Solution of 8.23.3, Leaf area using image processing — (Wilmes, unpublished)

> We want to determine the projected leaf area of plants using photos, and analyze whether the leaves grew significantly over the course of two days. The folder `CSB/r/data/leafarea/` contains images of plants at two time points (`t1` and `t2`). The data has been collected by Madlen.
>
> 1. Write a `for` loop that processes all images using the function `getArea` that is provided in `CSB/r/solutions/getArea.R`. The function accepts a single file name as argument, and returns the projected leaf area, measured in pixels. Your loop should record the leaf area for each image, and store it in the data frame `results`. To loop over all files, you can use the function `list.files` along with its pattern matching option, to produce a list of all the files with extension `.JPG` in the directory `CSB/r/data/leafarea/`. Work in your `sandbox` or change paths in the `getArea.R` function accordingly.
> 2. Plot the area of each plant as measured on time point 1 versus time point 2.
> 3. Determine if the plants significantly differ at time point 1 and 2 using a paired t-test.

If required, we first install the `EBImage` package, and load it.

```
# install the EBImage package: run the following two commands
# source("http://bioconductor.org/biocLite.R")
# biocLite("EBImage")
# note that you might need to install other libraries
# for the installation to work

# now load the library
library(EBImage)
```

We source the file `getArea.R`, to access the function `getArea`.

```
source("../solutions/getArea.R")
```

Let's create a data frame to record the data. We want to keep track of the file names, as they identify both the individual plant, and the time point. Our `results` data frame also needs a column to record the measured area. By default, `R` converts strings to factors. We want to maintain our JPG file names as type `character` and use the option `stringsAsFactors = FALSE`.

```
## create data frame to record results
results <- data.frame(JPG = character(), area = numeric(), stringsAsFactors = FALSE)
```

The function `list.files` allows to collect all file names in a directory, so we can loop over them. The documentation of the function describes a pattern matching option that allows us to select only the `.JPG` files.

```
files <- list.files("../data/leafarea", pattern = ".JPG")
```

Now, we can write our `for` loop. It should call the function `getArea` for each file, and store the area in our `results` data frame. We also make sure that our calculated area is of type `numerical`.

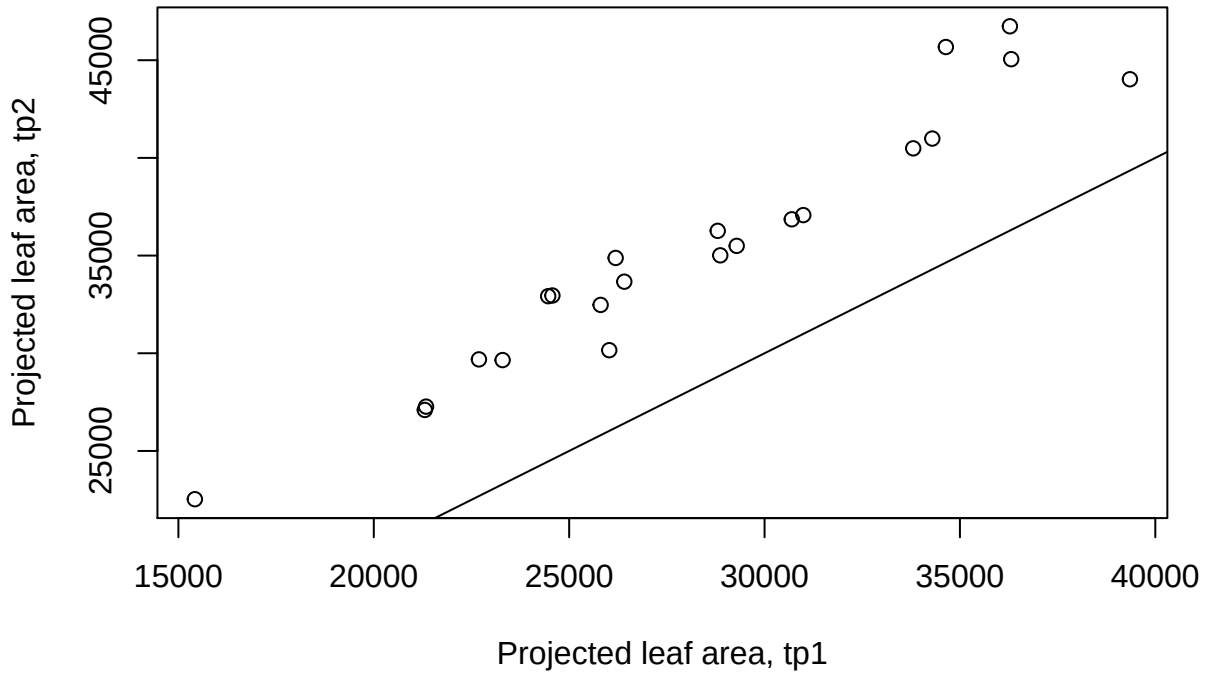
```
## run function getArea on all images
for (f in files) {
    area <- getArea(f)
    results[nrow(results) + 1, ] <- c(f, area)
}
results$area <- as.numeric(results$area)
```

In order to plot and analyze the data by time point, we need to extract the information from the file names and assign it to a new column in the results data frame.

```r
# extract time point information
results$tp <- substr(results$JPG, 1, 2)
results$tp <- as.factor(results$tp)
# extract plant information
results$plant <- sapply(results$JPG, function(x) unlist(strsplit(x, "[_]|[.]"))[2])
results$plant <- as.factor(results$plant)
```

Now, we can plot the area at time point 1 versus that at time point 2. Our data is not well organized to do this, as the area for both time points are stored in the column `area`. There are several ways to deal with this issue. For example, we can subset the the data frame using the information on the time point. Note that for this to be successful, we need the entries for each plant to be in the same order for both time points. Another option is the package `tidyr`, which has great power when it comes to data wrangling (which we will explore in the next chapter).

```r
# rearrange data in a new data frame
tp1 <- results[results$tp == "t1", ]$area
tp2 <- results[results$tp == "t2", ]$area
plot(tp2 ~ tp1, xlab = "Projected leaf area, tp1", ylab = "Projected leaf area, tp2")
abline(c(0,1)) # add the 1-to-1 line
```



At last, we can analyze our data using a one-sided, paired t-test, to see whether the leaves grew.

```r
## run the t-test
t.test(tp1, tp2, paired = TRUE, alternative = "less")
```

```
##
##  Paired t-test
##
## data:  tp1 and tp2
## t = -20.01, df = 21, p-value = 1.856e-15
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##       -Inf -6486.002
## sample estimates:
```

```
## mean of the differences
##               -7096.227
```