Knapsack Experiments

Michelle King, Maggie Casale

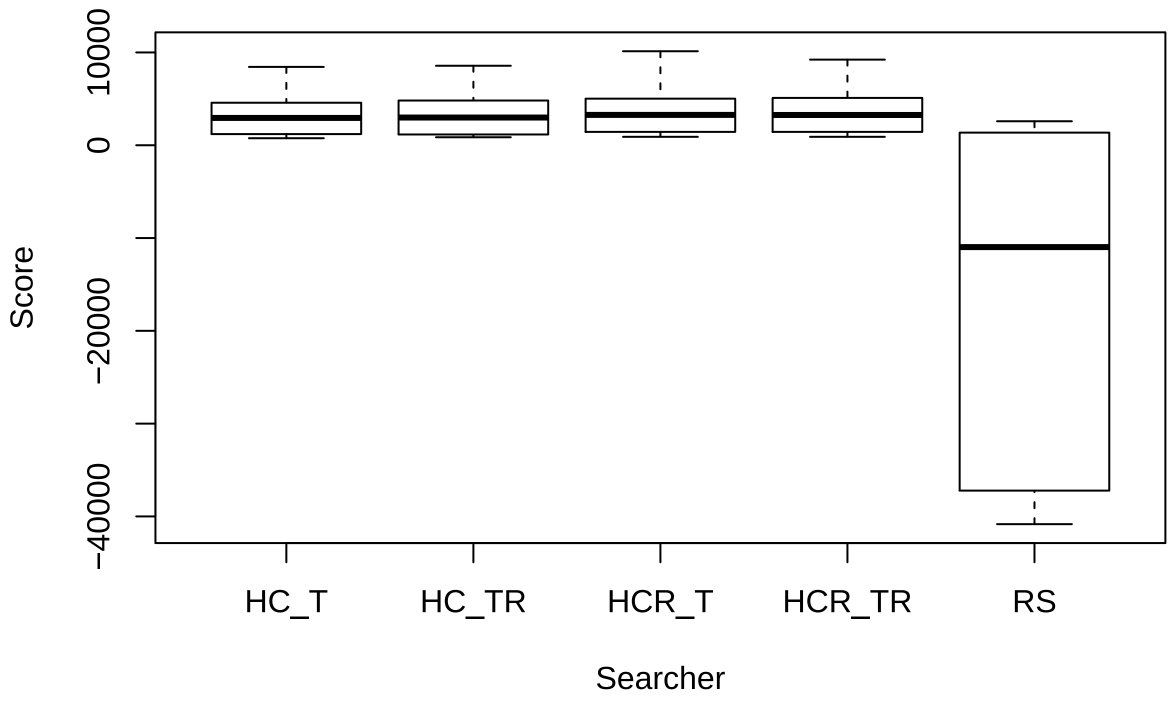
February 17, 2016

5 runs per treatment

First let's load up the data with just 5 runs per treatment.

```
data_5_runs <- read.csv("/home/casal033/ECAI/simple-search/data/output5run10000tries.txt", sep="",heade
```

- Now let's plot the score as a function of the search method with:
- HC_T: Hill Climber and basic Tweak function.
- HC_TR: Hill Climber and Tweak with Rates function.
- HCR_T: Hill Climber Random Restart and basic Tweak function.
- HCR_TR: Hill Climber Random Restart and Tweak with Rates function.



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

This plot definitely suggests that the end boxplot (`random_search`) does the worst. It also looks like `Hill Climber Random Restart` (the middle boxplot) is perhaps a little better than the rest. We should check, though, whether these differences are statistically significant, especially since it's not entirely clear whether the left and right plots are in fact that different.

Running a pairwise Wilcoxon Rank Sum Test allows us to compare each pair of approaches and see if there are statistically significant differences. We want to use a *pairwise* test because we're doing multiple comparisons, which increases the chances that one of them appears significant just because we got lucky. Pairwise tests correct for that, increasing our confidence in our results.

We're using the Wilcoxon test because our data (and EC results in general) typically is no where close to normally distributed, so we *don't* want to use things like standard t-tests that are based on assumptions of normality. Tests like Wilcoxon that are based on ranks and don't assume normality are generally much better choices for the kinds of data that EC systems generate.

```
pairwise.wilcox.test(data_5_runs$Score, data_5_runs$Search_method)
```

```
## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties

## Warning in wilcox.test.default(xi, xj, paired = paired, ...): cannot
## compute exact p-value with ties
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test
##
## data:  data_5_runs$Score and data_5_runs$Search_method
##
##        HC_T    HC_TR   HCR_T   HCR_TR
## HC_TR  1       -       -       -
## HCR_T  1       1       -       -
## HCR_TR 1       1       1       -
## RS     1.5e-05 9.5e-06 3.0e-07 3.0e-07
##
## P value adjustment method: holm
```

Don't sweat the warnings at the moment – these are mostly a function of only having 5 runs for each treatment at this point. The important things are the p-values in the ASCII art table.

These tell us that 'Random_Search' is *very* strongly different from all of the others:

- 1.5e-05 with `Hill_Climber_and_Tweaker`
- 9.5e-06 with `Hill_Climber_and_Tweaker_with_Rates`
- 3.0e-07 with `Hill_Climber_Random_Restarts_and_Tweaker` and `Hill_Climber_Random_Restarts_and_Tweaker_with`
  so those differences are *very* unlikely to be the result of random chance.

The difference between the various Hill Climber functions and Tweaker functions is less clear.

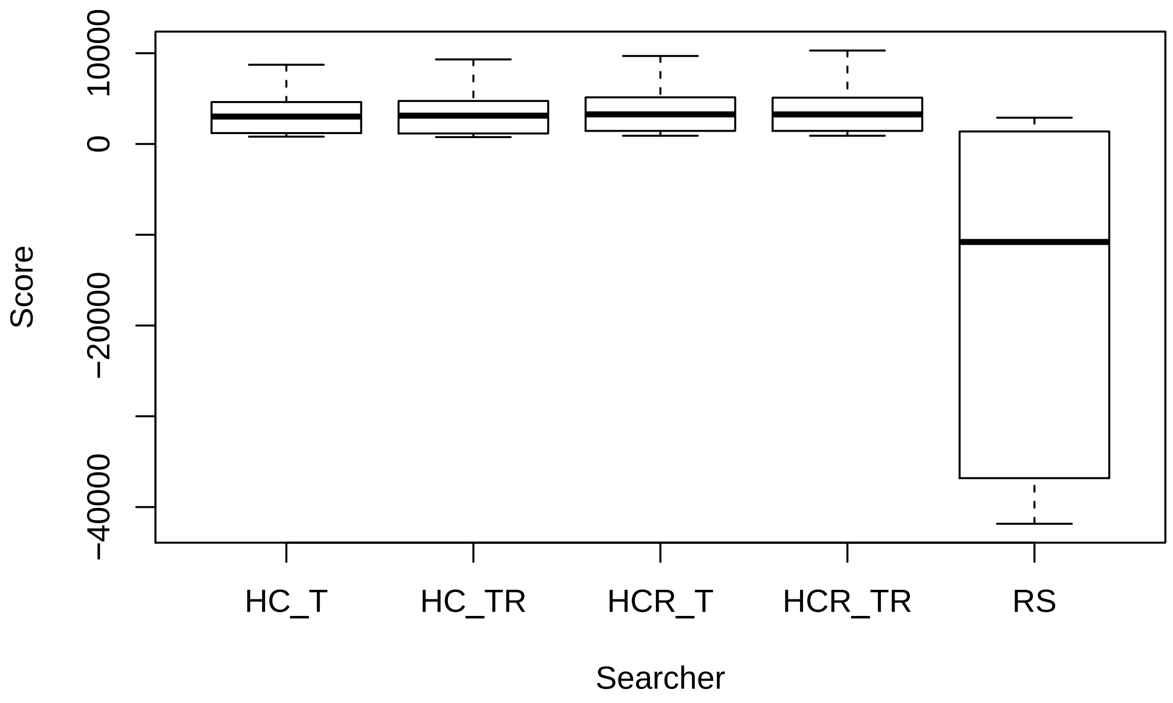
30 runs per treatment

One of the nifty things about evolutionary computation, though, is that we can usually do more runs, and doing more runs will often firm up weak p-values *if* there is indeed a meaningful difference. (And if there's not, then more runs will usually help clarify that as well.)

So let's load up data where we did *30* runs for each treatment, and plot those results.

```
data_30_runs <- read.csv("/home/casal033/ECAI/simple-search/data/output30run10000tries.txt", sep="",hea

plot(data_30_runs$Score ~ data_30_runs$Search_method,
     xlab="Searcher", ylab="Score")
```

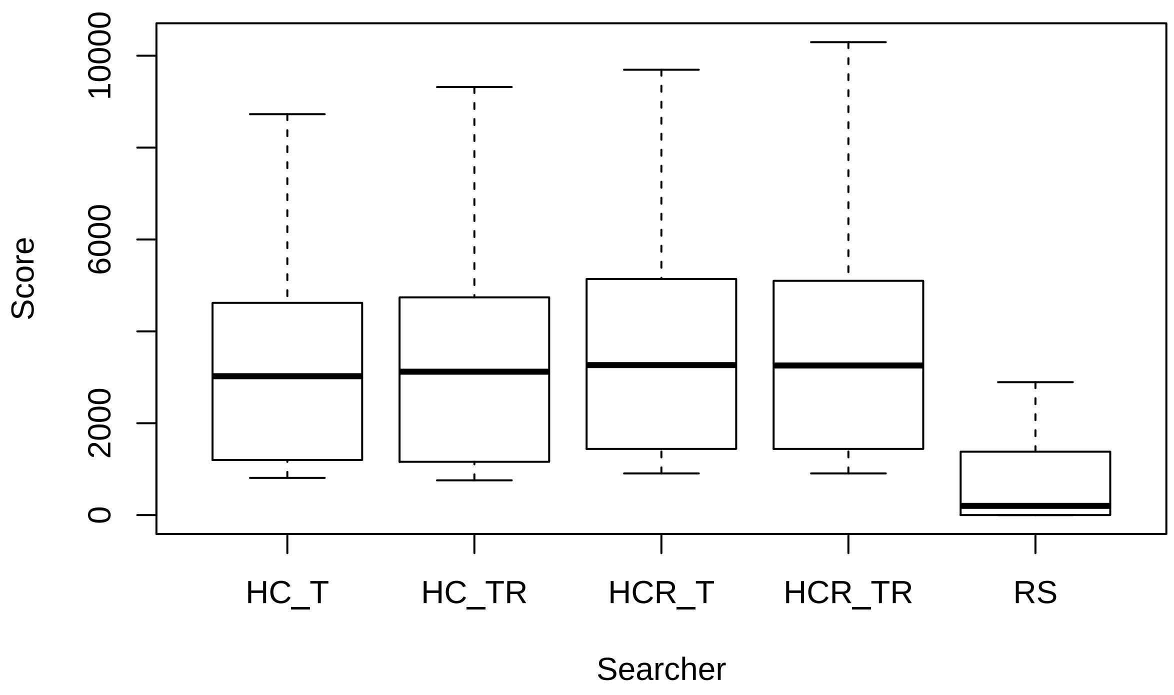


Yikes – there are negative values for some of the `random_Search` runs! This is because for some of the harder problems with smaller `Max_evals` the randomly generated knapsacks weren't able to find an answer outside of the "illegal" zone, so the best answer at the end of the run was still negative.

To make our comparisons a little more apples to apples, let's create a new column (`Non_negative_score`) that has negative scores converted to zeros. One of the very cool features of R is that you can add calculated columns like this quite easily.

```
data_30_runs$Non_negative_score = ifelse(data_30_runs$Score<0, 0, data_30_runs$Score)

plot(data_30_runs$Non_negative_score ~ data_30_runs$Search_method,
     xlab="Searcher", ylab="Score")
```



The general picture looks similar to the earlier situation, with the left boxplots definitely looking better than the `random_search`, but the situation between the rest of the left boxplots are not entirely clear. So let's run a test.

```
pairwise.wilcox.test(data_30_runs$Non_negative_score, data_30_runs$Search_method)
```

```
##
##  Pairwise comparisons using Wilcoxon rank sum test
##
## data:  data_30_runs$Non_negative_score and data_30_runs$Search_method
##
##        HC_T   HC_TR  HCR_T  HCR_TR
## HC_TR  1.000  -      -      -
## HCR_T  0.048  0.048  -      -
## HCR_TR 0.048  0.048  1.000  -
## RS     <2e-16 <2e-16 <2e-16 <2e-16
##
## P value adjustment method: holm
```

Now all the differences with `Random_Search` are strongly significant! ($< 2^{-16}$ is R's way of saying "Wow – that's as significant as I know how to possibly talk about!".) This is not uncommon if you've done a fair number of runs – if there's a meaningful difference you can do enough runs to make that super clear.

Comparing `Hill_Climber_and_Tweaker` with `Hill_Climber_and_Tweaker_with_Rates`, we get a p-value of 1. This tells us we're very sure of having the same outcome using either function combinations. We also see this with `Hill_Climber_Random_Restarts_and_Tweaker` and `Hill_Climber_Random_Restarts_and_Tweaker_with_Rates`.

When comparing `Hill_Climber_and_Tweaker` OR `Hill_Climber_and_Tweaker_with_Rates` with `Hill_Climber_Random_Restarts_and_Tweaker` OR `Hill_Climber_Random_Restarts_and_Tweaker_with_Rates` we get a p-value of 0.048 which suggests that either of the first two options or second two options will have the higher score outcome. We can see from the `non-negative` plot that the medians of the second two options have a slightly higher score than the first two options, so they are a tad more likely to have a higher score.

How do things change by problem?

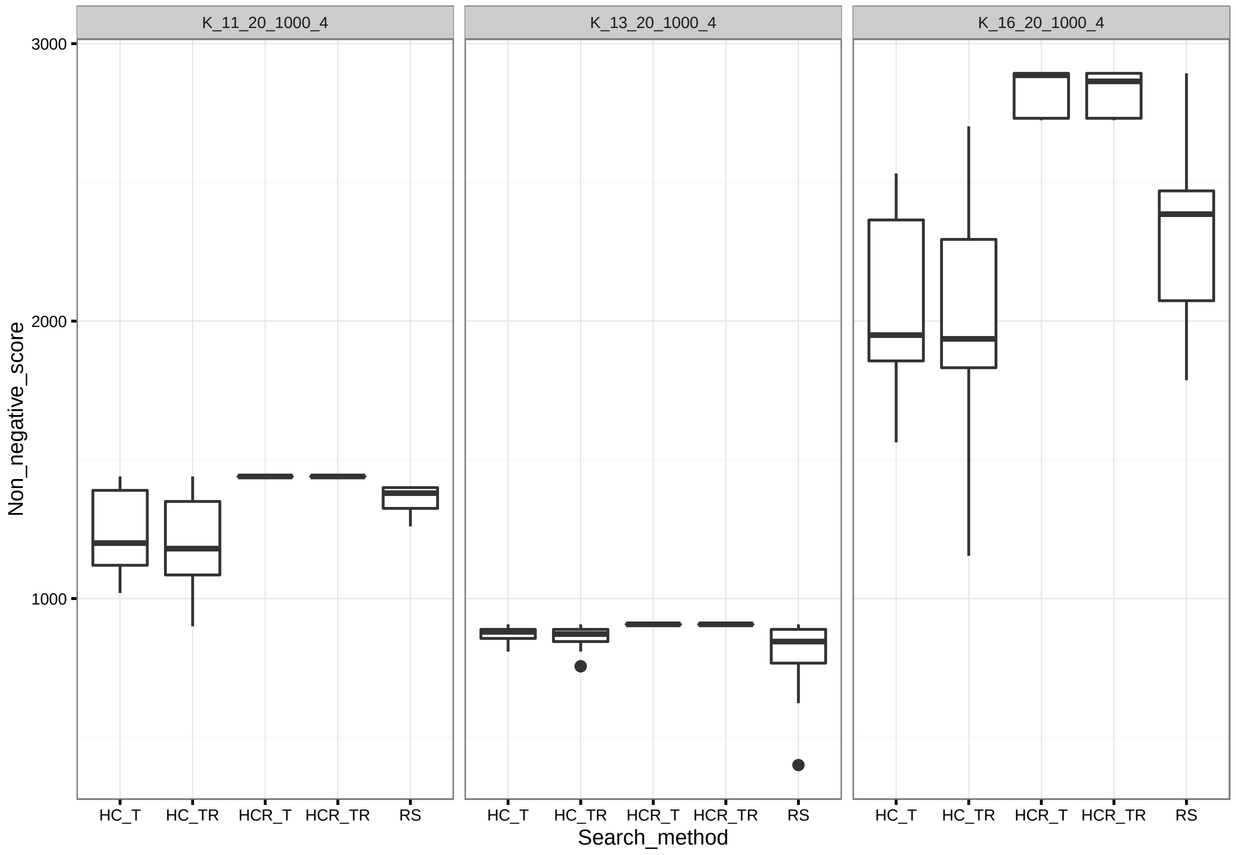
Let's load the `ggplot2` package

```
library("ggplot2")
```

and use it's nice faceting features to see how the differences change from problem to problem. Because I was a dope and didn't output the number of items as a column in my data, I'm going to use a pretty ugly `subset` to separate the 20 item cases from the 200 item cases; if I had more time I'd go back and regenerate the data with the number of items (and probably the "kind" of problem, i.e, the 11, 13, or 16 in the name) as a column so I could easily pull out runs with that property.

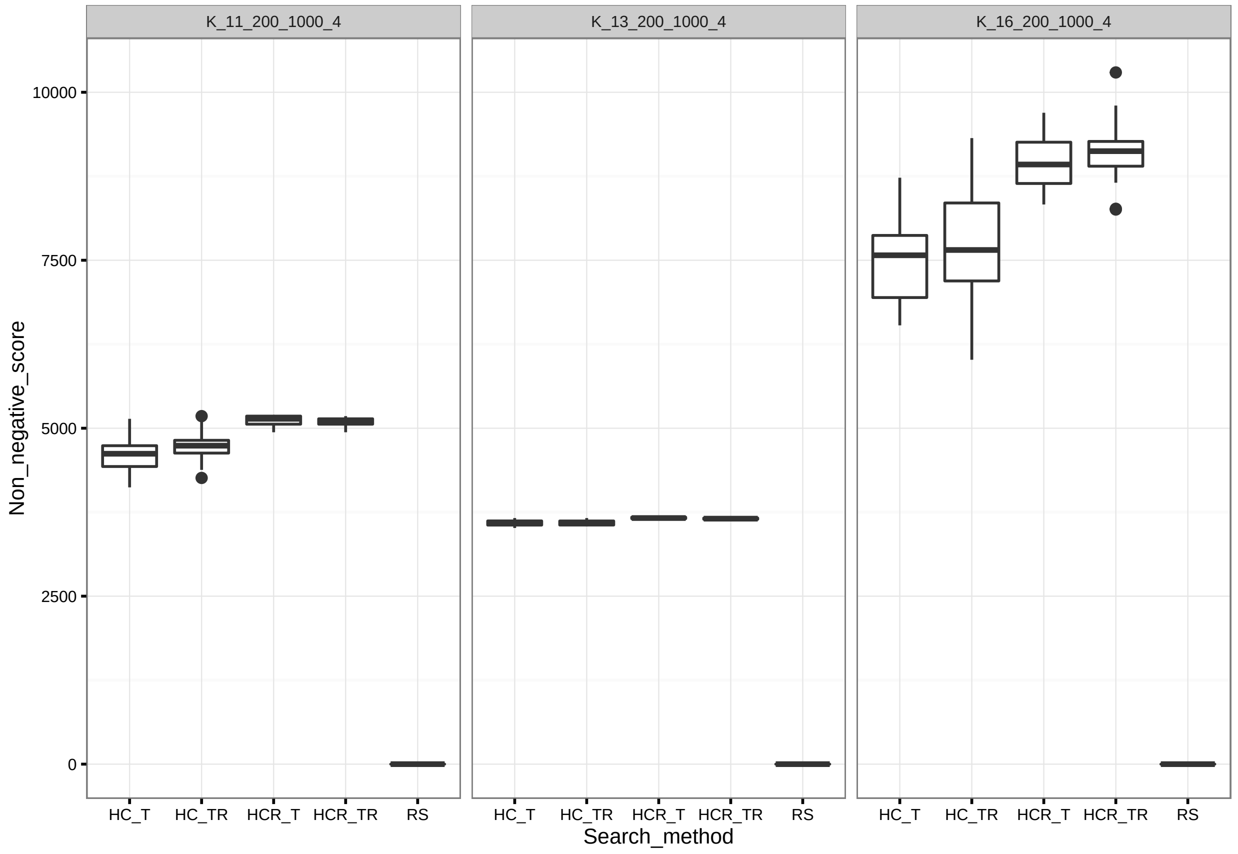
```
twenty_item_problems = subset(data_30_runs, Problem=="K_11_20_1000_4" | Problem=="K_13_20_1000_4" | Prol

ggplot(twenty_item_problems, aes(Search_method, Non_negative_score)) + geom_boxplot() + facet_grid(. ~ I
```



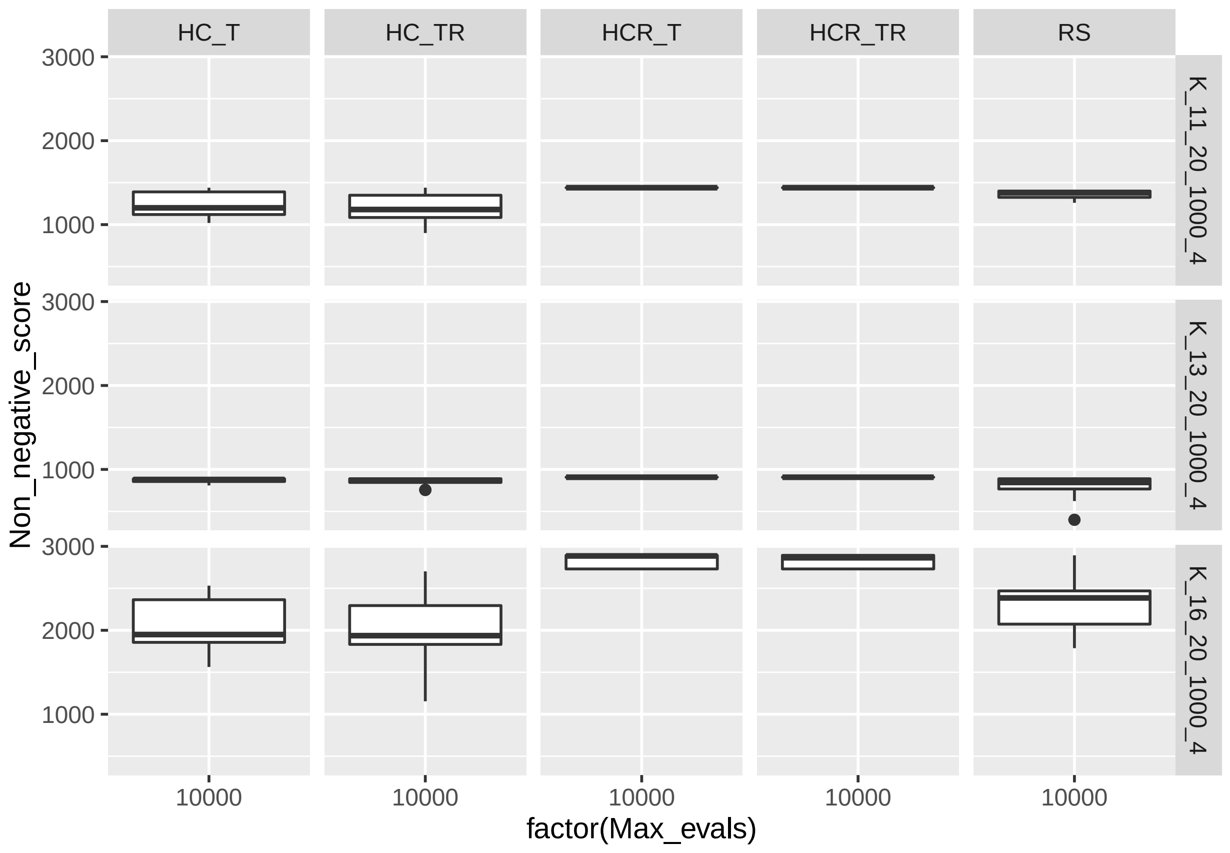
```
two_hundren_item_problems = subset(data_30_runs, Problem=="K_11_200_1000_4" | Problem=="K_13_200_1000_4

ggplot(two_hundren_item_problems, aes(Search_method, Non_negative_score)) + geom_boxplot() + facet_grid
```

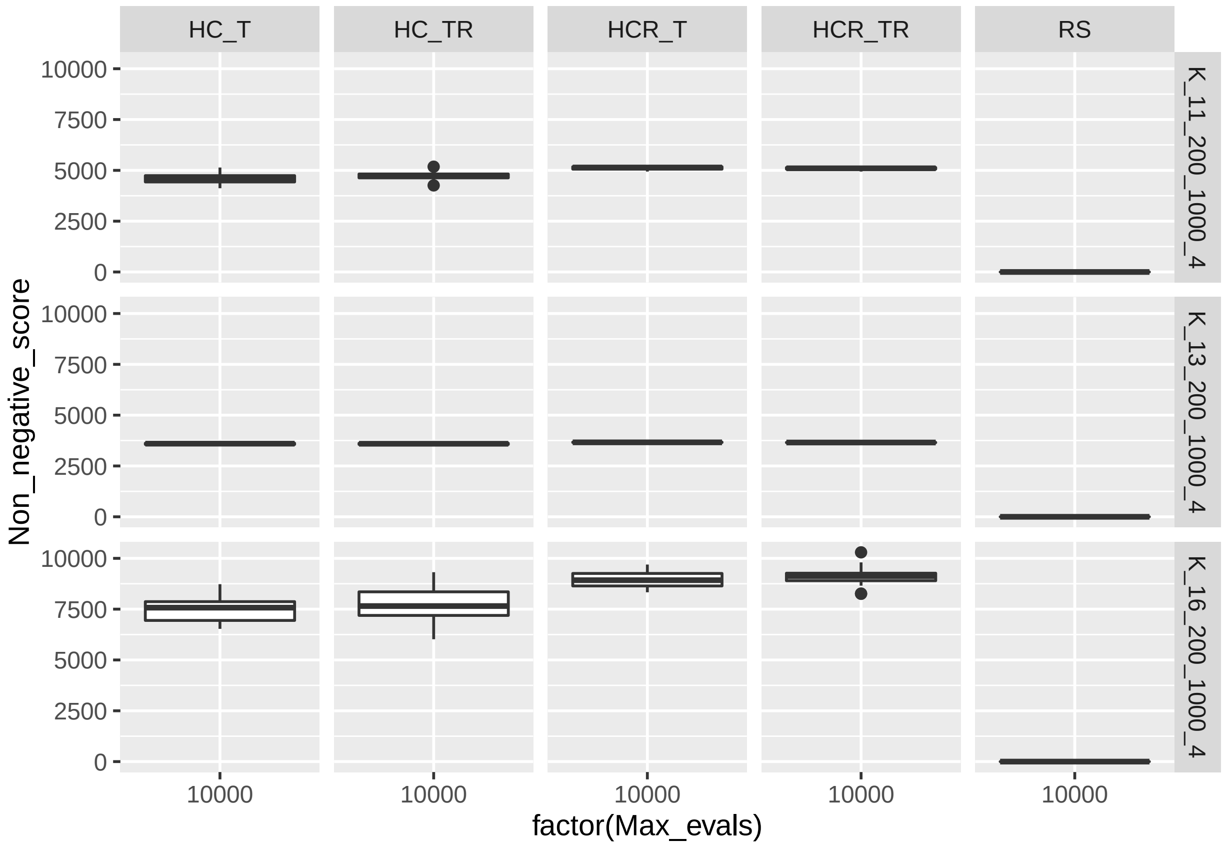


Or, to see it a different way:

```
ggplot(twenty_item_problems, aes(factor(Max_evals), Non_negative_score)) + geom_boxplot() + facet_grid(
```

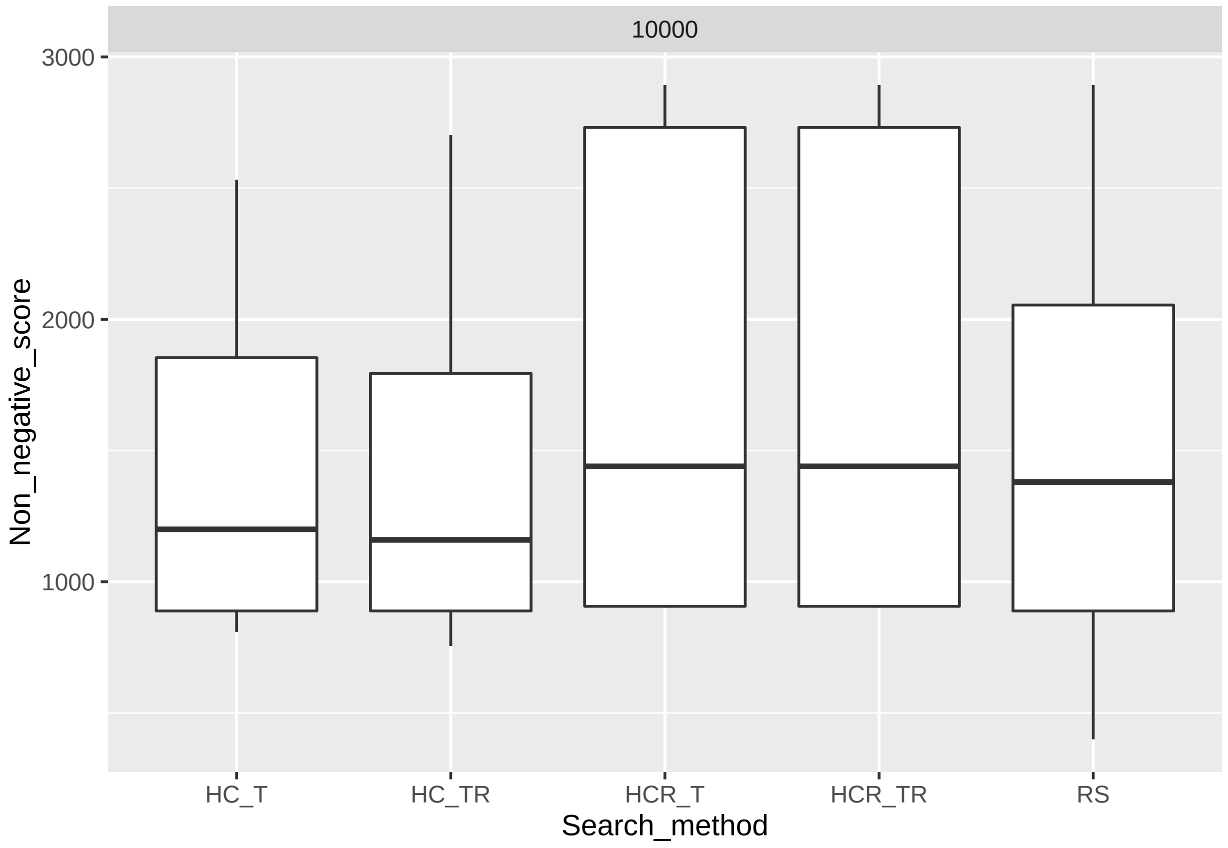


```
ggplot(two_hundren_item_problems, aes(factor(Max_evals), Non_negative_score)) + geom_boxplot() + facet_
```

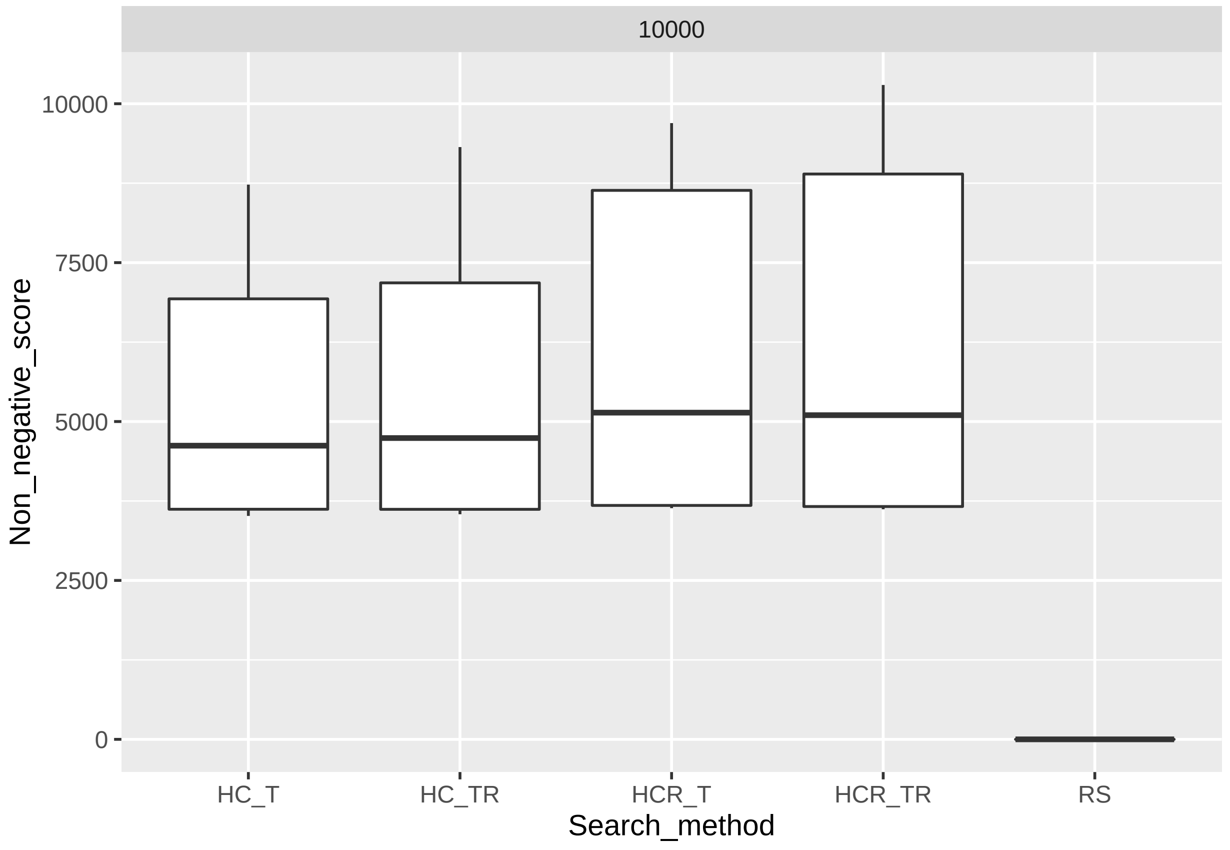


How do things change by maximum number of evals?

```
ggplot(twenty_item_problems, aes(Search_method, Non_negative_score)) + geom_boxplot() + facet_grid(. ~ 
```



```
ggplot(two_hundren_item_problems, aes(Search_method, Non_negative_score)) + geom_boxplot() + facet_grid
```

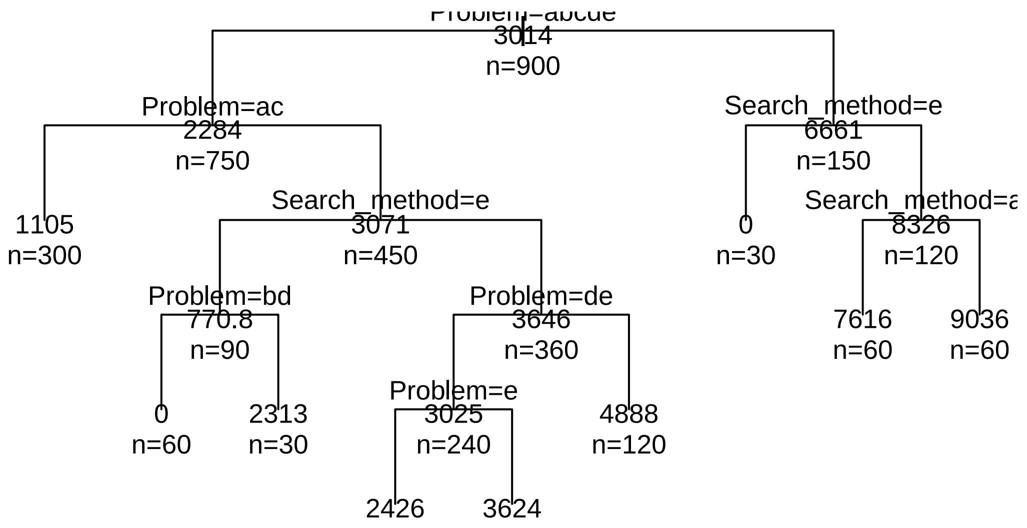


```
library("rpart")

rp <- rpart(Non_negative_score ~ Search_method + Problem + Max_evals, data=data_30_runs)
rp
```

```
## n= 900
##
## node), split, n, deviance, yval
##       * denotes terminal node
##
##  1) root 900 5978367000.0 3013.7690
##    2) Problem=K_11_20_1000_4,K_11_200_1000_4,K_13_20_1000_4,K_13_200_1000_4,K_16_20_1000_4 750 181965
##      4) Problem=K_11_20_1000_4,K_13_20_1000_4 300   19504530.0 1104.5100 *
##      5) Problem=K_11_200_1000_4,K_13_200_1000_4,K_16_20_1000_4 450 1104159000.0 3070.8840
##       10) Search_method=RS 90  109083900.0  770.8444
##         20) Problem=K_11_200_1000_4,K_13_200_1000_4 60          0.0    0.0000 *
##         21) Problem=K_16_20_1000_4 30    2127693.0 2312.5330 *
##       11) Search_method=HC_T,HC_TR,HCR_T,HCR_TR 360  399929200.0 3645.8940
##         22) Problem=K_13_200_1000_4,K_16_20_1000_4 240  113374400.0 3025.0920
##           44) Problem=K_16_20_1000_4 120   27017140.0 2426.0170 *
##           45) Problem=K_13_200_1000_4 120     223466.7 3624.1670 *
##         23) Problem=K_11_200_1000_4 120    9069650.0 4887.5000 *
##    3) Problem=K_16_200_1000_4 150 1764377000.0 6660.9400
##      6) Search_method=RS 30          0.0    0.0000 *
##      7) Search_method=HC_T,HC_TR,HCR_T,HCR_TR 120  100572100.0 8326.1750
##       14) Search_method=HC_T,HC_TR 60   30644480.0 7616.1330 *
##        15) Search_method=HCR_T,HCR_TR 60    9428566.0 9036.2170 *
```

```r
plot(rp, uniform=TRUE)
text(rp, use.n = TRUE, cex=0.8, all=TRUE)
```


Problem=abcde
3014
n=900
Problem=ac
2284
n=750
Search_method=e
6661
n=150
1105
n=300
Search_method=e
3071
n=450
0
n=30
Search_method=a
8326
n=120
Problem=bd
770.8
n=90
Problem=de
3646
n=360
7616
n=60
9036
n=60
0
n=60
2313
n=30
Problem=e
3025
n=240
4888
n=120
2426
3624