Project Report - Emily Palmer

Motivation

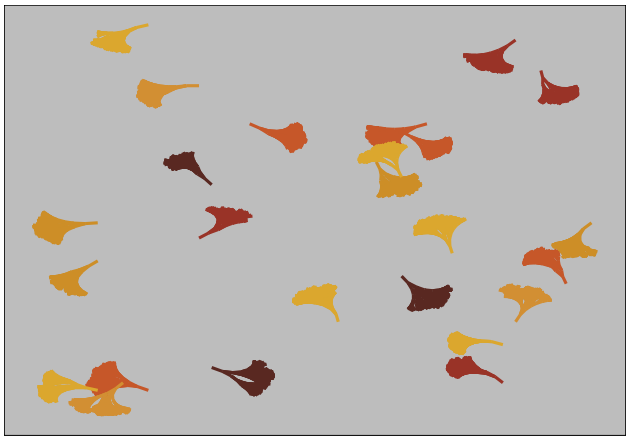
I initially had wanted to use this project to make tree landscape pictures. After generating and plotting several trees in one plot, essentially zoomed out trees, they looked more like leaves than evergreen trees that make up the mountainscapes of the pacific northwest, so I decided to make leaf plots instead.

Package and results

For this project I created the `leafart` package, which generates leaf/tree pictures.

This package contains two main functions `create_leaf_pile(param)` which creates a data frame containing multiple randomly generated leaves based on parameter restraints, and `plot_leaves()` which then plots the leaves. The two `get*_params()` functions easily create parameter lists with good defaults.

```
get_ginkgo_params() %>%
  create_leaf_pile() %>%
  plot_leaves()
```



The process to create these leaves and the helper functions that are combined to make the `create_leaf_pile()` are described below.

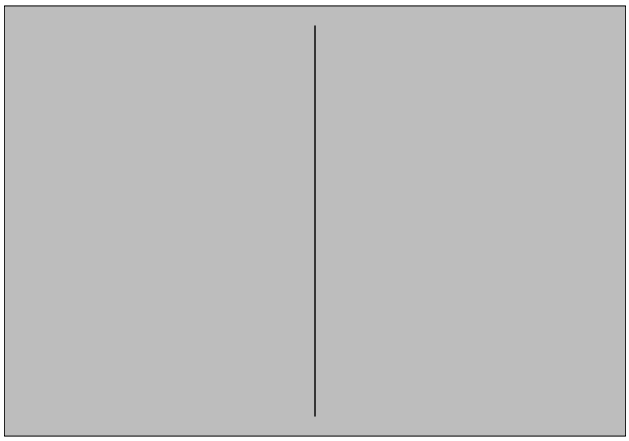
Growing a leaf

A leaf starts with a stalk. We consider this initial stalk to be the zeroth "layer". We keep track of the stalk start and endpoint so we can draw a line between them. This initial stalk also contains angle, length, and time information for future growth.

```
initial_leaf <- tibble::tibble(
    x_0 = 0, y_0 = 0,
    x_1 = x_0, y_1 = y_0 + 1,
    angle = 90, length = 1L, iter_n = 1L
  )
```

This format makes it harder to plot. The function `rake_leaves()` will transform the leaf into something better suited to plotting. The `plot_leaves()` function will plot a leaf after it is "raked", returning an `x` and

y column which are the aesthetics used for the plotting function. It also keeps track of a `step` column which is used as a grouping variable that keeps track of what layer the points are in which is used to decide which dots to connect.



Now lets create some branches. The way the leaf/tree branches depends on the parameter values we pass to the growing function. These parameters could be single values, or a vector of values to be sampled from.

We need to know the number of branches to branch off from, the angle to branch, and how long the new branches should be.

The function `get_params()` will give us a list of parameter values. We can specify what values we want, or we can use its defaults. It will default to making one tree. Since we just want to make a tree with 1 additional layer now, we will also specify that.

To grow one branch, we use the function `one_branch()`. This function creates a new list that contains the growth of the new branch.

```
params <- get_params(n_layer = 1, split = 2)
one_branch(initial_leaf, params)
```

```
## # A tibble: 1 x 7
##     x_0   y_0    x_1   y_1 angle length iter_n
##   <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>  <int>
## 1     0     1 -0.139  1.79   100    0.8      2
```

To grow all the branches in this second layer we use the function `grow_leaf_layers()`. This function iterates the function `one_branch()` over the number of new branches we want. This combines each branch (row) into one data frame that contains one row per new branch. Since we specified we want 2 splits, we will get 2 new branches.

```
grow_leaf_layers(initial_leaf, params)
```

```
## # A tibble: 2 x 7
##     x_0   y_0    x_1   y_1 angle length iter_n
##   <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>  <int>
## 1     0     1 -0.308  1.85   110    0.9      2
## 2     0     1  0.4    1.69    60    0.8      2
```

The function `grow_leaf()` will do this entire process, creating the initial stalk and growing the new layer (depending on the parameters). It accumulates calls to `grow_leaf_layers()` to do so.

```
grow_leaf(params)
```

```
## [[1]]
## # A tibble: 1 x 7
```

```
##     x_0   y_0   x_1   y_1 angle length iter_n
##   <int> <int> <dbl> <dbl> <dbl>  <int>  <int>
## 1     6    30     6    31    90      1      1
##
## [[2]]
## # A tibble: 2 x 7
##     x_0   y_0   x_1   y_1 angle length iter_n
##   <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <int>
## 1     6    31  6.45  31.8    60    0.9      2
## 2     6    31  5.69  31.8   110    0.9      2
```

This resulting list is not in a form we can plot it. We have to clean up the data frame to be easier to plot.

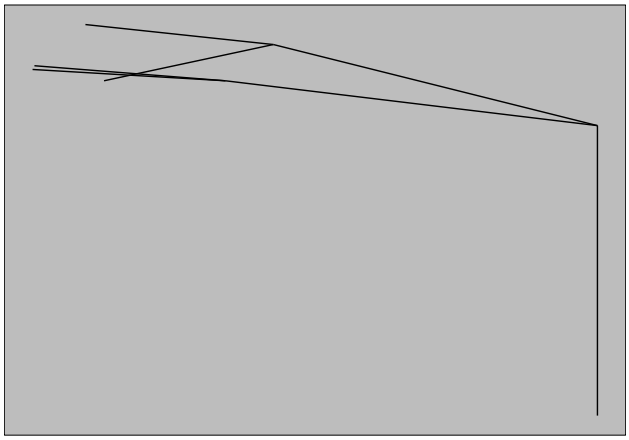
We now also have to keep track of which layer the branch belongs to so the graphing knows which points to connect.

```
get_params(n_layer = 1,
                     scale = .5,
                     angle = 90,
                     split = 2) %>%
  grow_leaf(params) %>%
  rake_leaves()
```

```
## Warning in if (init_location == "random") {: the condition has length > 1 and
## only the first element will be used
```

```
## # A tibble: 6 x 3
##           x     y  step
##       <dbl> <dbl> <int>
## 1  0.        0        1
## 2  6.12e-17  1        1
## 3  6.12e-17  1        2
## 4 -6.09e- 2  1.50     2
## 5  6.12e-17  1        3
## 6 -4.83e- 1  1.13     3
```

```
get_params(nleaves = 1,
           n_layer = 2,
           scale = .5,
           angle = 90,
           split = 2) %>%
  grow_leaf() %>%
  rake_leaves() %>%
  plot_leaves()
```



Now we have two layers. We repeat this growing process at the endpoint of each newly grown branch, until we have fully grown out the tree.

Note that I h

Attempts at speeding up runtime

Creating and plotting the leaves will take a moment to run, especially if we are creating many leaves which each have many layers. I tried to improve this run time by instead of generating each tree individually, we generate a smaller number of distinct trees, and then replicate that smaller number of trees to create the full leaf pile. These duplicated trees then need to be also randomly rotated and placed, using the functions `rotate_leaf()` and `move_leaf()`. However, this process actually took longer. This exploration can be found in the document (here)[/../topic_exploration/07_Timing_Tests.md]

Tests

Examples

Examples of running this code can be found in the package README. All functions have associated help pages created from roxygen. Users can also utilize the shiny app to explore the code as well.

Acknowledgements and sources

Big thank you to Charlotte Wickham for advice and help with code. Thanks to the members of OSSSO for being a great study group. Finally thanks to Danielle Navaro's flametree package for providing inspiration and and ideas for the .