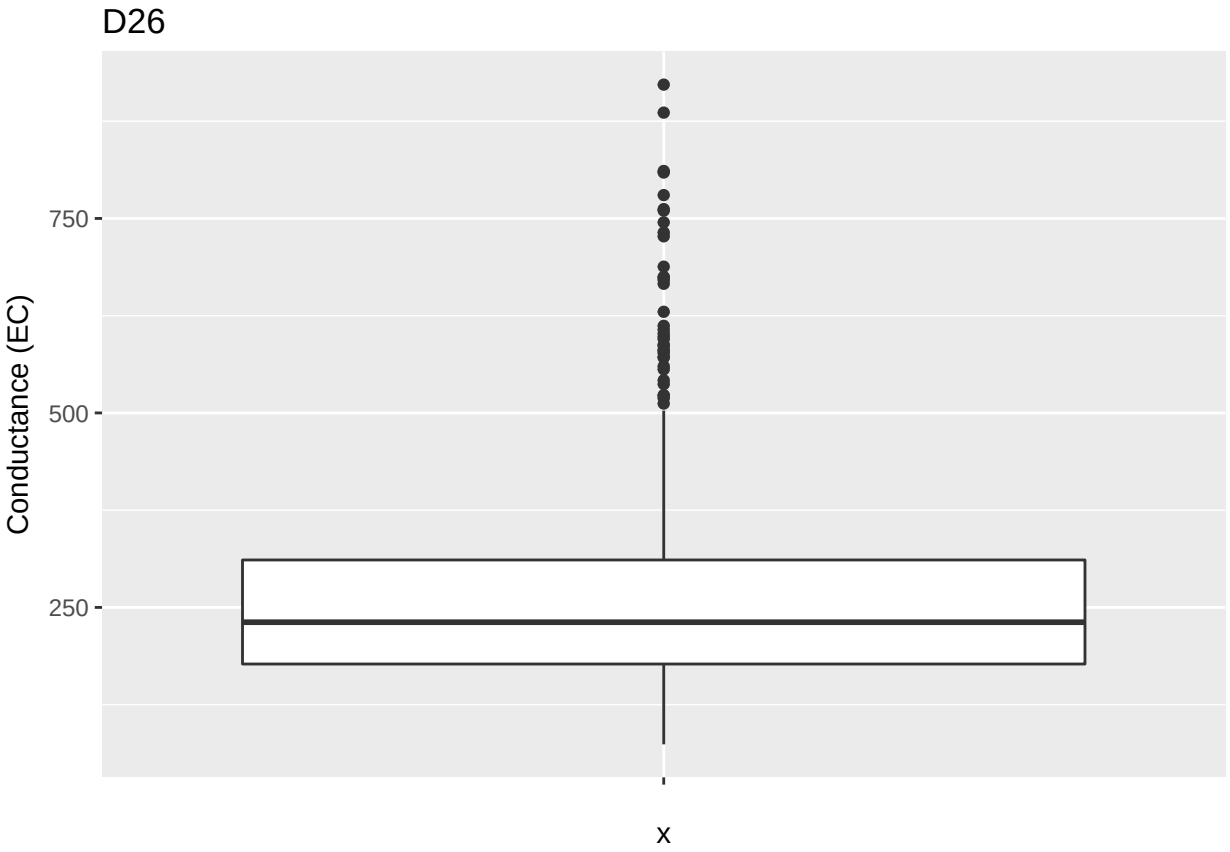

D15
Conductance (EC)
6000
4000
2000
0
x

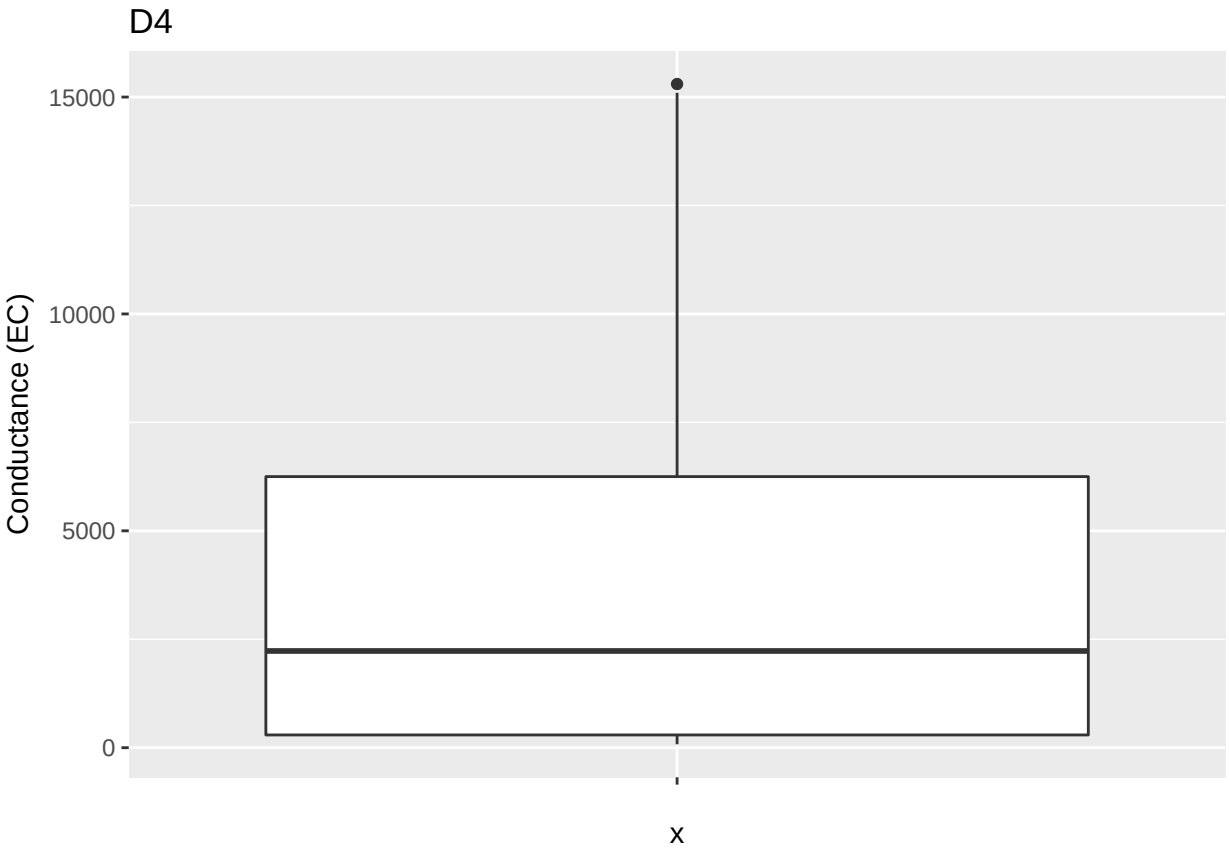

D16
Conductance (EC)
6000
4000
2000
0
x

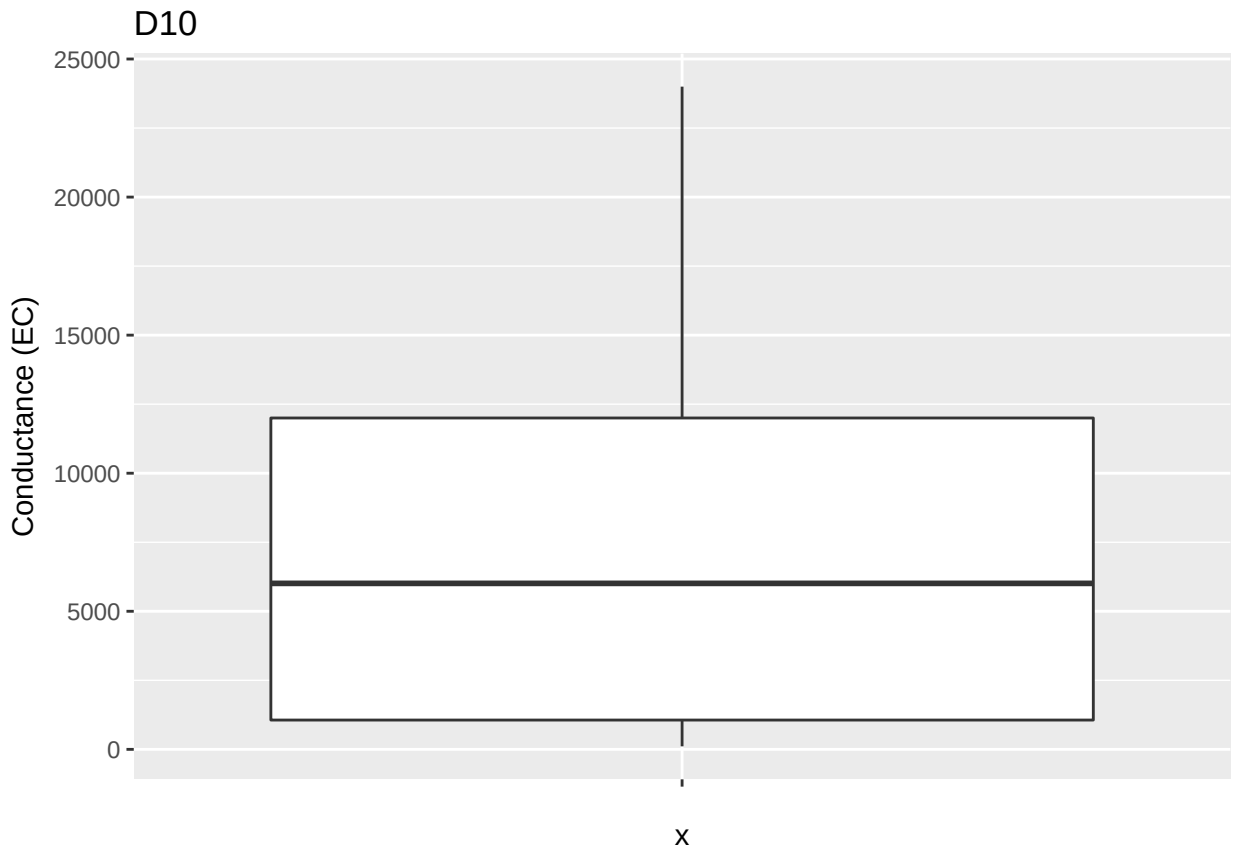

D19
2500
2000
1500
1000
500
Conductance (EC)
x

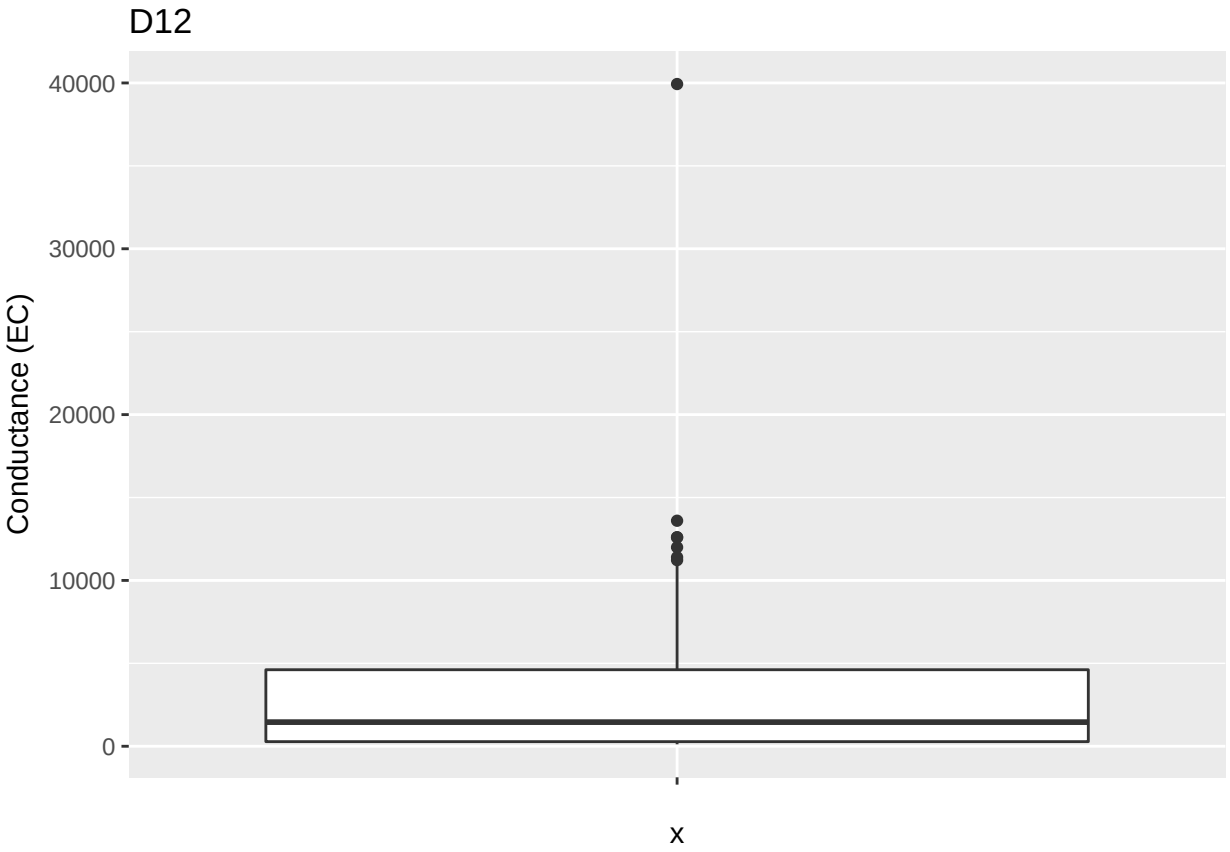

D22
Conductance (EC)
30000
20000
10000
0
x

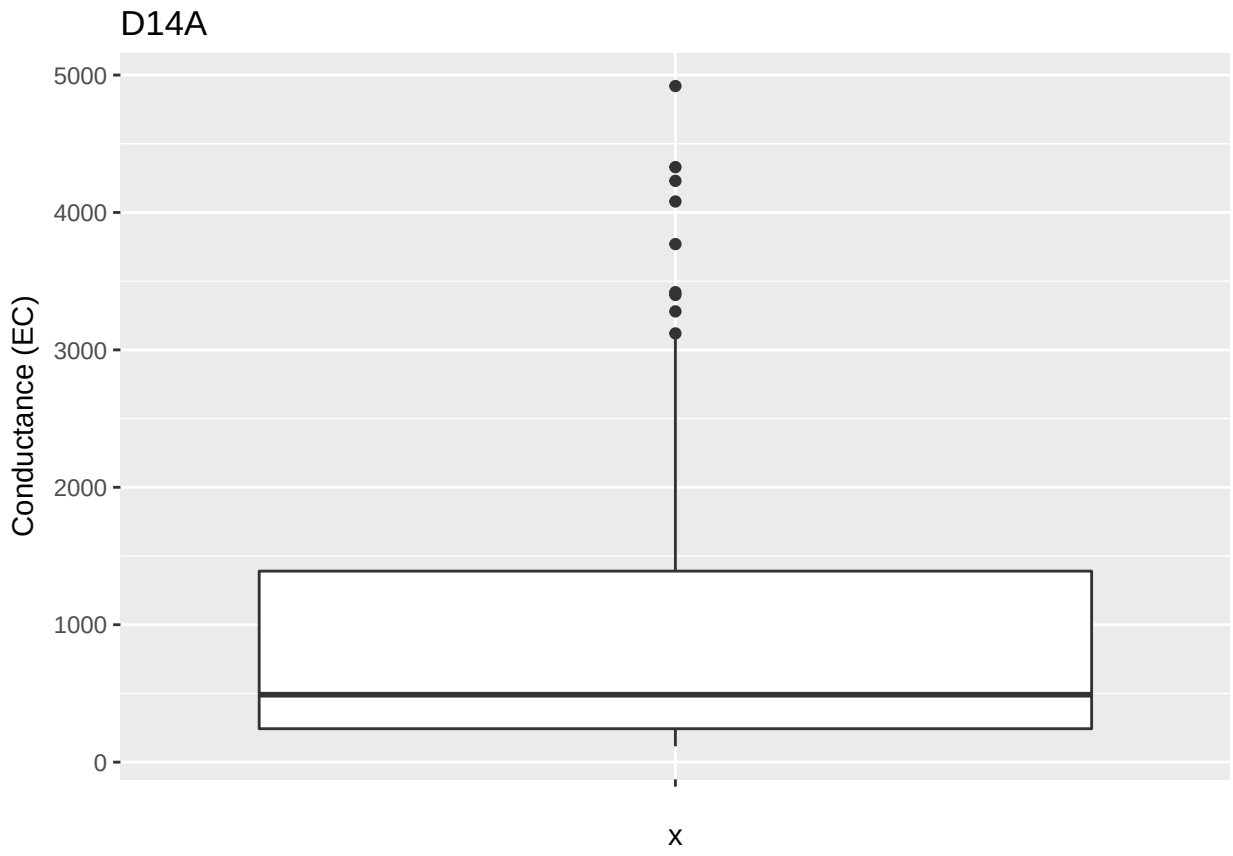

D24
4000
3000
2000
1000
0
Conductance (EC)
x

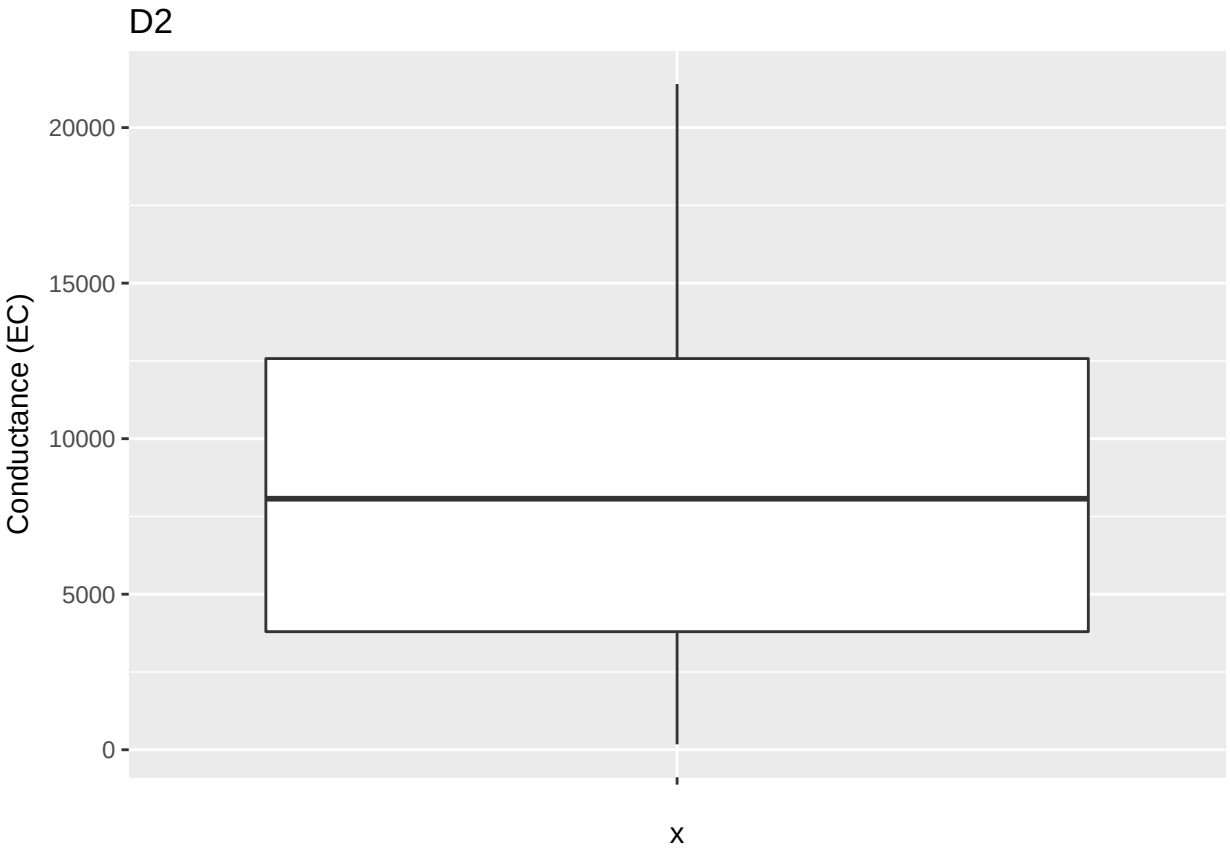

D26
Conductance (EC)
750
500
250
x

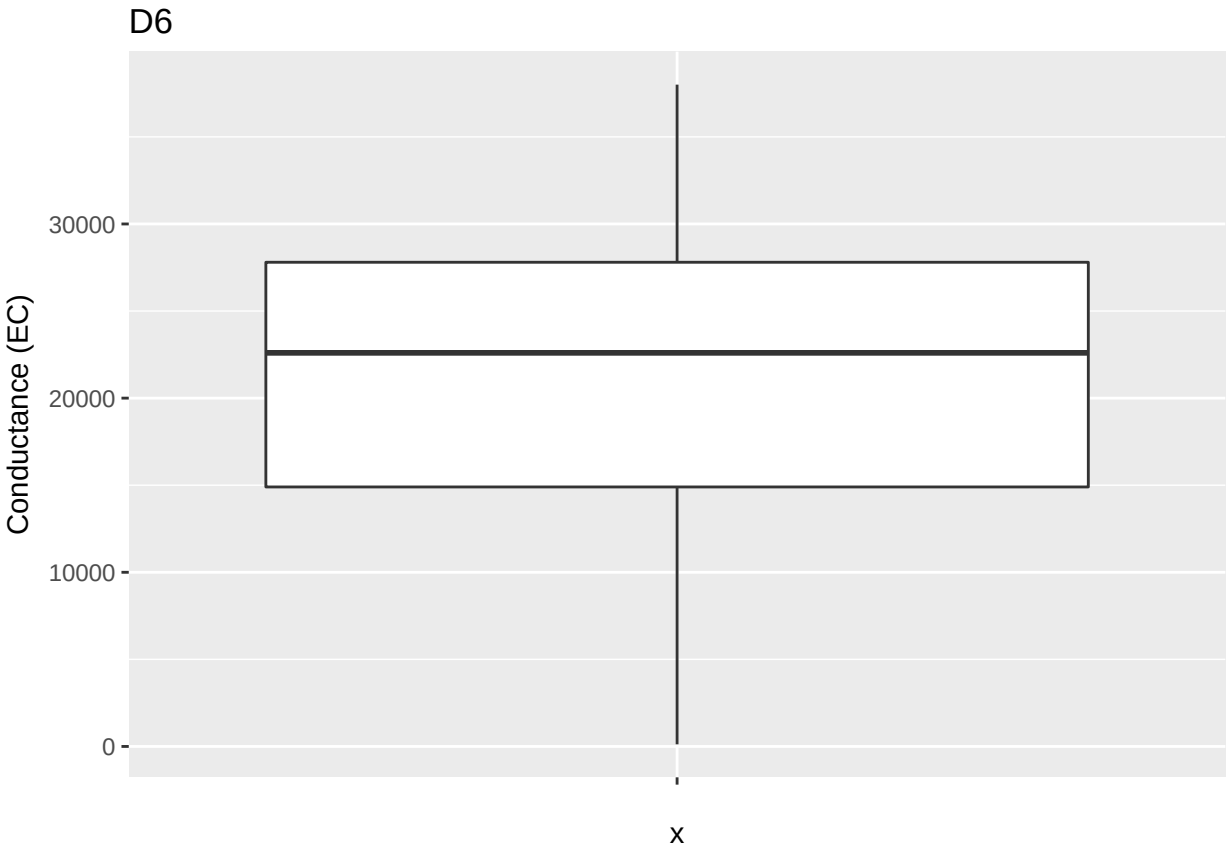

D4
15000
10000
5000
0
Conductance (EC)
x

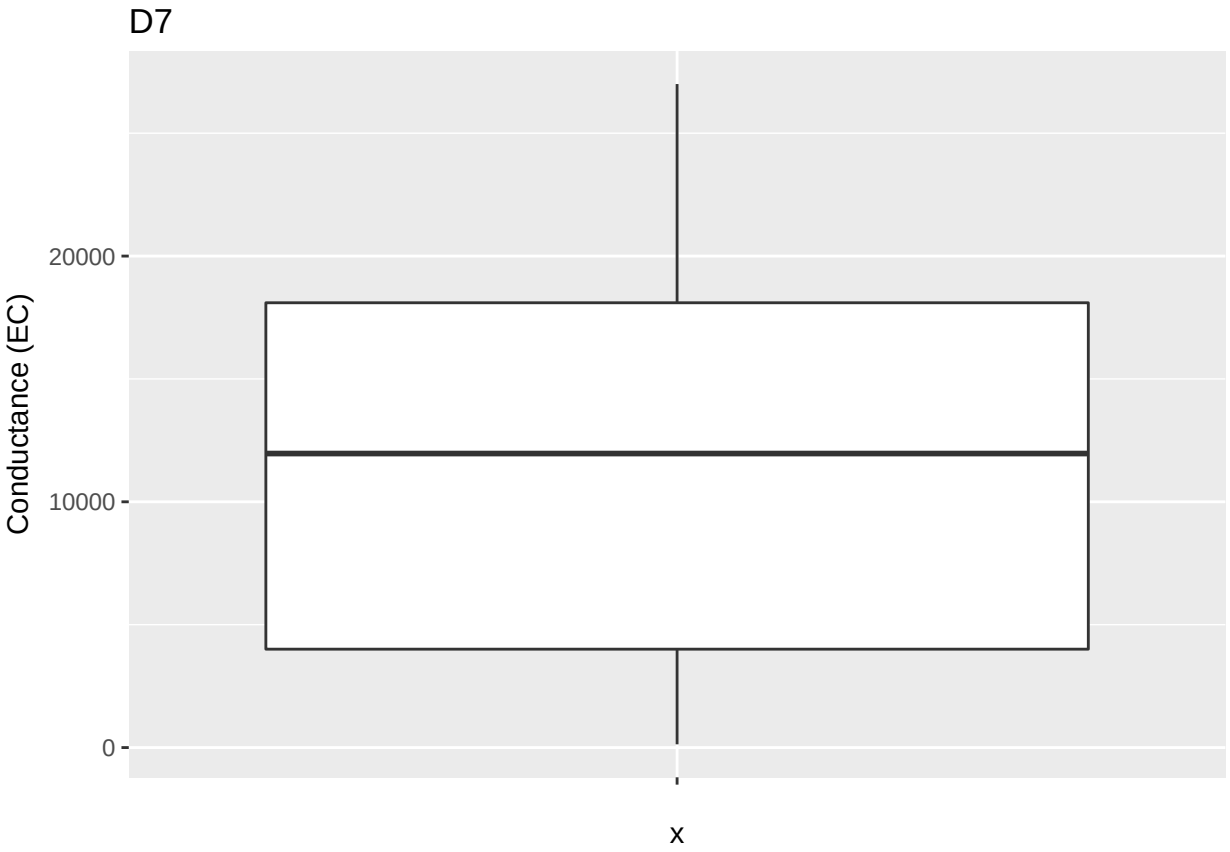

D10
25000
20000
15000
10000
5000
0
Conductance (EC)
x

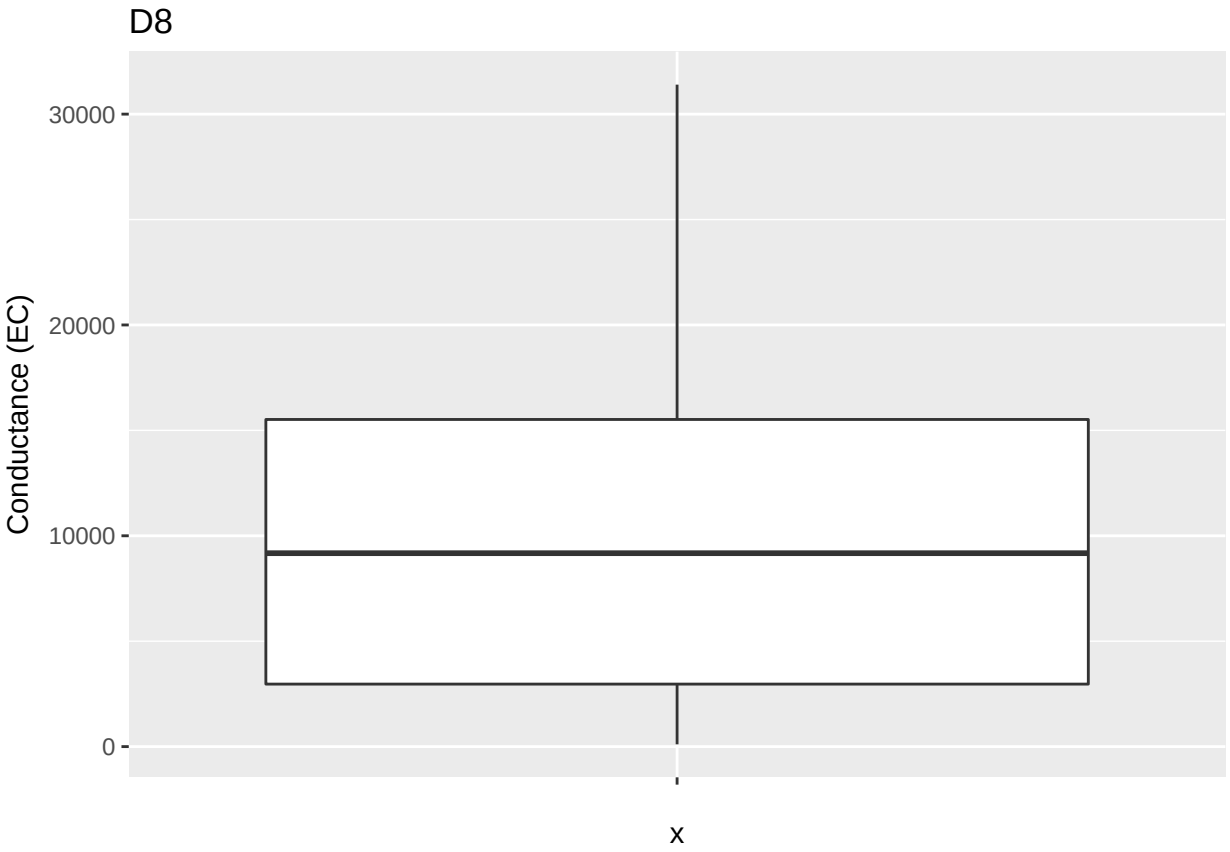

D12
40000
30000
20000
10000
0
Conductance (EC)
x

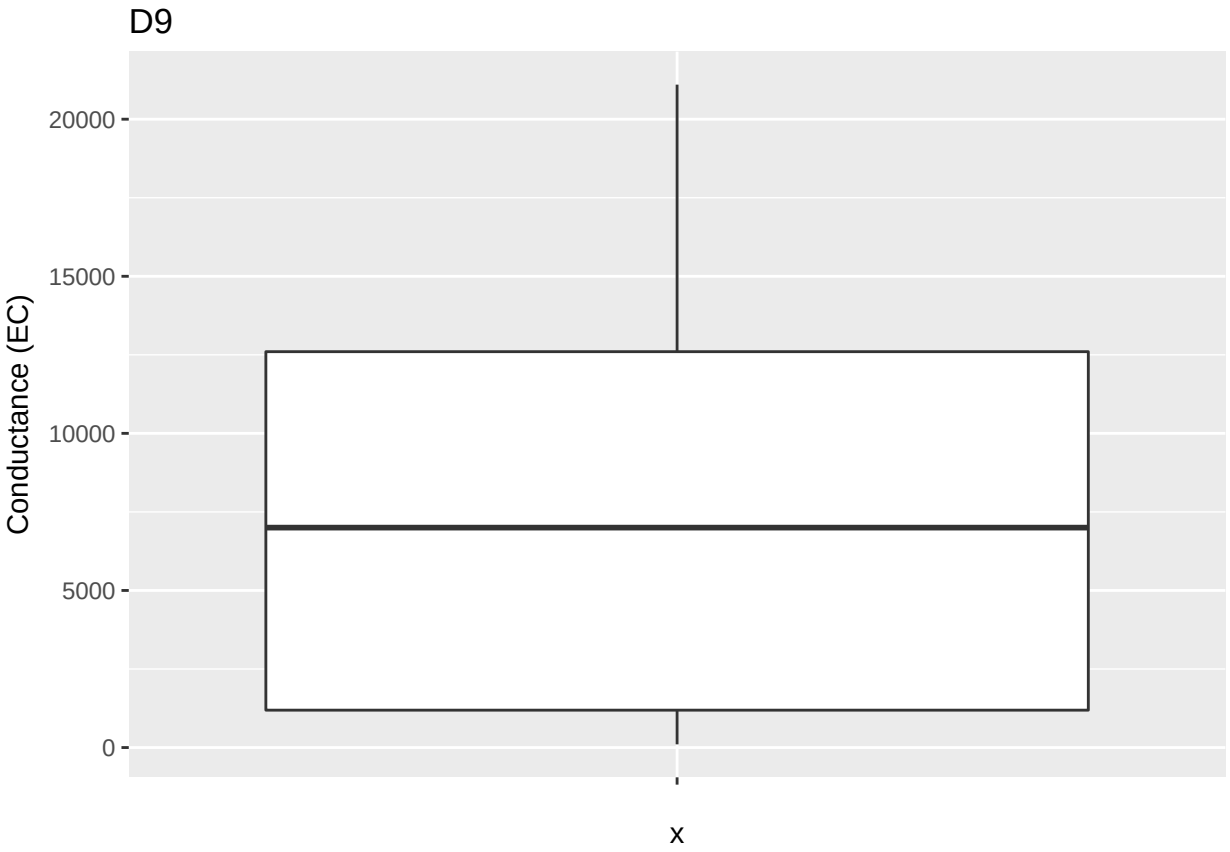

D14A
5000
4000
3000
2000
1000
0
Conductance (EC)
x

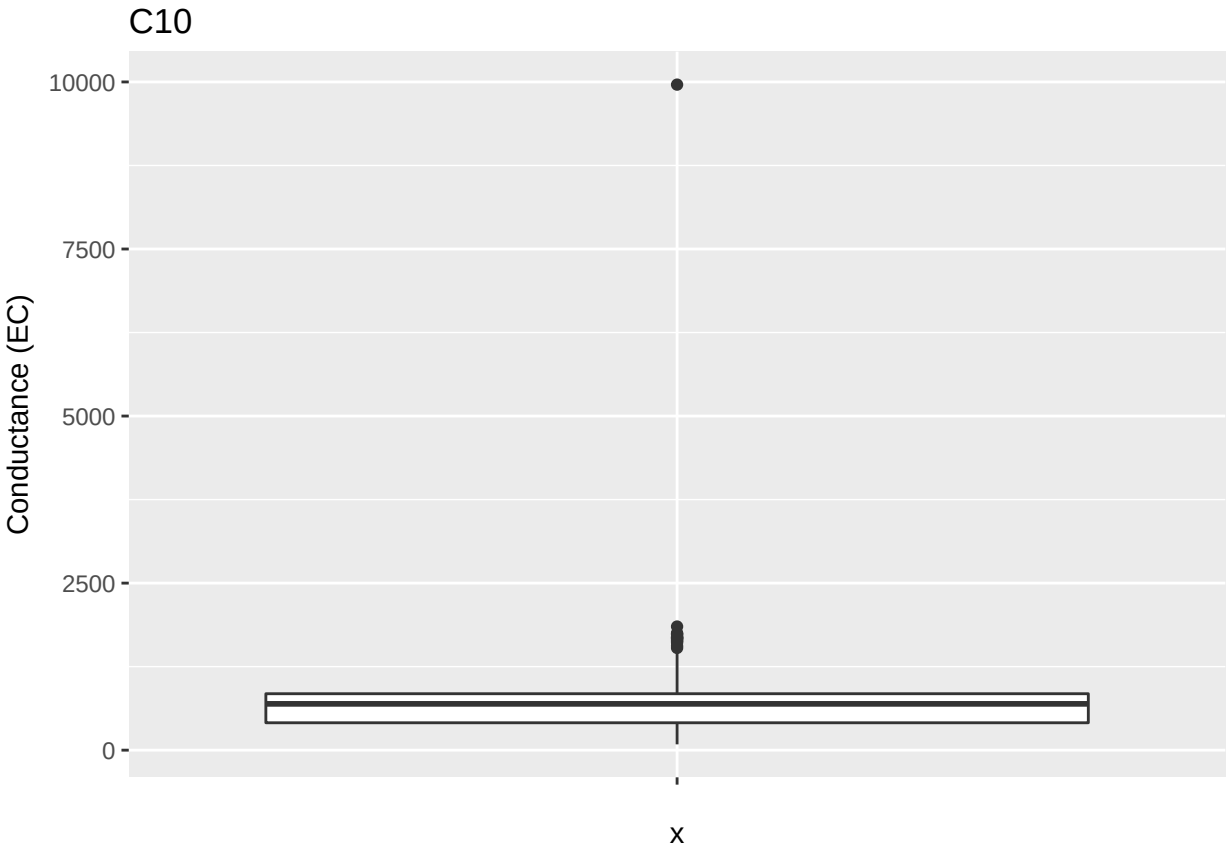

D2
Conductance (EC)
20000
15000
10000
5000
0
x

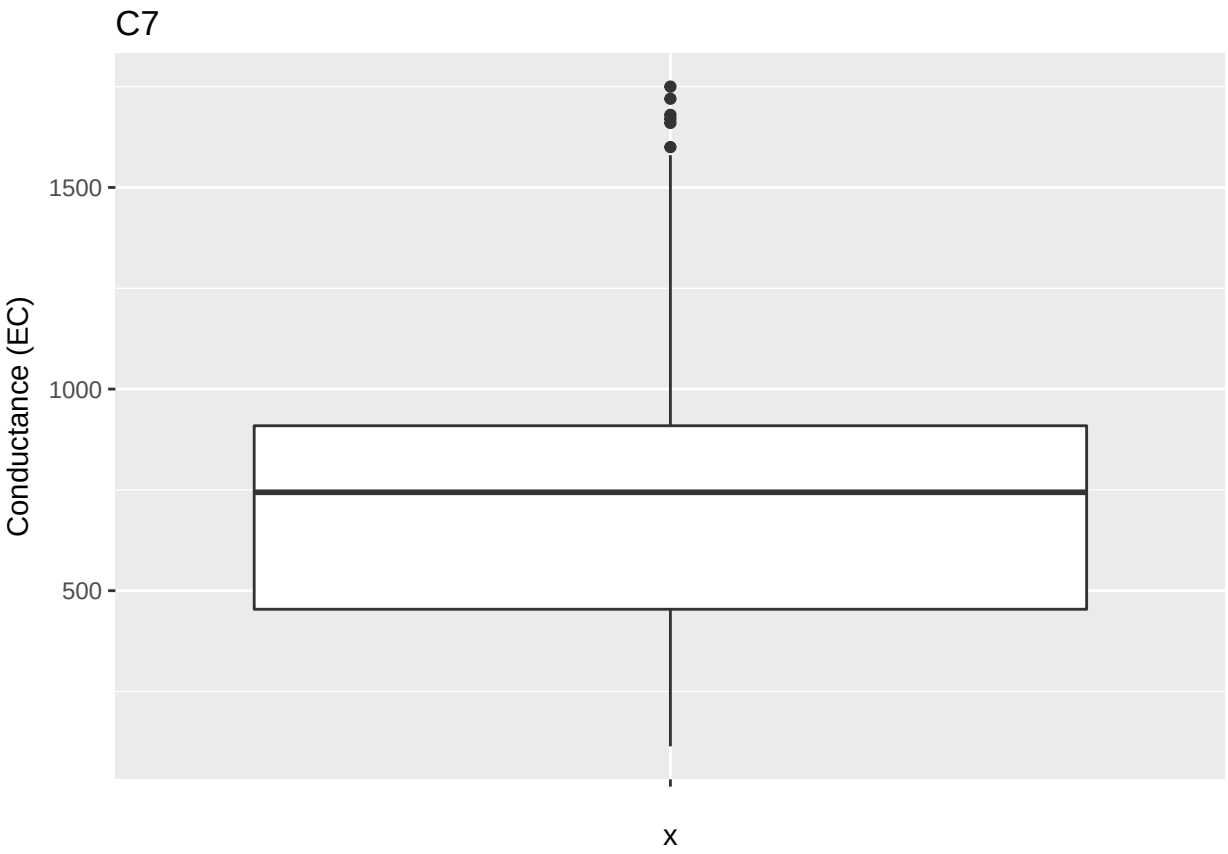

D6
Conductance (EC)
30000
20000
10000
0
x

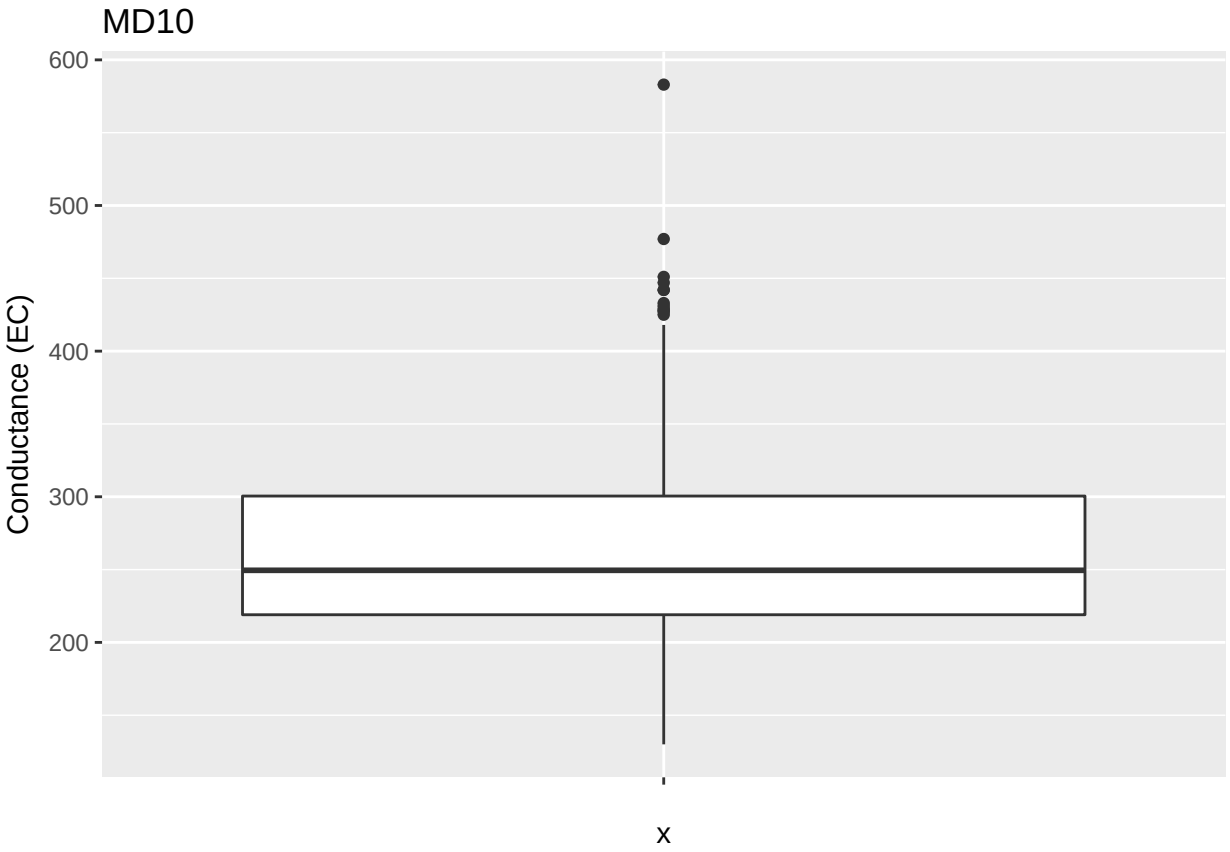

D7
Conductance (EC)
20000
10000
0
x

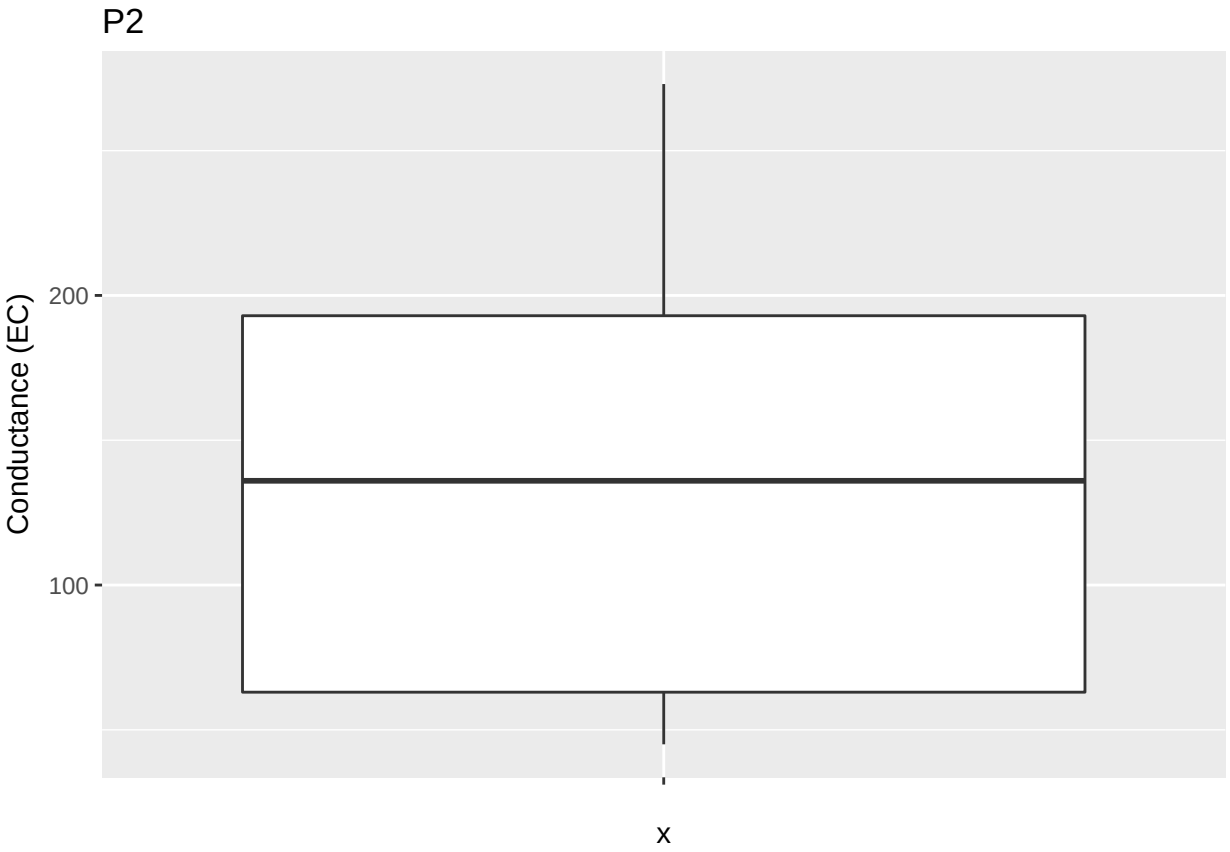

D8
30000
20000
10000
0
Conductance (EC)
x

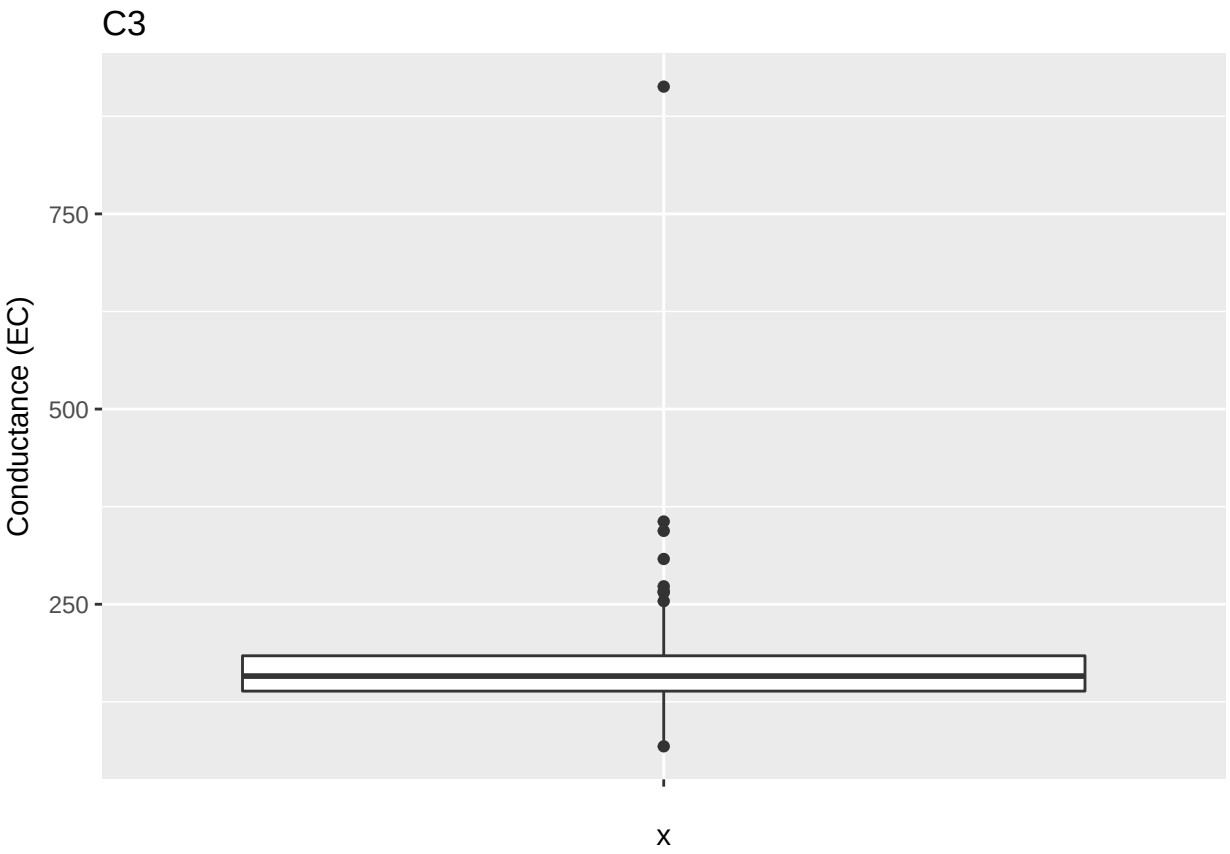

D9
Conductance (EC)
20000
15000
10000
5000
0
x

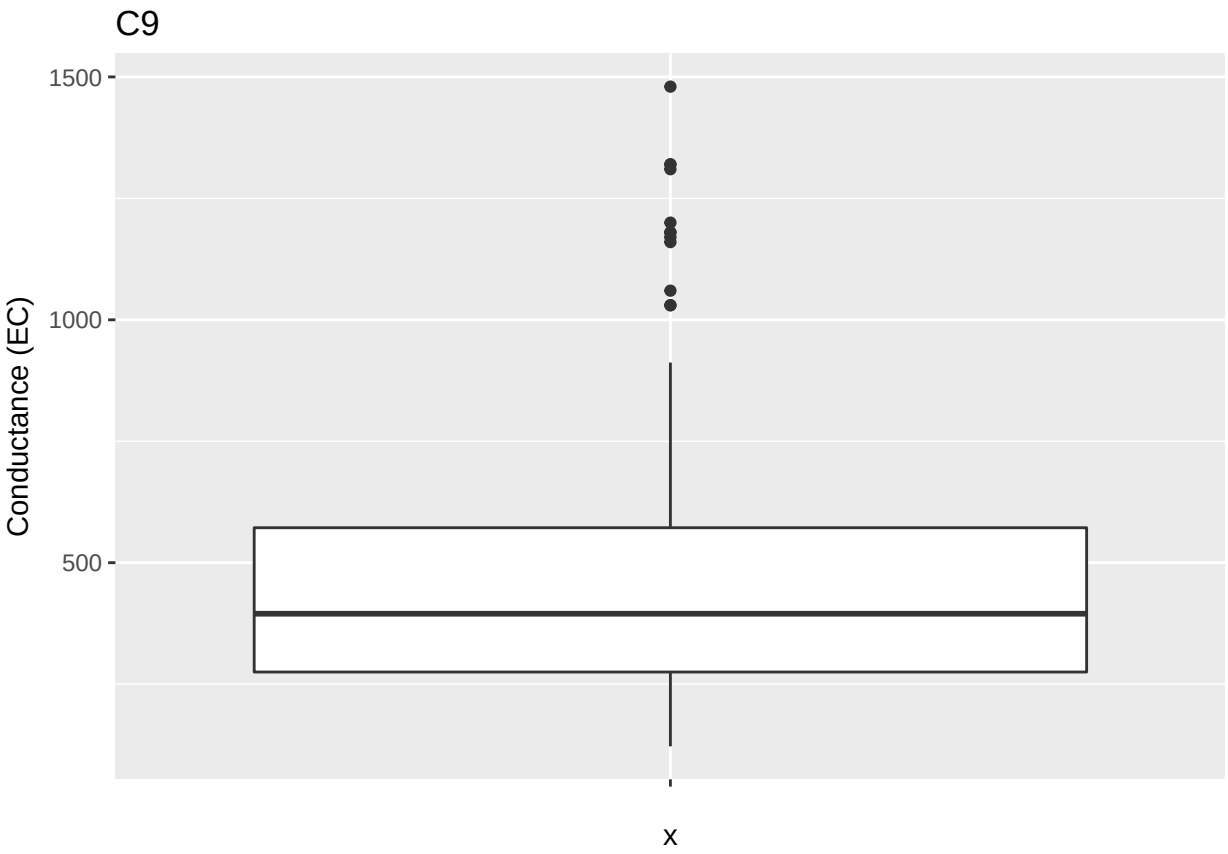

C10
10000
7500
5000
2500
0
Conductance (EC)
x

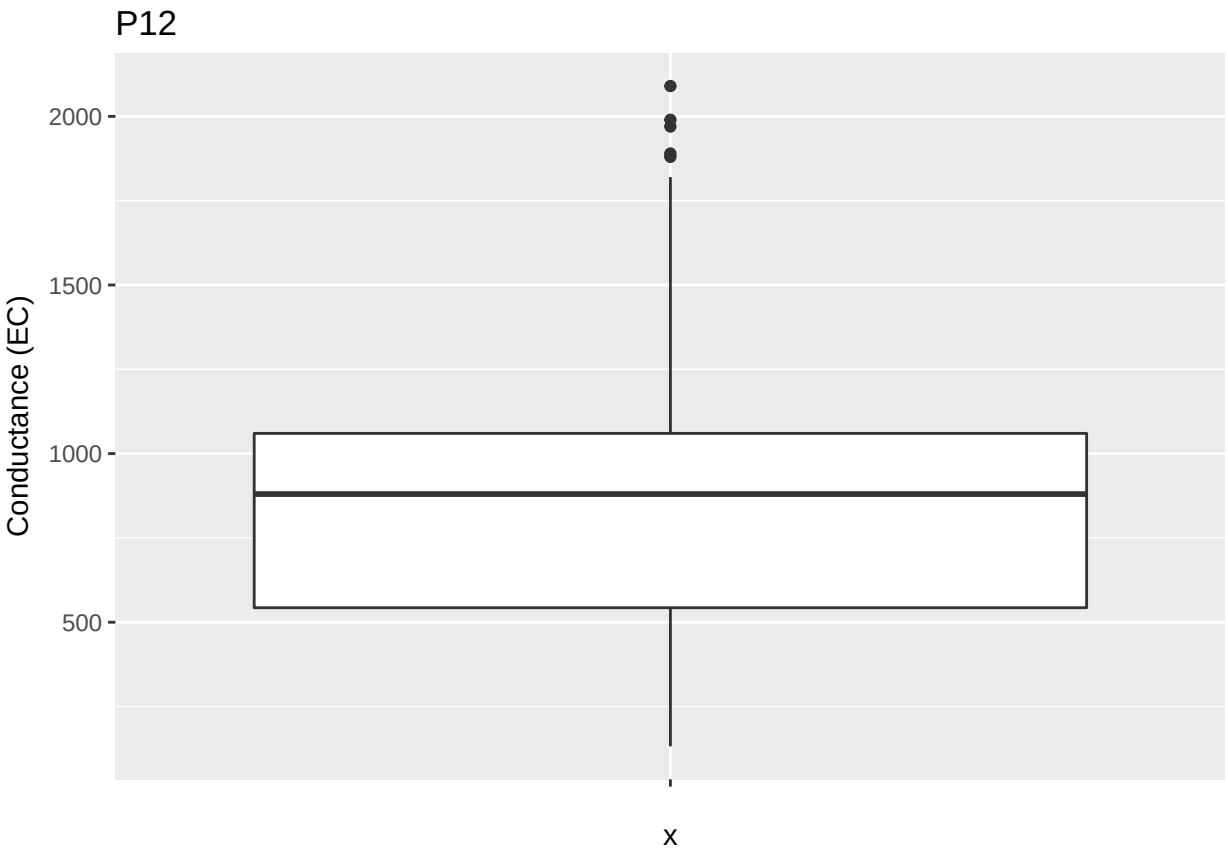

C7
Conductance (EC)
1500
1000
500
x

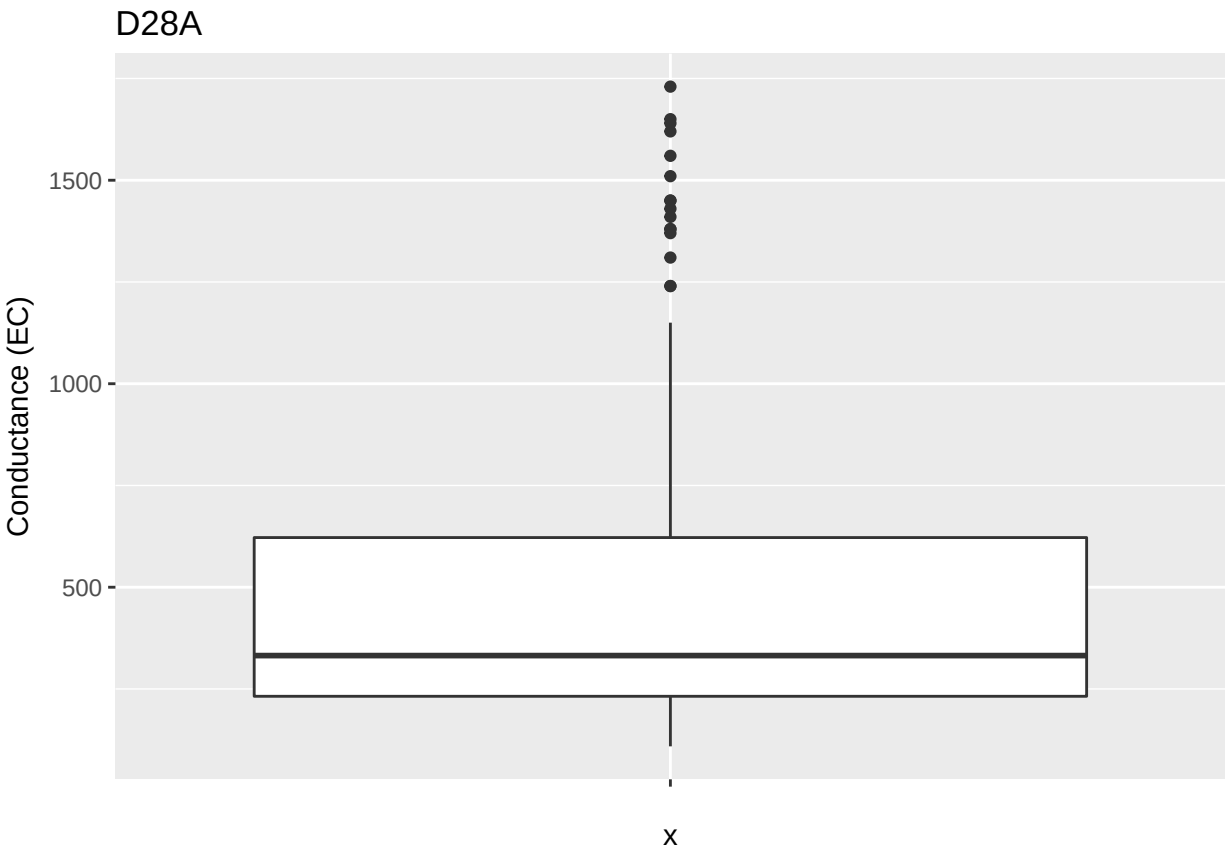

MD10
600
500
400
300
200
Conductance (EC)
x

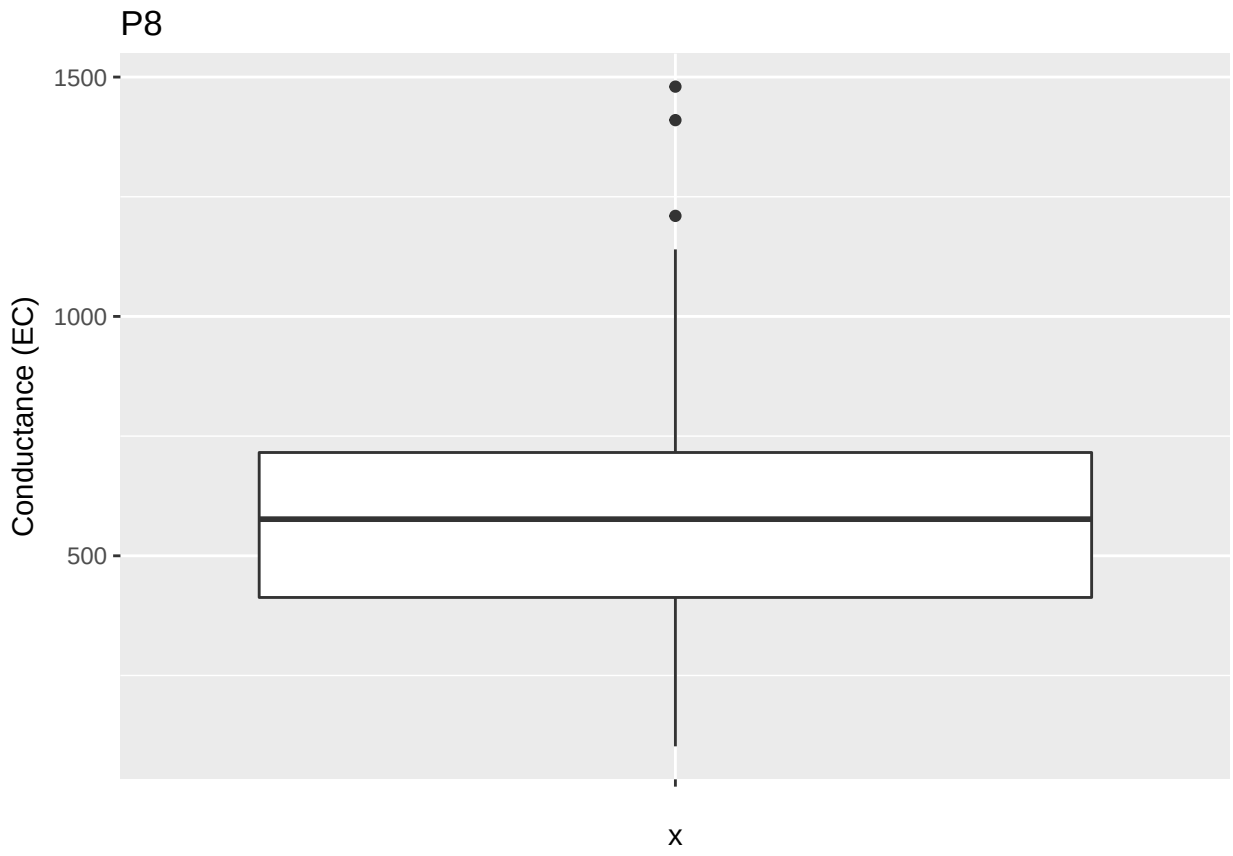

P2
Conductance (EC)
200
100
x

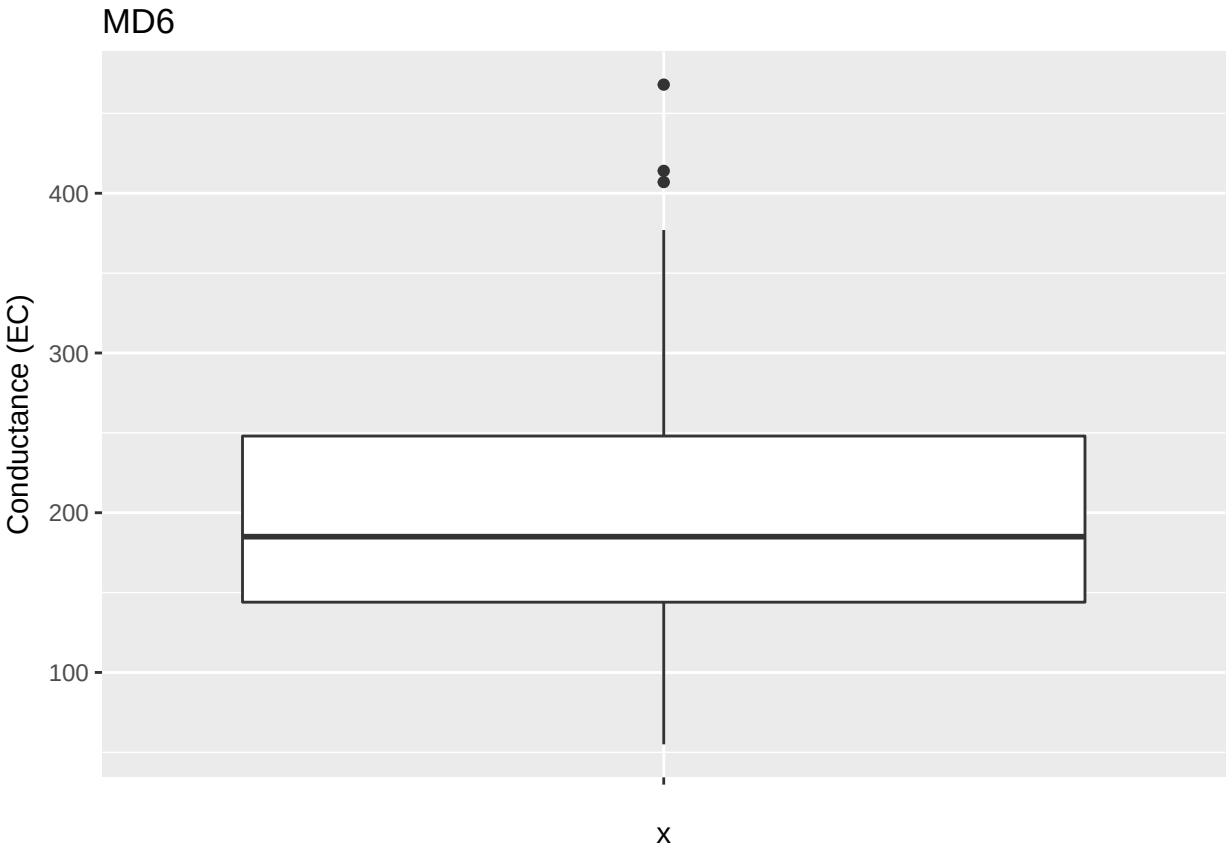

C3
Conductance (EC)
750
500
250
x

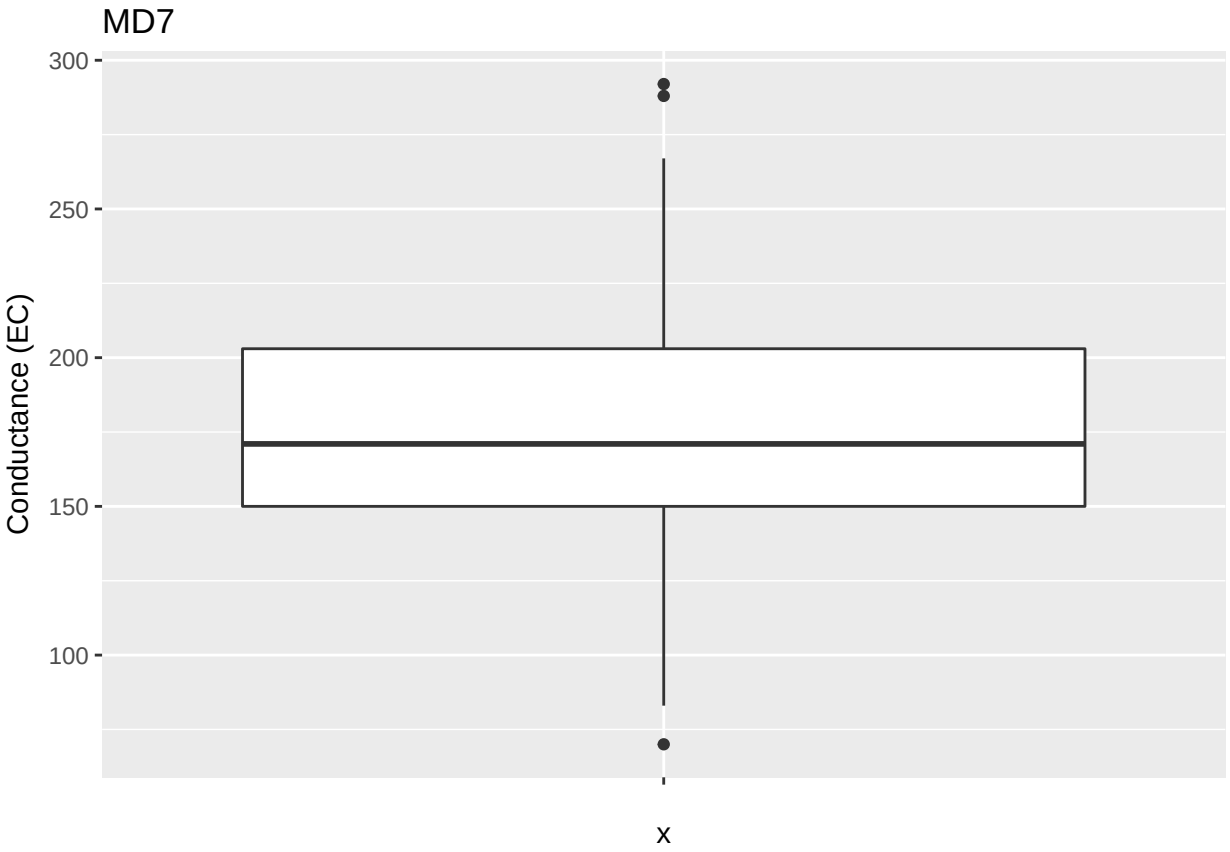

C9
1500
1000
500
Conductance (EC)
x

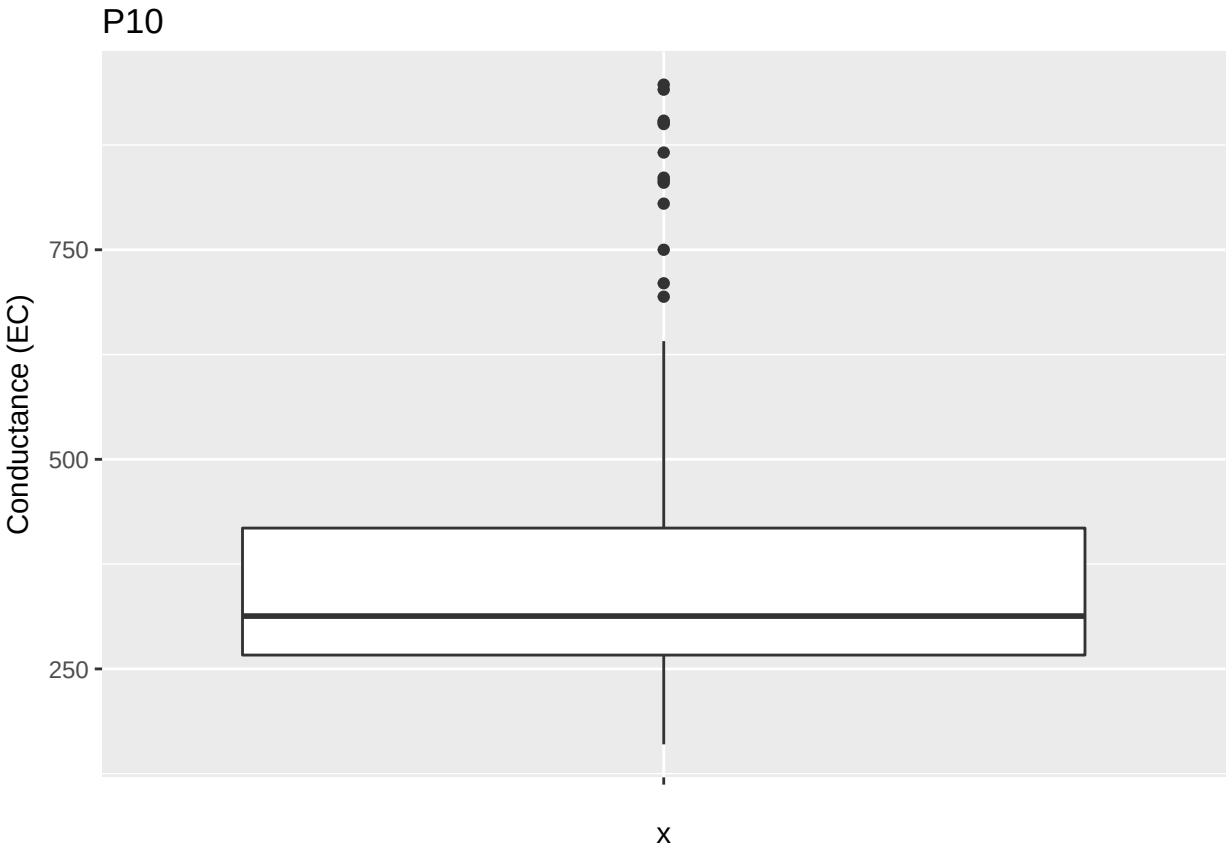

P12
2000
1500
1000
500
Conductance (EC)
x

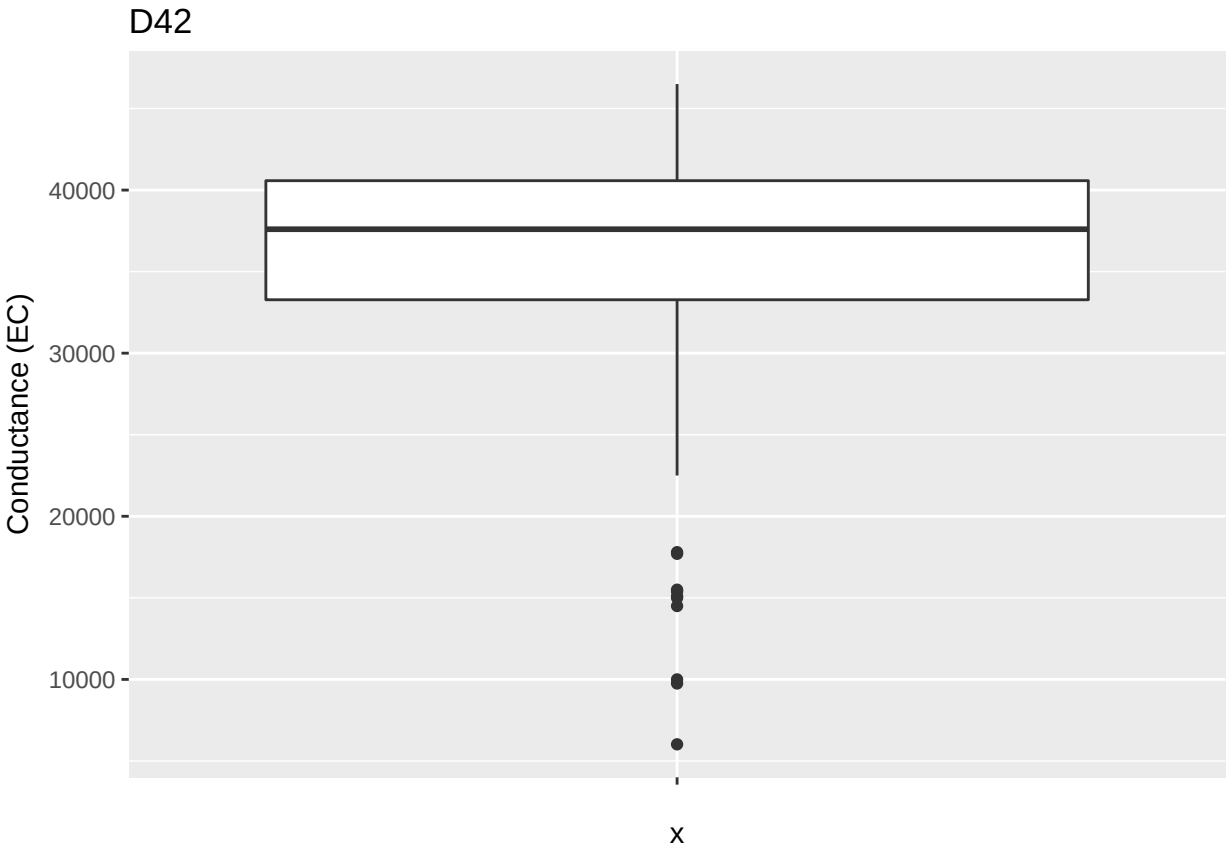

D28A
Conductance (EC)
1500
1000
500
x

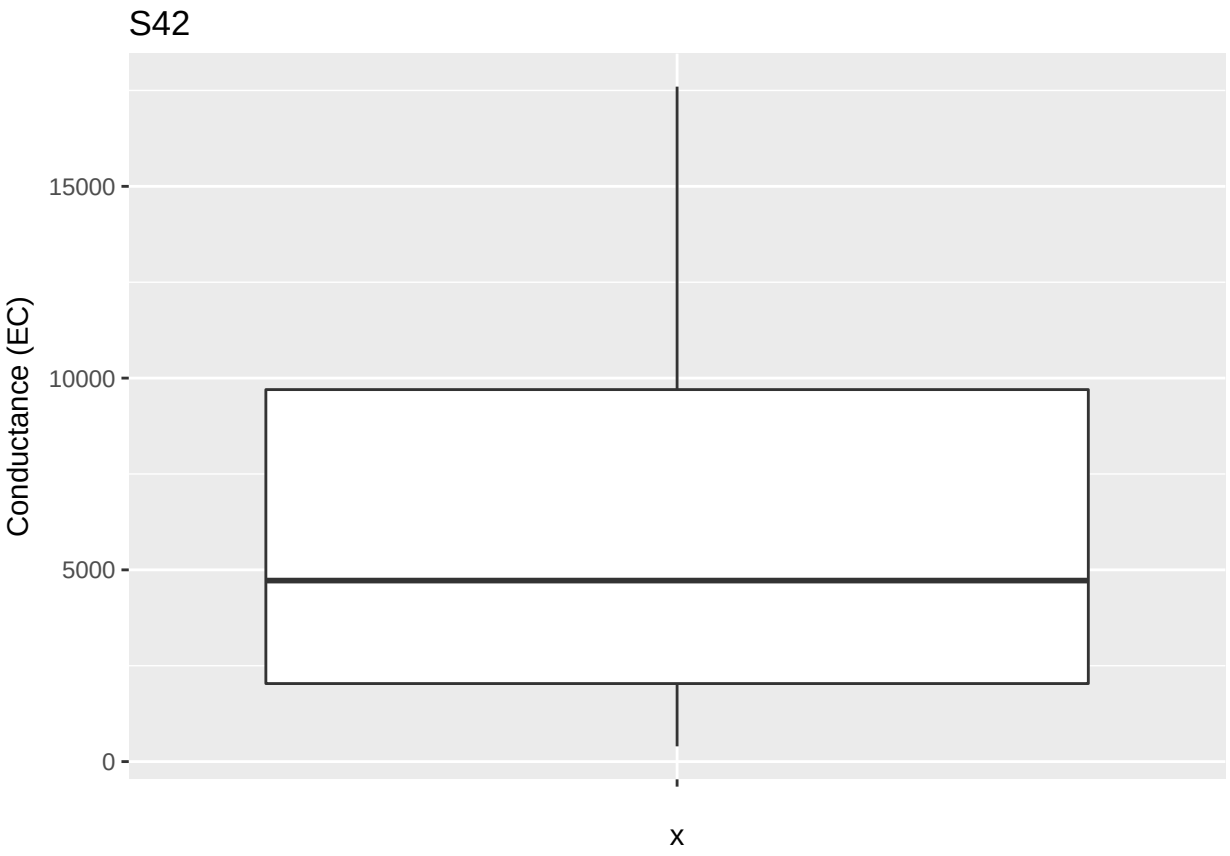

P8
1500
1000
500
Conductance (EC)
x

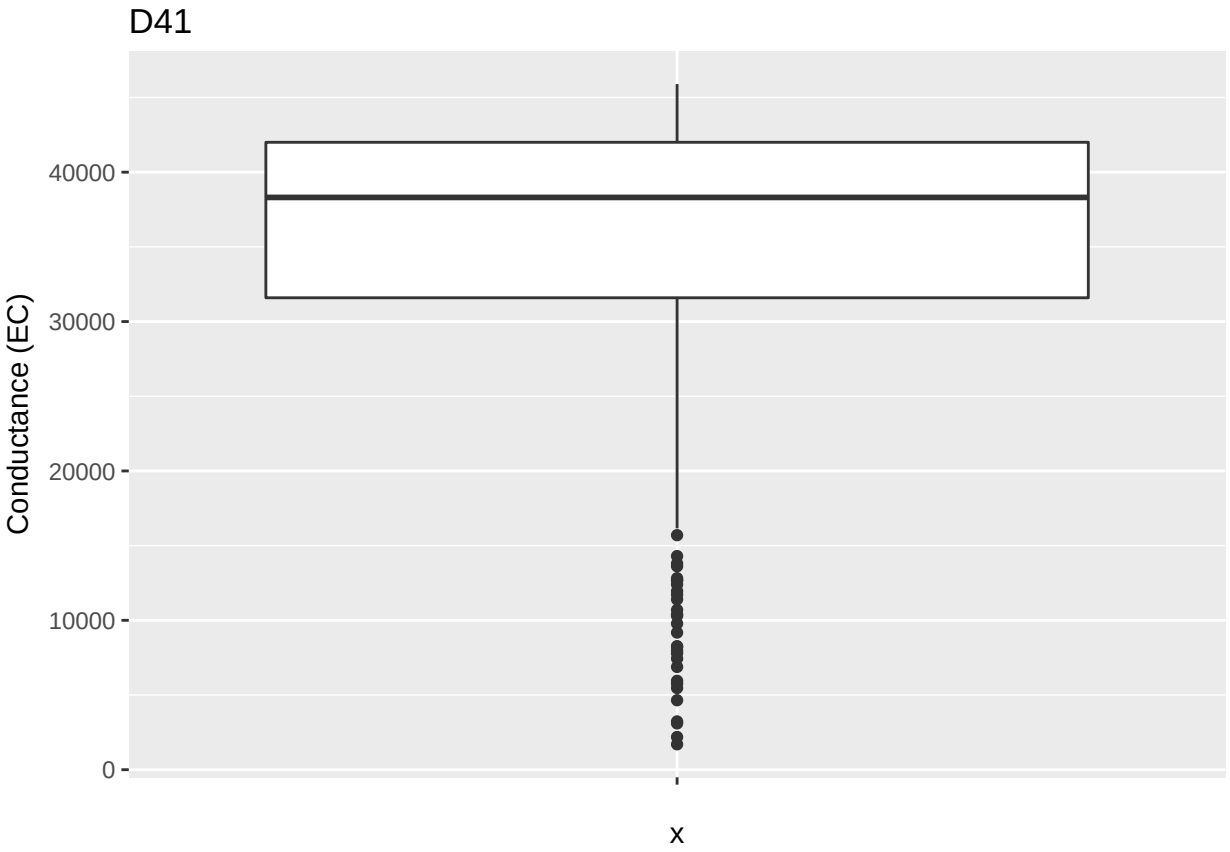

MD6
Conductance (EC)
400
300
200
100
x

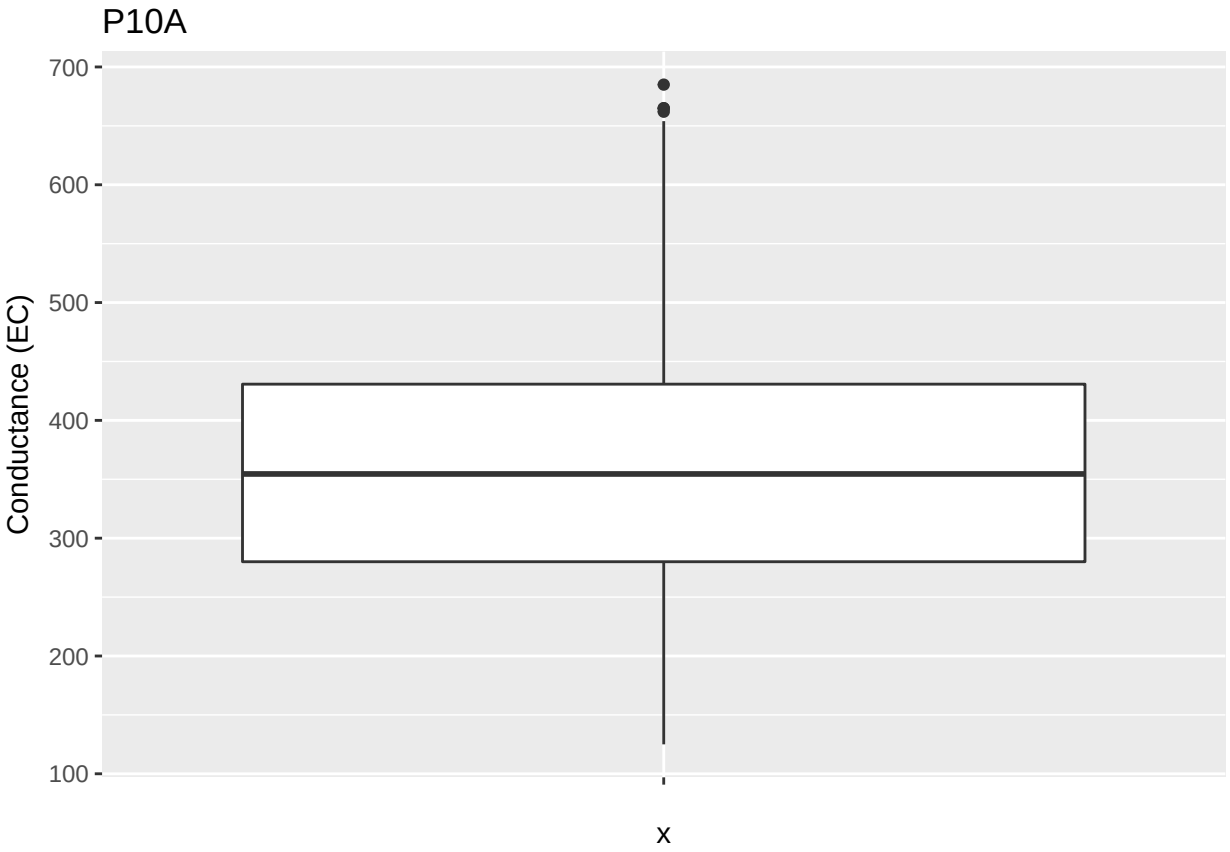

MD7
300
250
200
150
100
Conductance (EC)
x

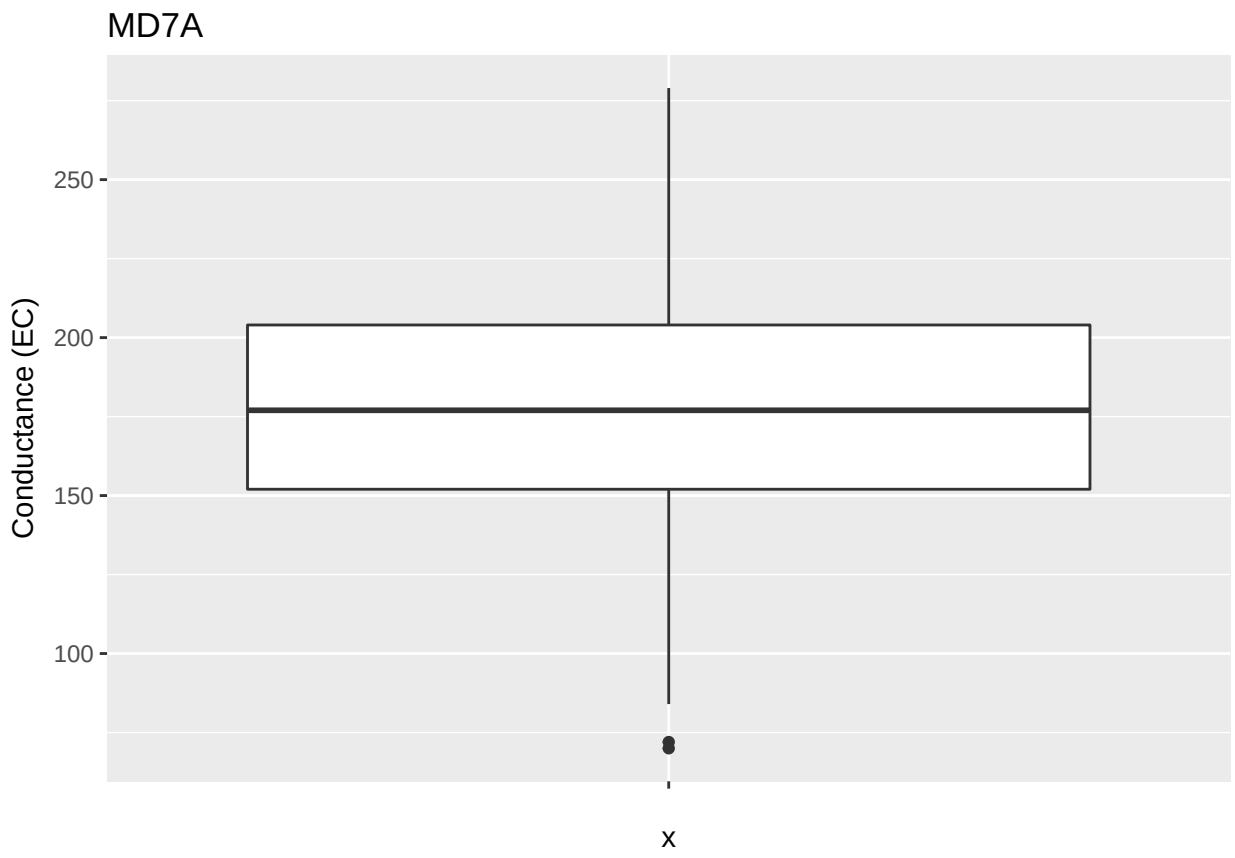

P10
Conductance (EC)
750
500
250
x

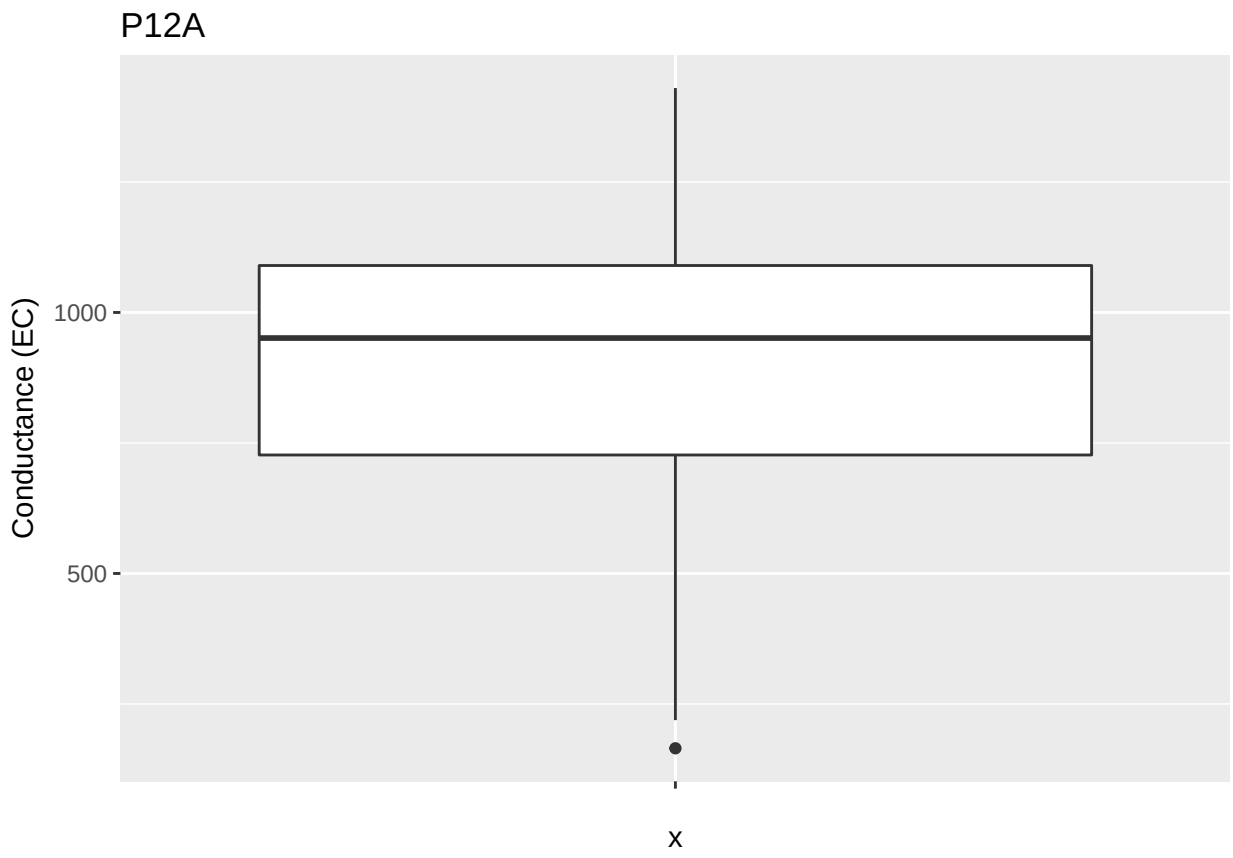

D42
Conductance (EC)
40000
30000
20000
10000
x

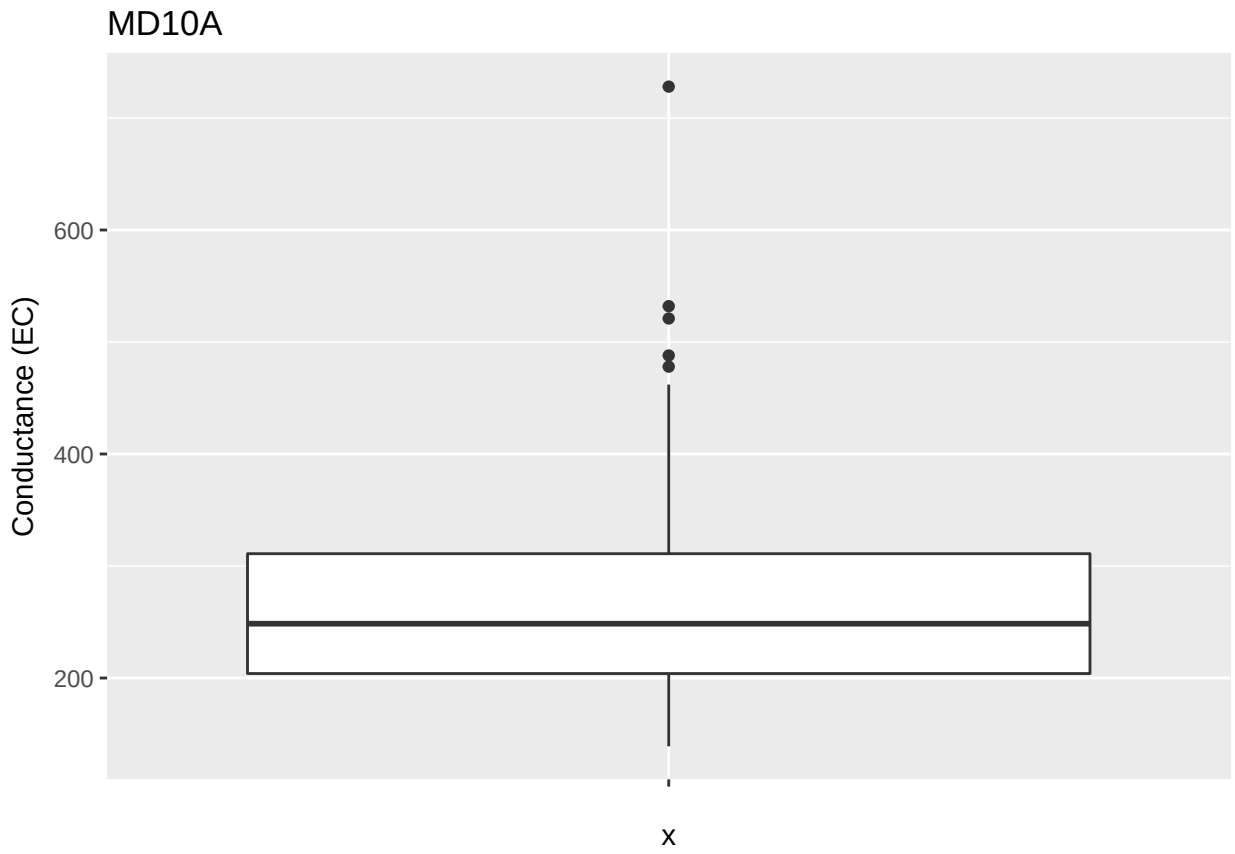

S42
15000
10000
5000
0
Conductance (EC)
x

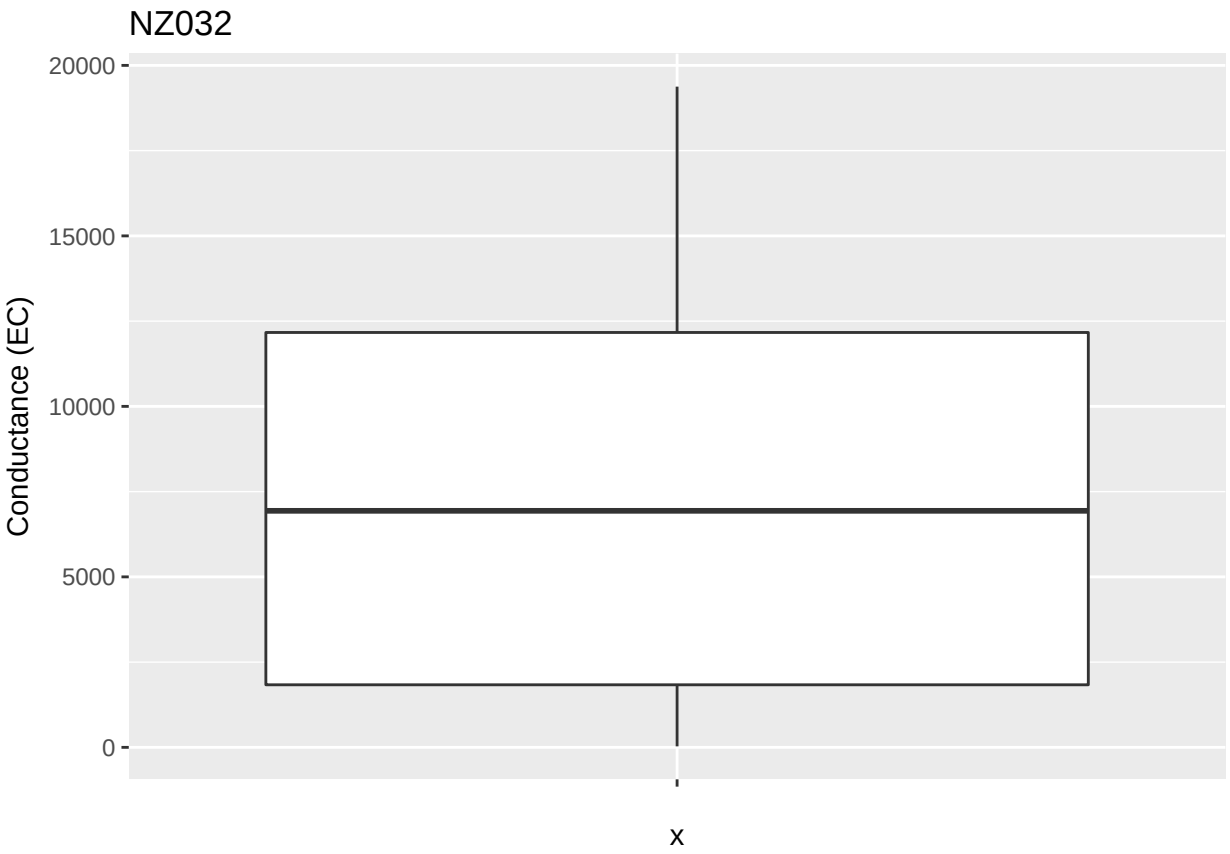

D41
Conductance (EC)
40000
30000
20000
10000
0
x

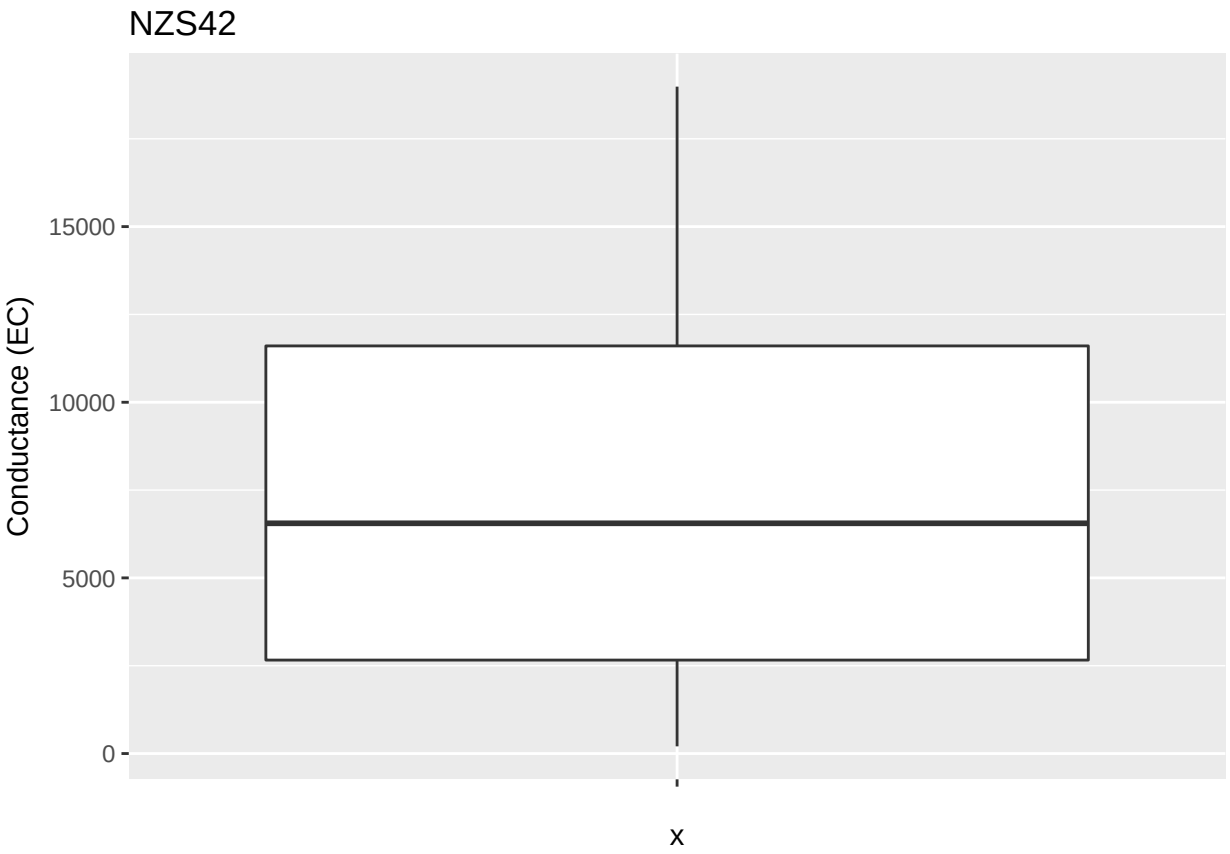

P10A
700
600
500
400
300
200
100
Conductance (EC)
x

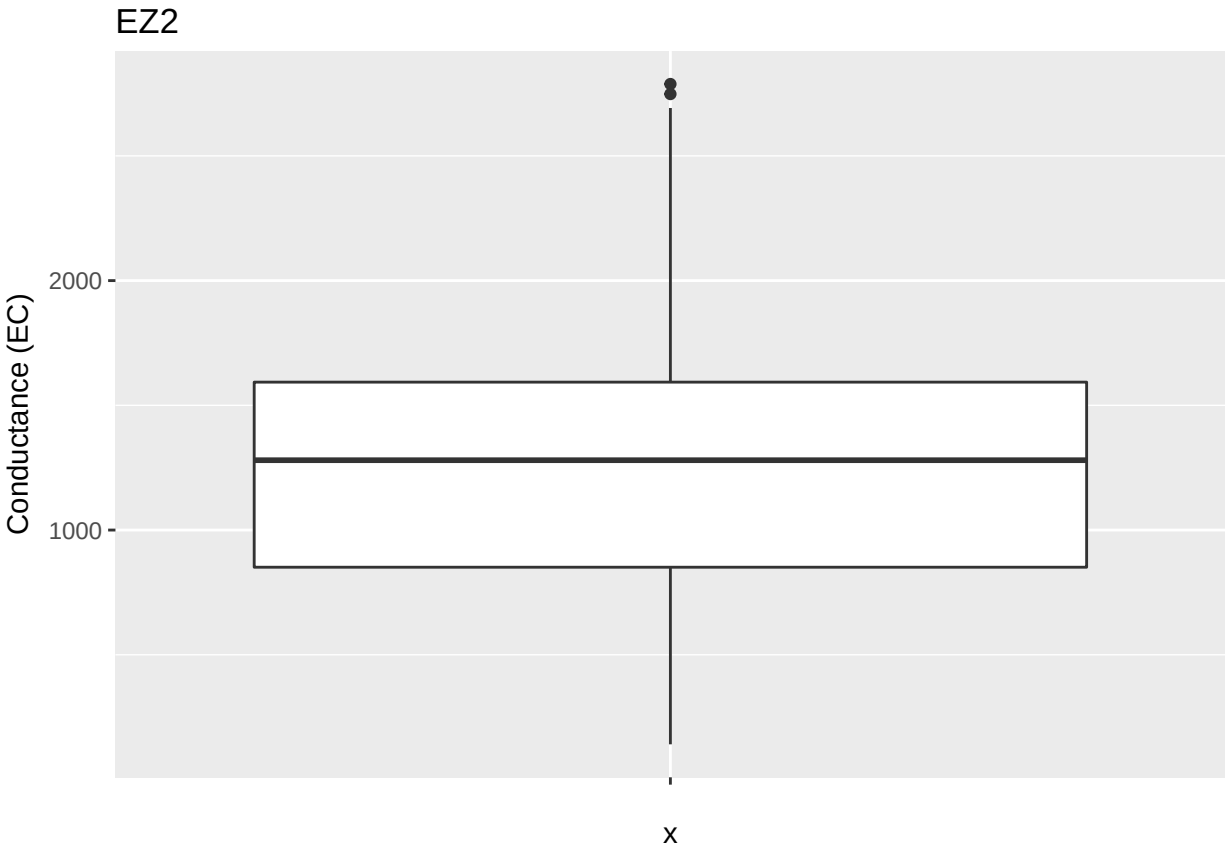

MD7A
Conductance (EC)
250
200
150
100
x

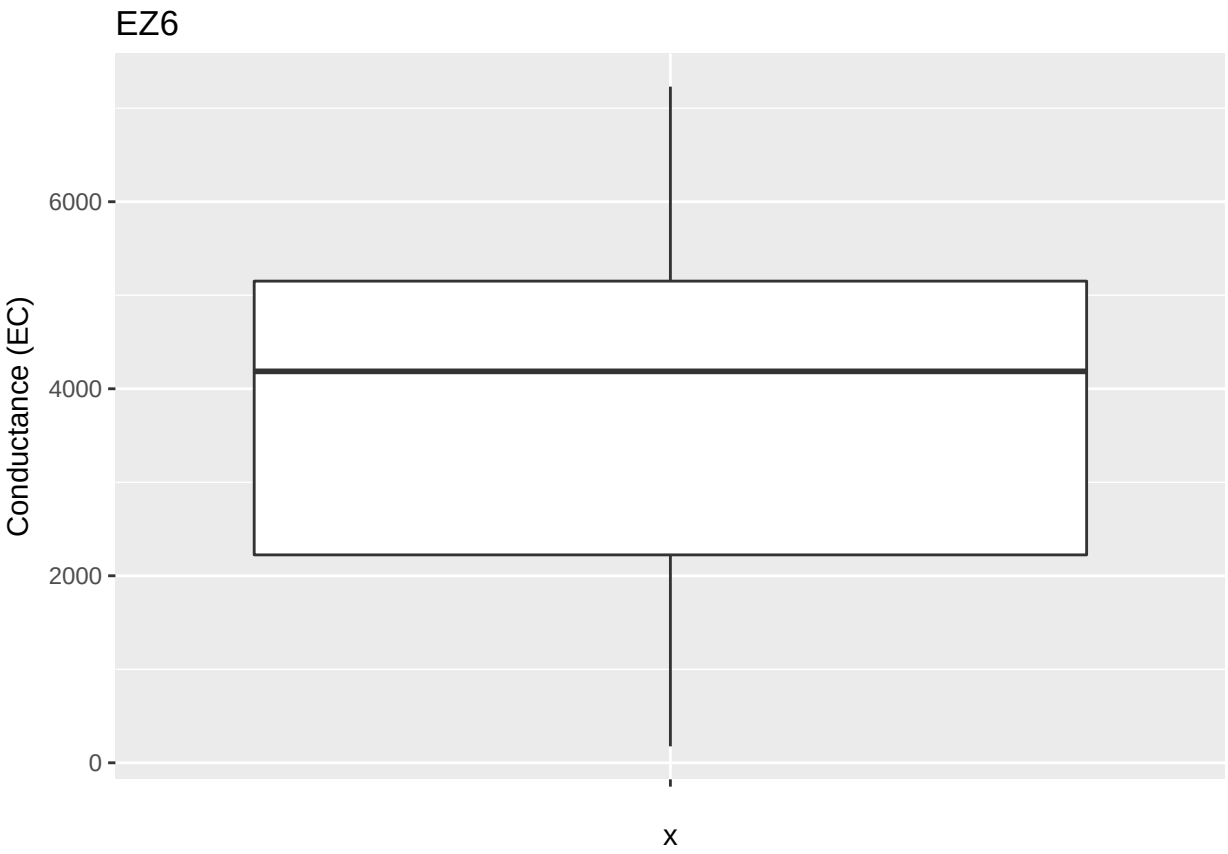

P12A
Conductance (EC)
1000
500
x

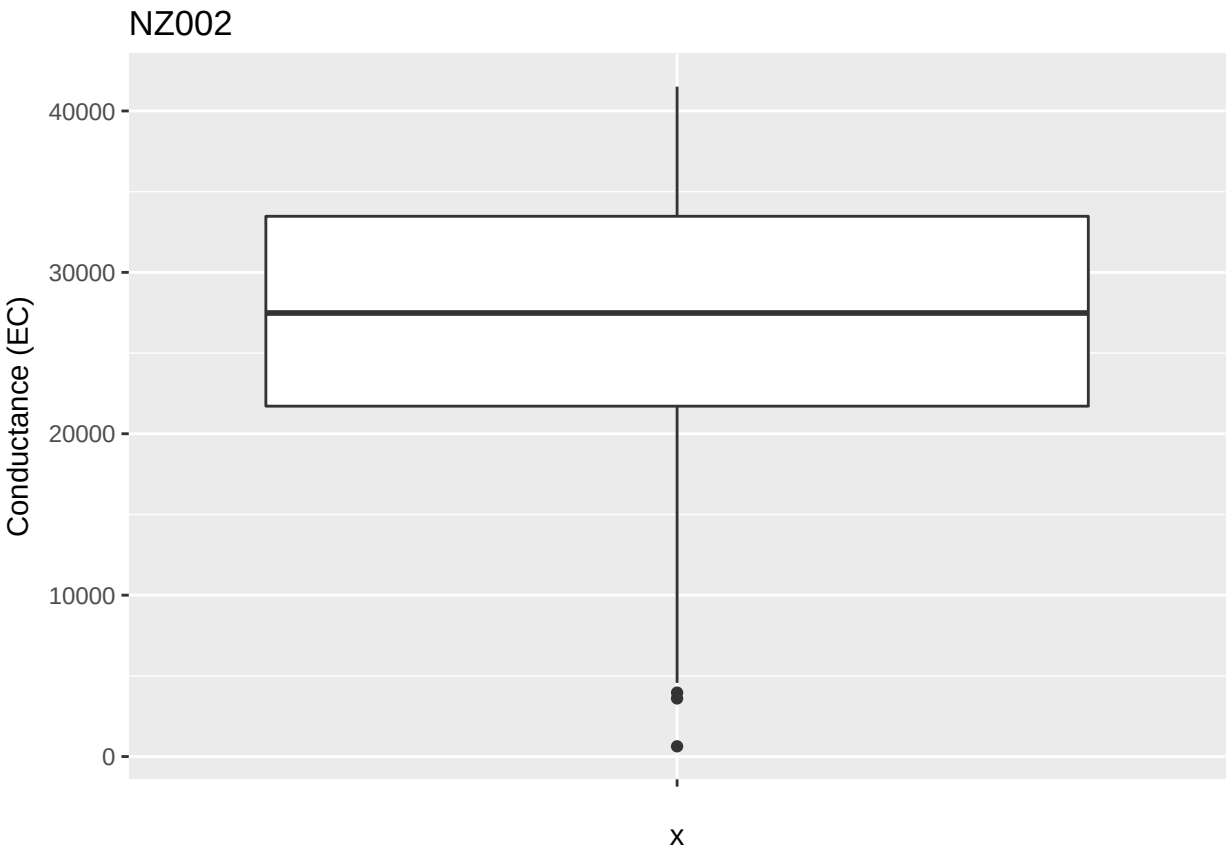

MD10A
Conductance (EC)
600
400
200
x

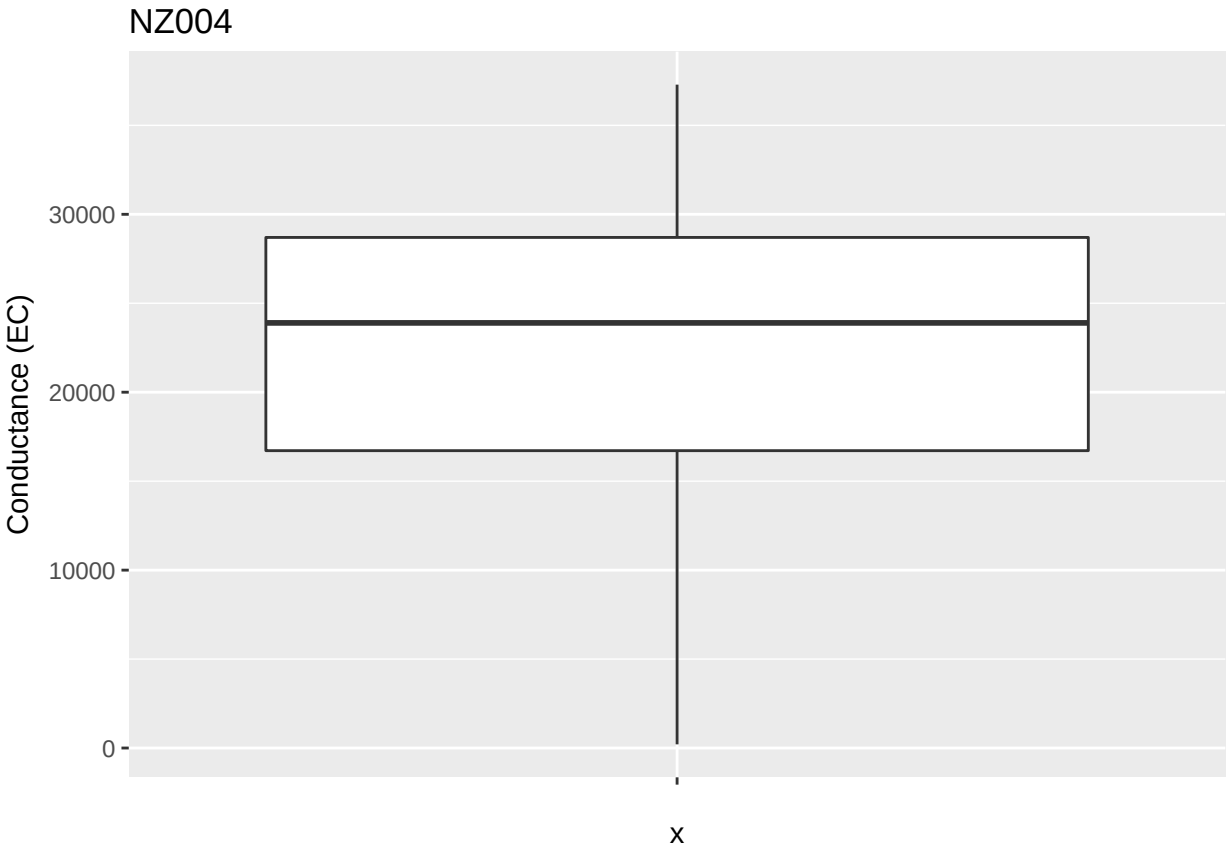

NZ032
20000
15000
10000
5000
0
Conductance (EC)
x

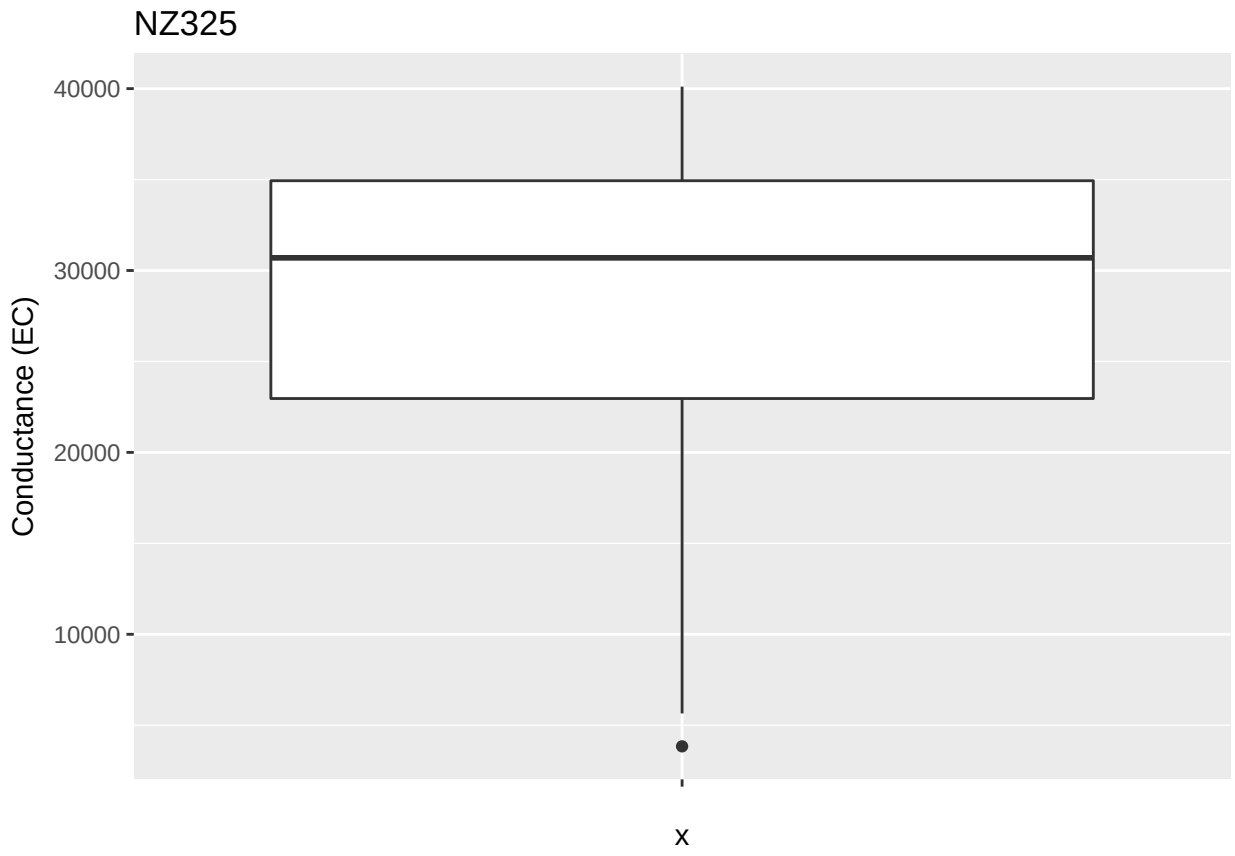

NZS42
Conductance (EC)
15000
10000
5000
0
x

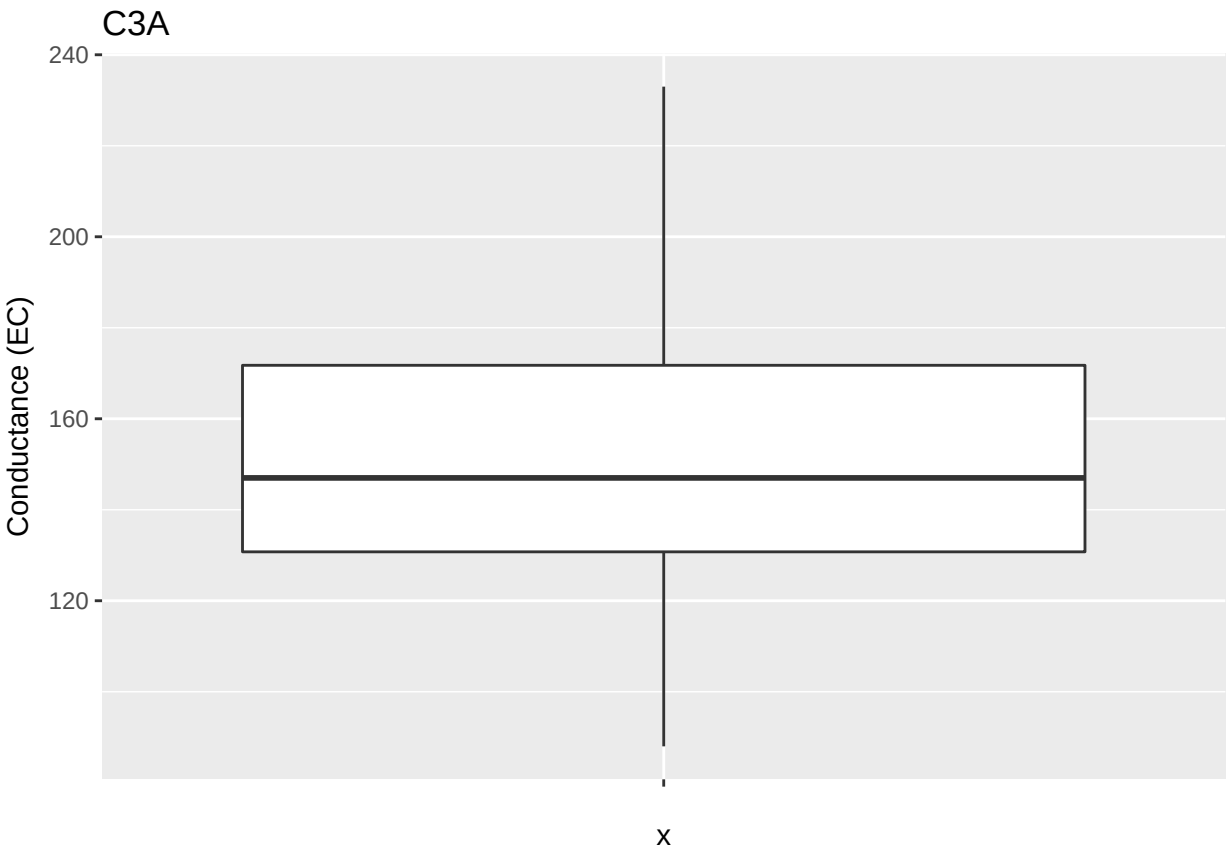

EZ2
Conductance (EC)
2000
1000
x

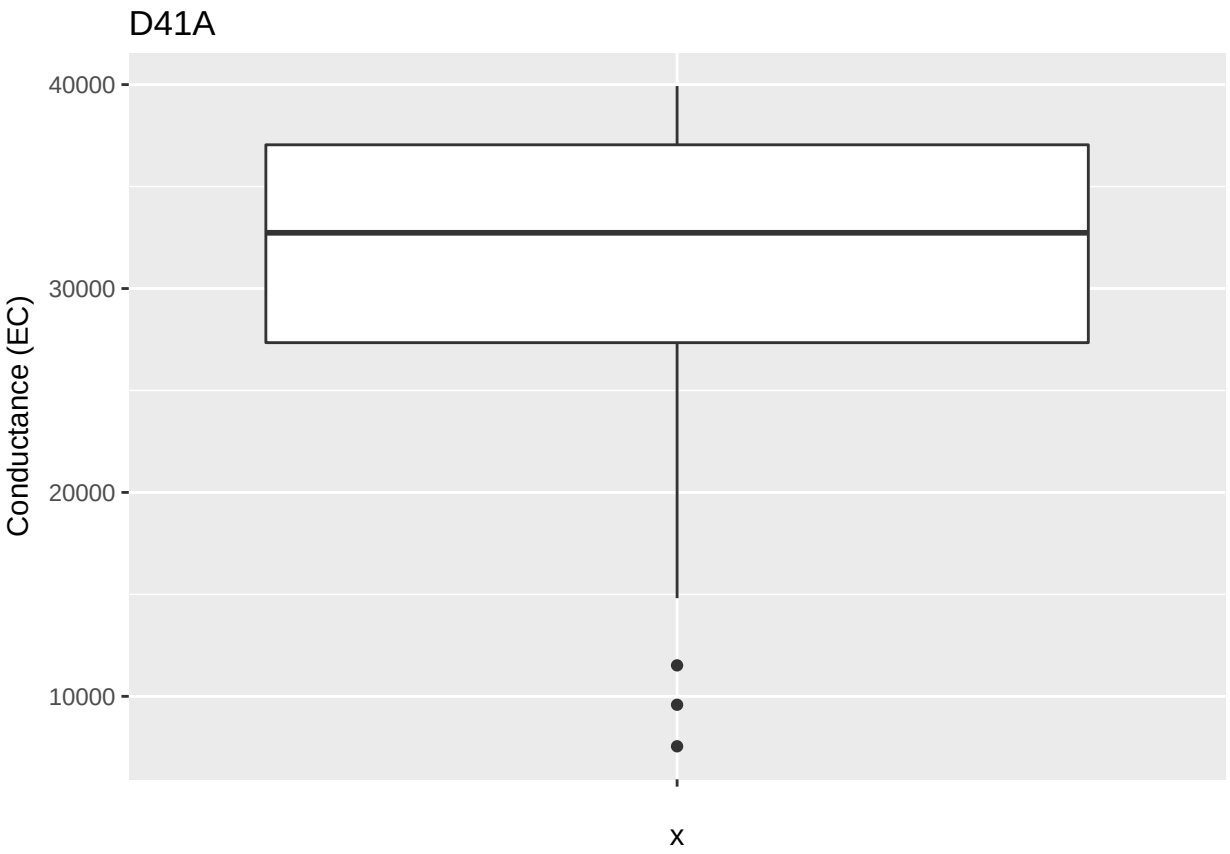

EZ6
Conductance (EC)
6000
4000
2000
0
x

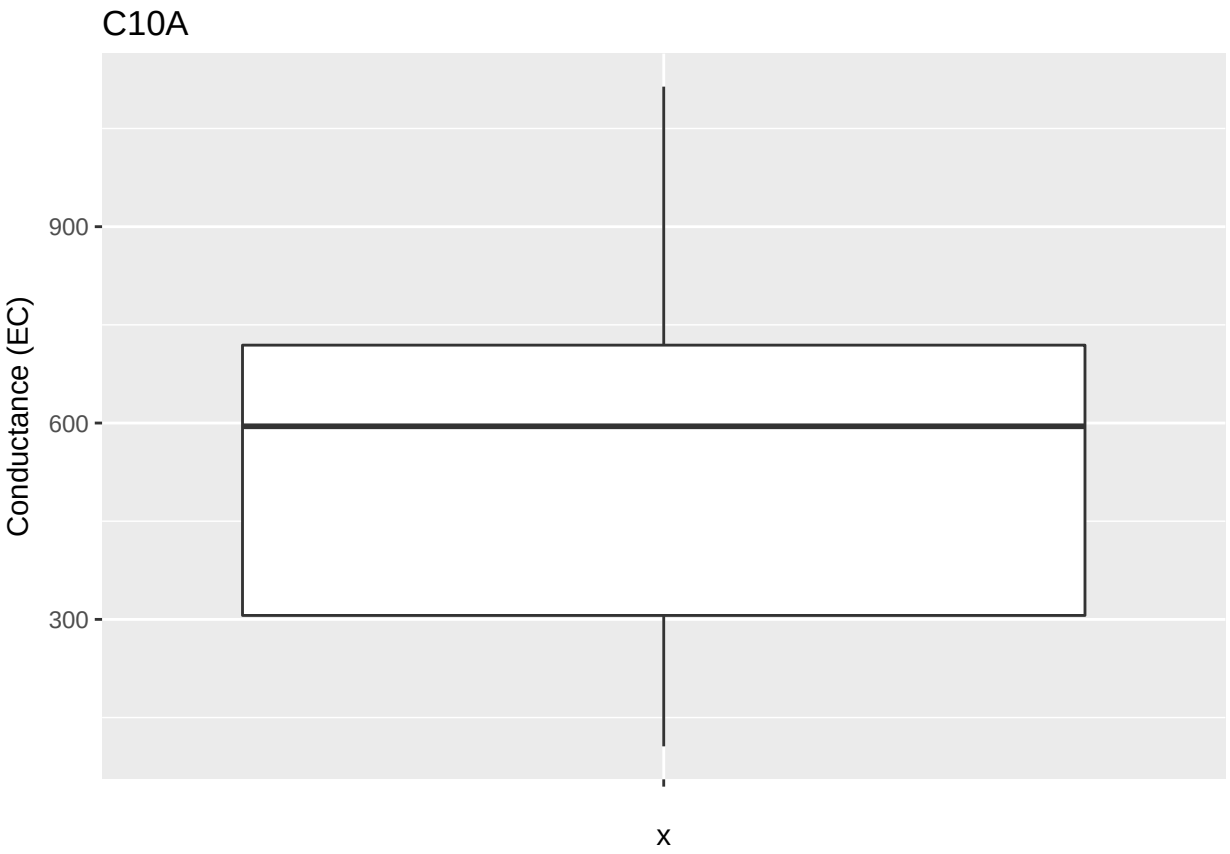

NZ002
40000
30000
20000
10000
0
Conductance (EC)
x

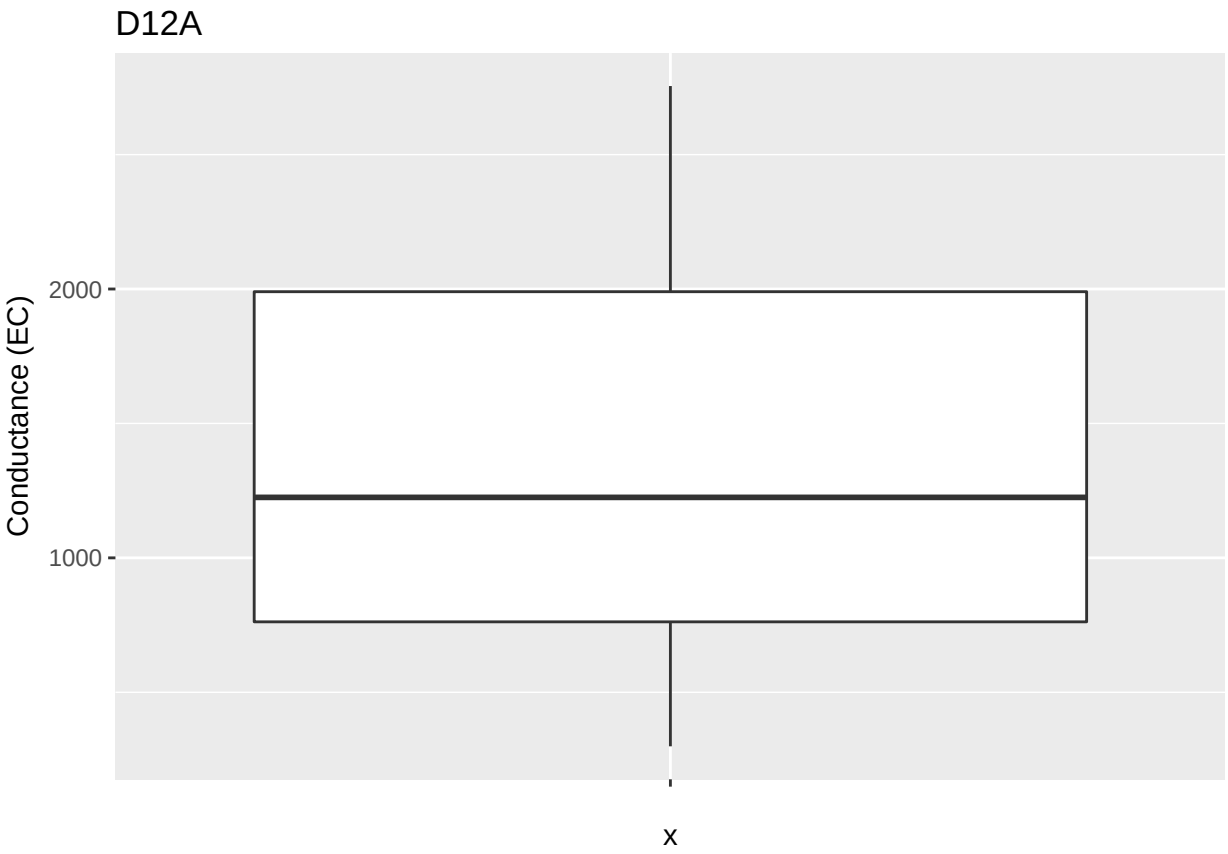

NZ004
Conductance (EC)
30000
20000
10000
0
x

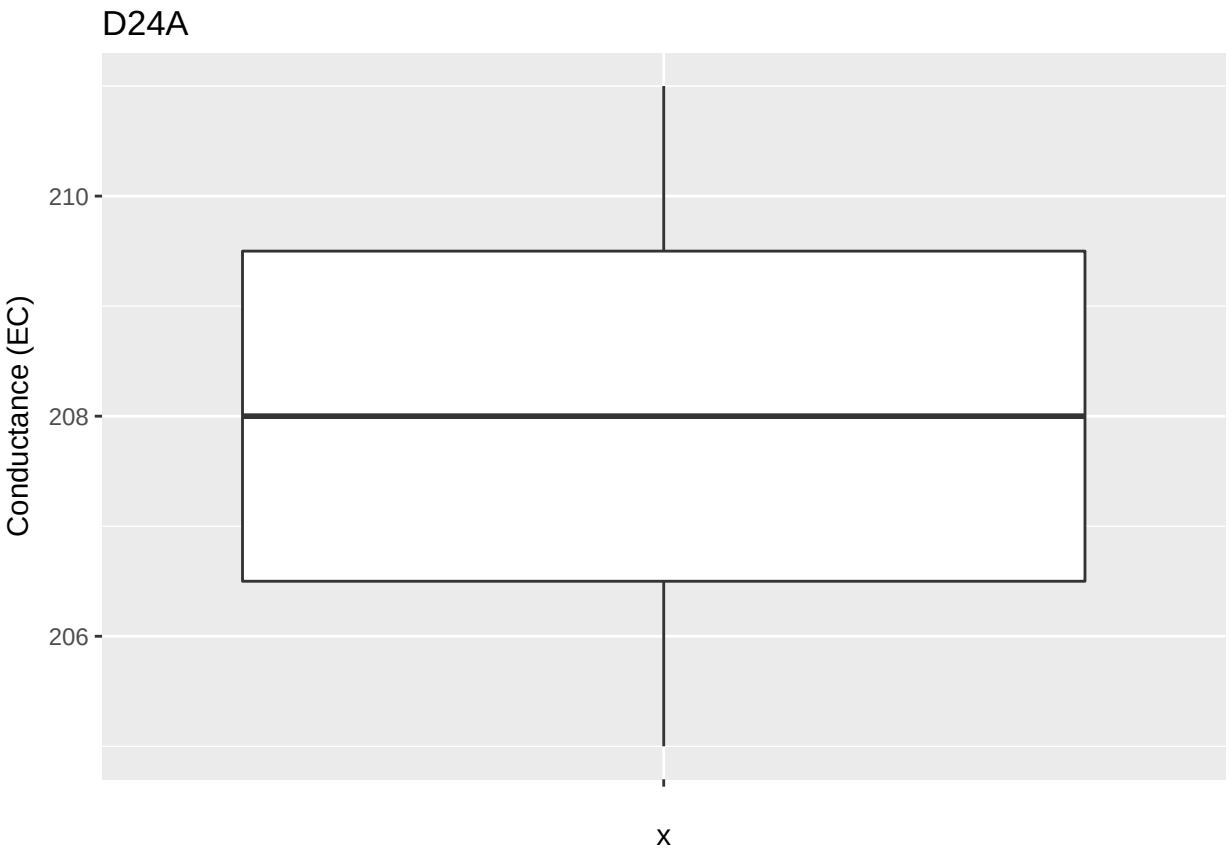

NZ325
40000
30000
20000
10000
Conductance (EC)
x

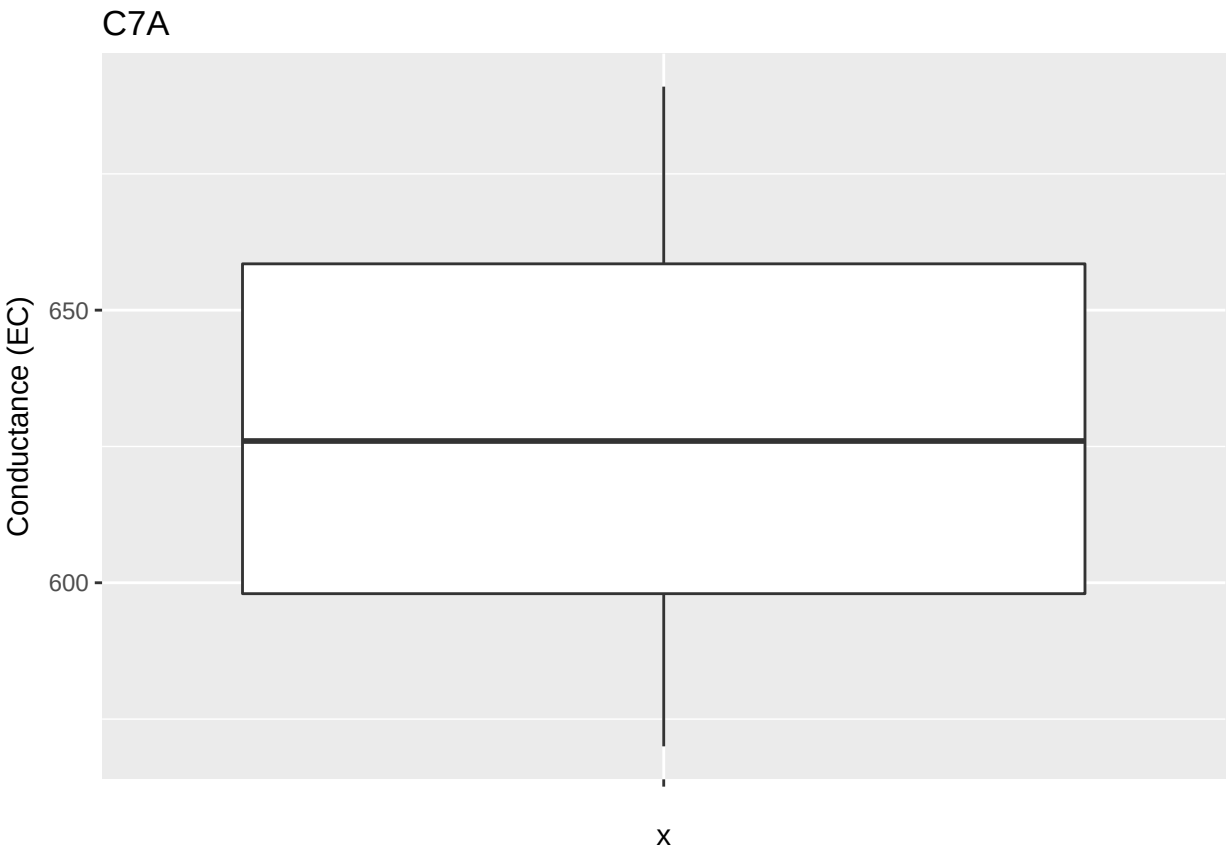

C3A
240
200
160
120
Conductance (EC)
x

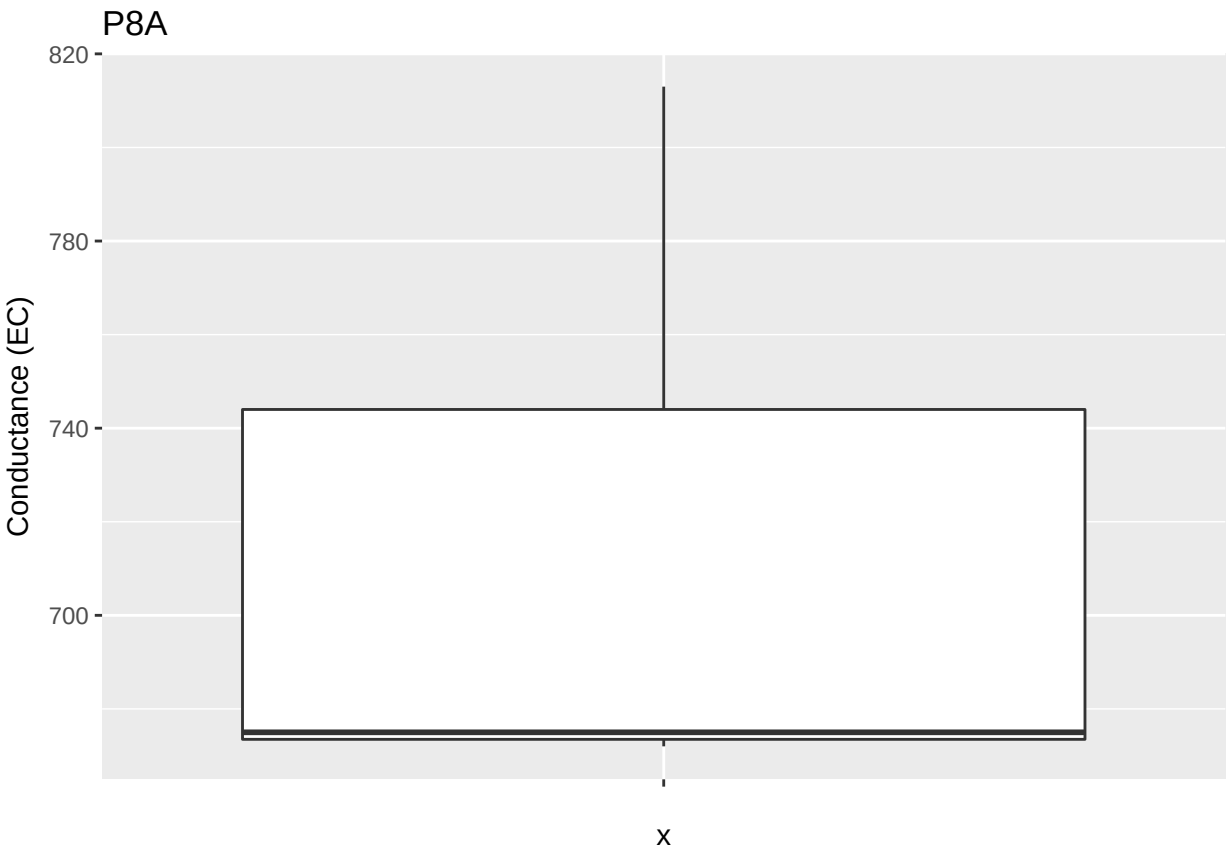

D41A
40000
30000
20000
10000
Conductance (EC)
x

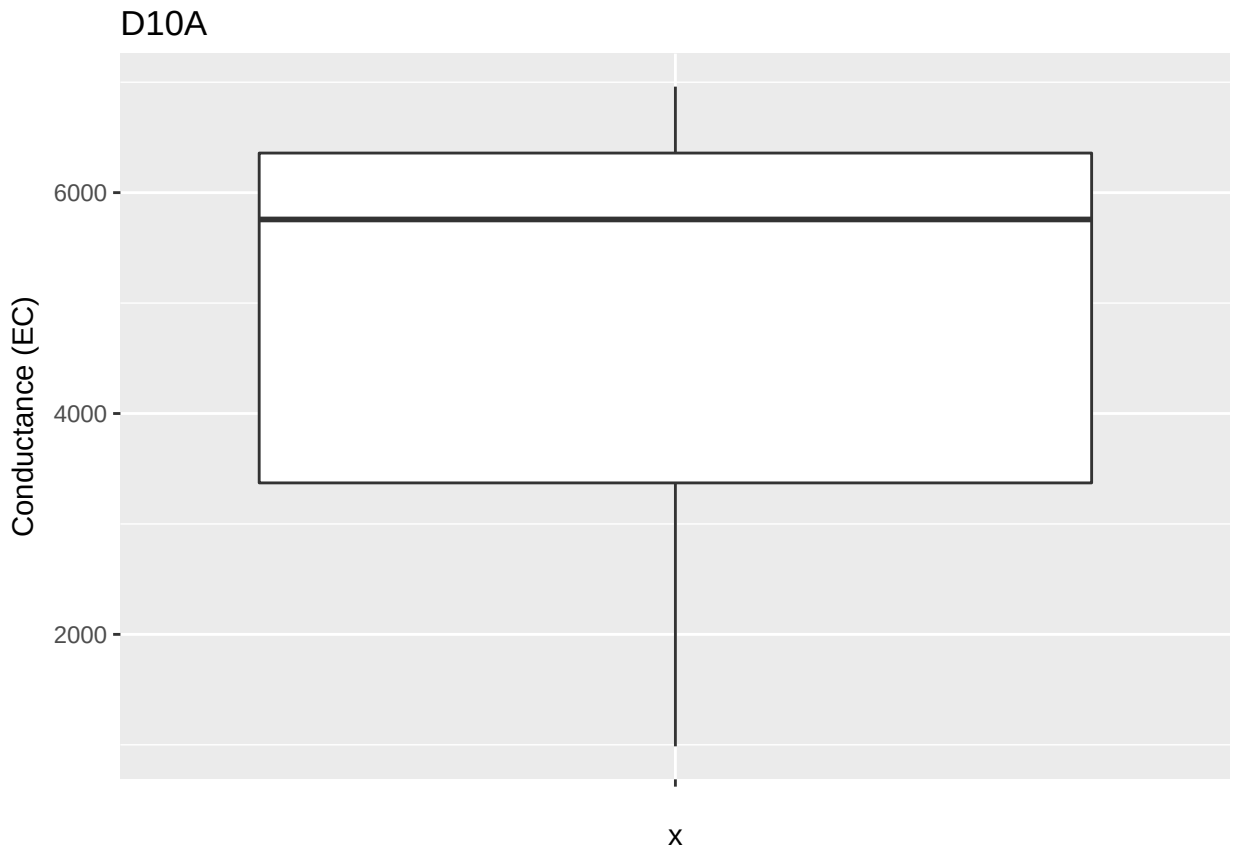

C10A
Conductance (EC)
900
600
300
x

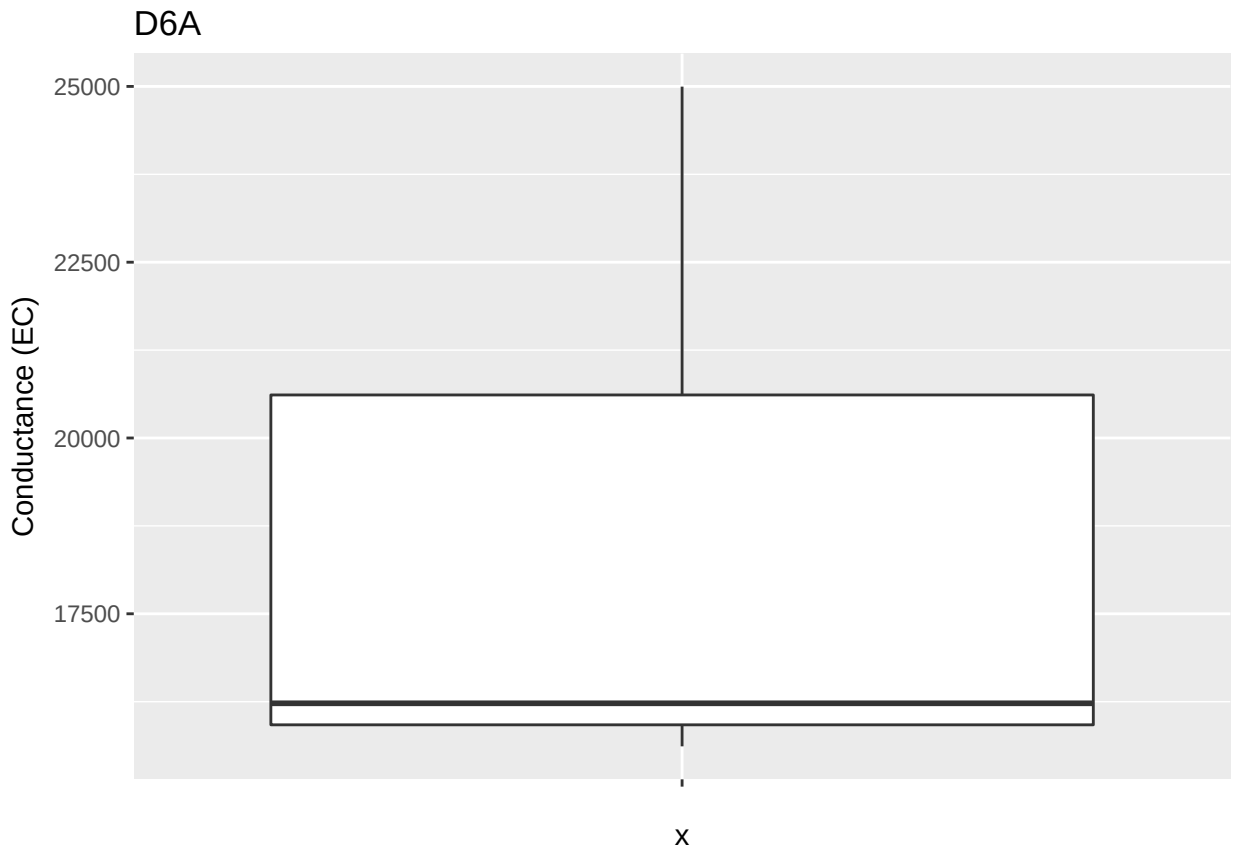

D12A
Conductance (EC)
2000
1000
x


D24A
Conductance (EC)
210
208
206
x

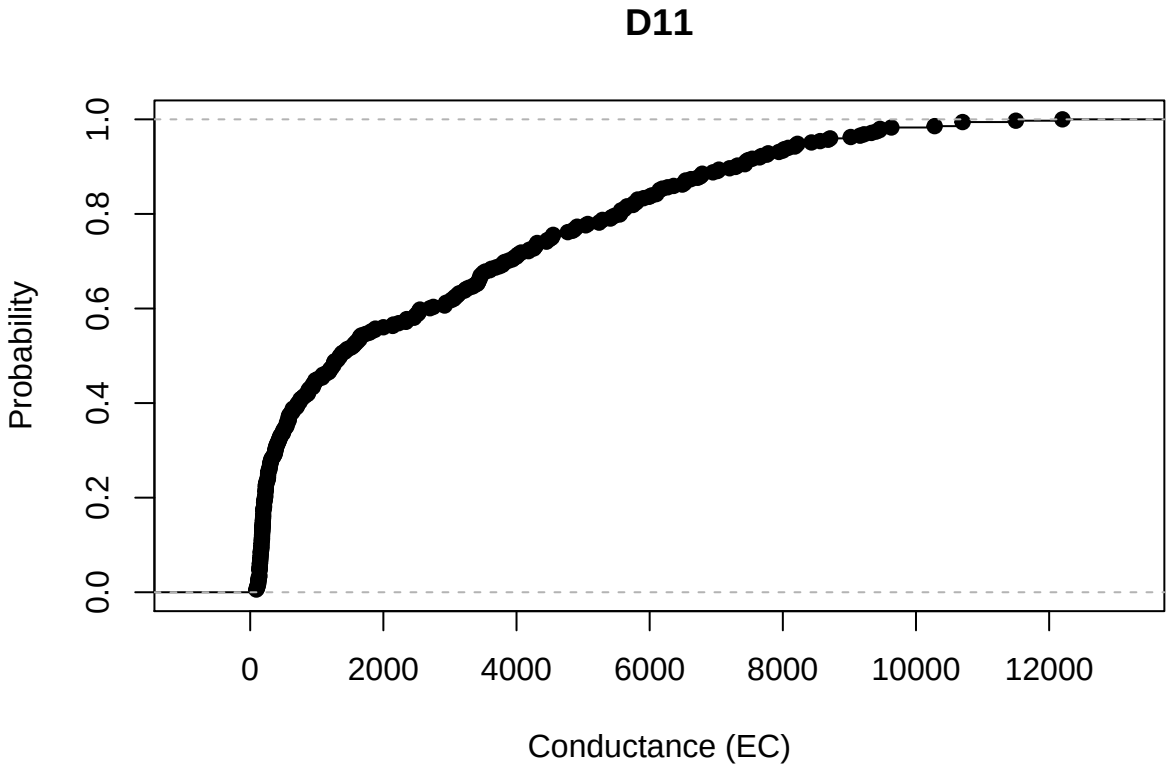

C7A
Conductance (EC)
650
600
x

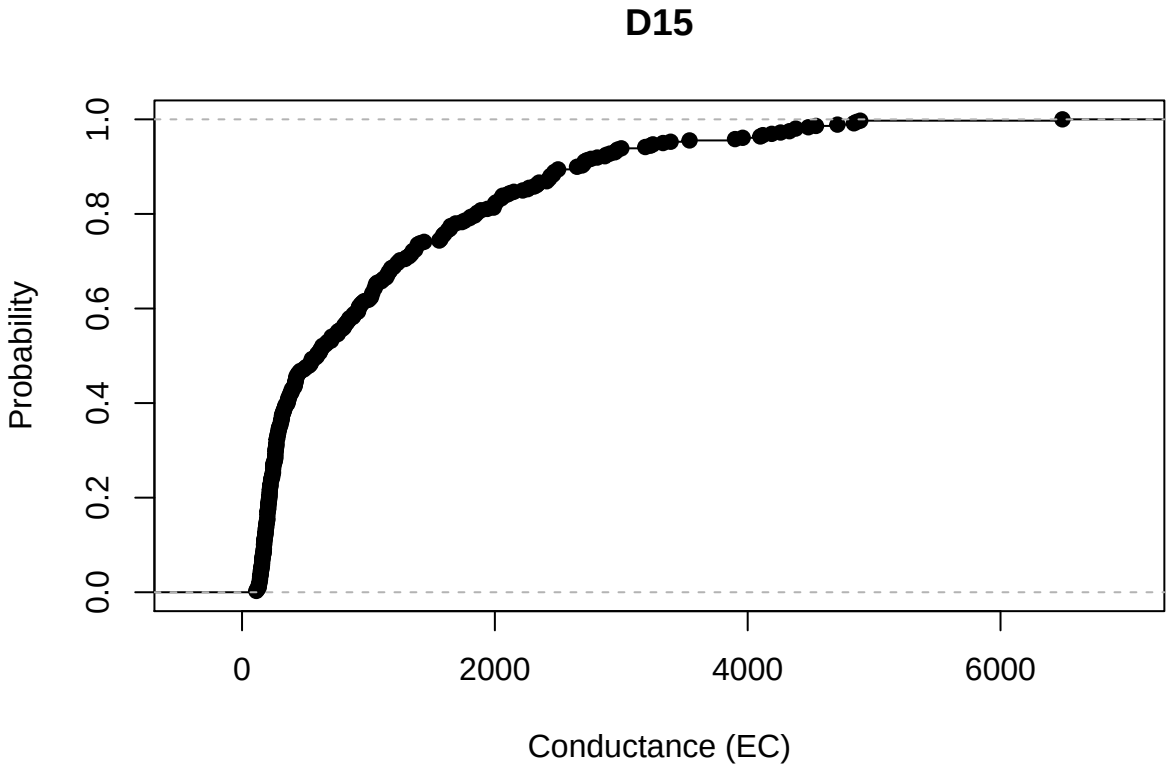

P8A
820
780
740
700
Conductance (EC)
x

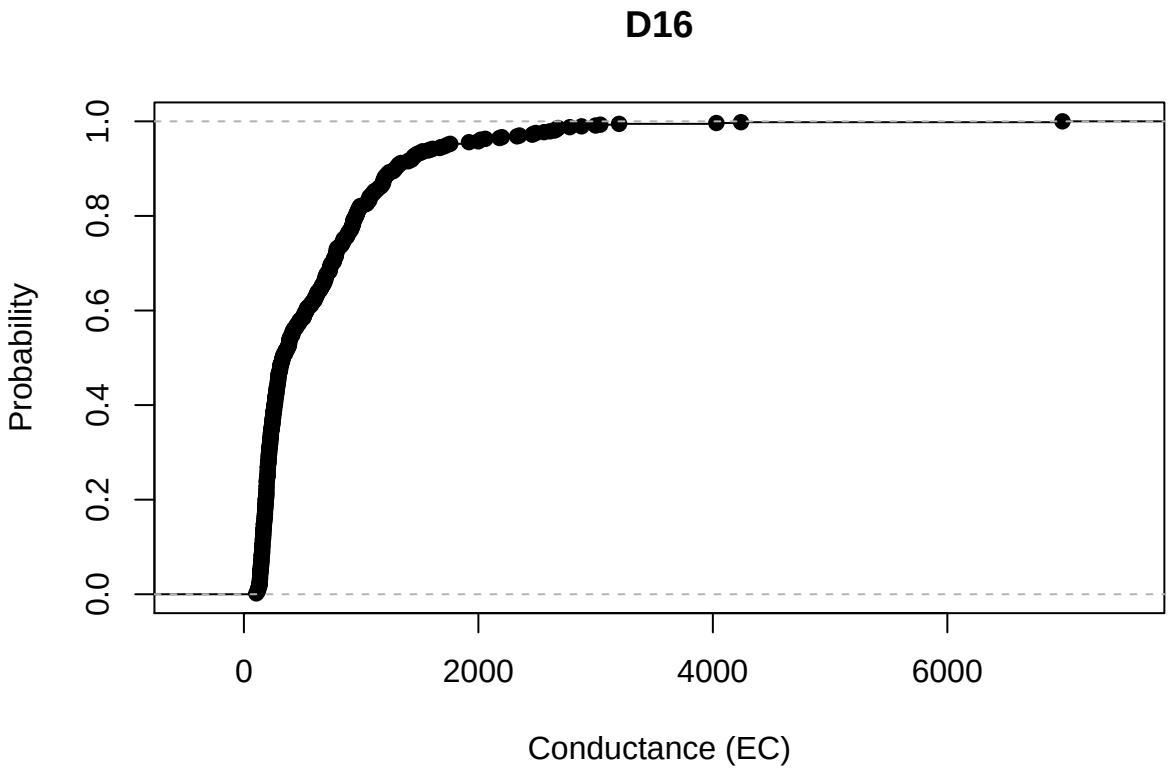

D10A
Conductance (EC)
6000
4000
2000
x

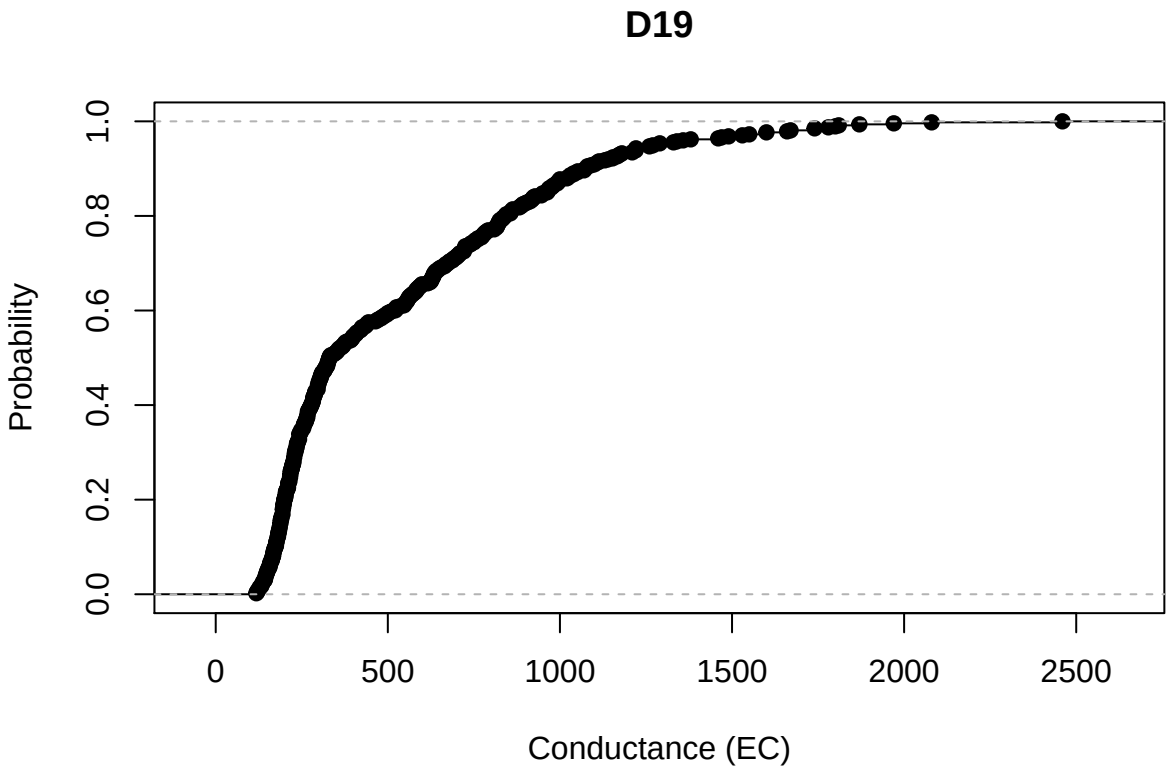


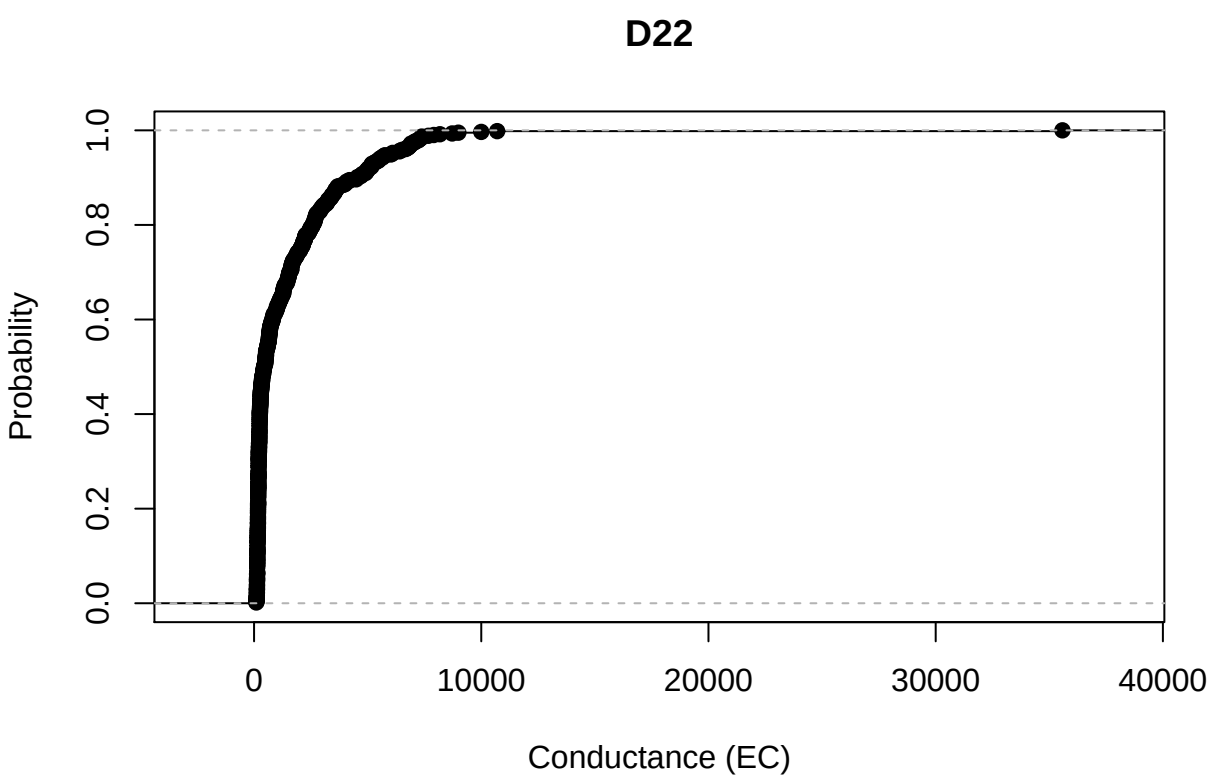

```
## cummulative density plot
sites <- unique(wQ$StationCode)
try(for (i in 1:length(sites)){
   mySub <- subset(wQ,StationCode == sites[i] ,select = `Conductance (EC)` )
    cdf_fun <- ecdf(mySub$`Conductance (EC)`)
    plot(cdf_fun, xlab = "Conductance (EC)", ylab = "Probability", main = sites[i])
})
```

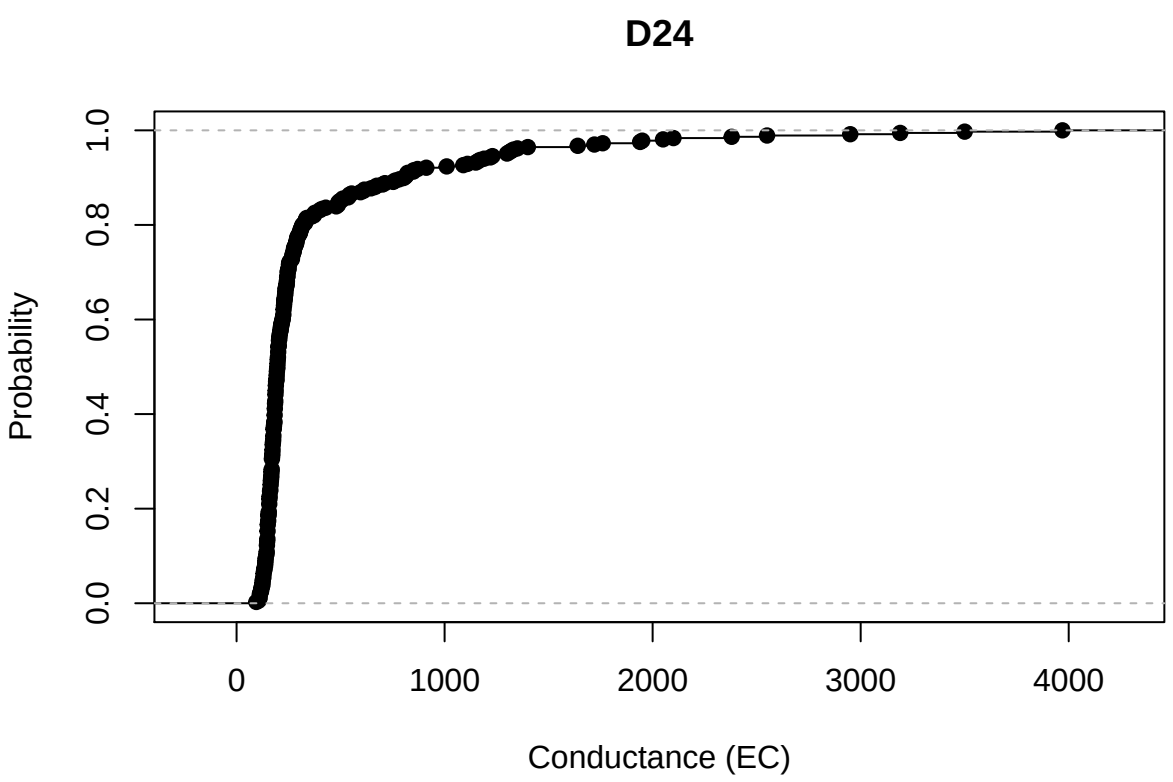

D11
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000
10000
12000
Conductance (EC)

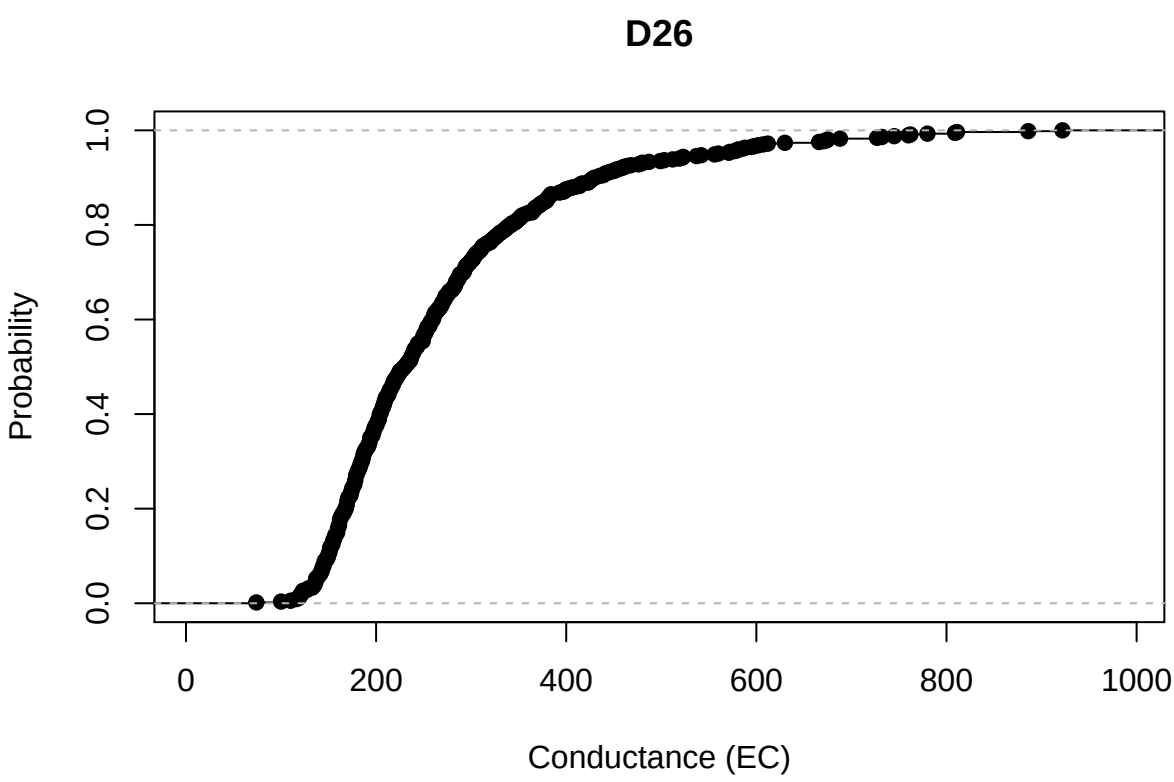

D15
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
Conductance (EC)

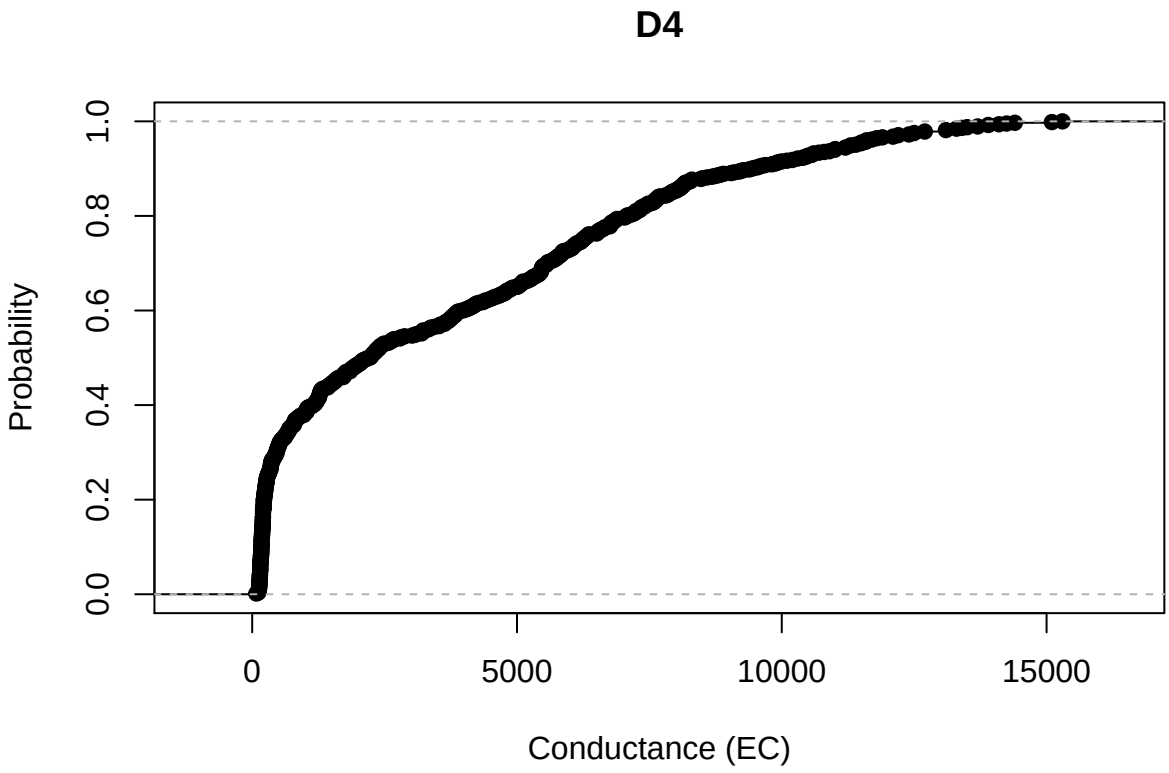

D16
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
Conductance (EC)

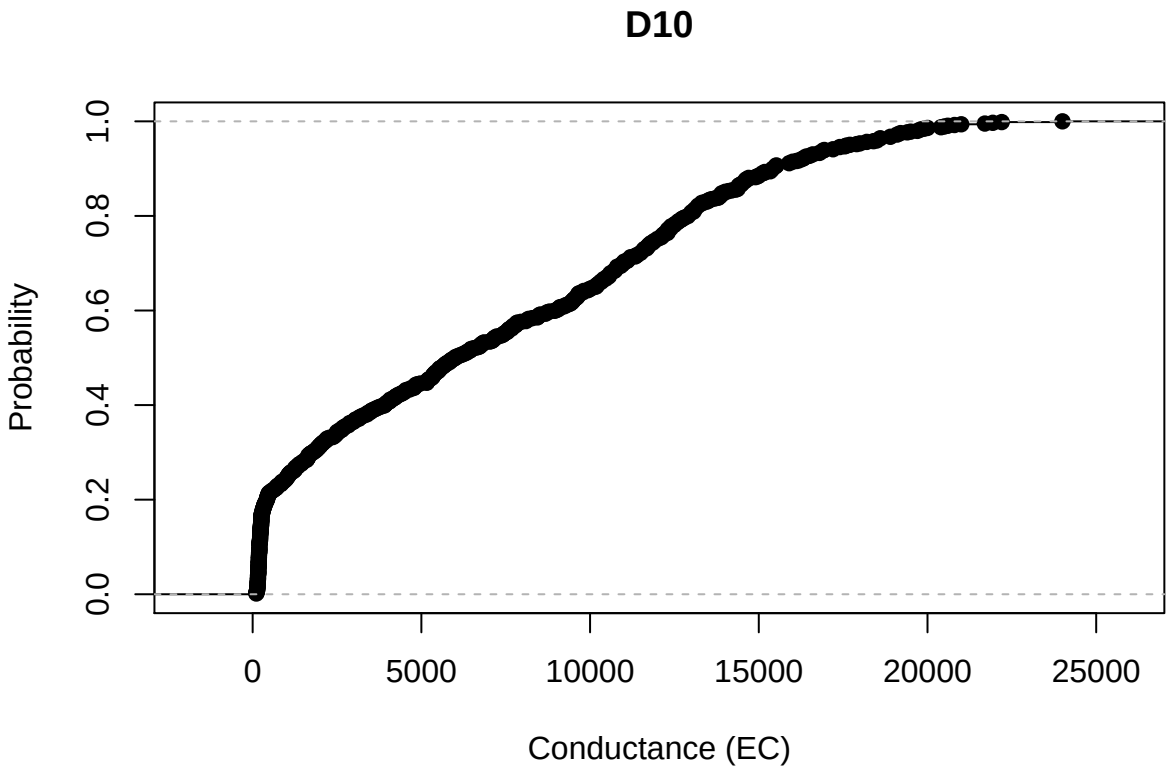

D19
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
Conductance (EC)

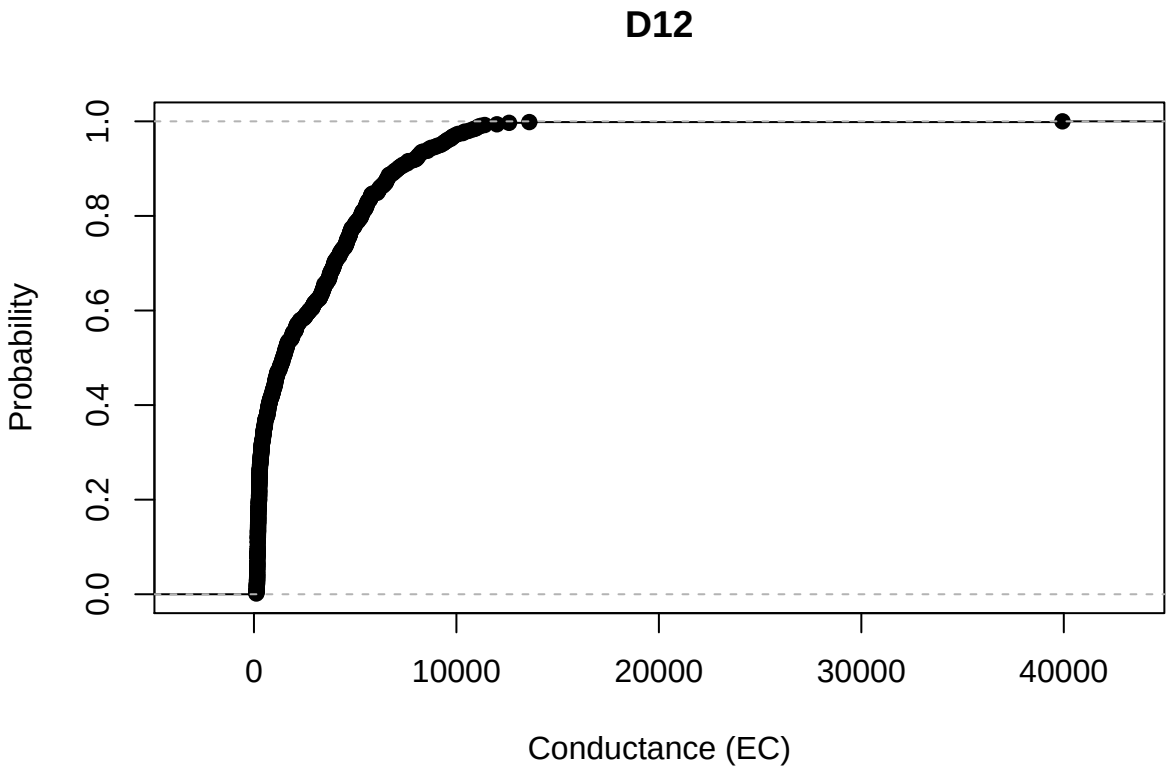

D22
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
30000
40000
Conductance (EC)

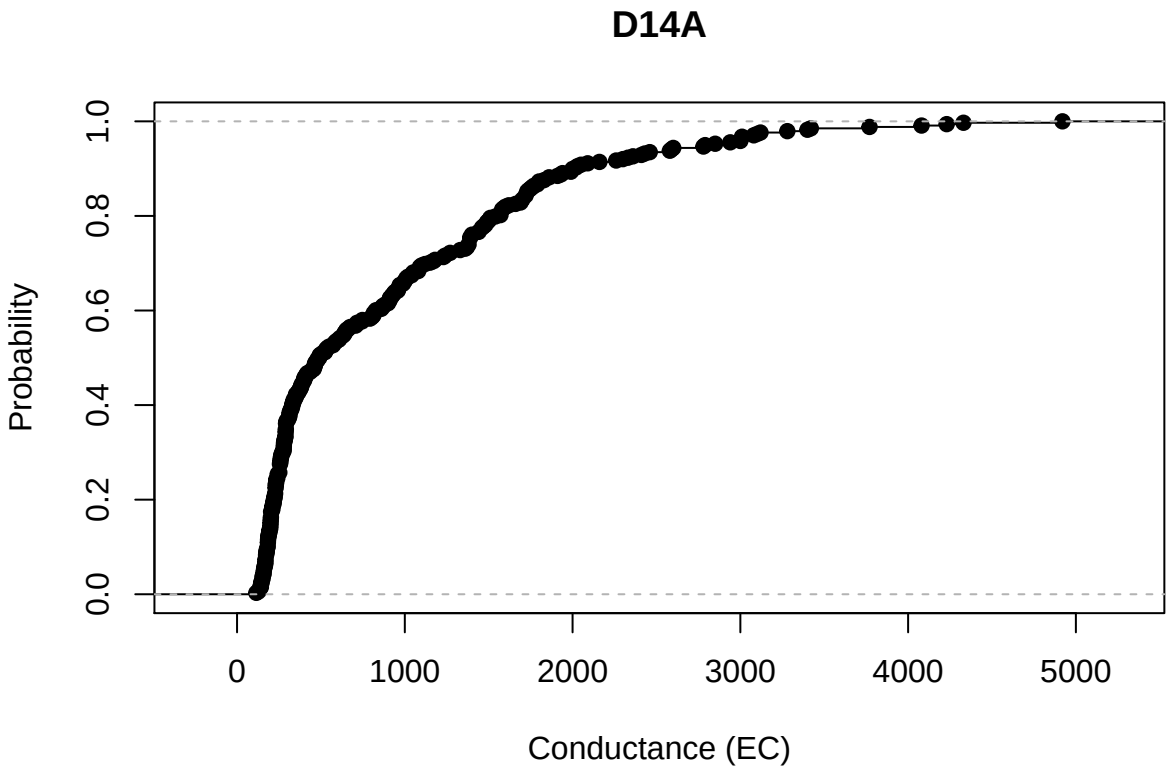

D24
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
1000
2000
3000
4000
Conductance (EC)

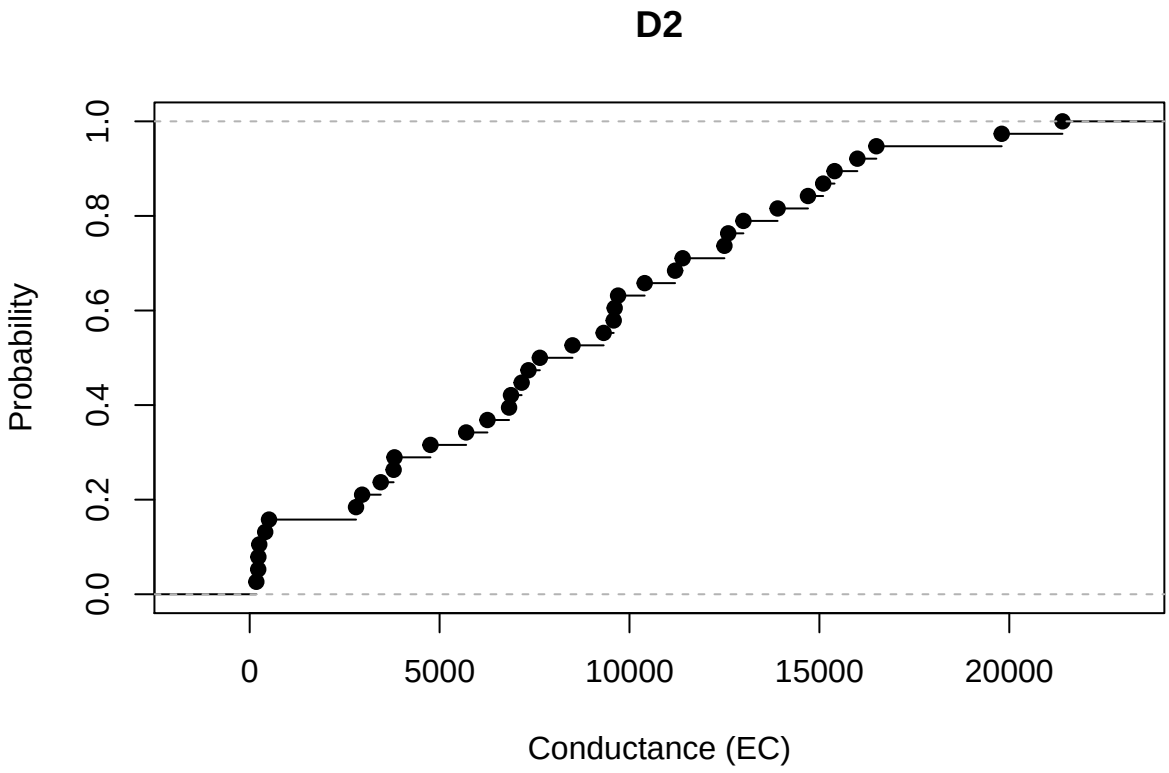

D26
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
Conductance (EC)

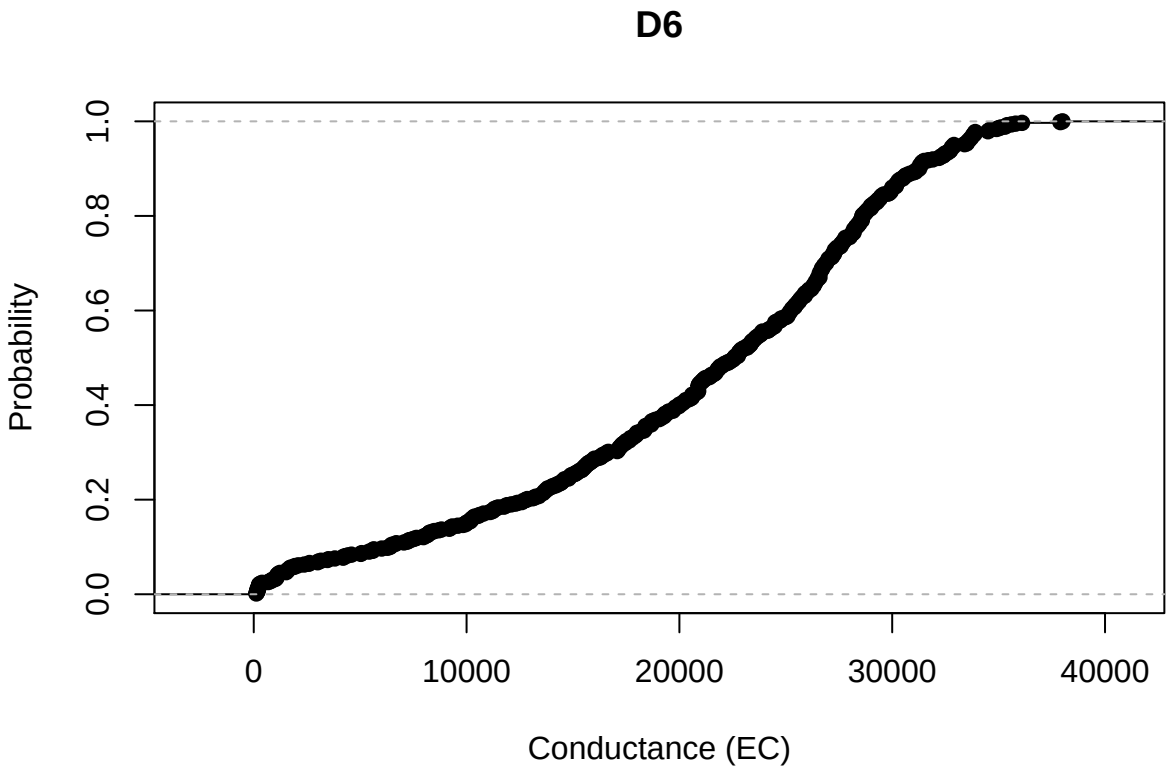

D4
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
5000
10000
15000
Conductance (EC)

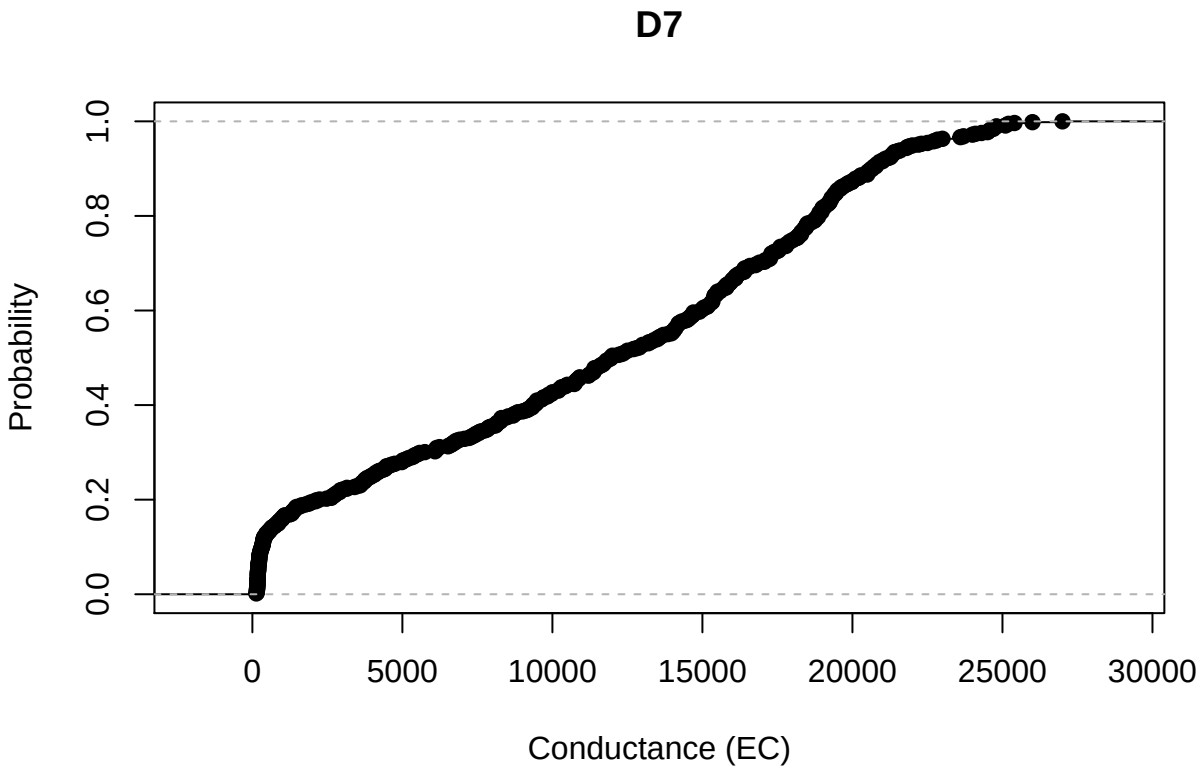

D10
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
5000
10000
15000
20000
25000
Conductance (EC)

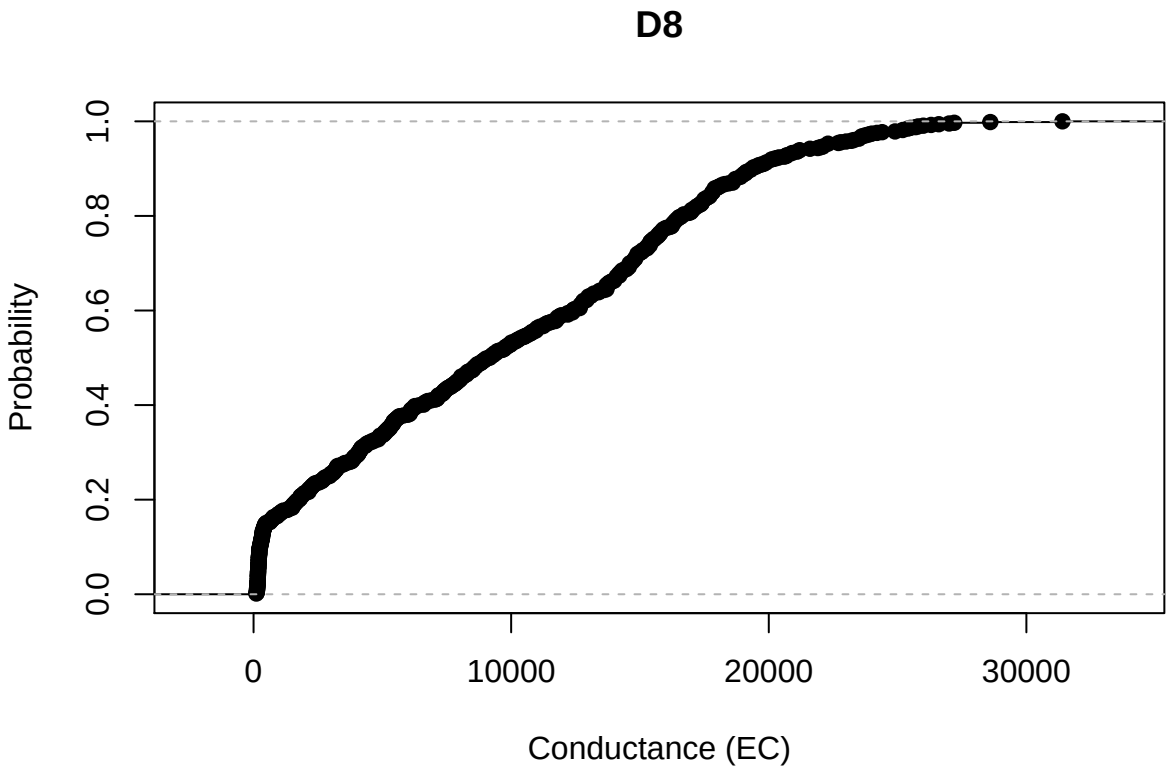

D12
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
30000
40000
Conductance (EC)

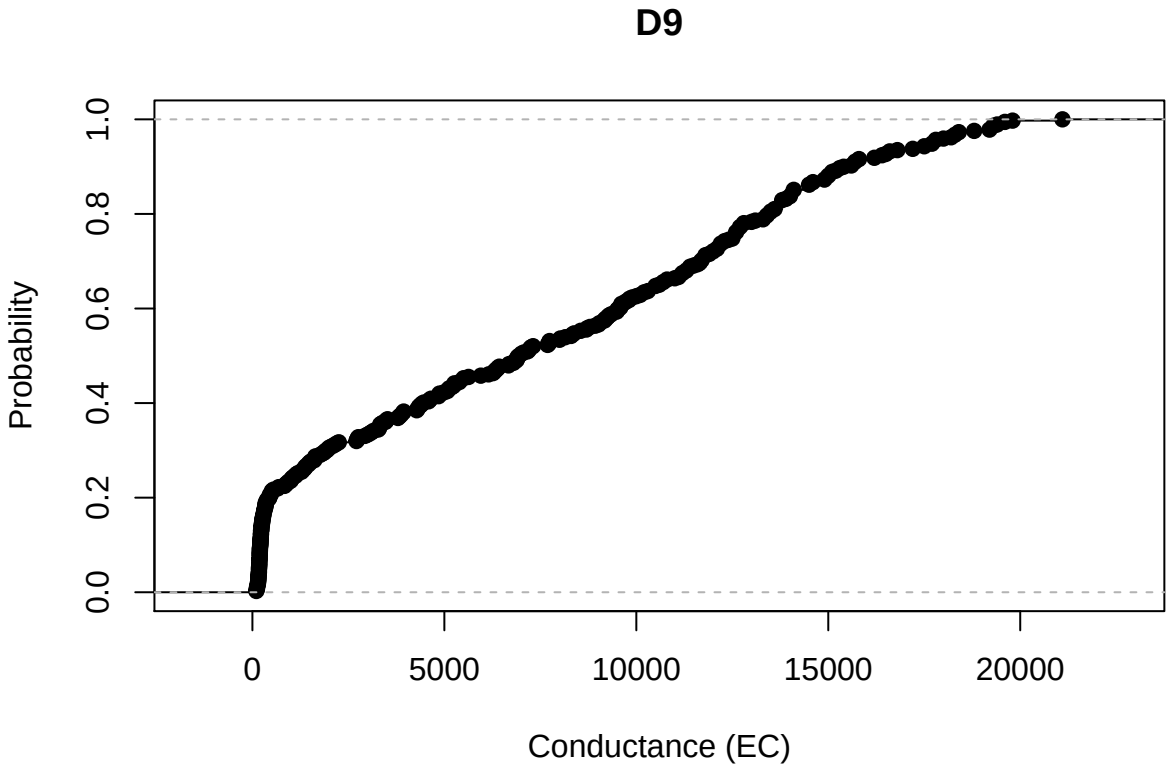

D14A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
1000
2000
3000
4000
5000
Conductance (EC)

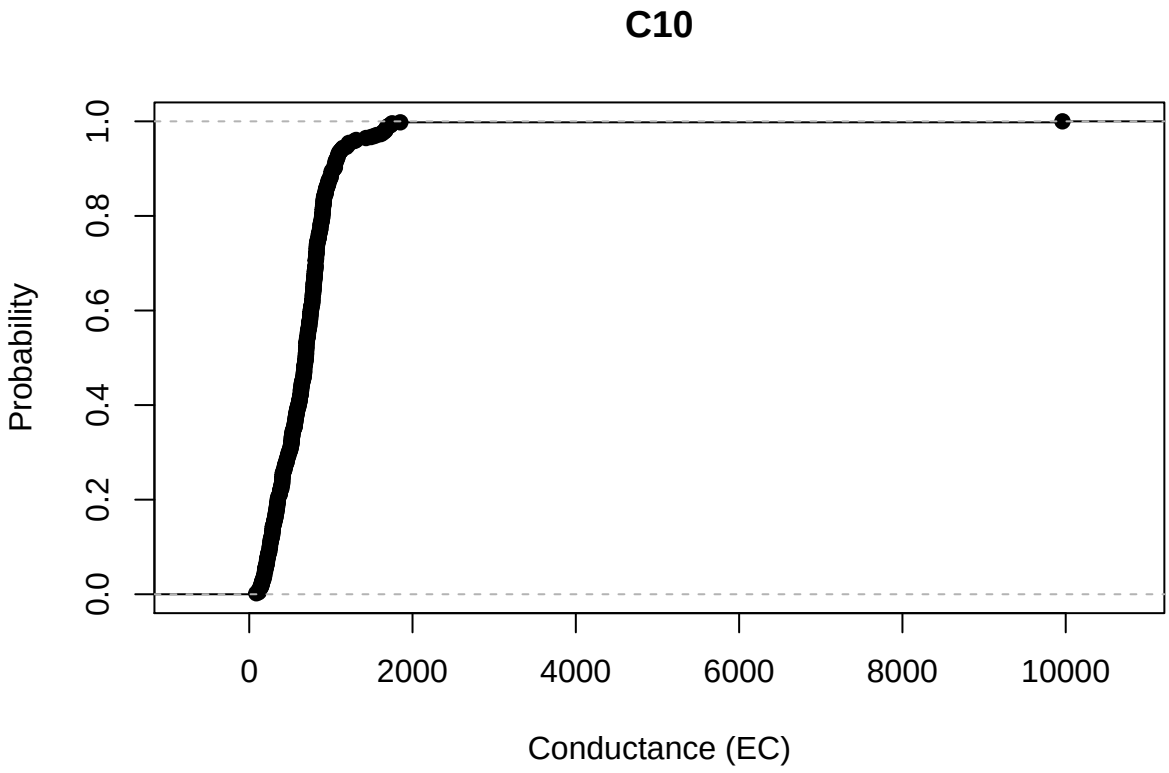

D2
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
5000
10000
15000
20000
Conductance (EC)

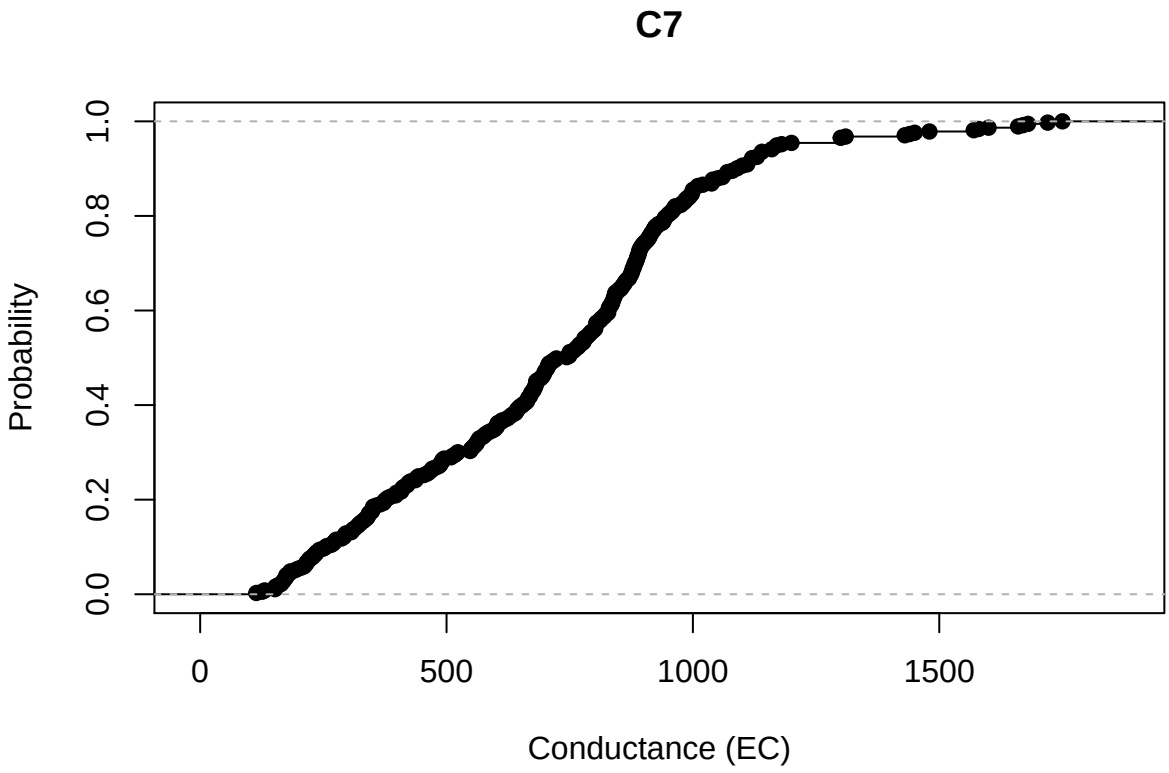

D6
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
30000
40000
Conductance (EC)

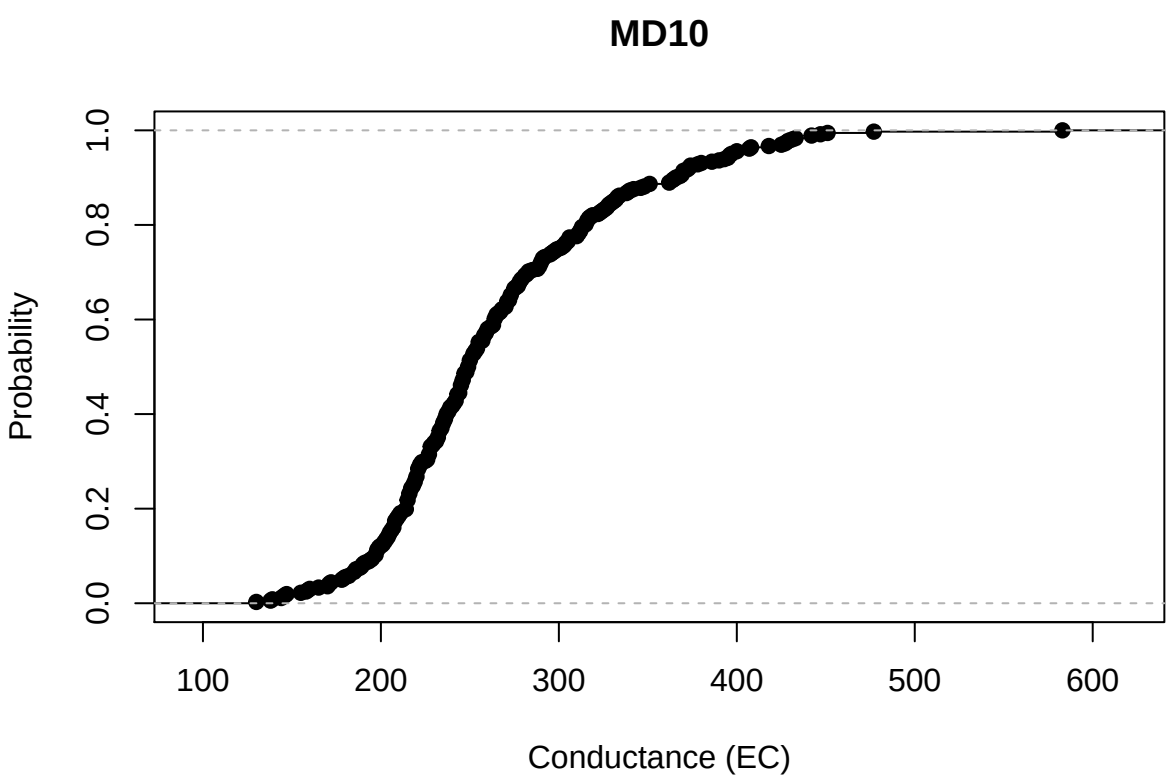

D7
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
5000
10000
15000
20000
25000
30000
Conductance (EC)

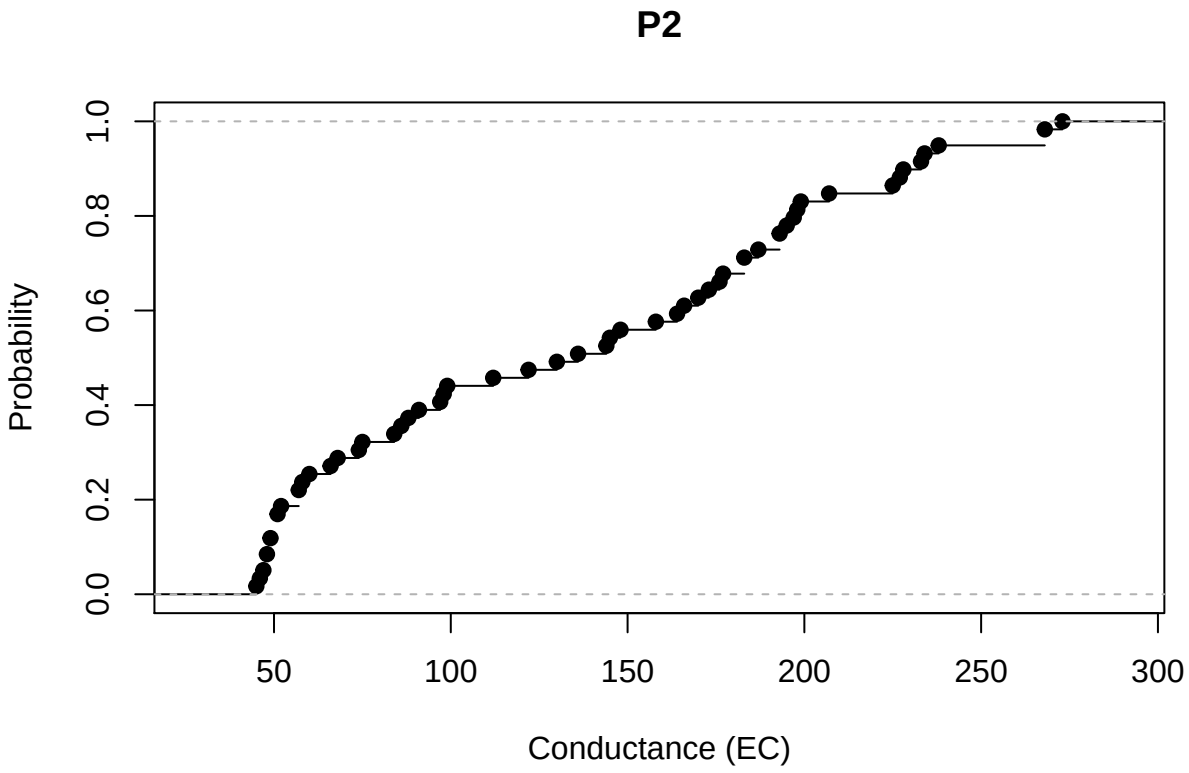

D8
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
30000
Conductance (EC)

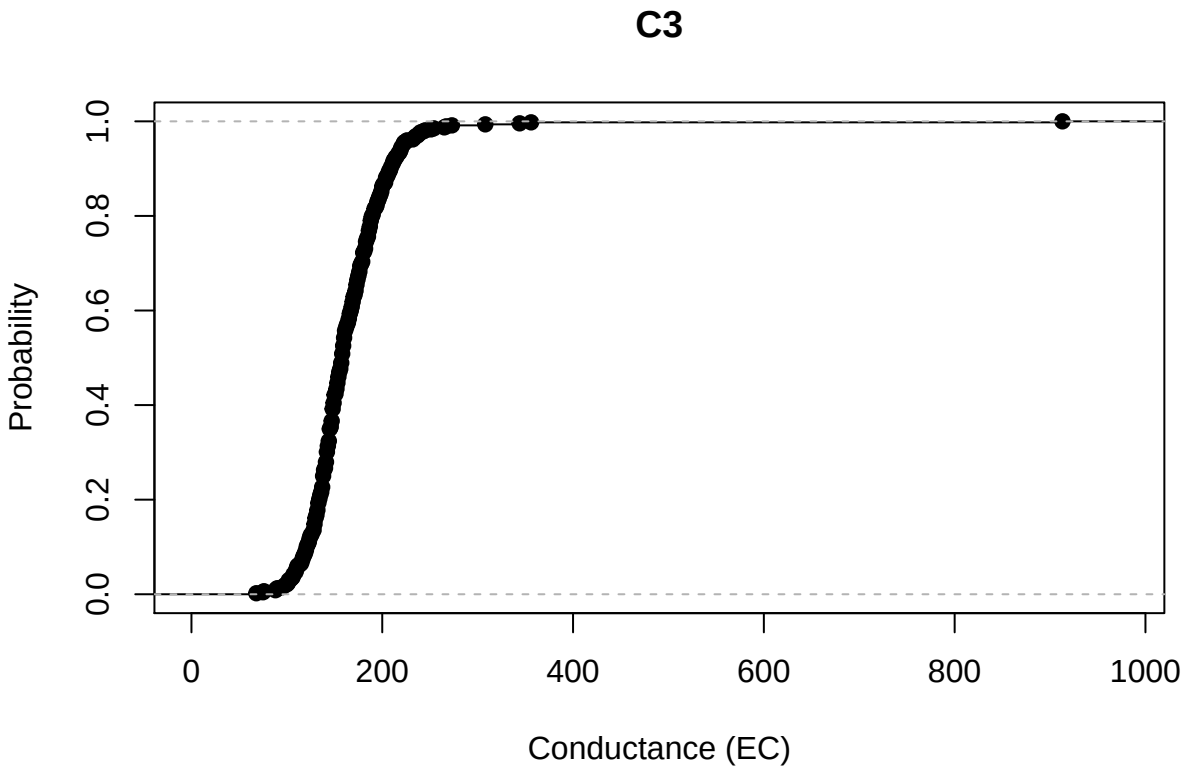

D9
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
5000
10000
15000
20000
Conductance (EC)

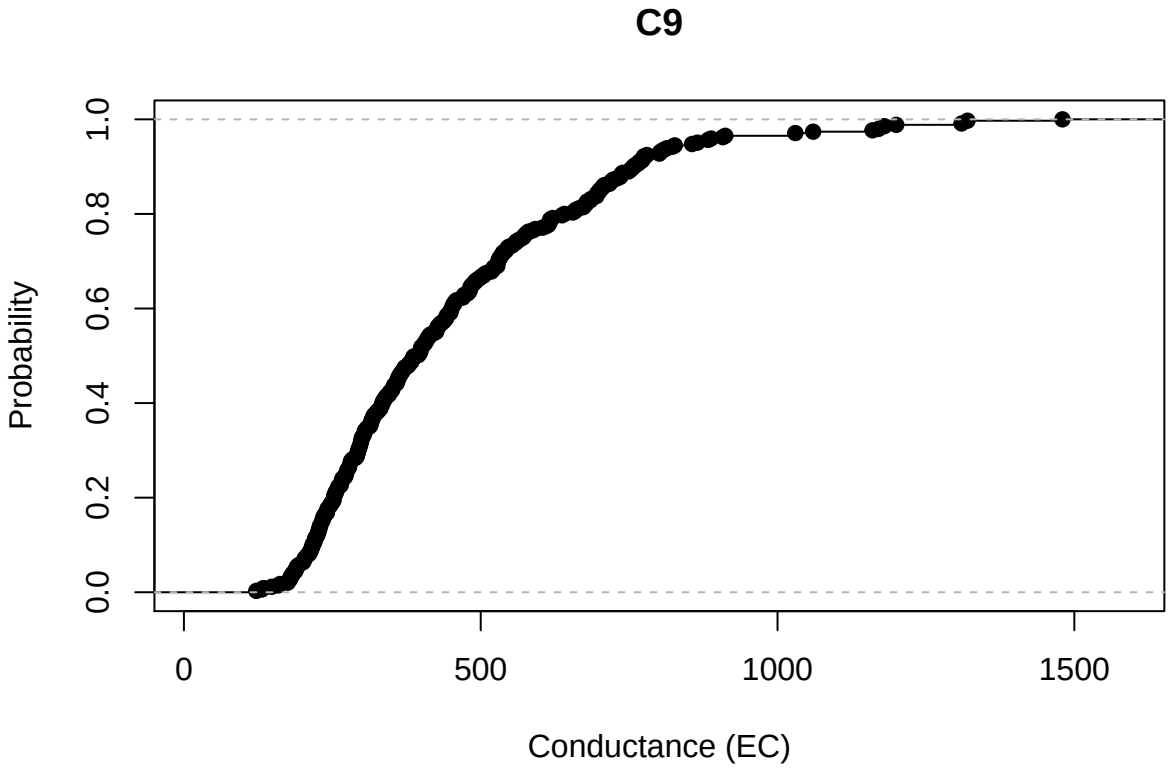

C10
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000
10000
Conductance (EC)

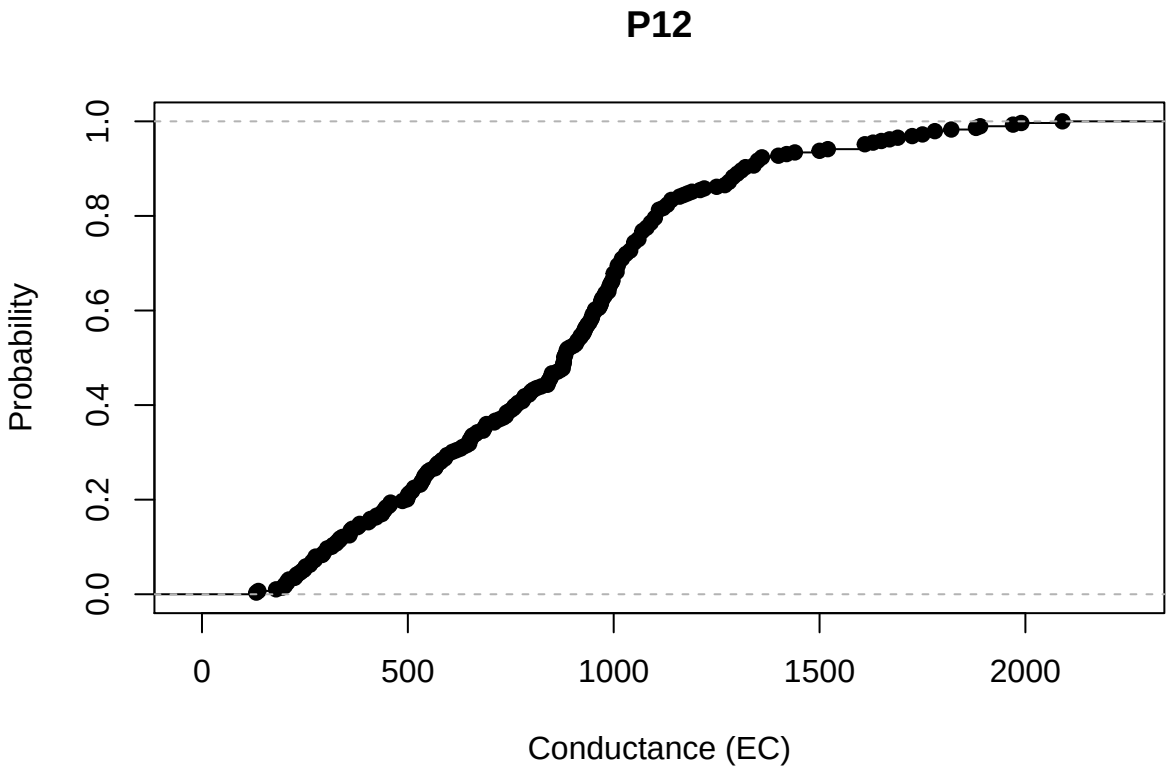

C7
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
Conductance (EC)

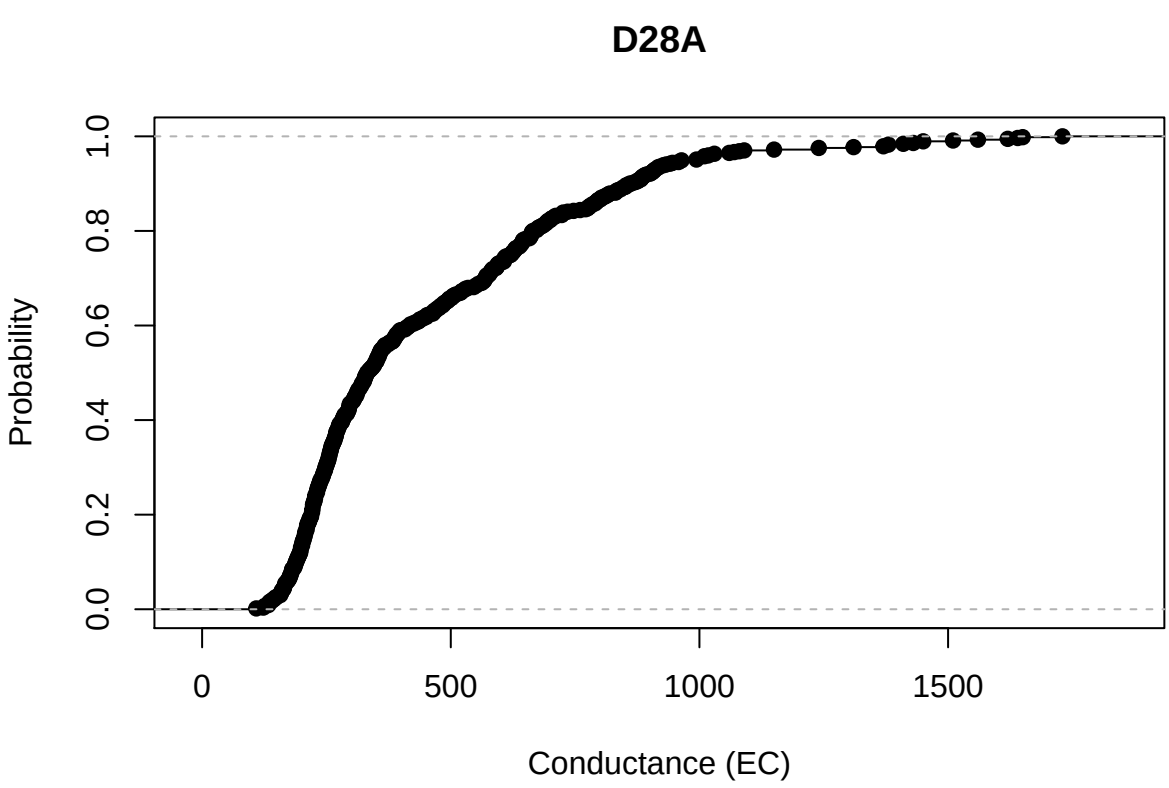

MD10
Probability
0.0
0.2
0.4
0.6
0.8
1.0
100
200
300
400
500
600
Conductance (EC)

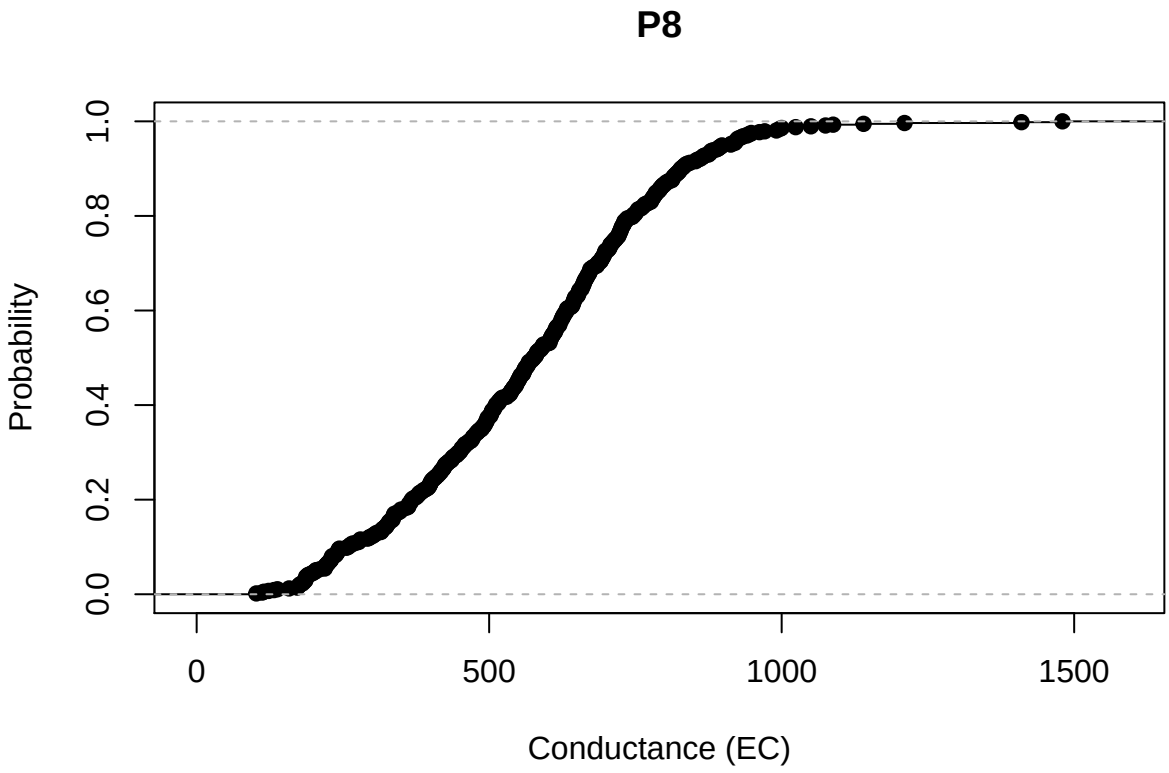

P2
Probability
0.0
0.2
0.4
0.6
0.8
1.0
50
100
150
200
250
300
Conductance (EC)

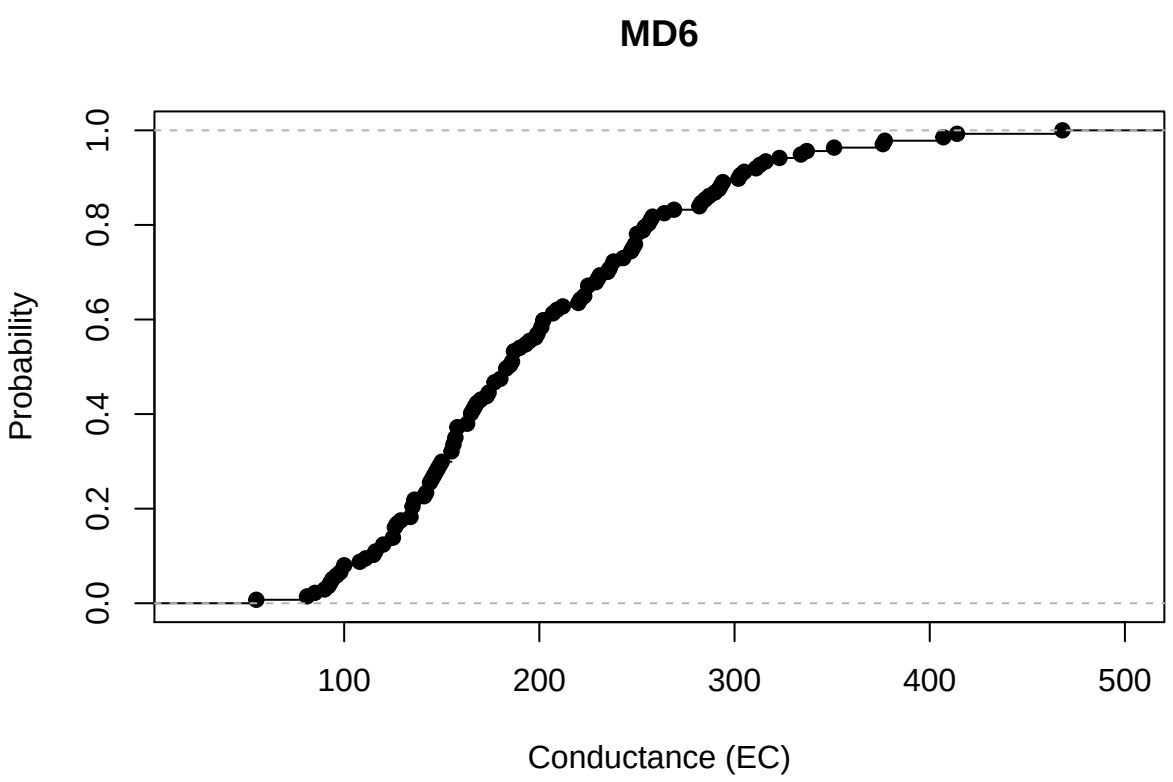

C3
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
Conductance (EC)

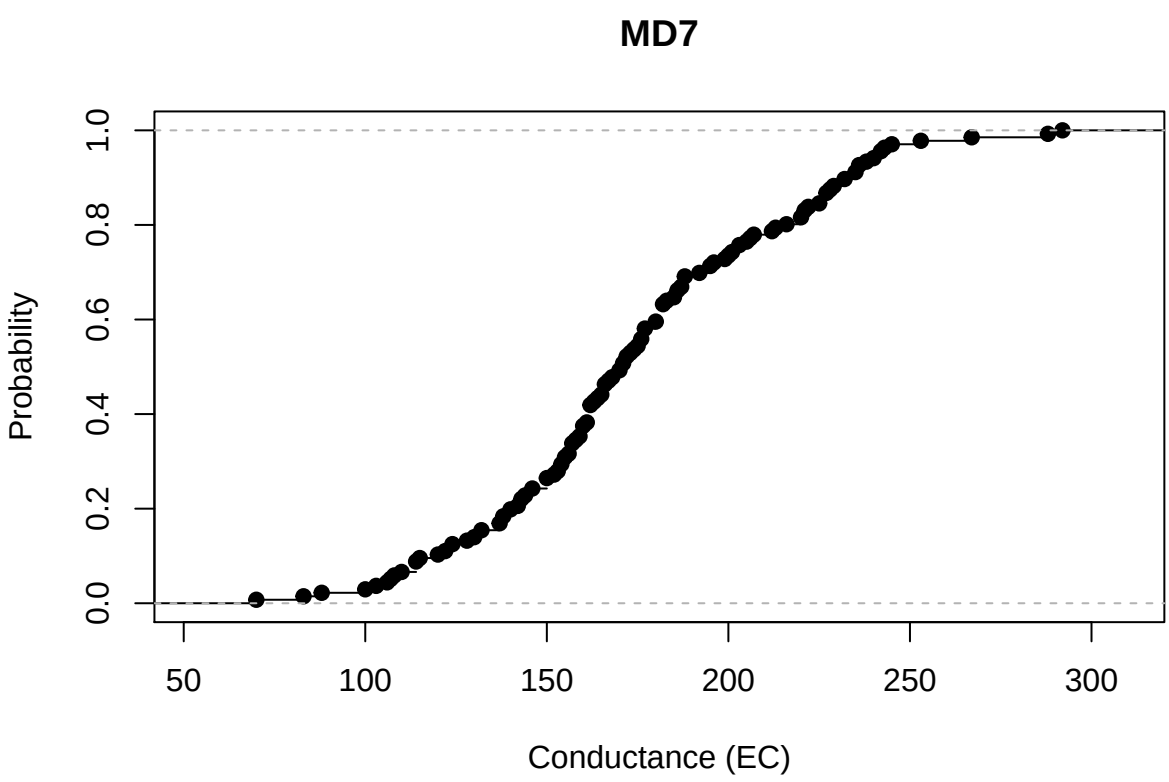

C9
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
Conductance (EC)

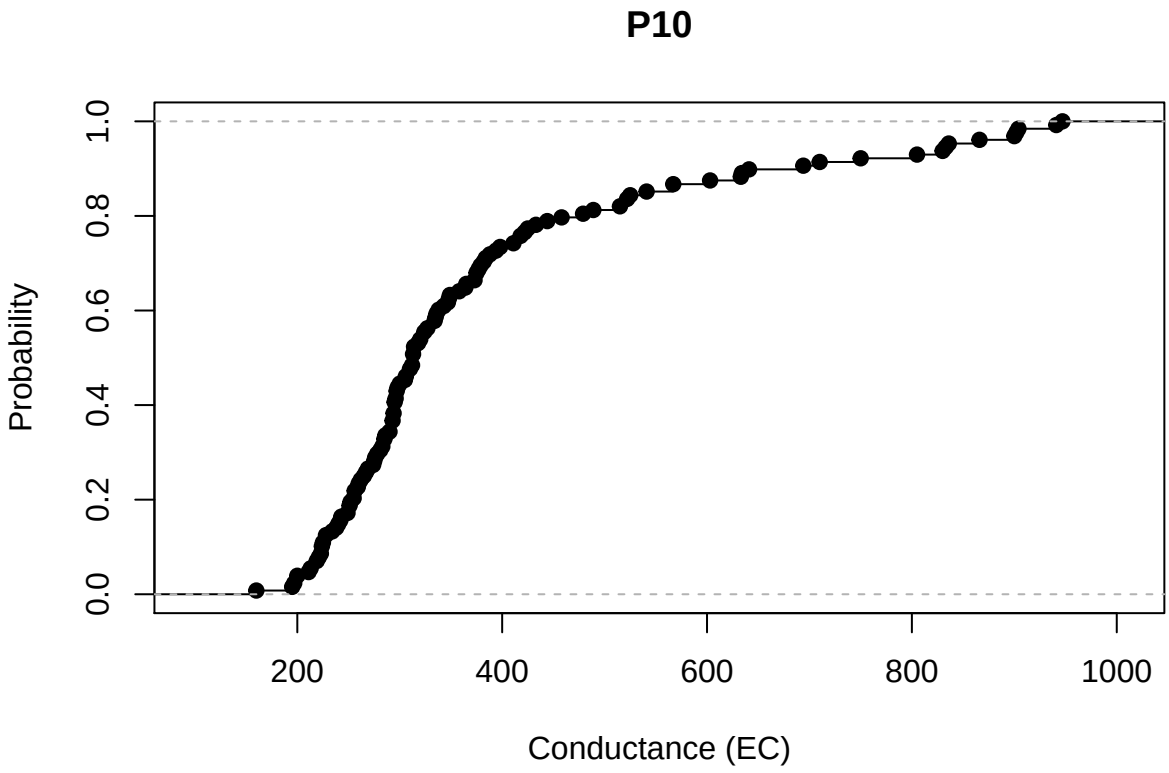

P12
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
Conductance (EC)

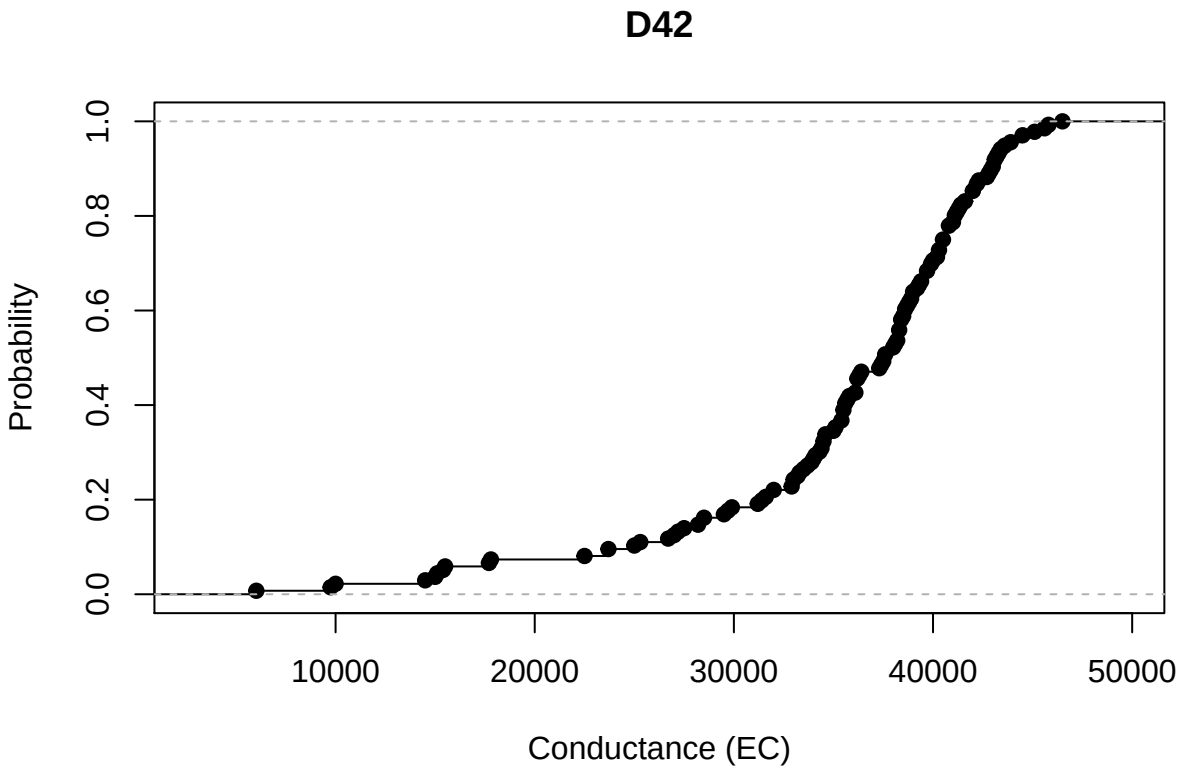

D28A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
Conductance (EC)

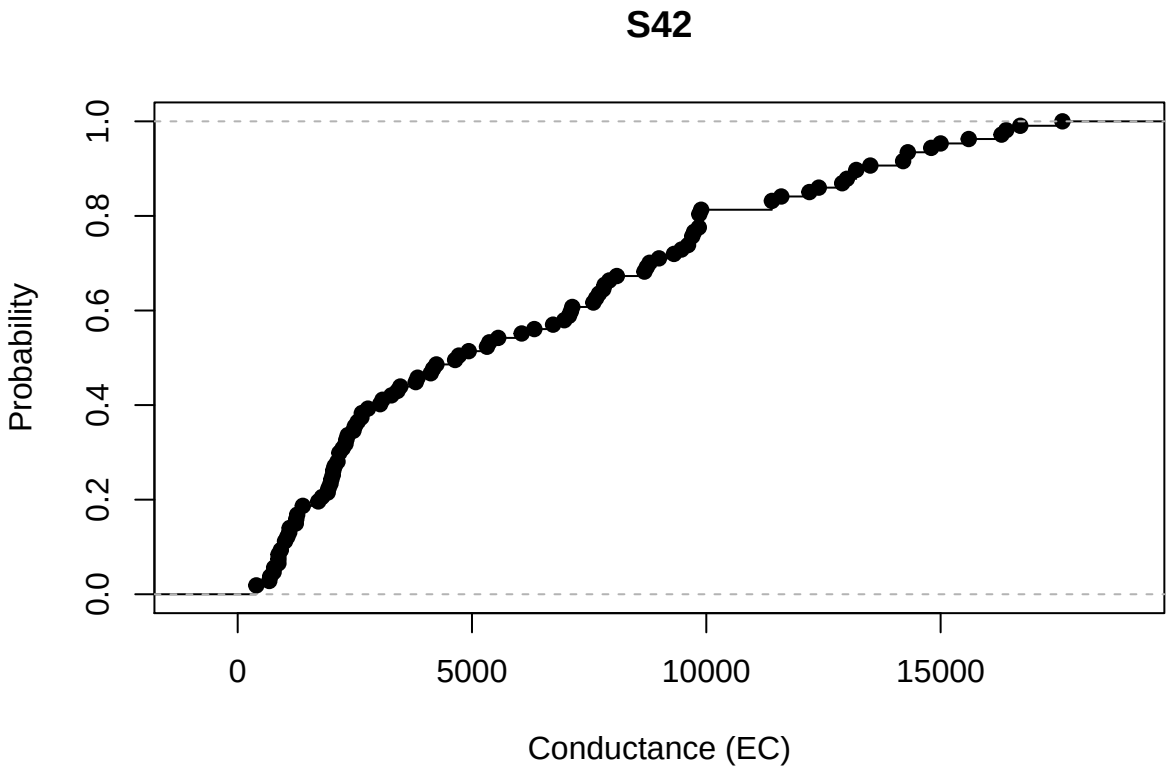

P8
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
Conductance (EC)

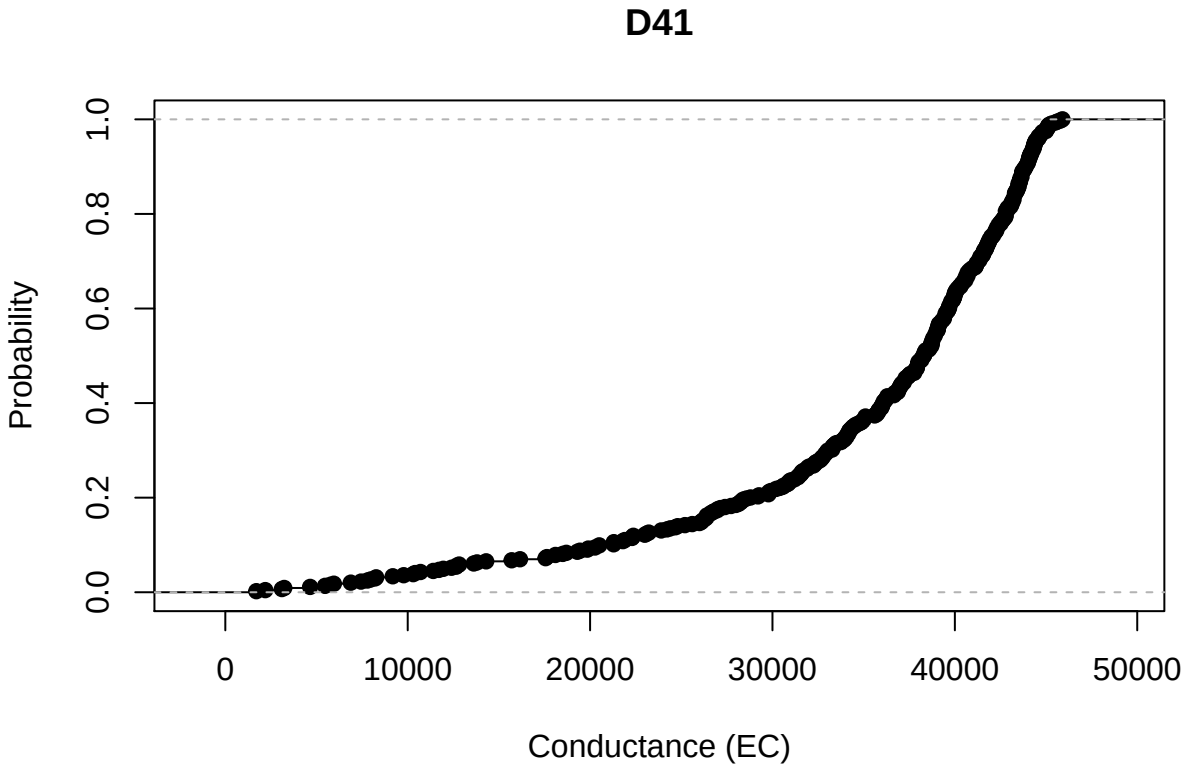

MD6
Probability
0.0
0.2
0.4
0.6
0.8
1.0
100
200
300
400
500
Conductance (EC)

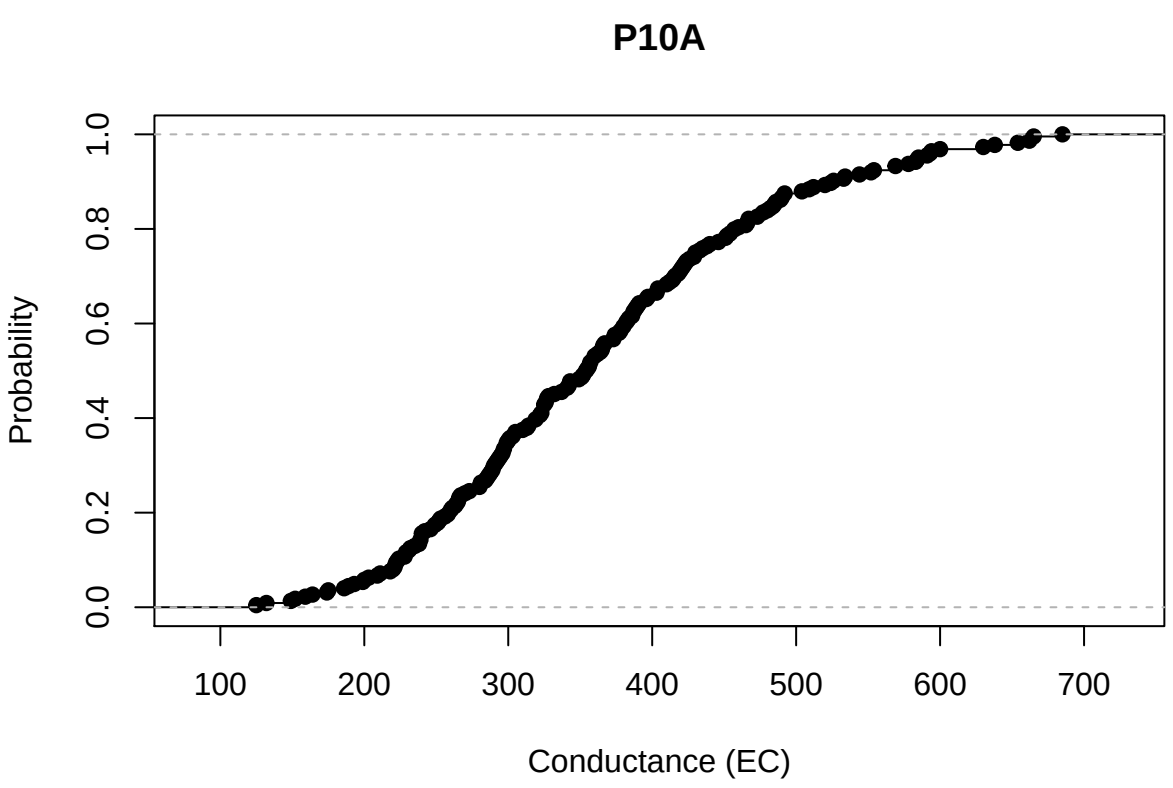

MD7
Probability
0.0
0.2
0.4
0.6
0.8
1.0
50
100
150
200
250
300
Conductance (EC)

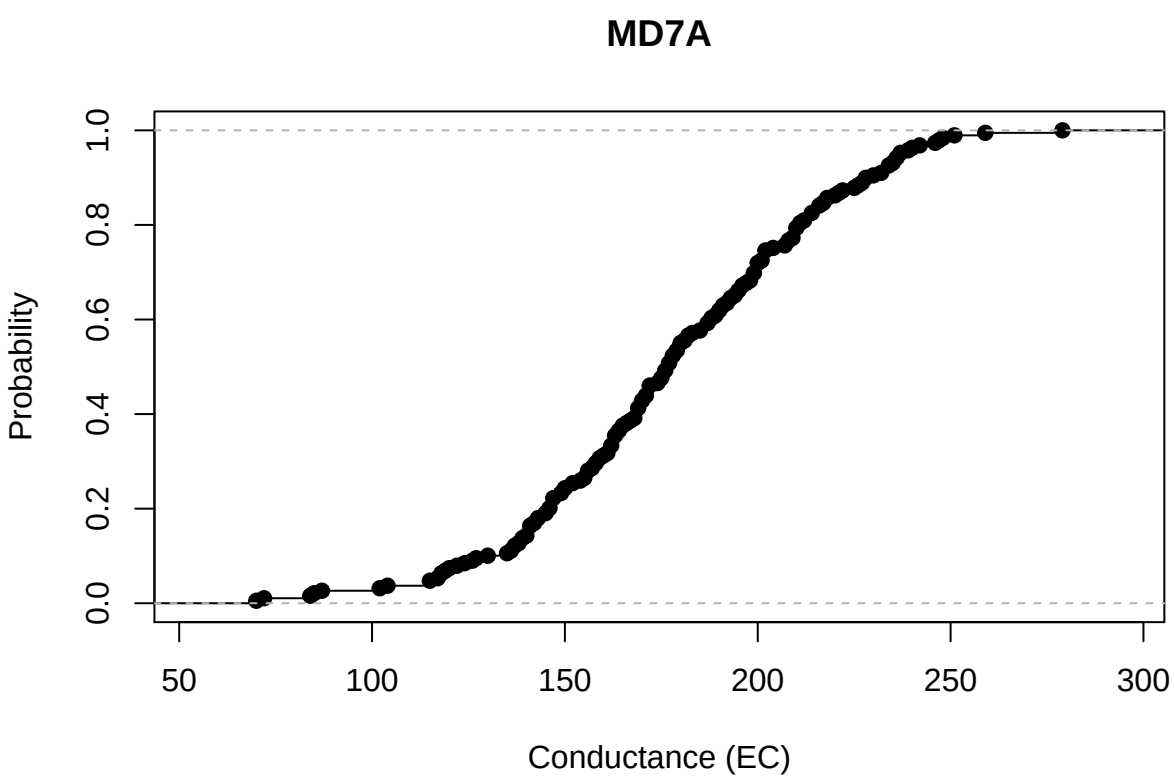

P10
Probability
0.0
0.2
0.4
0.6
0.8
1.0
200
400
600
800
1000
Conductance (EC)

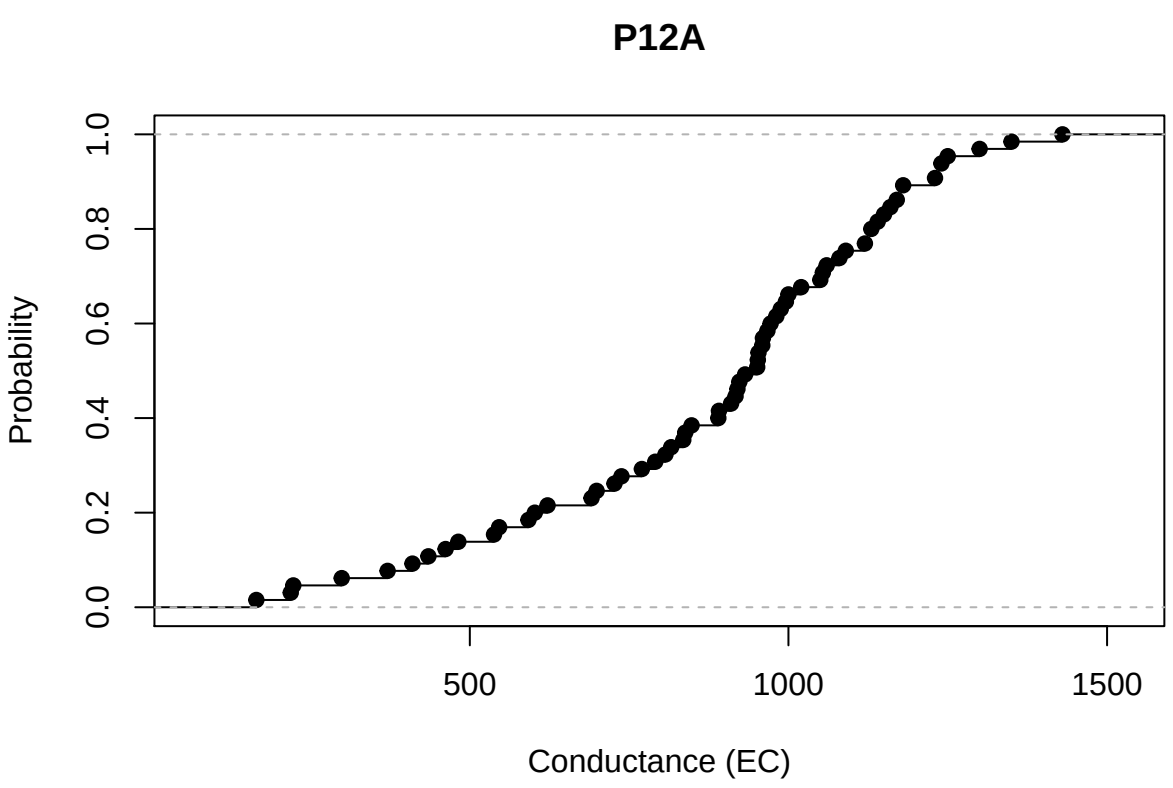

D42
Probability
0.0
0.2
0.4
0.6
0.8
1.0
10000
20000
30000
40000
50000
Conductance (EC)

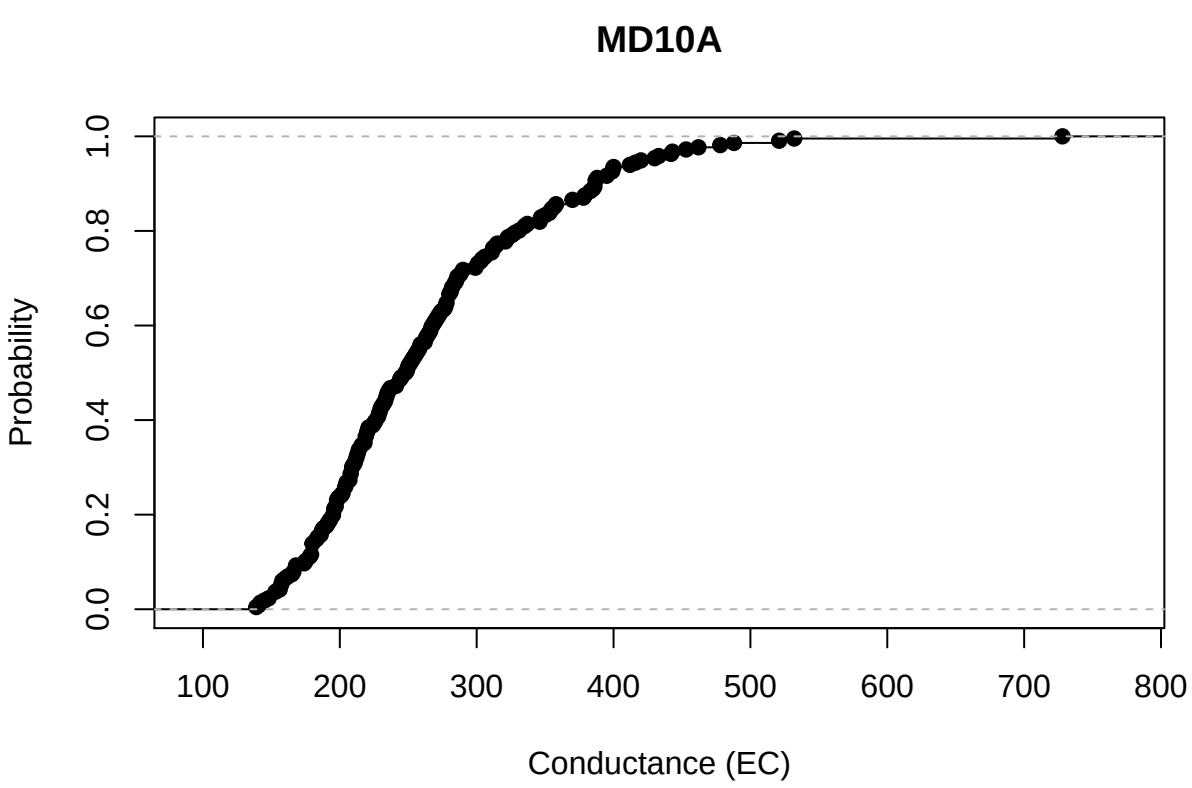

S42
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
5000
10000
15000
Conductance (EC)

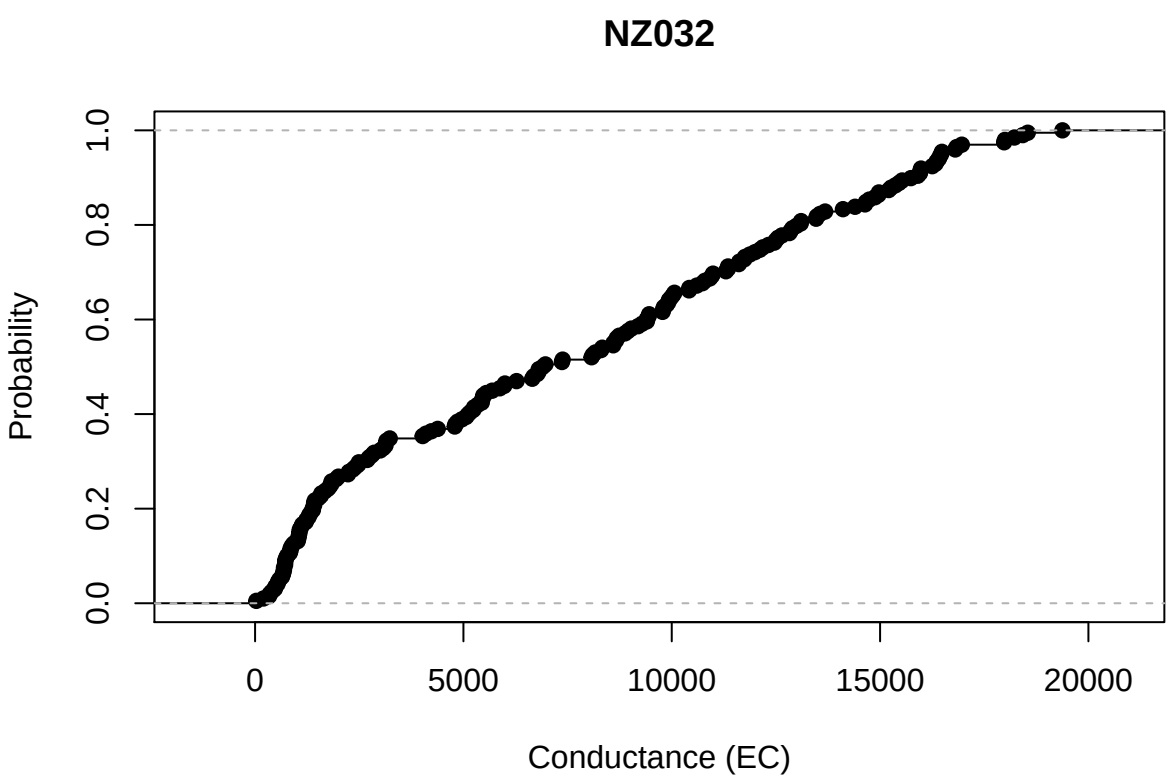

D41
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
30000
40000
50000
Conductance (EC)

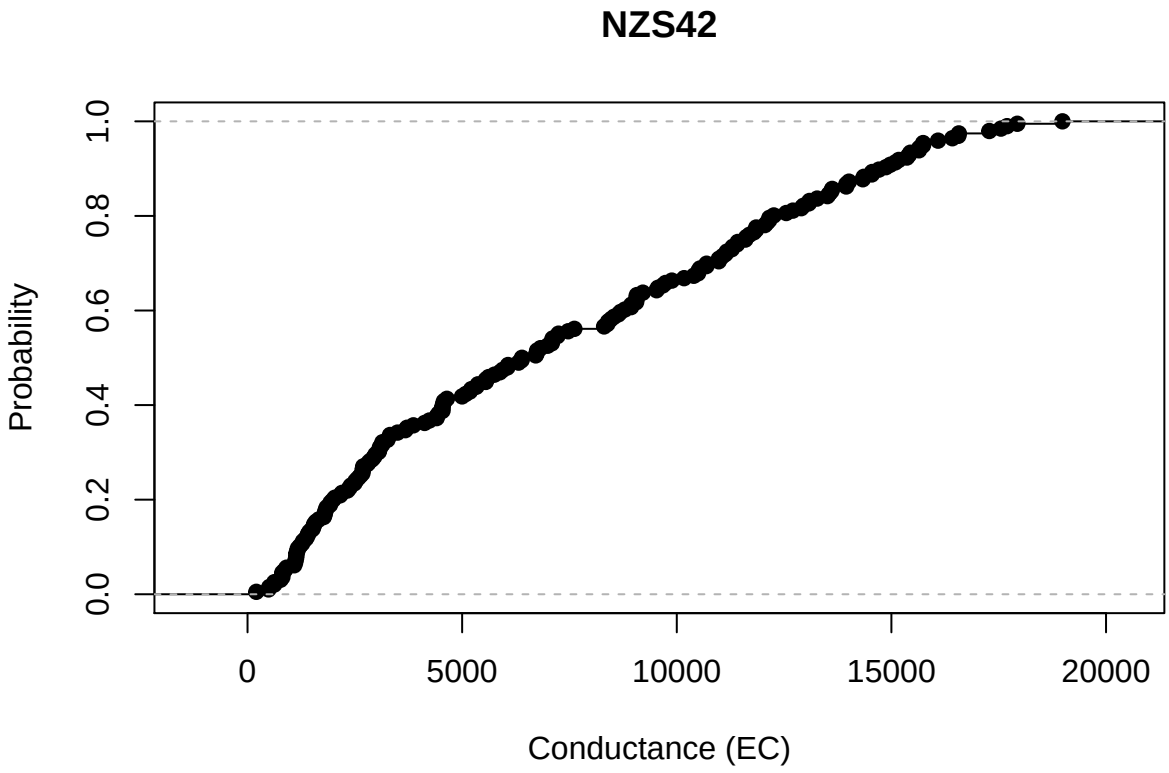

P10A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
100
200
300
400
500
600
700
Conductance (EC)

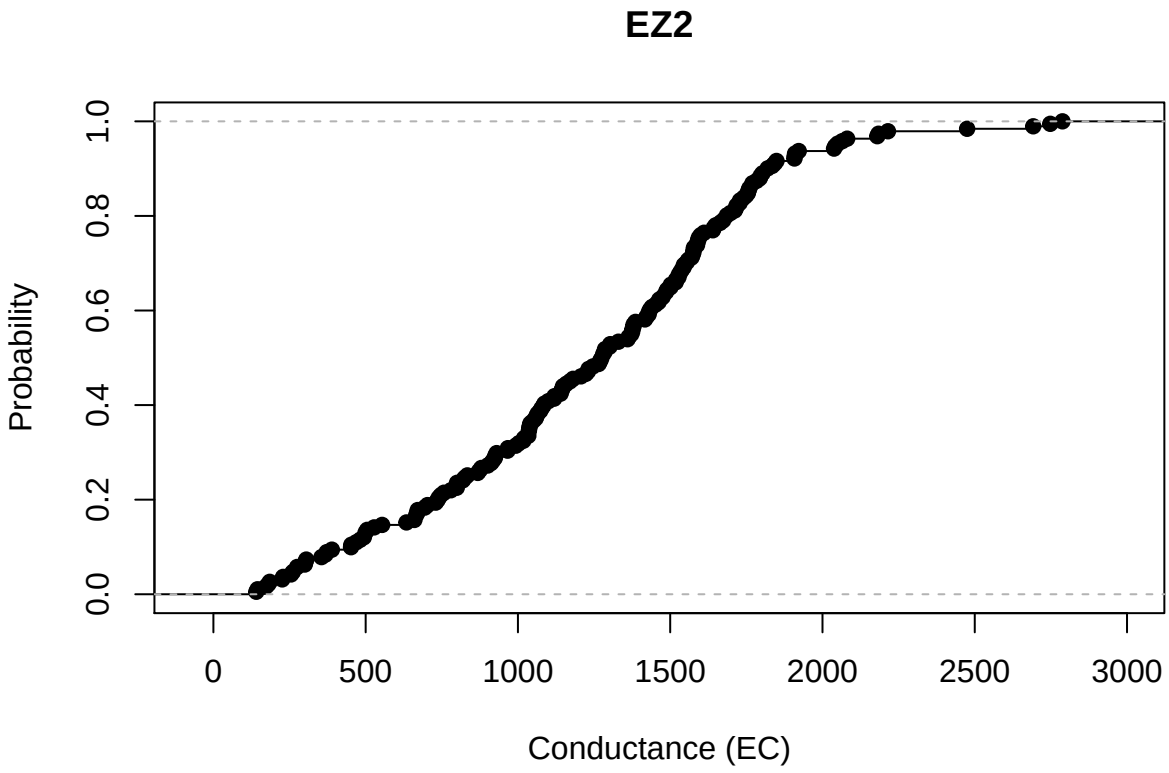

MD7A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
50
100
150
200
250
300
Conductance (EC)

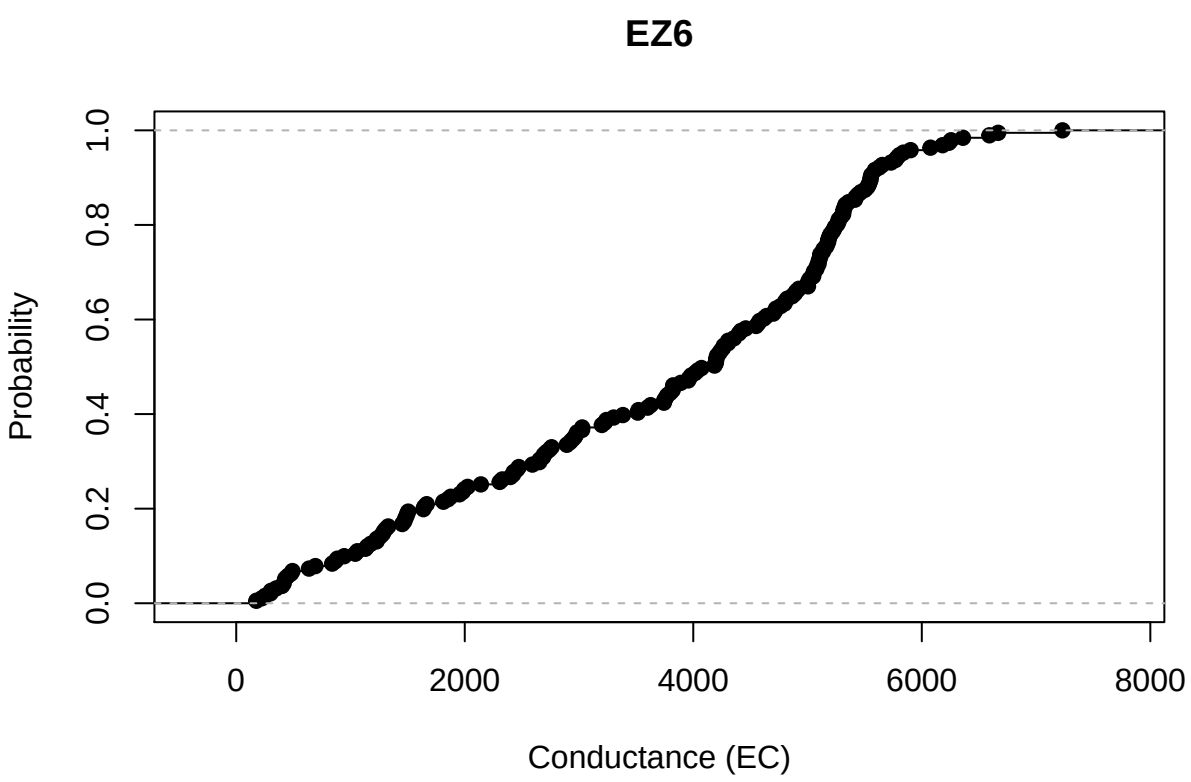

P12A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
500
1000
1500
Conductance (EC)

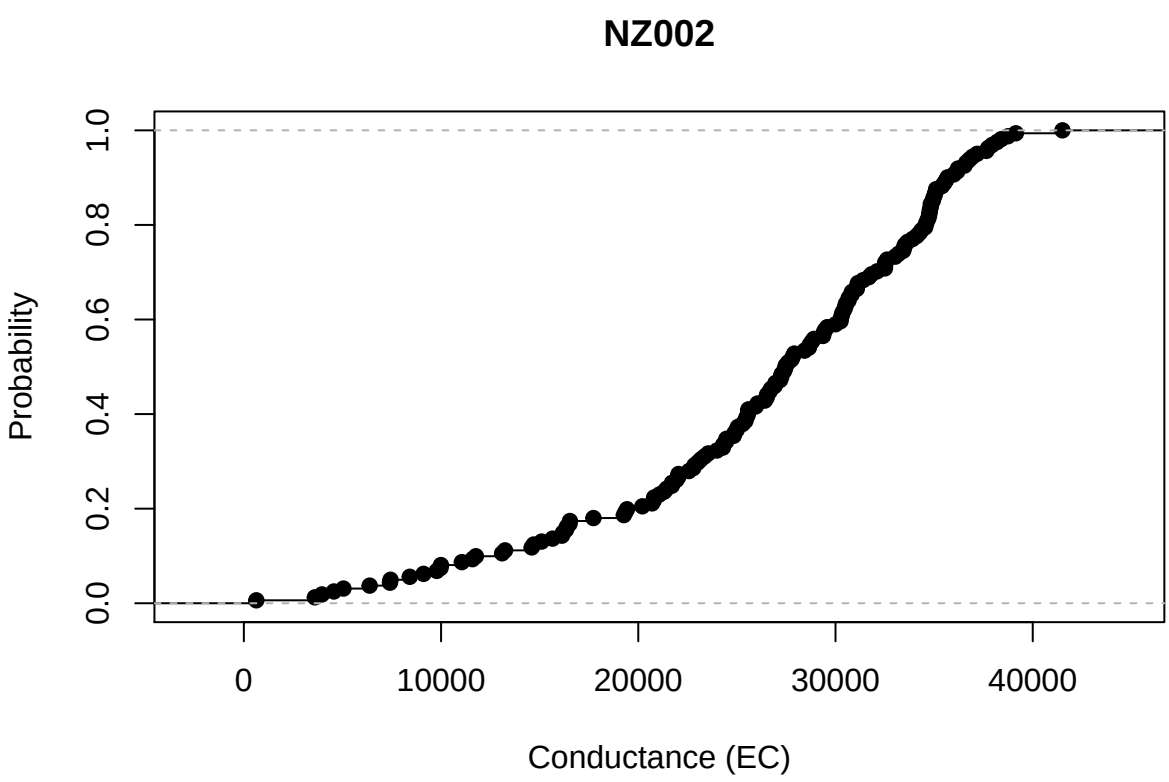

MD10A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
100
200
300
400
500
600
700
800
Conductance (EC)

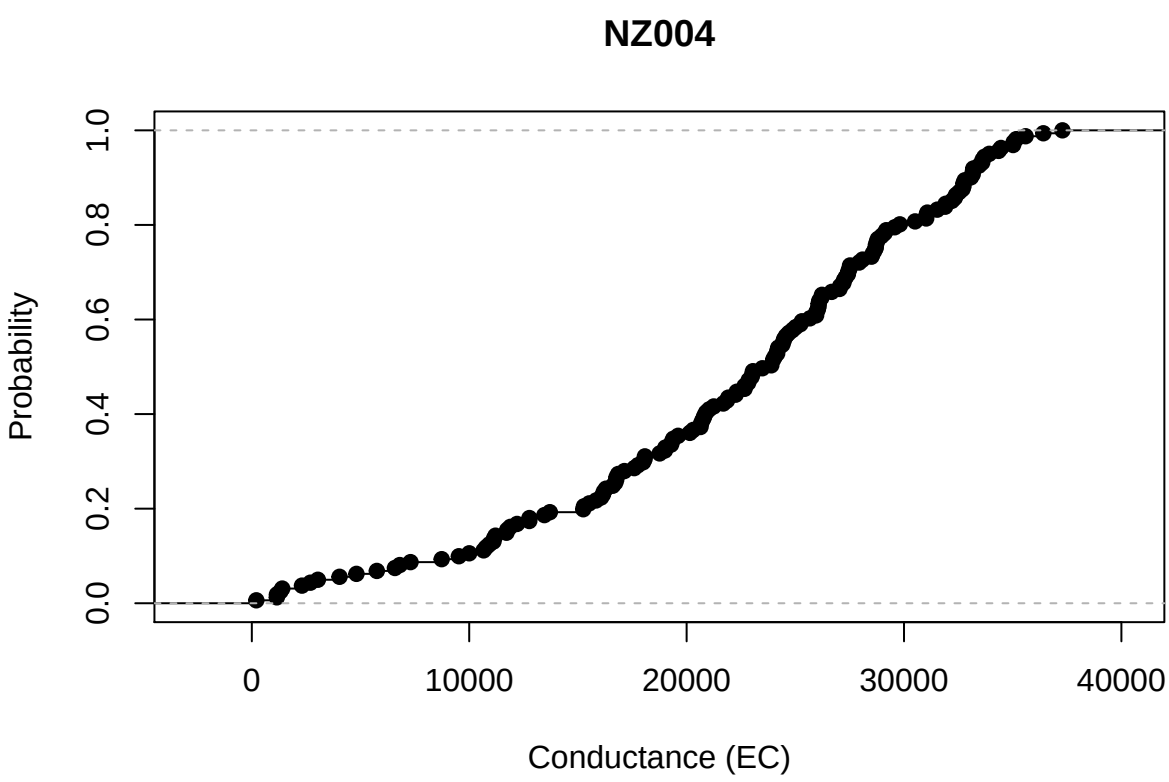

NZ032
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
5000
10000
15000
20000
Conductance (EC)

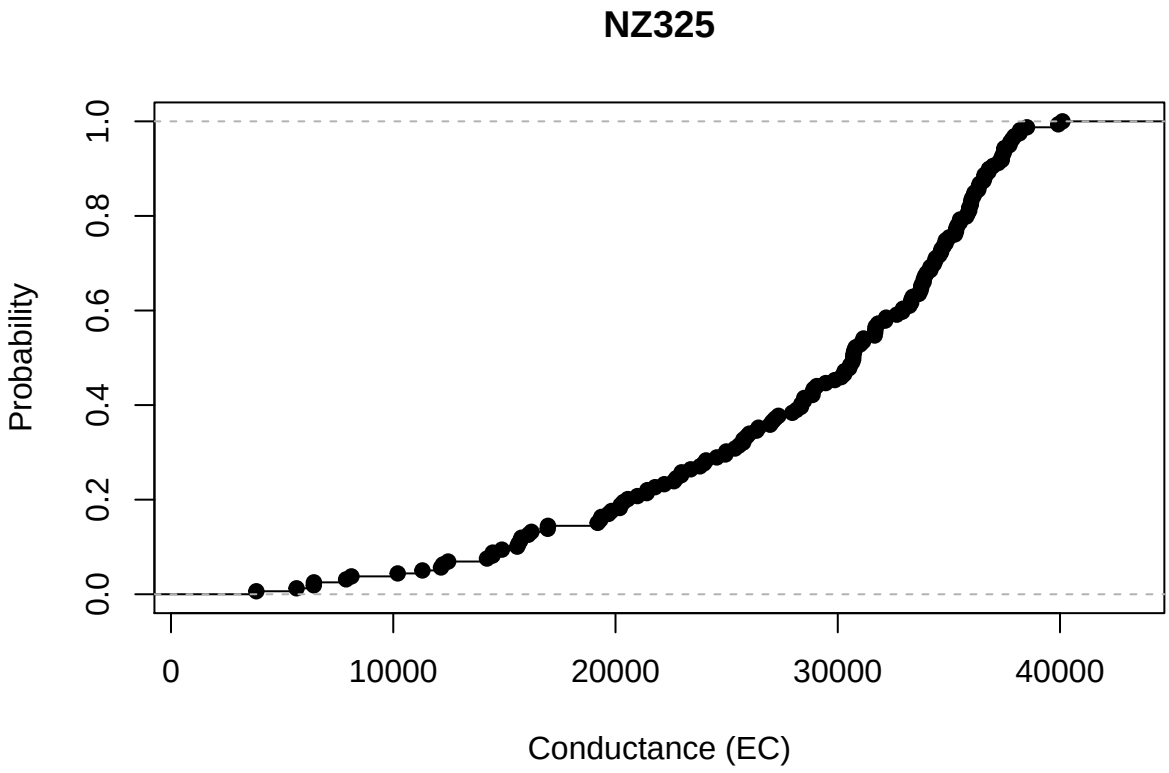

NZS42
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
5000
10000
15000
20000
Conductance (EC)

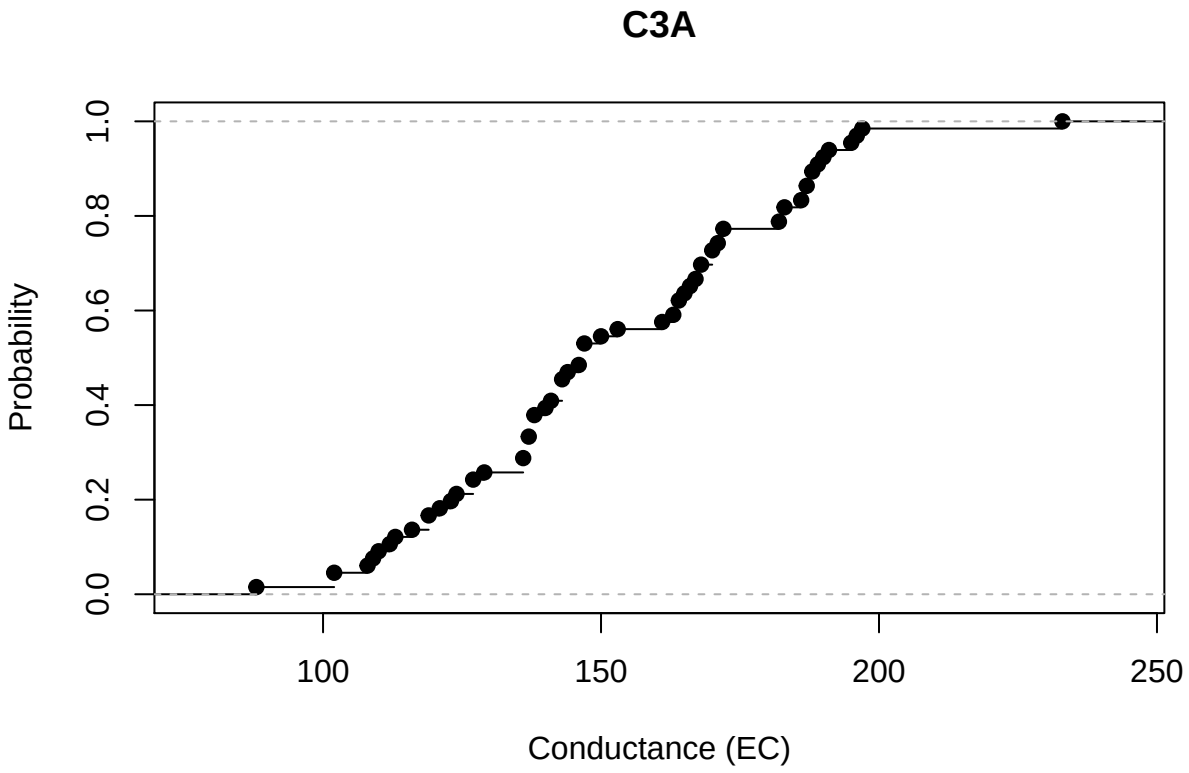

EZ2
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
500
1000
1500
2000
2500
3000
Conductance (EC)

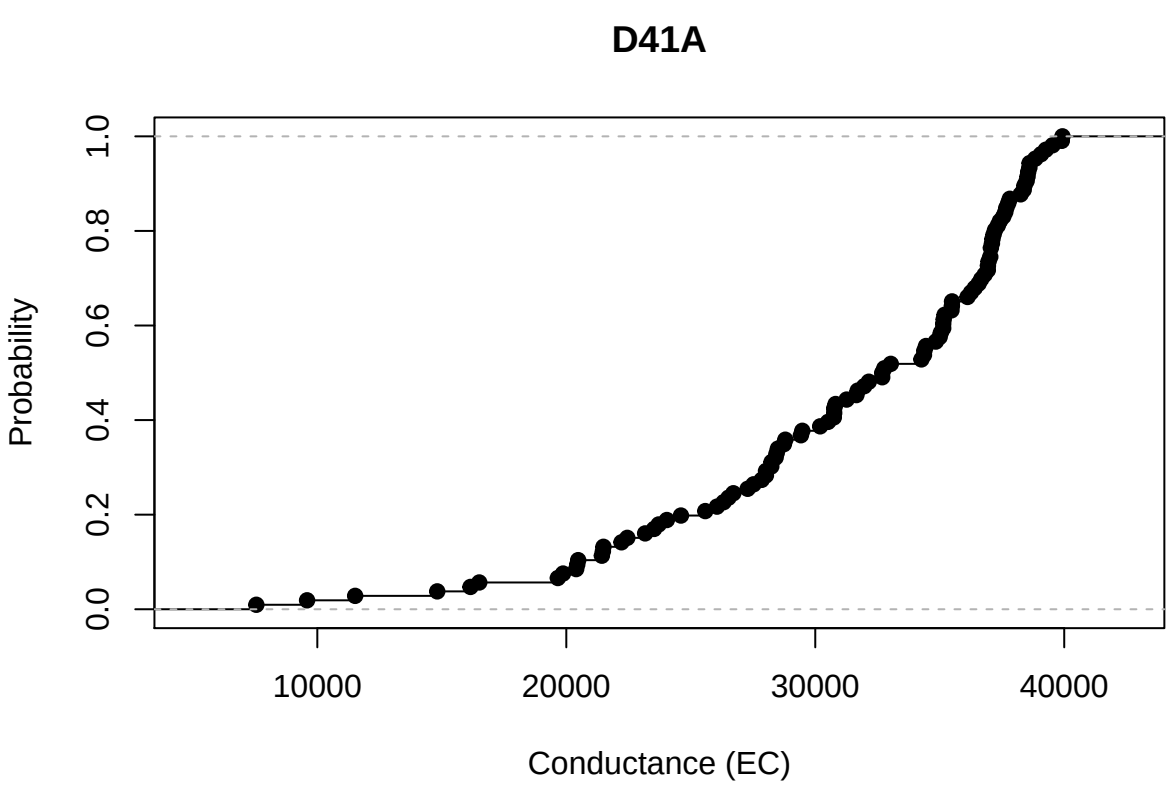

EZ6
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000
Conductance (EC)

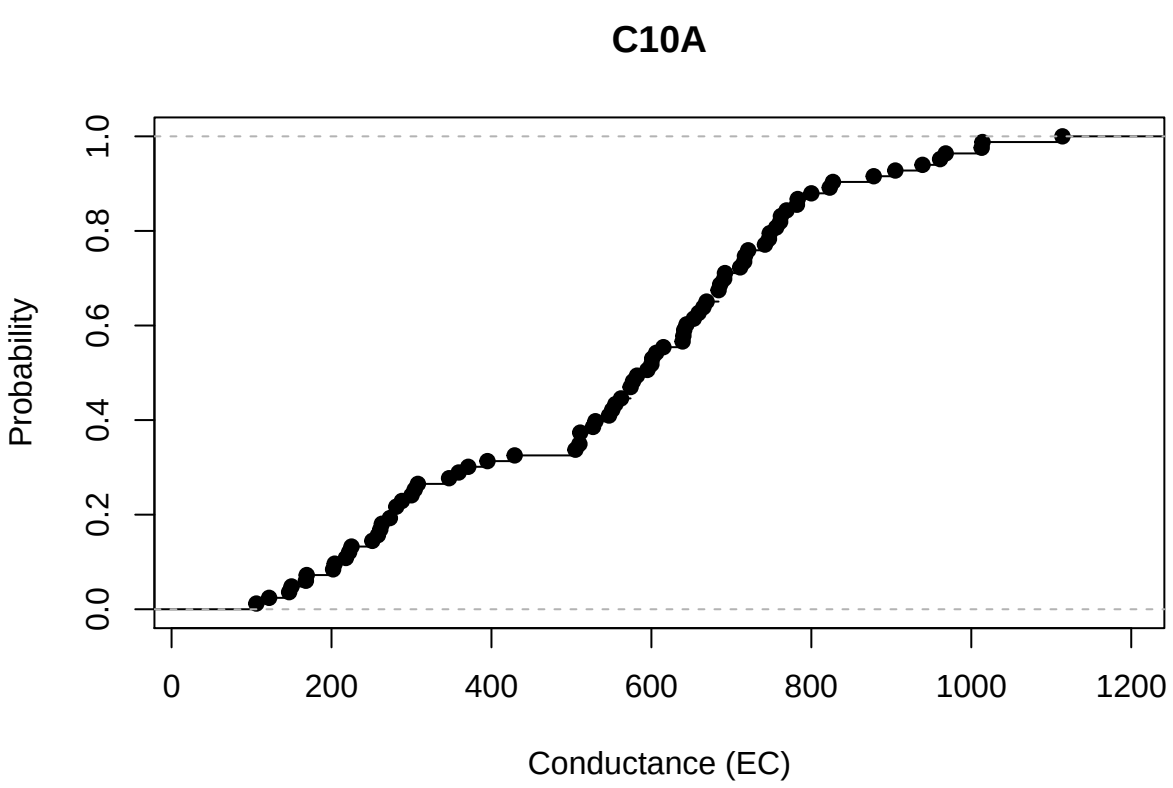

NZ002
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
30000
40000
Conductance (EC)

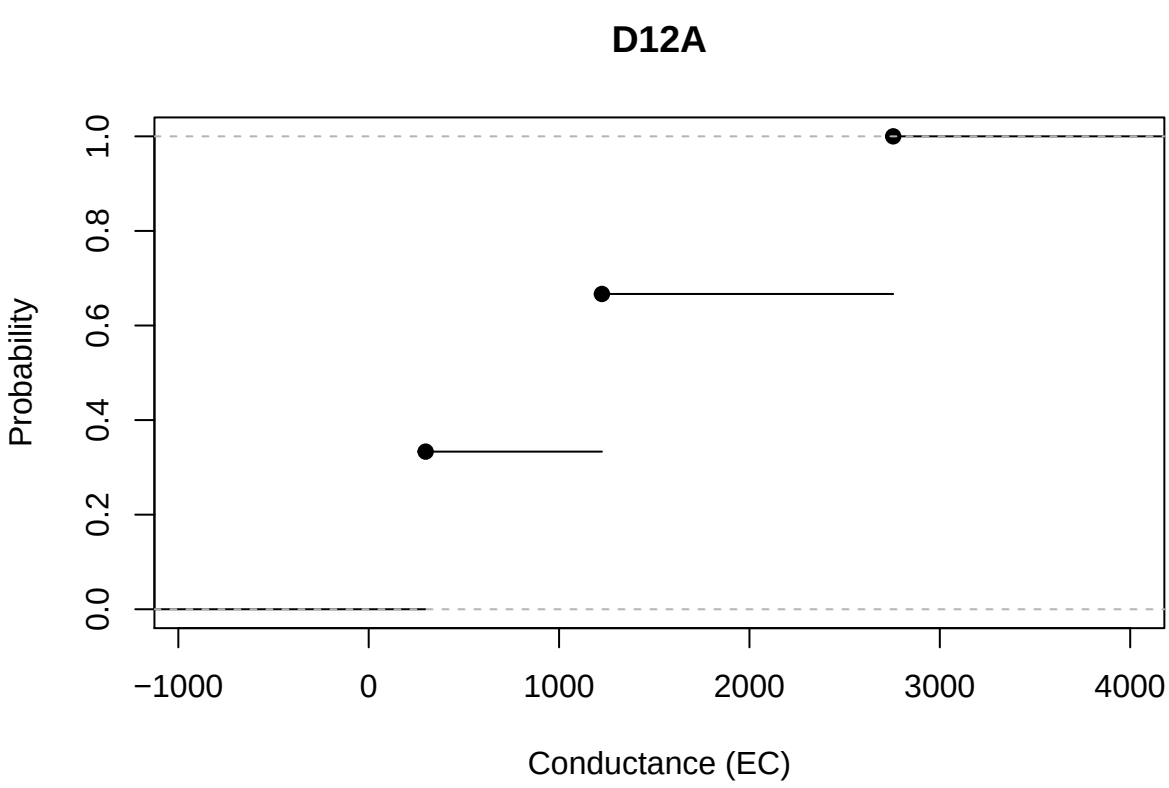

NZ004
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
30000
40000
Conductance (EC)

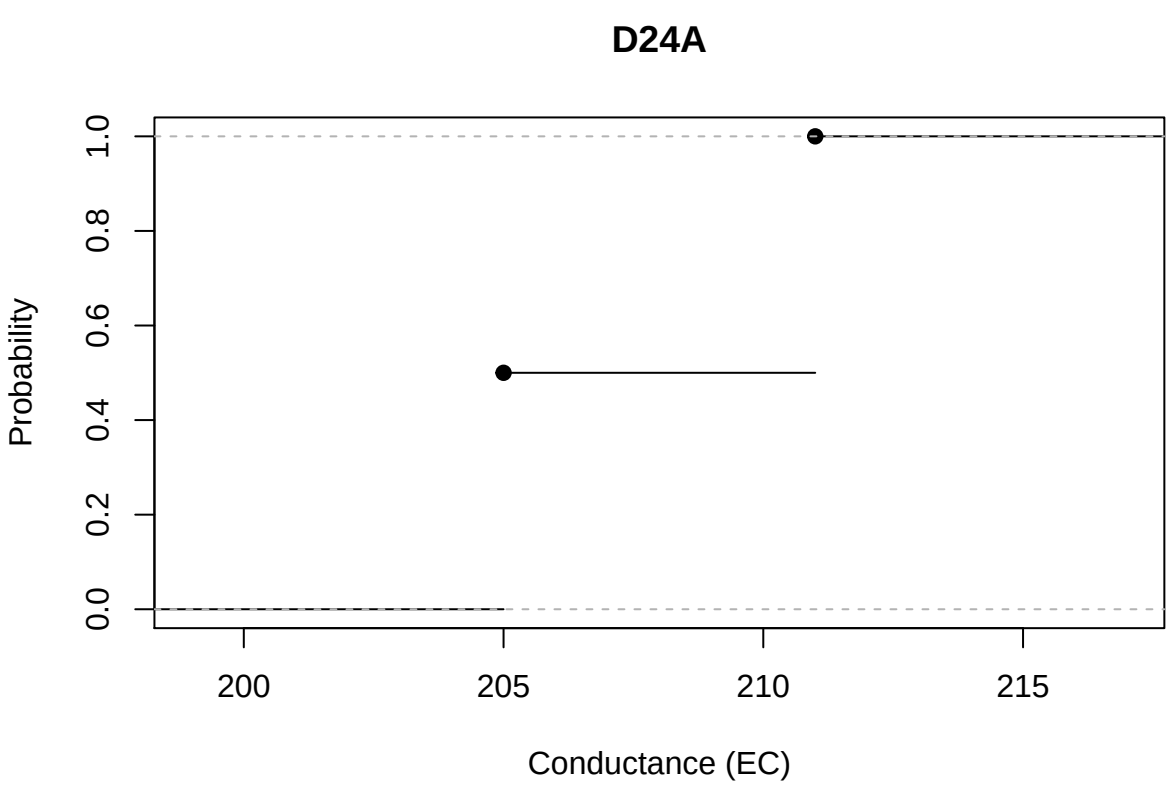

NZ325
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
10000
20000
30000
40000
Conductance (EC)

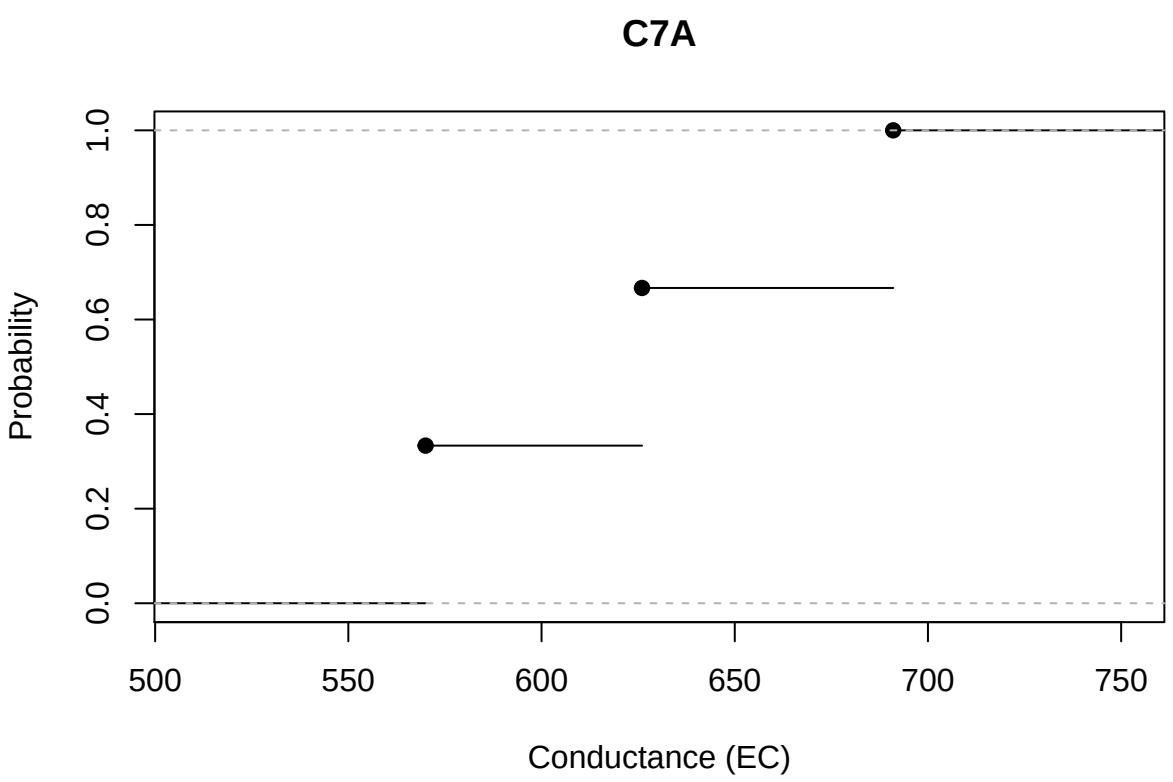

C3A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
100
150
200
250
Conductance (EC)

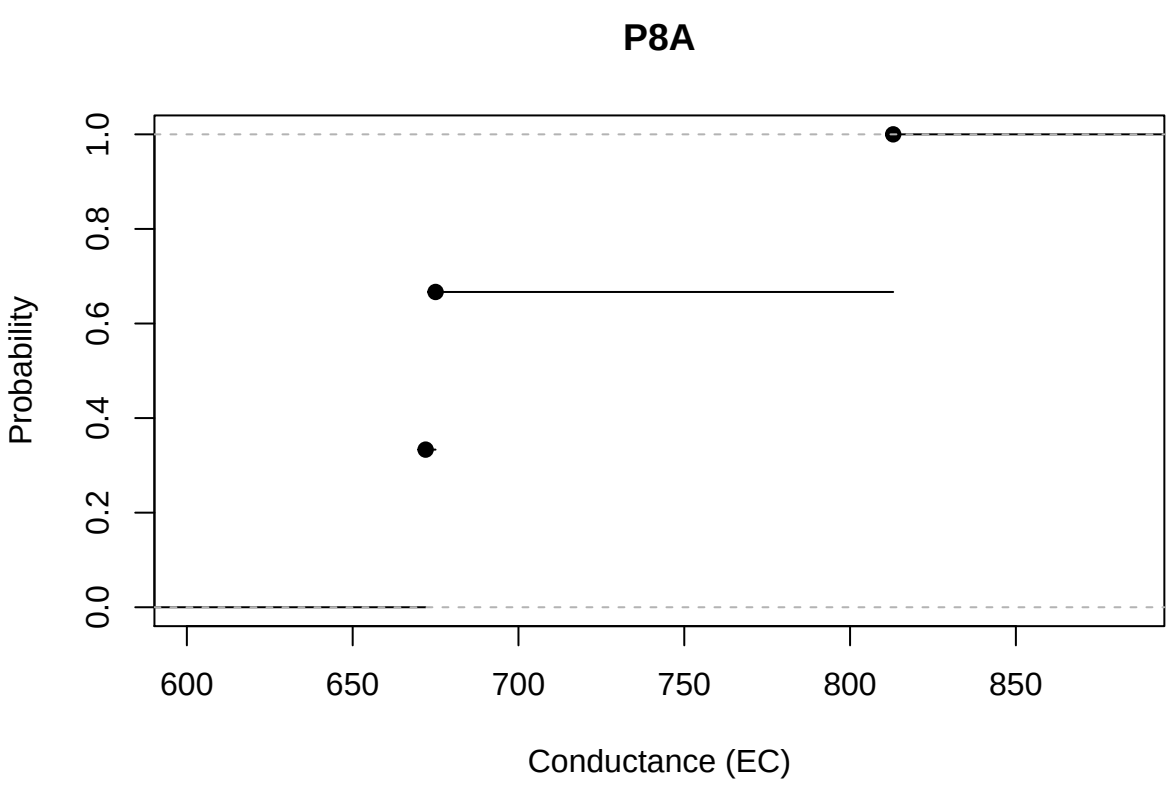

D41A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
10000
20000
30000
40000
Conductance (EC)

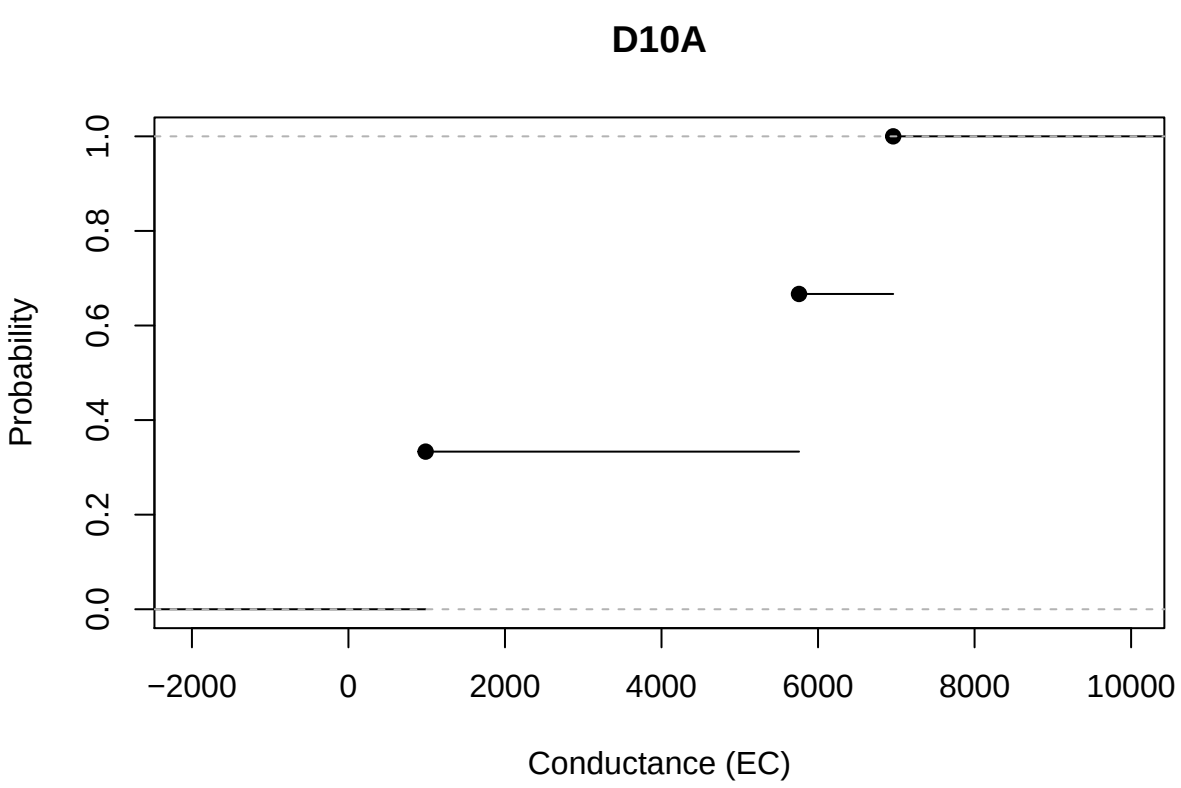

C10A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1000
1200
Conductance (EC)

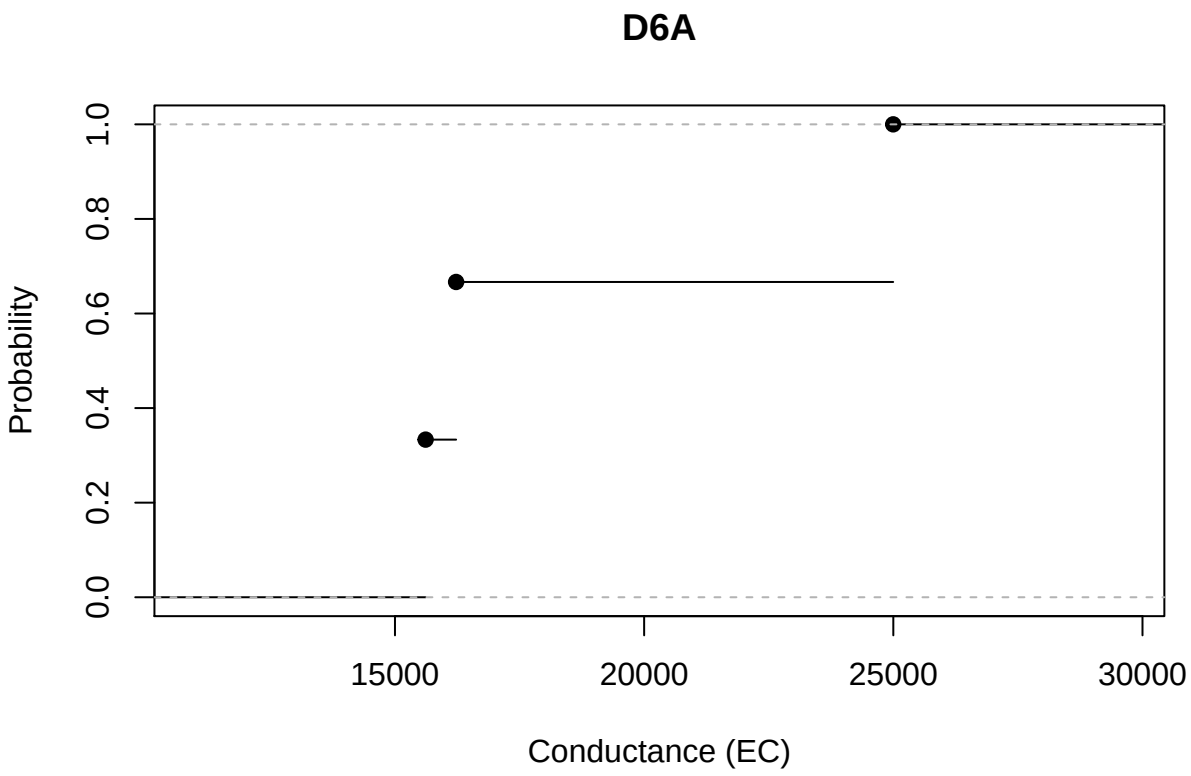

D12A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
−1000
0
1000
2000
3000
4000
Conductance (EC)


D24A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
200
205
210
215
Conductance (EC)


C7A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
500
550
600
650
700
750
Conductance (EC)


P8A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
600
650
700
750
800
850
Conductance (EC)


D10A
Probability
0.0
0.2
0.4
0.6
0.8
1.0
−2000
0
2000
4000
6000
8000
10000
Conductance (EC)

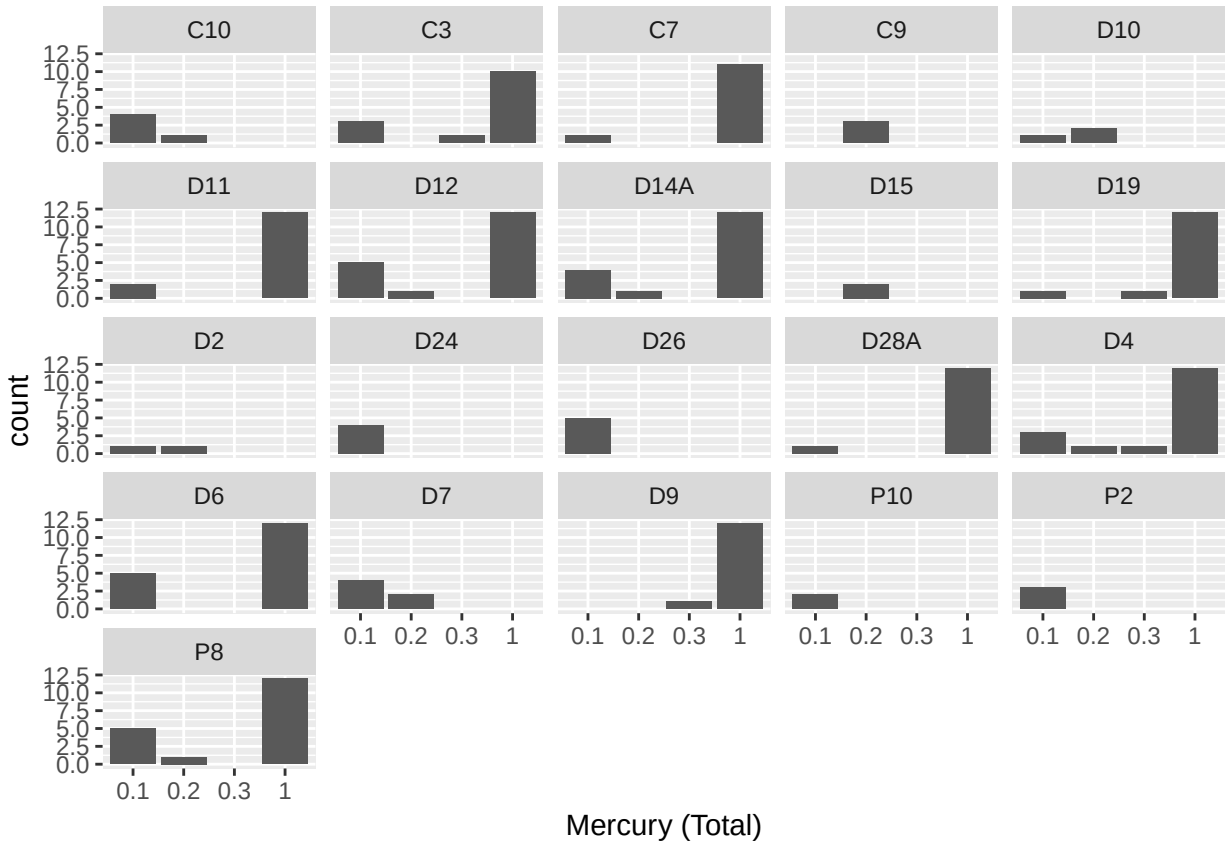


7. Compare the mercury concentrations from the station with the highest concentration and the lowest concentration using a Q-Q plot. Also plot two other types of graphs. Describe the similarities and differences in Mercury. What characteristics are evidet in each graph?

If we look at a frequence plot of all Mercury data we can see that there is no station with the highest [hg] or lowest [hg]. From these plots we can see two things, 1) There is no single station or pair of stations to select 2) Excluding NA's, there are only 4 values avaliable for mercury concentration: 0.1,0.2,0.3,1.0. Several of the stations, such as P10 only have one value of mercury recorded, which is problematic seeing as a QQPlot is used to check the validity of a distribution (normal QQ for normal distibution) assumption for a given dataset . Therefore, in cases were there is only one value for a given station the QQplot would confirm that the data is not normally distributed. When you look at the QQplot it somewhat looks bi-modal with some very high values and some very low values spanning across the whole dataset. Additionally, you may want to group the stations by some other variable to get a better understand of what is going on. Perhaps location or year.

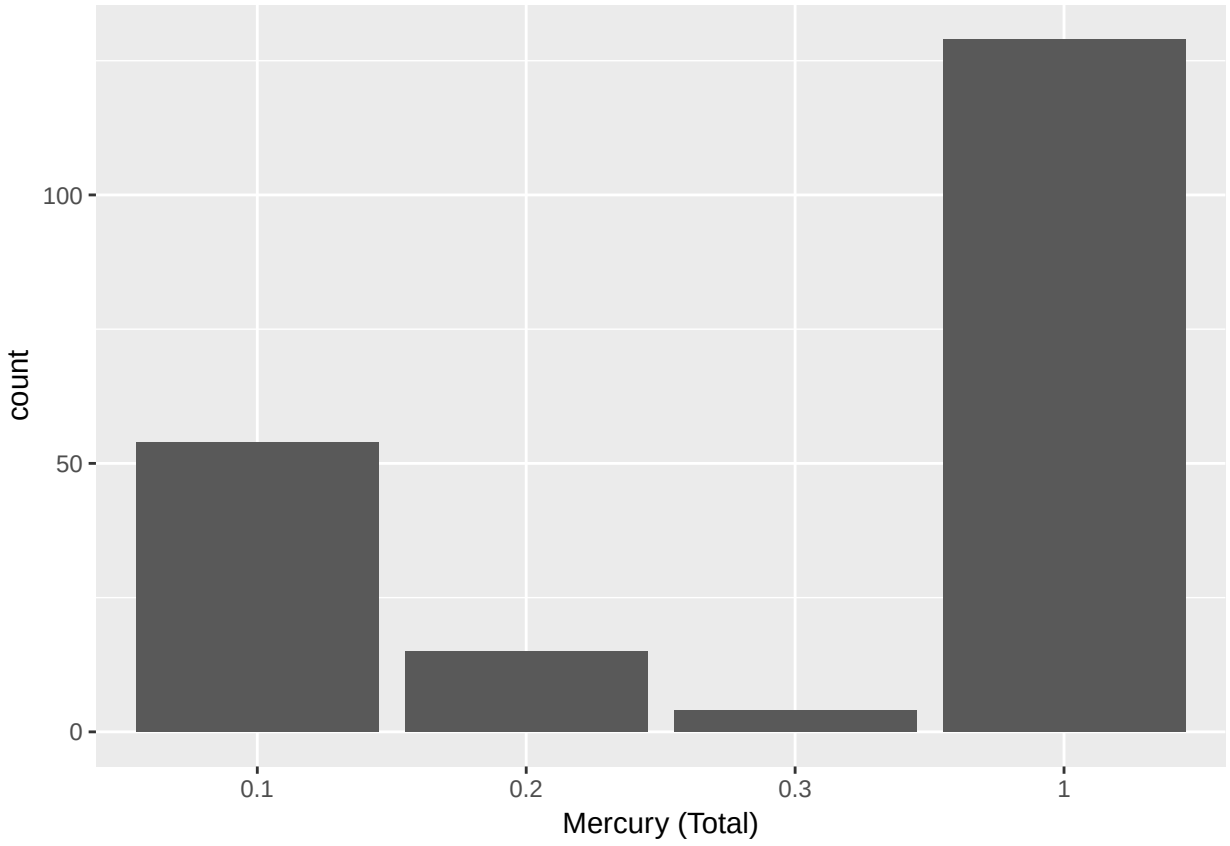
```
## Historgram of all the stations together
# not  very effective to look at.
wQ %>%
  select(`Mercury (Total)`,StationCode) %>%
  na.omit() %>%
  ggplot(aes(`Mercury (Total)`)) +
    geom_histogram(stat ="count")+
      facet_wrap(~StationCode)
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```
# histrogram all data
wQ %>%
  select(`Mercury (Total)`,StationCode) %>%
  na.omit() %>%
  ggplot(aes(`Mercury (Total)`)) +
    geom_histogram(stat ="count")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



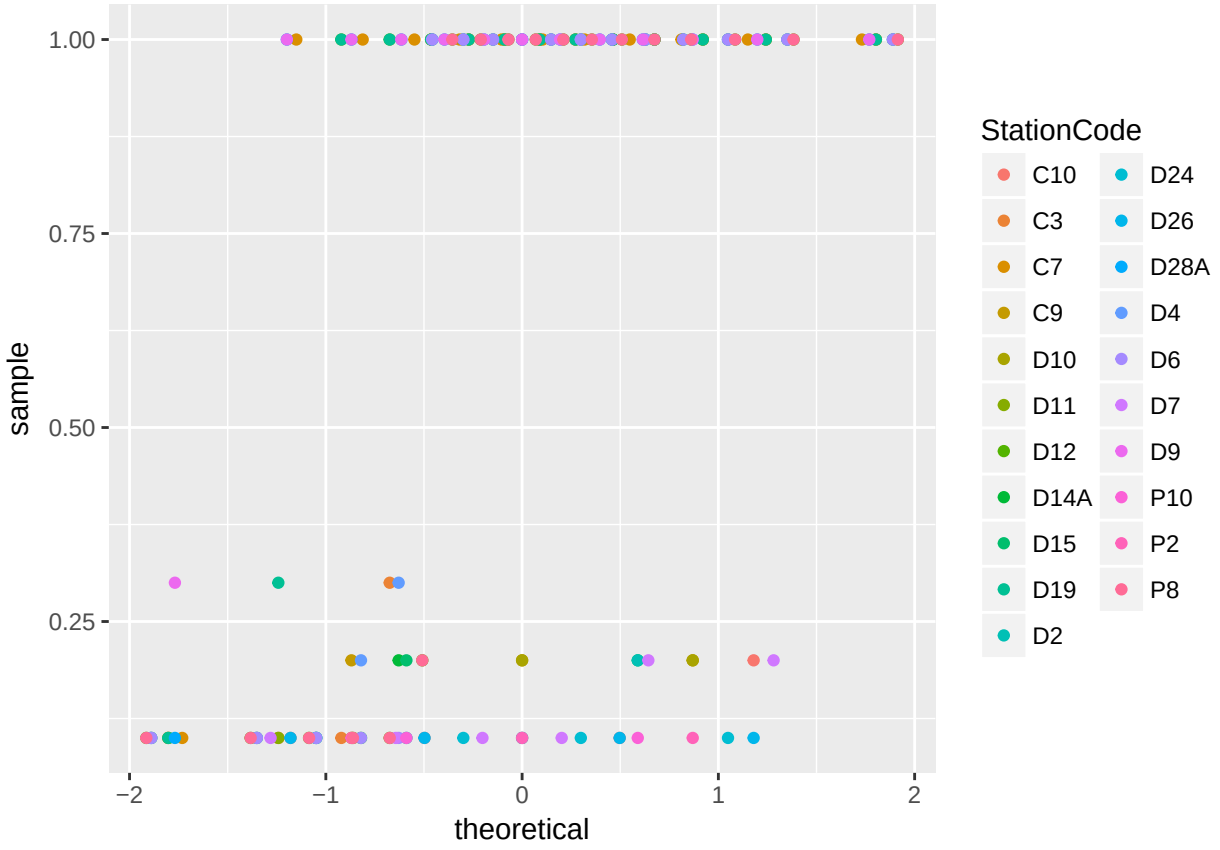

```
# Verify that some stations only have one value
wQ %>%
  filter(StationCode == "P10") %>%
  select(`Mercury (Total)`) %>%
  unique()
```

```
## # A tibble: 2 x 1
##   `Mercury (Total)`
##   <chr>
## 1 <NA>
## 2 0.1
```

Normally when there is somewhat of a break in the middle of the QQplot one would say that the data are bi-modal. how meaningful is that here?

```
## QQplot of all the stations
wQ %>%
  select(`Mercury (Total)`,StationCode) %>%
  ggplot(mapping = aes(sample = as.numeric(`Mercury (Total)`), col = StationCode)) +
  stat_qq()
```

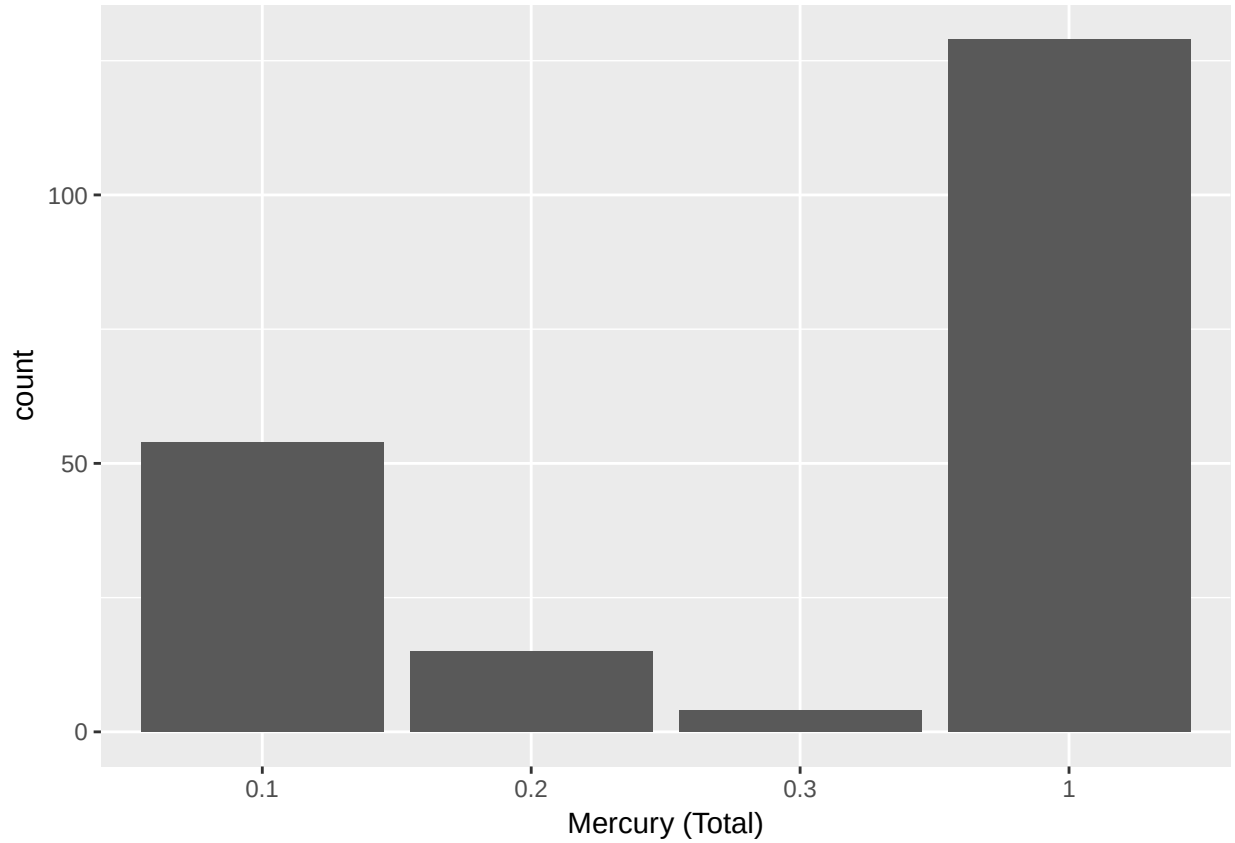
```
## Warning: Removed 30318 rows containing non-finite values (stat_qq).
```



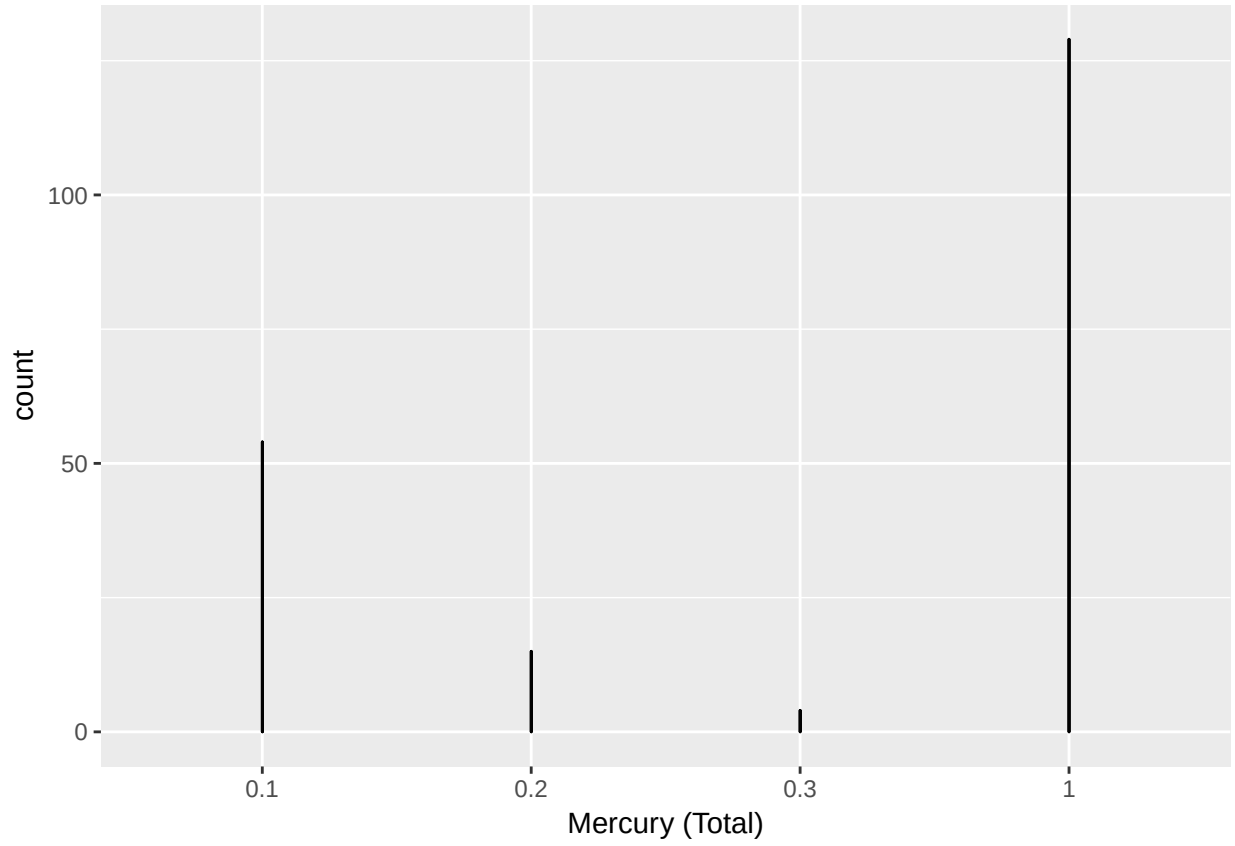

```
#group_by(StationCode) %>%
```

```
# histogram of all the stations together
wQ %>%
  select(`Mercury (Total)`,StationCode) %>%
  na.omit() %>%
  ggplot(aes(`Mercury (Total)`)) +
    geom_histogram(stat ="count")
```

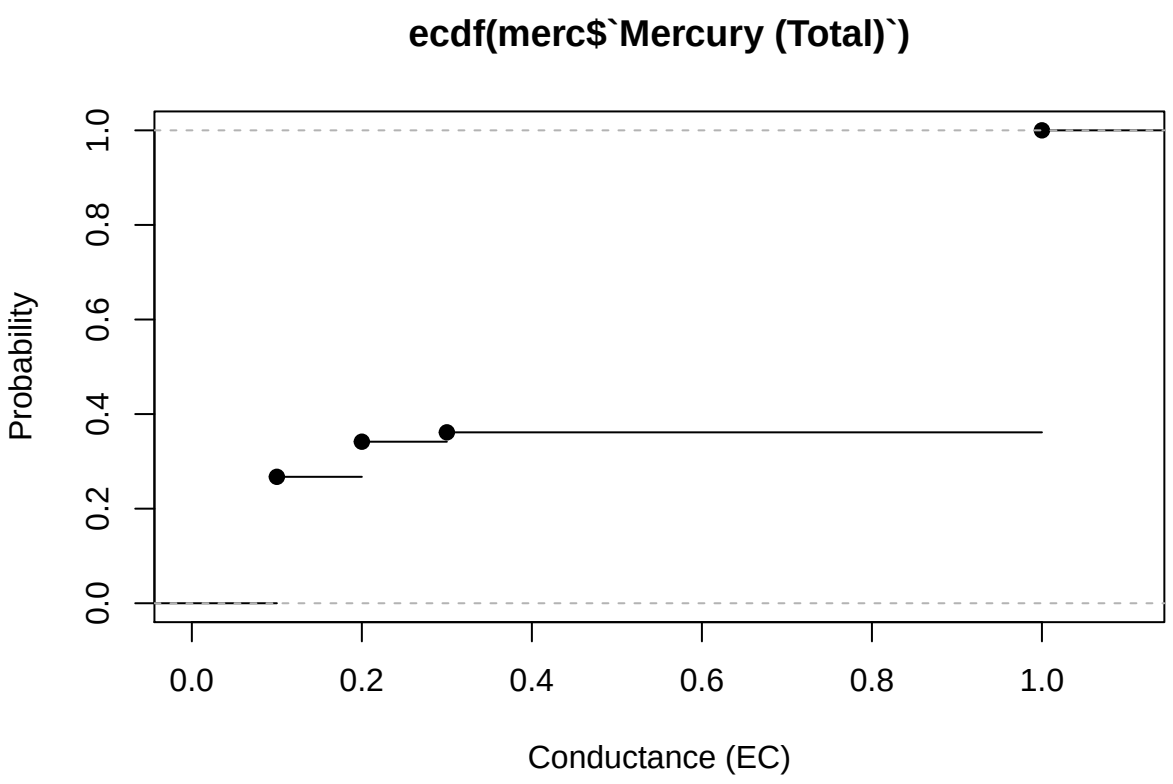
```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```




```
# if we try to make a denisty plot which will use a guassian kernel to smooth the data by default
# there is no smoothing.
wQ %>%
  select(`Mercury (Total)`,StationCode) %>%
  na.omit() %>%
  ggplot(aes(`Mercury (Total)`)) +
    geom_density(stat ="count")
```




```
## cummulative

merc <- wQ %>%
  select(`Mercury (Total)`,StationCode) %>%
  na.omit()

cdf_fun <- ecdf(merc$`Mercury (Total)`)
plot(cdf_fun, xlab = "Conductance (EC)", ylab = "Probability")
```



random notes to self . . . ignore

- distriubtion linear space linear distributed
- you know it its skewed high and low
- you see if it is high and low
- plot by date: difference in method
- is mercury increasing over time
- continuous data
- 78-88 - drought, el nino 82/83 el nino
- not all the time series are complete in the dataset