Aspect Ratio Study

Eduardo F. Vernier

November 2017

Statistical Assessment of Layout Quality

Introduction

In this document we'll be asking a few questions on how to test the quality of layout, i.e. aspect ratios, of rectangular treemap techniques.

The most common procedure on the literature involves choosing a few datasets, running test on the algorithm being proposed plus on a few other popular techniques, and plotting/ranking the mean and (sometimes) standard deviation of the produced layouts.

This methodology has a few flaws:

- Authors might "cherry pick" the results that go into the paper.
- Authors might choose to only test datasets whose nature favor their technique.
- Authors might use a very small test cases.
- Authors might choose only to show statistics measurements (mean, median, max) that favor their results.
- And so on…

The purpose of this document is to propose an *unbiased methodology* for quality of layout analysis. And the base for this methodology will be statistics and probability.

Background

First we need to understand the Central Limit Theorem and how the sampling distribution of a function can give us an estimate of the function value for the whole population.

Let's imagine that we want to find out the mean yearly income of the whole population of the world.

If it were possible, we could interview every person alive, ask them their income, and plot it like so:

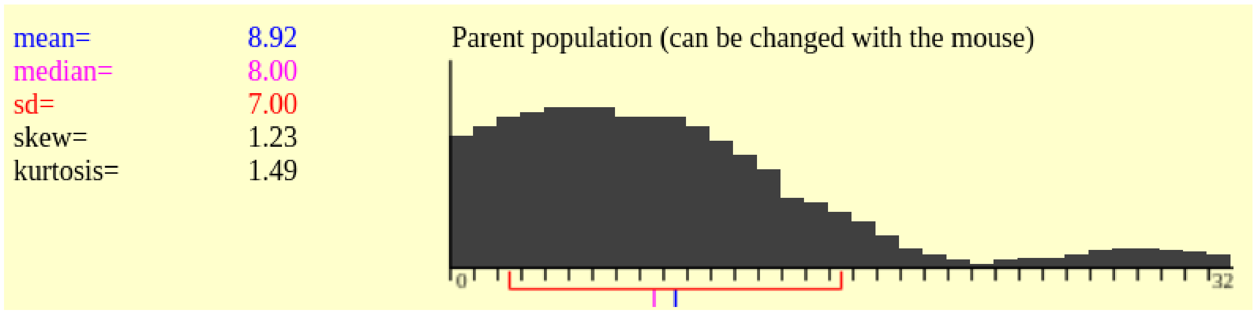
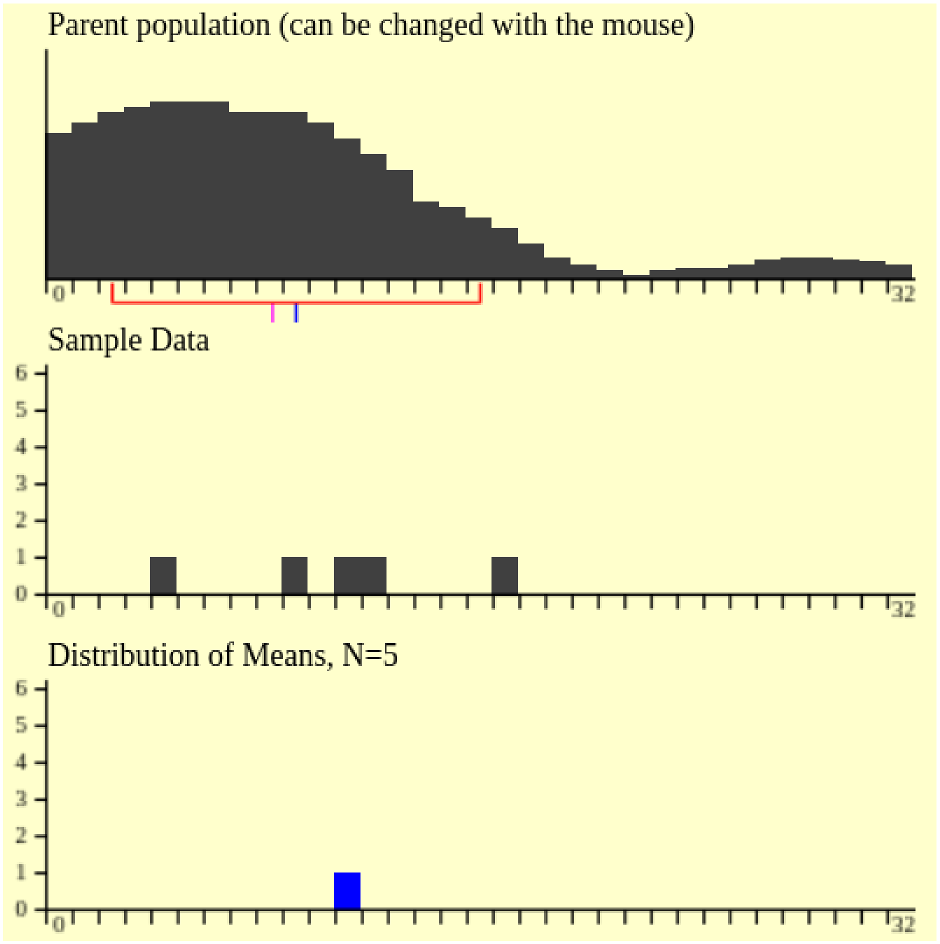


Figure 1:

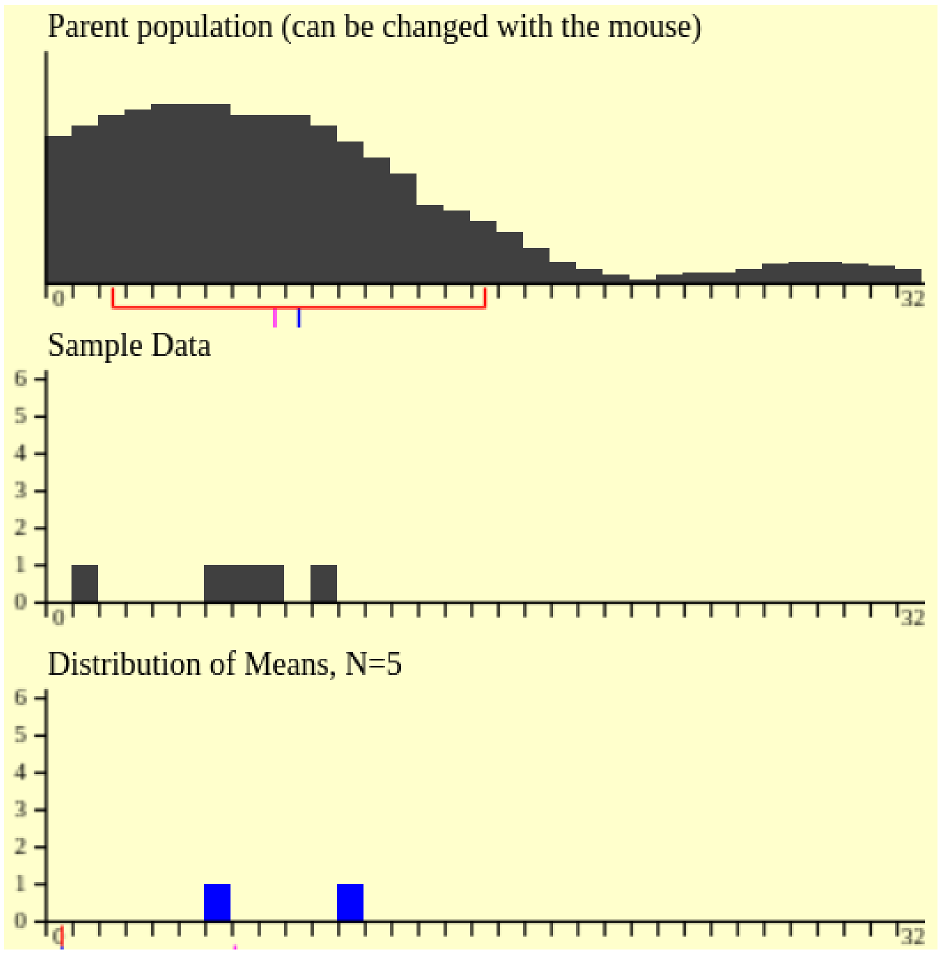
We then can compute the mean (which is what we were after) and other statistics if we want.

Since that is impossible, we'll have to try out best to **estimate** the mean of the population.

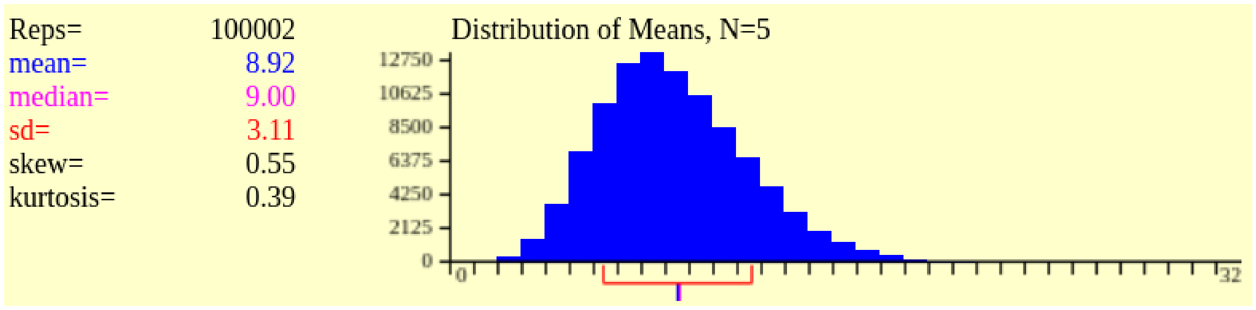
One way we can do it is by taking a random sample x of size N from the population and computing the mean of the sample $\overline{x}$.


Parent population (can be changed with the mouse)
0
32
Sample Data
6
5
4
3
2
1
0
0
32
Distribution of Means, N=5
6
5
4
3
2
1
0
0
32

We can take another random sample from the population of same size N and do the same again.


Parent population (can be changed with the mouse)
0
32
Sample Data
Distribution of Means, N=5

If we keep doing it, we will end up with a list of sample means. If we plot a histogram of these means, for a large number of samples, we'll end up with a distribution like this:



Reps=	100002
mean=	8.92
median=	9.00
sd=	3.11
skew=	0.55
kurtosis=	0.39

Figure 2:

The *mean of the distribution of* ***means*** will be an approximation of the population mean. This is the Central Limit Theorem + The Law of Large Numbers at work.

Just to recap:

> The Law of Large Numbers basically tells us that if we take a sample (n) observations of our random variable & avg the observation(mean)– it will approach the expected value E(x) of the random variable.
>
> The Central Limit Theorem, tells us that if we take the mean of the samples (n) and plot the

frequencies of their mean, we get a normal distribution! And as the sample size (n) increases –> approaches infinity, we find a normal distribution.

We can see that our normal curve has a large standard deviation. If the want to be more precise with our prediction, we can use a larger N.

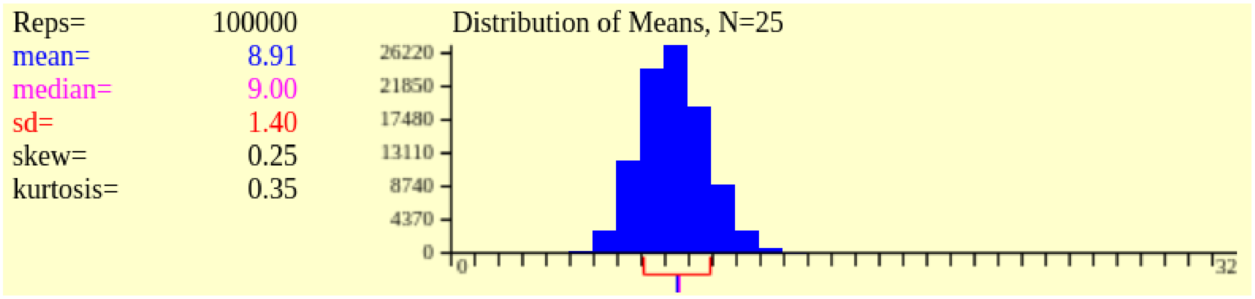
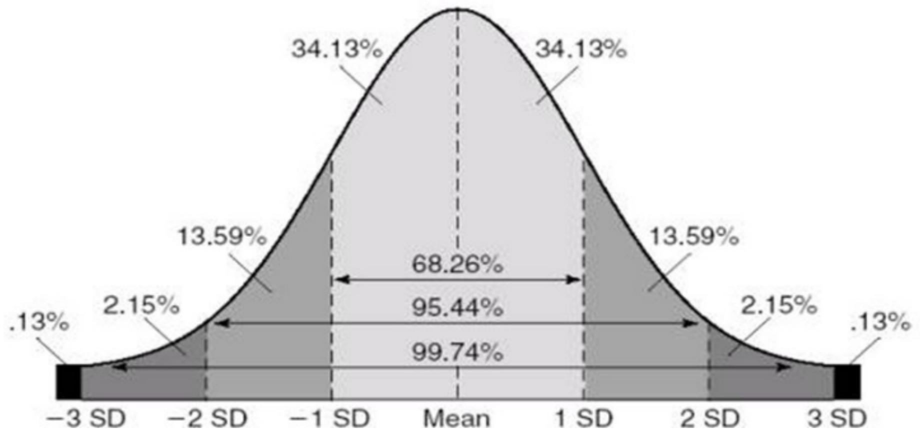


Figure 3: $N = 25$

Using the area under a normal curve (Z table), we can make claims about our statistic within a confidence interval.



For example, since we can claim:

> There is a 68.26% chance that the mean income of the world population is in the range $[7.51, 10.41]$ (1 stardard deviation away from mean).
>
> There is a 95.44% chance that the mean income of the world population is in the range $[6.1, 11.71]$ (2 standard deviations away from mean).

I know that these intervals seem a bit loose, but increasing the sample size N reduces the interval size.

I did this long introduction for a very specific reason: These statistical concepts work not only for distribution of means, we can use it with any other function we like (need a statistician to back me up here), for example, the standard deviation, variance, or even **average aspect ratio of treemaps**.

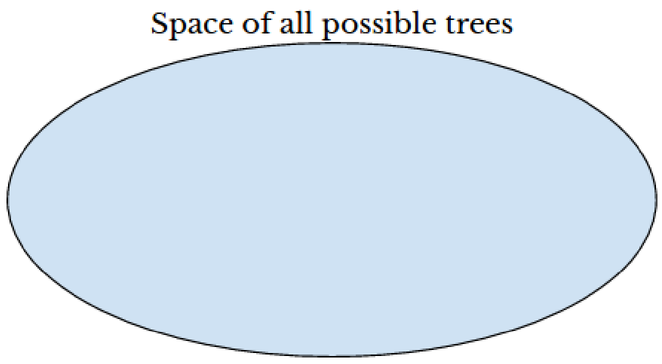
Imagine now the space of all possible weighted trees that can be inputted into a treemap algorithm.

I'm not even sure how to characterize this space, but imagine that we can run our treemap algorithm in every single tree that exists in this space. After that we can compute the mean aspect for all trees, AR variance, etc. This would be the most definite test of the quality of layout of an algorithm. We basically ran it through every possible input and computed an aggregation metric.

We know that we can't do that, as this space is infinite. But, if we can find a way to randomly sample this space, and for every sample compute the metrics we are interested in, just like we did with the world population, we will be able to *estimate* what the space mean (or other metric) would be!

Methodology

Let's use this ellipse to represent the space (or population) of all possible trees.



Then we randomly sample it with a sample size N of 5, for example, for each tree t in the sample x we compute it's corresponding treemap, and plot the distribution of aspect ratios and compute its mean.

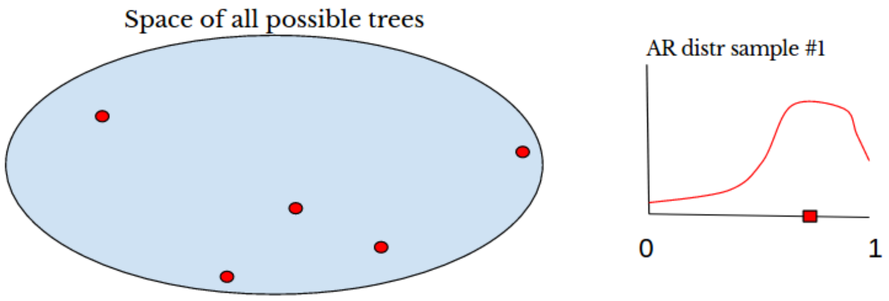


Figure 4:

After that we take another random sample and do the same.

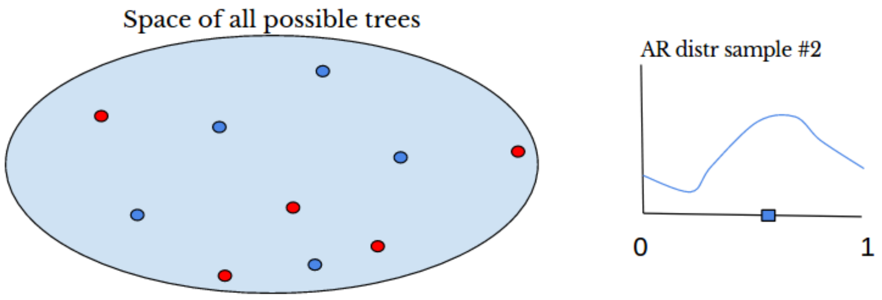


Figure 5:

After repeating the procedure a large number of times, we will end up with a *distribution of sample aspect ratio means.* And the mean of this distribution will be a fair estimator for the average aspect ratio that a technique will produce for any possible input tree.

If all this is correct, we just developed a methodology for testing treemap techniques that is independent of datasets, and all those ill intended actions that we mention in the introduction wouldn't affect the technique ranking/quality score.

Now there is one key element that we glossed over: How do we randomly sample the tree space? I have some ideas, but I need to think a little bit more about them.

References:

https://www.khanacademy.org/math/statistics-probability/sampling-distributions-library/sample-means/v/central-limit-theorem

https://www.khanacademy.org/math/statistics-probability/sampling-distributions-library/sample-means/v/sampling-distribution-of-the-sample-mean

http://onlinestatbook.com/stat_sim/sampling_dist/index.html

A New Metric For Quality Measurements

We just defined a new methodology for quality measurement, but it's a little bit over the top. It is a nice idea, but we still need to test how it behaves in practice.

Today I talked to a visiting professor about the problem and he proposed a very straight forward metric for measuring how good treemap layouts are.

What he proposed was the following: Imagine we have a 2d plane where the x axis is the longer side of a rectangle, and the y axis is the smaller size of a rectangle. Then we take a treemap and plot each cell as point. Intuitively we could say that the ideal case would be to have all points in the the line $x = y$.

Let's see what a Squarified treemap of 2505 cells looks like:

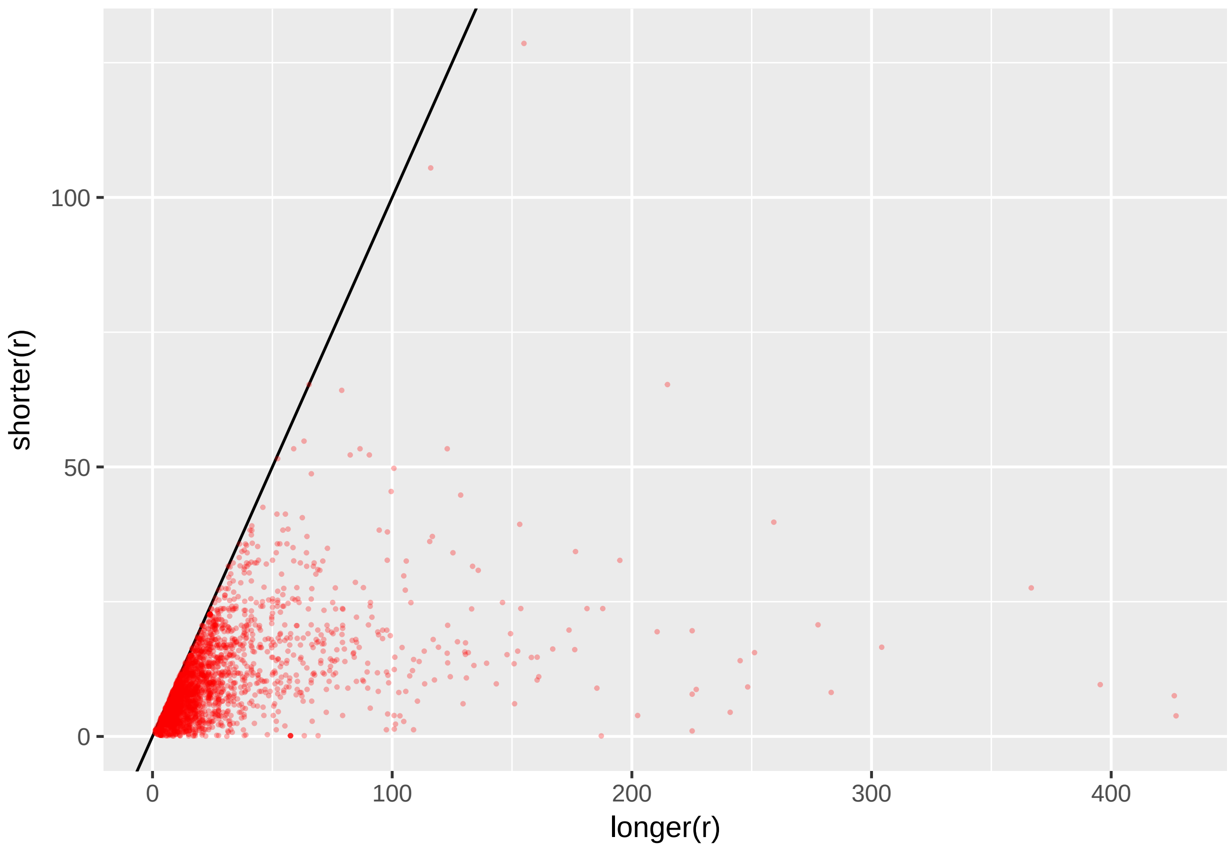
First let's read the rectangles.

```r
library(dplyr)
library(magrittr)
library(ggplot2)
# Read rectangles from file
filename = "/home/eduardo/Desktop/treemap-analysis/rectangles/sqr/cpython/t320.rect"
cls = c("character","numeric", "numeric", "numeric", "numeric")
df = read.csv(filename, colClasses=cls, stringsAsFactors=FALSE, head=FALSE, sep=" ")
n_rectangles = nrow(df)
# Extract column of width and height
width  = df[, "V4"]
height = df[, "V5"]

df = df %>%
  mutate(width = as.numeric(V4), height = as.numeric(V5)) %>%
  mutate(l = pmax(width, height)) %>%
  mutate(s = pmin(width, height))
```

And now plot the result.

```r
ggplot(df, aes(df$l, df$s)) +
  geom_abline(slope=1, intercept=0) +
  geom_point(colour = "red", size = 0.3, alpha = 0.3) +
  xlab("longer(r)") + ylab("shorter(r)")
```



Then for each point, we compute the distace to the $x = y$ line. (and divide by the number of points)

```
df = df %>%
  mutate(dist_point_line = abs(l + s) / sqrt(2))

sum(df[, "dist_point_line"]) / nrow(df)
```

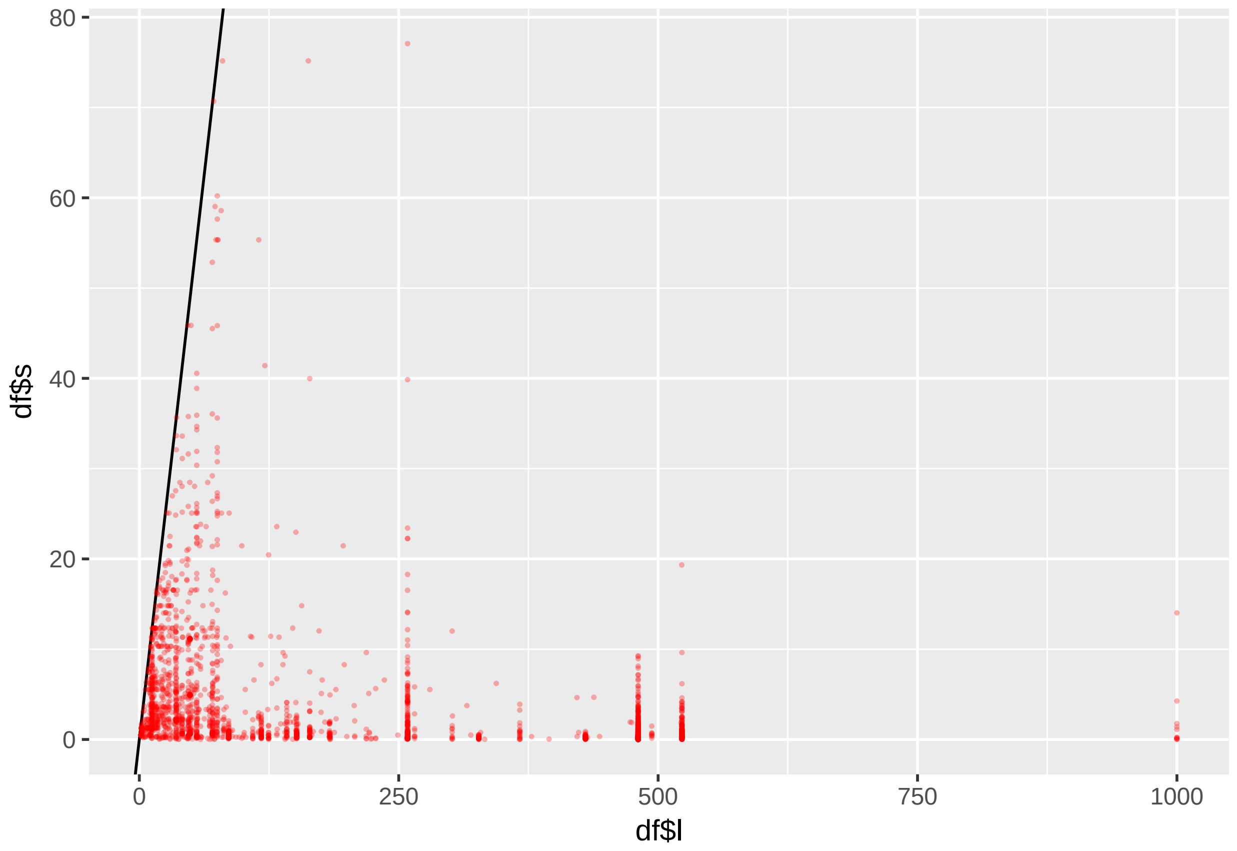
```
## [1] 26.29428
```

Ok, we got a score of 26.3. Let's see how Slice and Dice performs for the same input tree.

```
filename = "/home/eduardo/Desktop/treemap-analysis/rectangles/snd/cpython/t320.rect"
cls = c("character","numeric", "numeric", "numeric", "numeric")
df = read.csv(filename, colClasses=cls, stringsAsFactors=FALSE, head=FALSE, sep=" ")
n_rectangles = nrow(df)
# Extract column of width and height
width  = df[, "V4"]
height = df[, "V5"]

df = df %>%
  mutate(width = as.numeric(V4), height = as.numeric(V5)) %>%
  mutate(l = pmax(width, height)) %>%
  mutate(s = pmin(width, height))
```

```
ggplot(df, aes(df$l, df$s)) +
  geom_abline(slope=1, intercept=0) +
  geom_point(colour = "red", size = 0.3, alpha = 0.3)
```



```r
df = df %>%
  mutate(dist_point_line = abs(l + s) / sqrt(2))

sum(df[, "dist_point_line"]) / nrow(df)
```

```
## [1] 148.83
```

Slice and Dice scored 148.83, a much higher result.

This idea borrows heavily on the concept of Errors and Residuals. I'll run some tests on all datasets for all technique and see the results.

https://en.wikipedia.org/wiki/Errors_and_residuals