Introduction to R

Roma Klapaukh

8 May 1515

What is R?

- R is a programming language for doing statistics
- It has good libraries for doing statistical tests and analysis

Starting R

At ECS

Starting R is very easy:

```
need R
R
```

At home

```
R
```

Using RStudio

RStudio is an open source IDE for R

Loading packages

The best thing about R is its libraries. Using R without any libraries, while good, does not let you get the most out of it.

ggplot2 is a library for drawing charts

```
library(ggplot2)
```

dplyr is a library for manipulating tables of data

```
library(dplyr)
```

Scatterplot3d is library for drawing 3D scatter plots

```
library(scatterplot3d)
```

rgl is for making interactive 3d plots

```
library(rgl)
```

Using R

Let’s look at an example of using R to do an analysis of experimental data.

We will use the results of Alex’s experiments, which he has run on the ECS grid.

Create Filenames

Alex has sensibly given all of his files names in a fixed pattern: artificialOptimumResults/out*N*.stat

This can be broken up into a prefix, number, and suffix.

```
prefix <- "artificialOptimumResults/out"
suffix <-".stat"
```

The *N* range from 1 to 50.

```
numbers <- 1:50
```

We stick all three parts together to get all the filenames.

```
filenames <- paste(prefix, numbers, suffix, sep="")
```

Read in a file

We need a function to read in files that that can do several extra things:

- Deal with missing files (because of the grid)
- Ignore the last line (which is the final solution)
- Add a column to remember which file the table came from (the source)

```
readMyFile <- function(name){
  table <- tryCatch(
           read.table(name, nrows=500),
          error = function(x) return(NULL),
          warning = function(x) return(NULL)
        )
   if(!is.null(table)){
     table$source = name
   }
  return(table)
}
```

Read in all the files

We need to apply the readMyFile function to all the filenames

```
files = lapply(filenames, readMyFile)
```

But some of these tables didn't read, and so are NULL. We need to remove them

```
files = files[sapply(files, function(x) !is.null(x))]
```

And we only want to use 30 of the ones which succeeded

```
files = files[1:30]
```

Now that we have all the tables we want, let's turn them all into a single table

```
data = bind_rows(files)
```

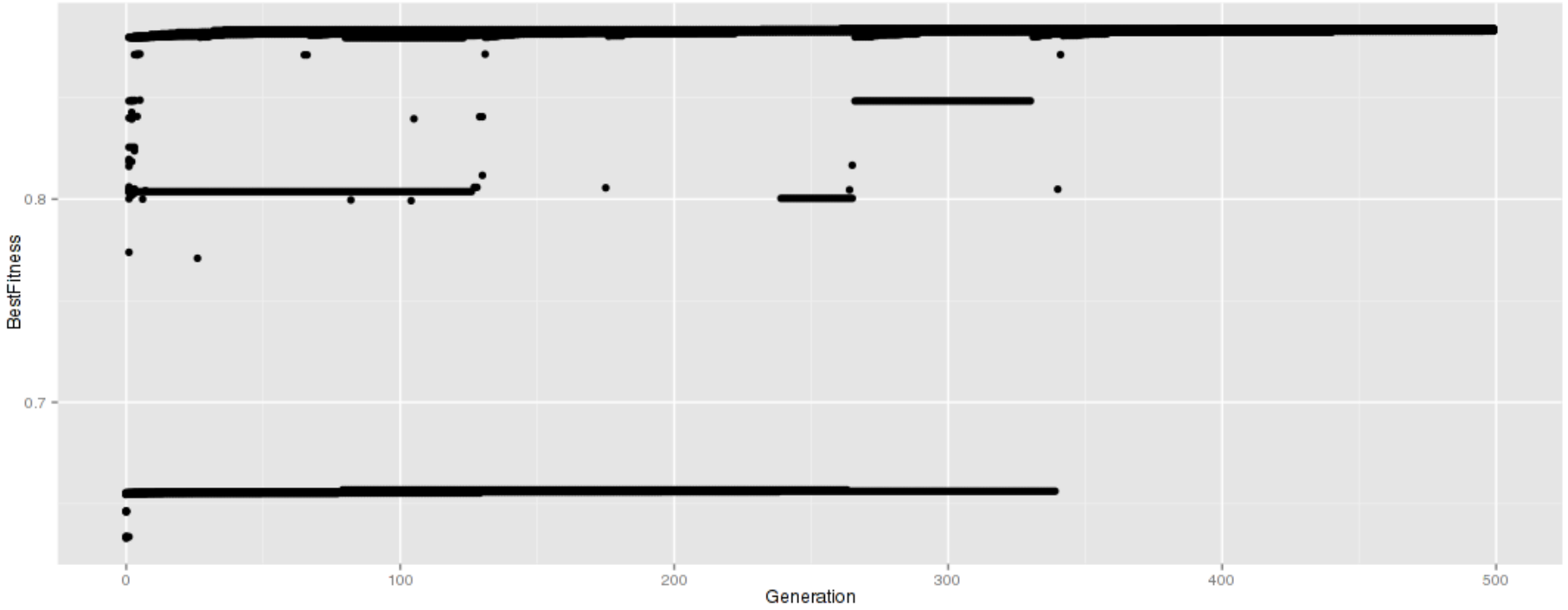
Let's give the columns nice names (because there aren't any in the file)

```
colnames(data) <- c("Generation", "SetUpTime", "RunTime", "AverageFitness",
                    "BestFitnessThisGen", "BestFitness", "X","Y","source")
```

Plotting convergence (1)

The first thing we can do just plot fitness over time

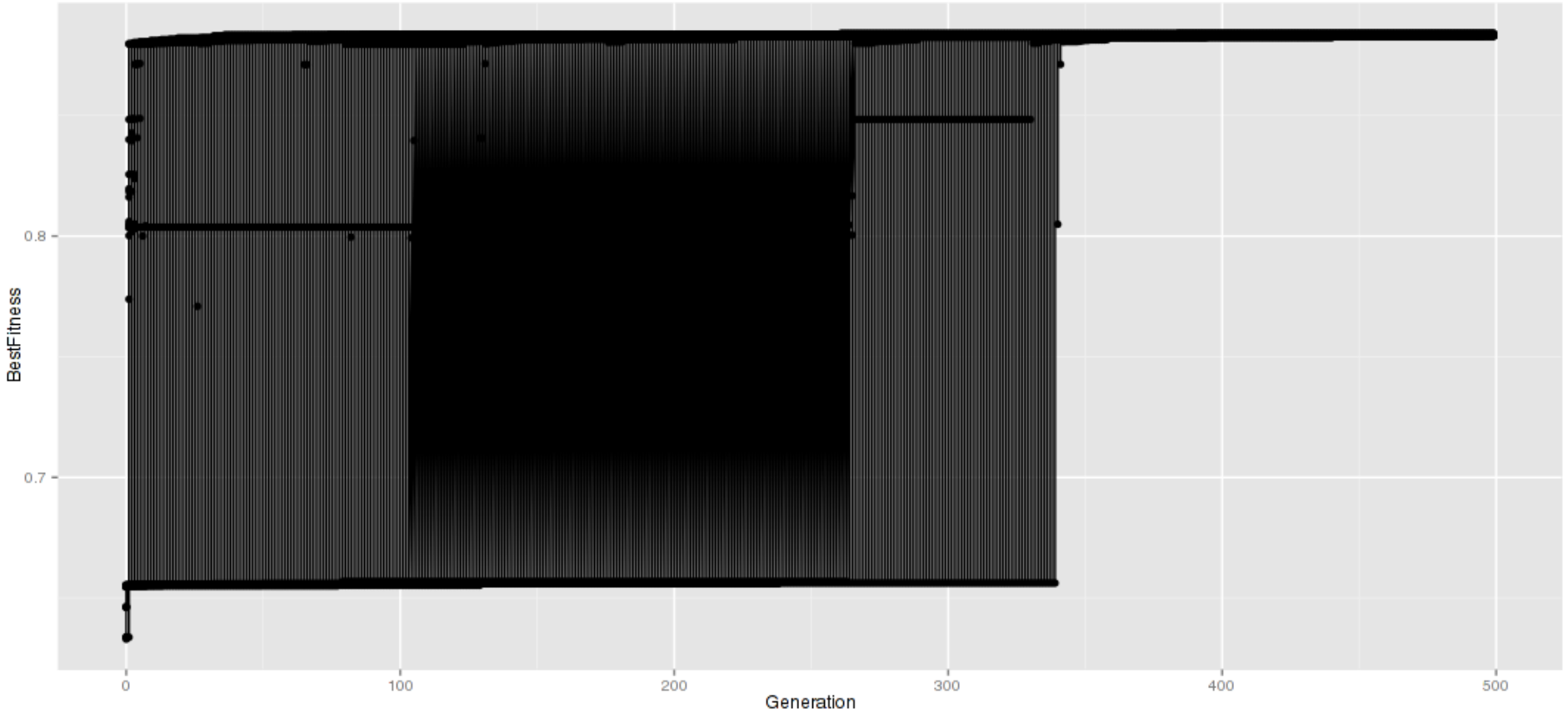
```
ggplot(data, aes(x=Generation, y = BestFitness)) + geom_point()
```



Plotting convergence (2)

Let’s add in lines

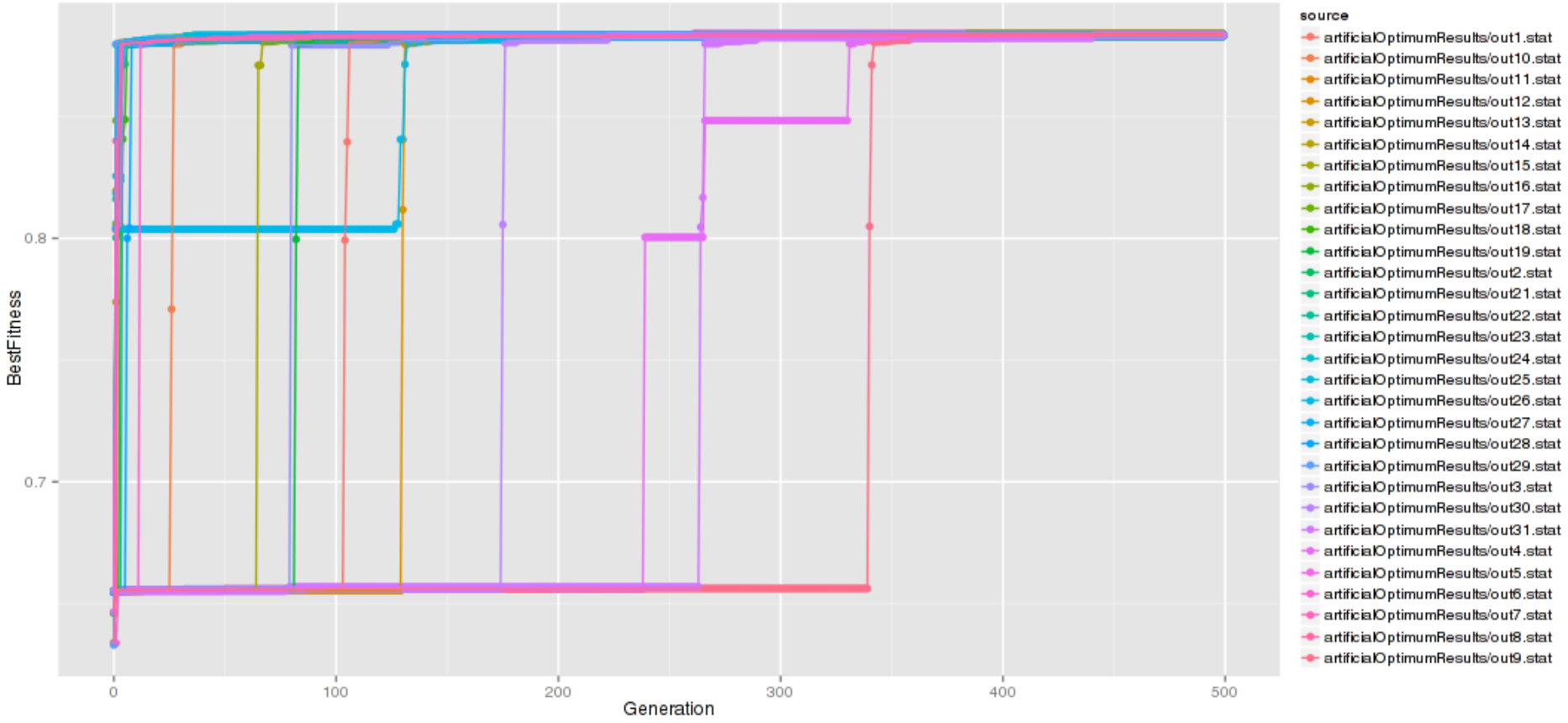
```
ggplot(data, aes(x=Generation, y = BestFitness)) + geom_point() + geom_line()
```



Plotting convergence (3)

Colour the line by source file (test run)

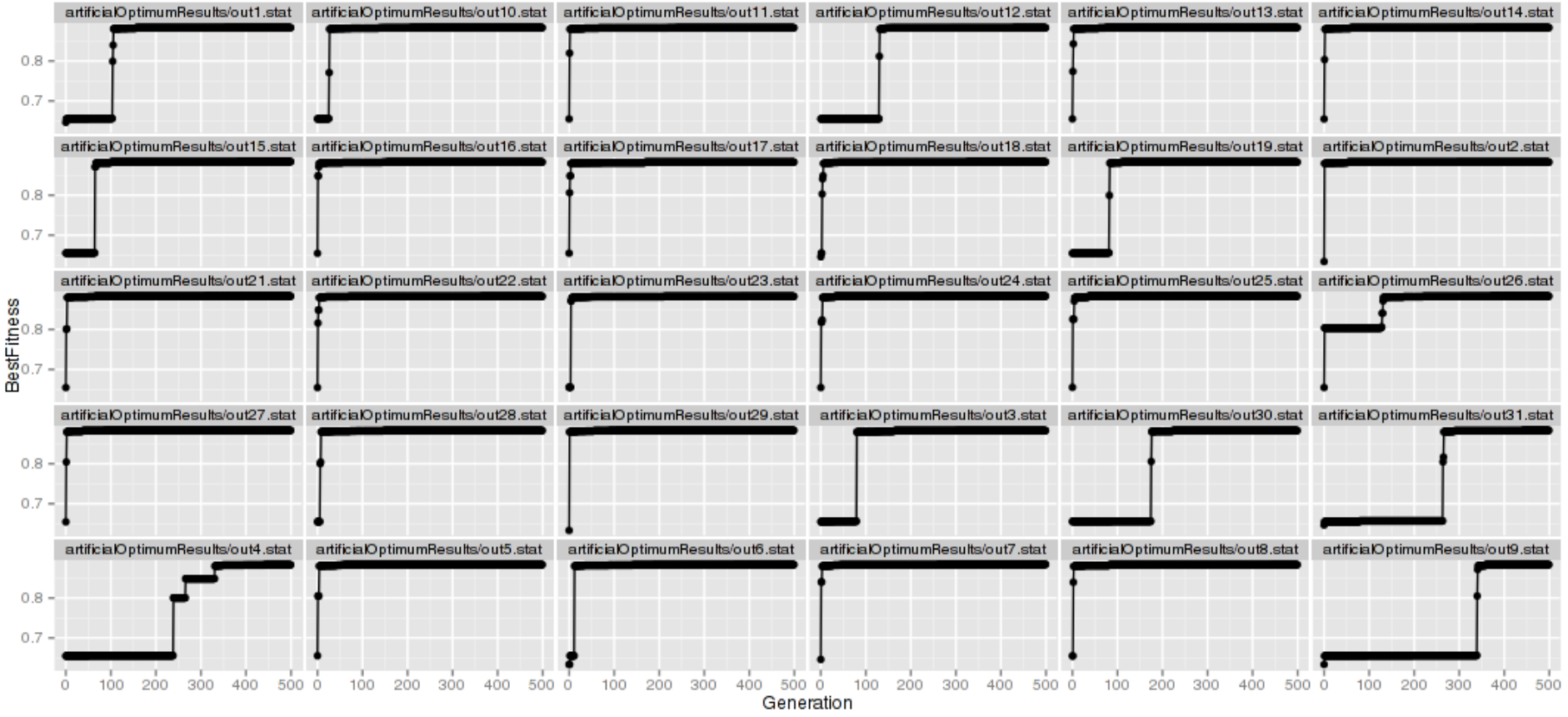
```
ggplot(data, aes(x=Generation, y = BestFitness, color=source)) + geom_point() +
  geom_line()
```



Plotting convergence (4)

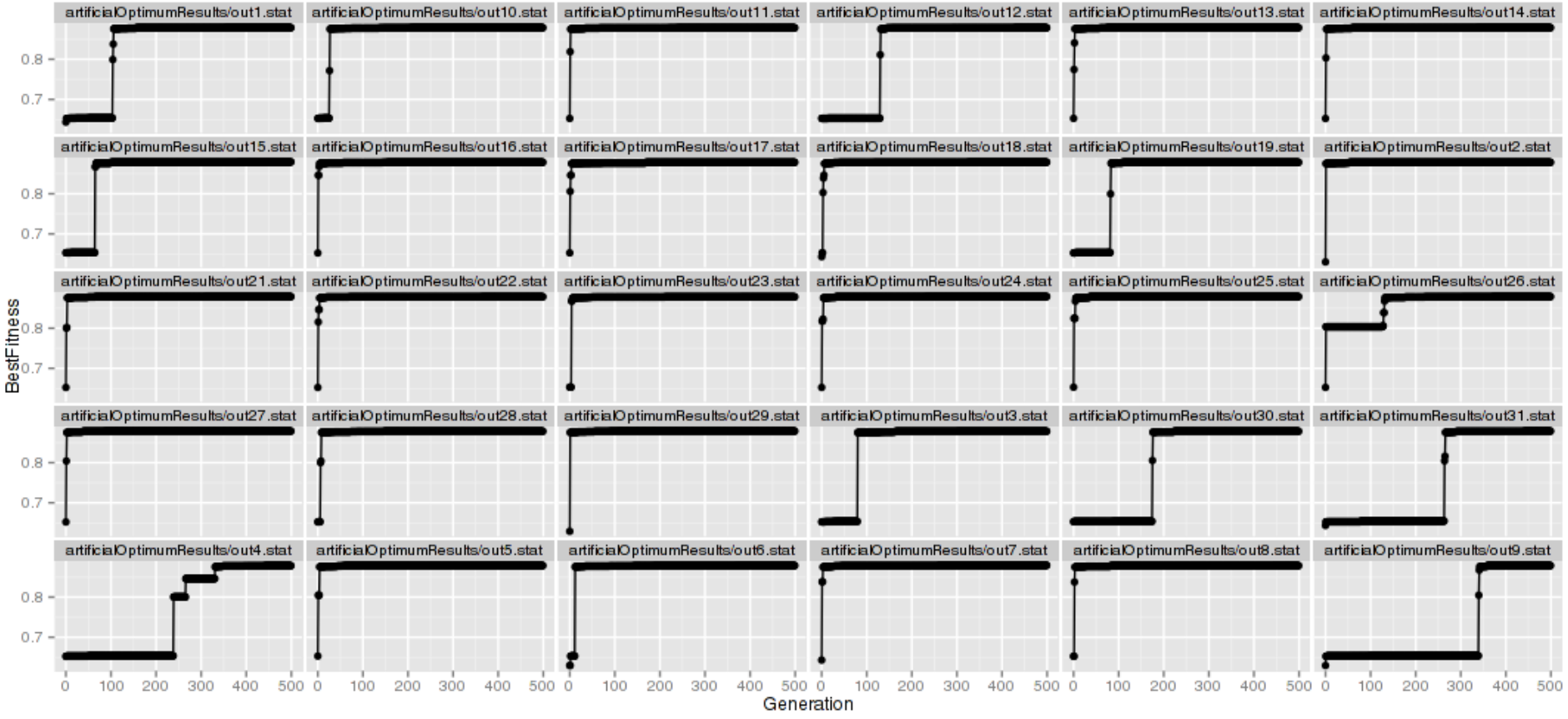
Make a separate plot for each source file (test run)

```
ggplot(data, aes(x=Generation, y = BestFitness)) + geom_point() + geom_line() +
  facet_wrap(~source)
```



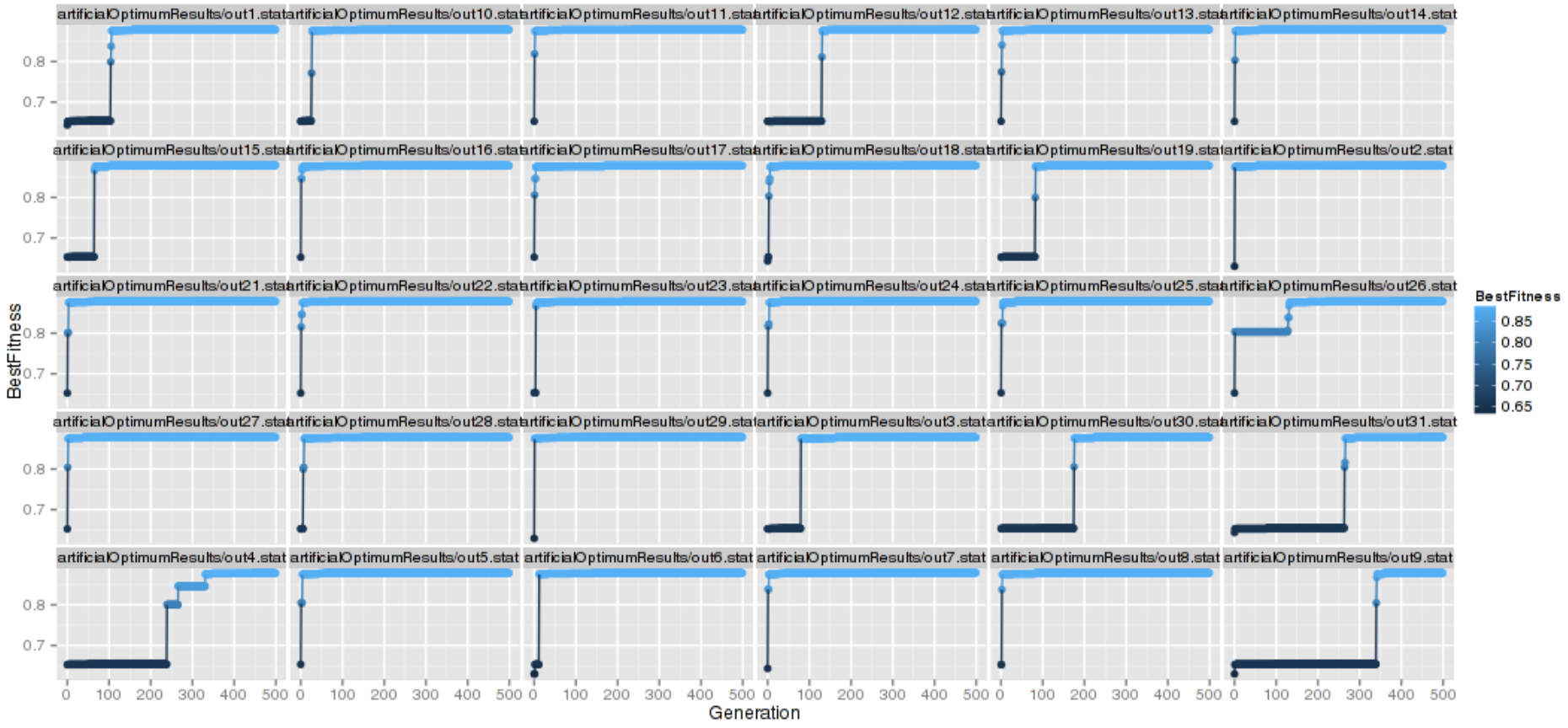
Plotting convergence (5)

```
ggplot(data, aes(x=Generation, y = BestFitness)) + geom_point() + geom_line() +
  scale_y_sqrt() + facet_wrap(~source)
```



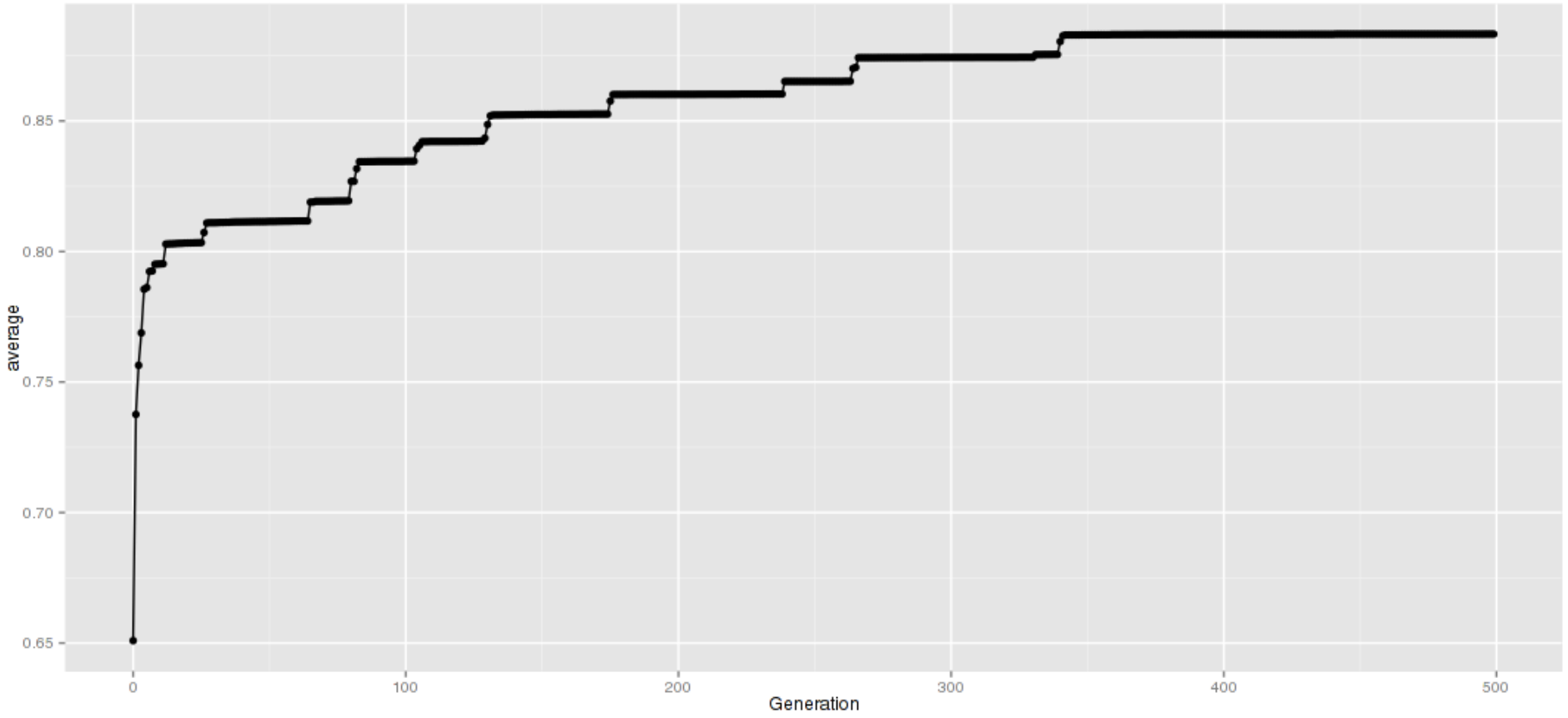
Plotting convergence (6)

```
ggplot(data, aes(x=Generation, y = BestFitness, color=BestFitness)) + geom_point() +
  geom_line() + scale_y_sqrt() + facet_wrap(~source)
```



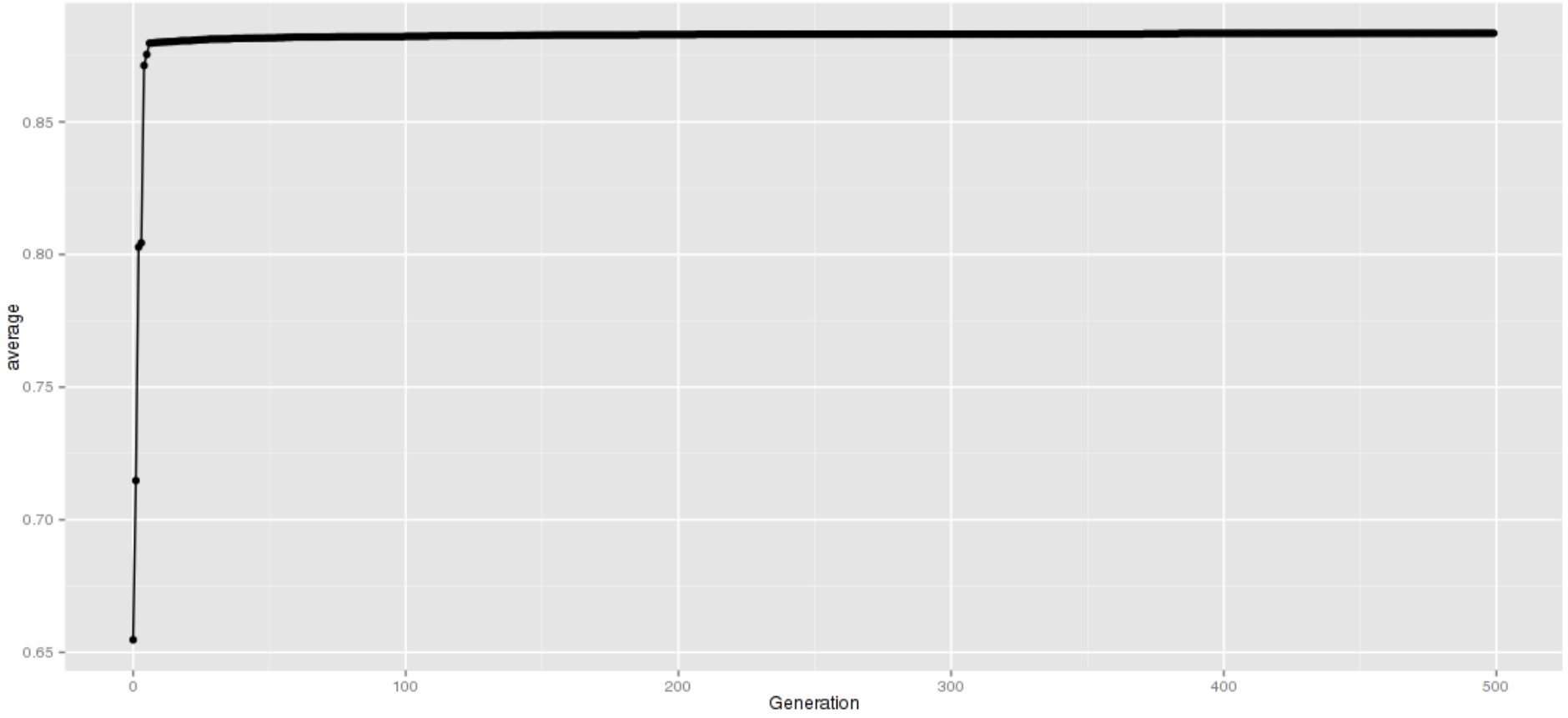
Plot the average

```
newData = data %>%
        group_by(Generation) %>%
        summarise(average = mean(BestFitness))

ggplot(newData, aes(x=Generation, y=average)) + geom_point() + geom_line()
```



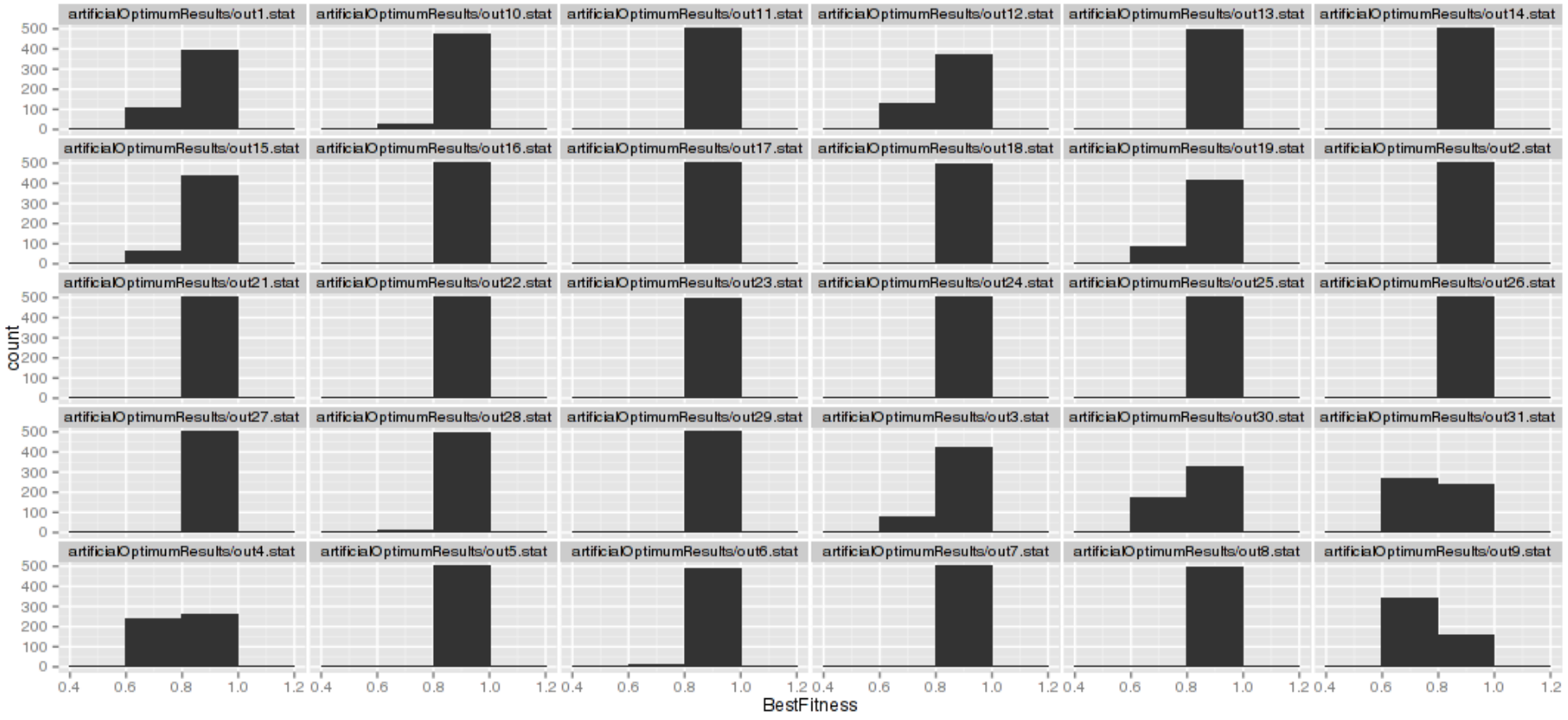
Plot the median

```
newData = data %>%
        group_by(Generation) %>%
        summarise(average = median(BestFitness))

ggplot(newData, aes(x=Generation, y=average)) + geom_point() + geom_line()
```



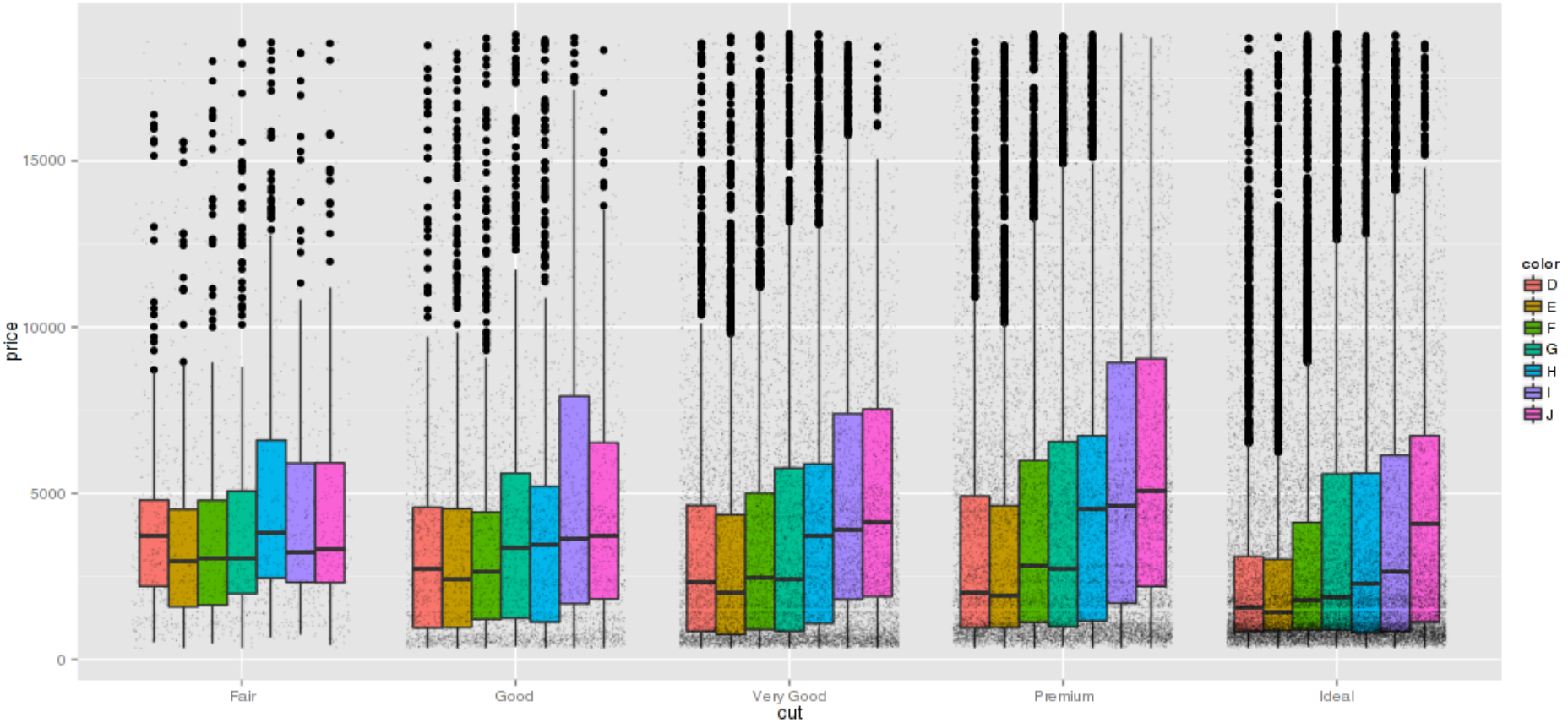
plot the distribution

```
ggplot(data, aes(x=BestFitness)) + geom_histogram(binwidth=0.2) + facet_wrap(~source)
```



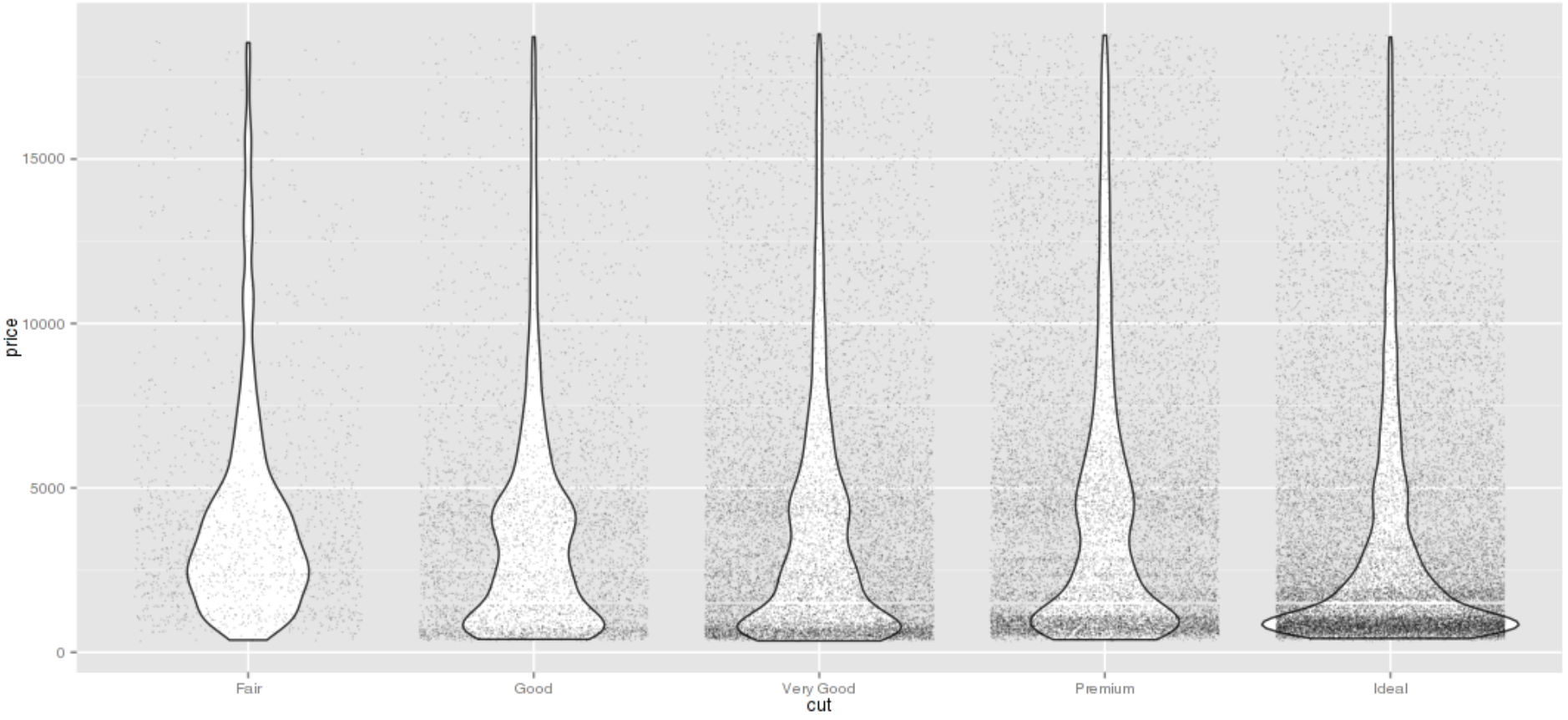
Plot boxplots

```
ggplot(diamonds, aes(x=cut, y = price, fill = color)) + geom_boxplot() +
    geom_jitter(alpha=0.2,size=0.1)
```



Plot a violin

```
ggplot(diamonds, aes(x=cut , y = price)) + geom_violin() +
  geom_jitter(alpha=0.2,size=0.1)
```



Significance testing

We want to do a significance test. Fortunately, that is very easy in R.

```
wilcox.test(data$BestFitness,data$AverageFitness)
```

```
    Wilcoxon rank sum test with continuity correction

data:  data$BestFitness and data$AverageFitness
W = 202192196, p-value < 2.2e-16
alternative hypothesis: true location shift is not equal to 0
```

```
t.test(data$BestFitness,data$AverageFitness)
```

```
    Welch Two Sample t-test

data:  data$BestFitness and data$AverageFitness
t = 52.2739, df = 27572.67, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.04799139 0.05173054
sample estimates:
mean of x mean of y
0.8582629 0.8084020
```

Pareto front

Deepak has a pareto front of solutions he wants to plot 

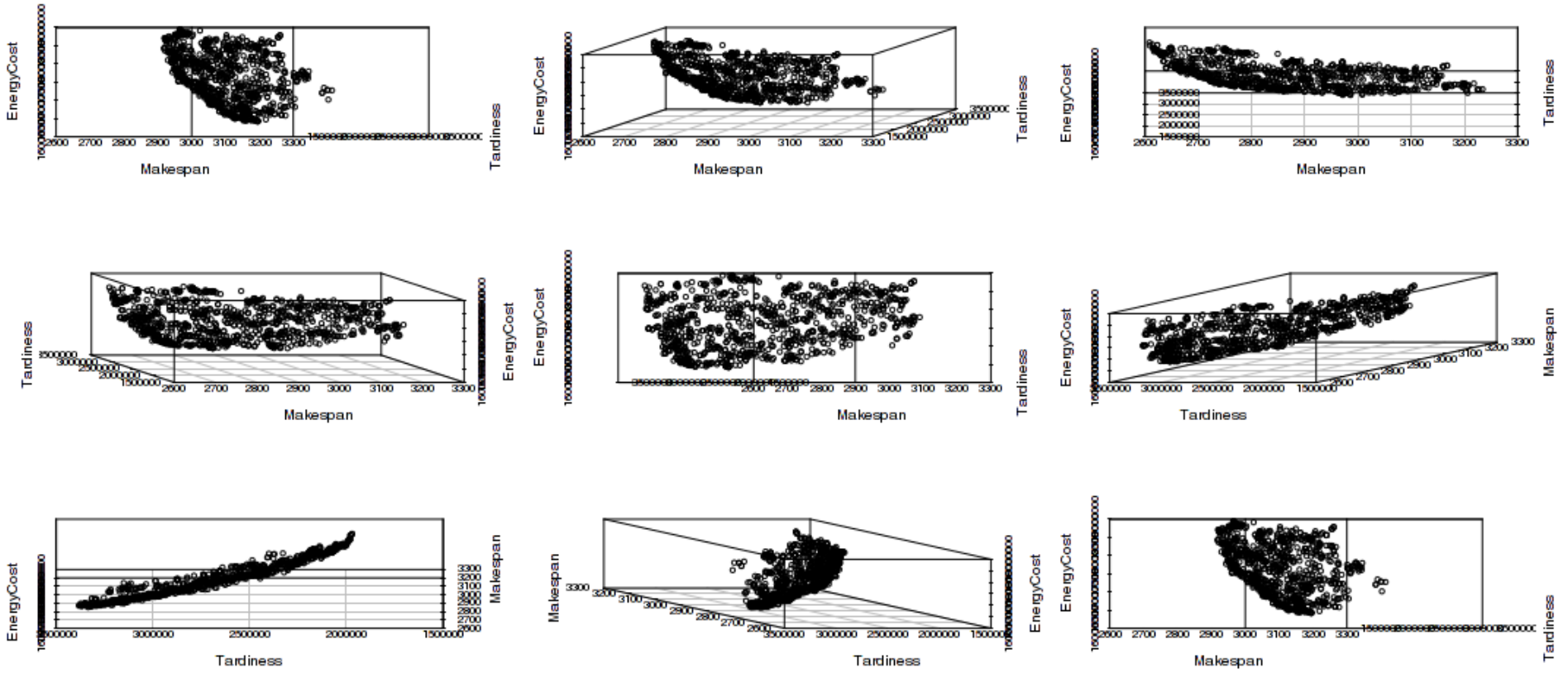
```
front = read.table("paretofront-MTE.stat",head=T)

head(front)
```

```
  Makespan Tardiness EnergyCost
1  2608.44   3150822   18657403
2  2608.94   3124107   18550323
3  2613.64   3113757   18612588
4  2614.74   3123718   18419247
5  2617.08   3161501   18240228
6  2617.40   3125426   18387053
```

3D Plotting

We can plot the 3D front in 3D directly.

```
par(mfrow=c(3,3))
x=sapply(seq(0,360,length.out=9), function(x) scatterplot3d(front, angle=x))
```



Interactive 3D

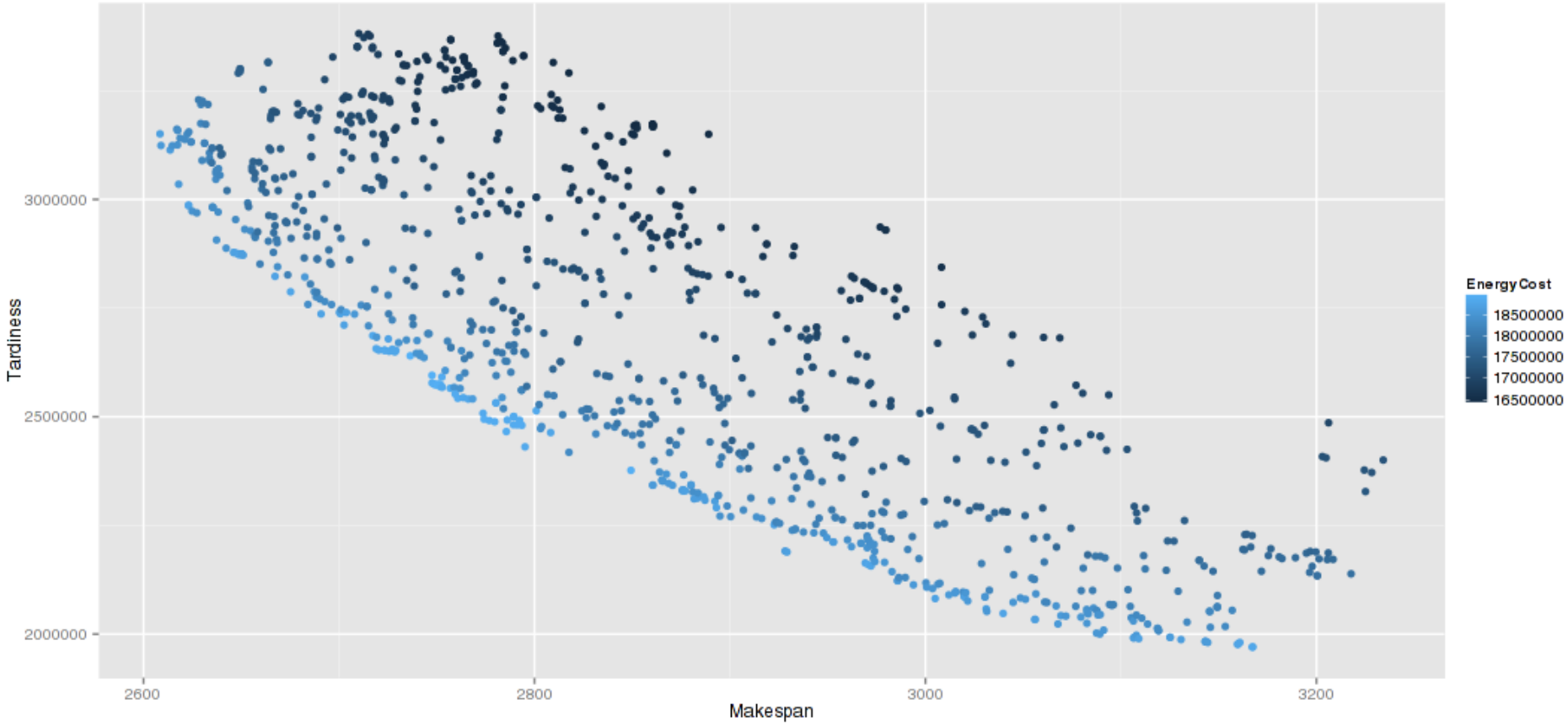
We can make an interactive 3D plot, but unfortunately we can't do this in the slides

```
plot3d(front)
```

2D plotting (1)

We can draw a simple 2D projection using colour to show the third variable

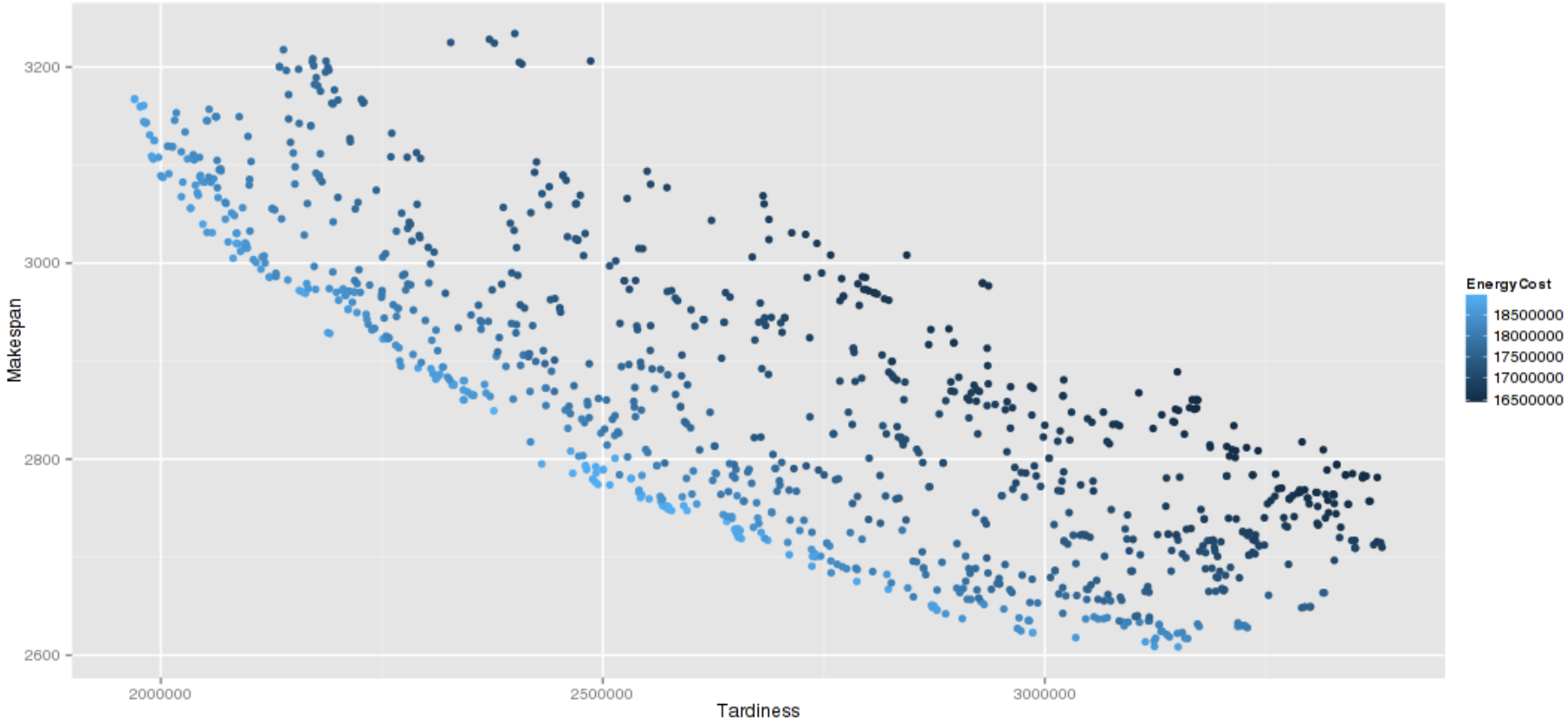
```
ggplot(front, aes(x=Makespan, y=Tardiness, colour=EnergyCost)) + geom_point()
```



2D plotting (2)

We can rotate this image

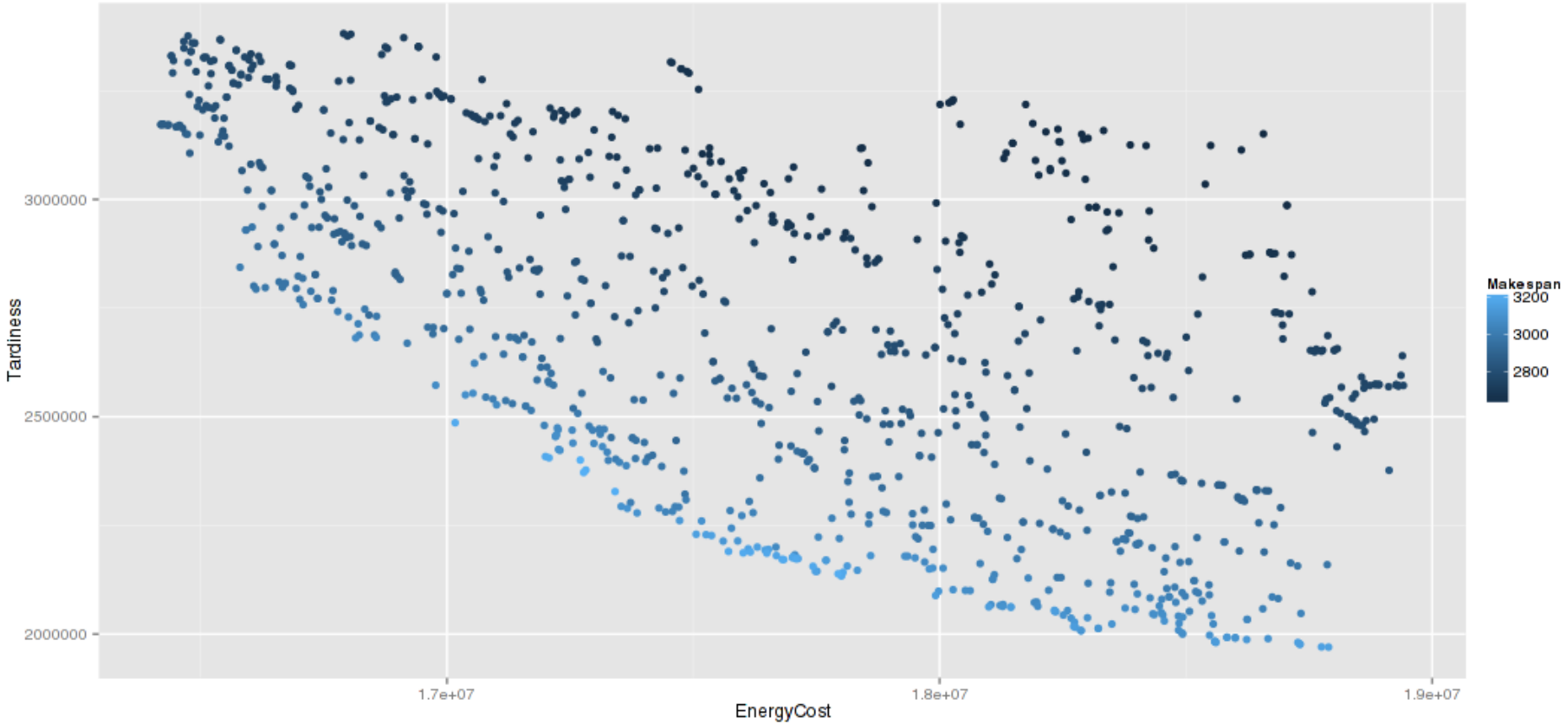
```
ggplot(front, aes(y=Makespan, x=Tardiness, colour=EnergyCost)) + geom_point()
```



2D plotting (3)

And rotate again

```
ggplot(front, aes(colour=Makespan, y=Tardiness, x=EnergyCost)) + geom_point()
```



Questions?