

# Rq2 reproduction

Jeffrey M. Young

15 May 2020

In this walk-through we'll perform the plot generation and significance testing for rq2. We assume you have read the `rq1_rq3` and `sigTest` walk-throughs, in that order.

## Preliminaries

First, libraries, note that I have silenced the shadowing warnings from R:

```
library(ggplot2)    #plotting
library(dplyr)      #dataframe manipulation
library(tidyr)      #dataframe manipulation
library(broom)      #for the tidy function
library(scales)     #for scientific function
library(latex2exp)  #for TeX in Labels
library(ggpubr)     #for stat_cor
```

Load the data, and perform some simple munging to select relevant columns

```
finResultsFile <- "../data/fin_comp_data.csv"
autoResultsFile <- "../data/auto_comp_data.csv"

finData <- read.csv(file=finResultsFile) %>%
  mutate(Algorithm = as.factor(Algorithm), Config = as.factor(Config))

autoData <- read.csv(file=autoResultsFile) %>%
  mutate(Algorithm = as.factor(Algorithm), Config = as.factor(Config))

finDF <- finData %>% mutate(data = "Fin") %>%
  select(Mean, Algorithm, CompressionRatio, data, ChcCount, PlainCount, Config)

autoDF <- autoData %>% mutate(data = "Auto") %>%
  select(Mean, Algorithm, CompressionRatio, data, ChcCount, PlainCount, Config)
```

## Visualization

Now the sharing ratio must be calculated, this is performed via a `spread` and `mutate` operation. `spread` transforms long data into wide data, note that it has been recently deprecated and replaced with `pivot_wider`, see the `tidyverse` documentation for details here. We use `spread` just for convenience and familiarity. Now we find the sharing ratio for each Algorithm, and construct a data frame for each dataset:

```
## finding the sharing ratios for auto
autoPtoP <- autoDF %>% group_by(Algorithm, Config) %>%
  filter(Algorithm == "v-->p" || Algorithm == "p-->p") %>%
  spread(Algorithm, Mean) %>% mutate(MeanRatio = (`v-->p` / `p-->p`)) %>%
  select(-"v-->p", -"p-->p") %>% mutate(Algorithm = "p\u27f6p")
```

```

autoVtoV <- autoDF %>% group_by(Algorithm, Config) %>%
  filter(Algorithm == "v-->p" || Algorithm == "v-->v") %>%
  spread(Algorithm, Mean) %>% mutate(MeanRatio = (`v-->p` / `v-->v`)) %>%
  select("-v-->v", "-v-->p") %>% mutate(Algorithm = "v\u27f6v")

autoPtoV <- autoDF %>% group_by(Algorithm, Config) %>%
  filter(Algorithm == "v-->p" || Algorithm == "p-->v") %>%
  spread(Algorithm, Mean) %>% mutate(MeanRatio = (`v-->p` / `p-->v`)) %>%
  select("-v-->p", "-p-->v") %>% mutate(Algorithm = "p\u27f6v")

## construct auto data frame
autoSharedDF <- rbind(autoPtoP, autoVtoV, autoPtoV)

```

and for financial:

```

## finding the sharing ratios for fin
finPtoP <- finDF %>% group_by(Algorithm, Config) %>%
  filter(Algorithm == "v-->p" || Algorithm == "p-->p") %>%
  spread(Algorithm, Mean) %>% mutate(MeanRatio = (`v-->p` / `p-->p`)) %>%
  select("-v-->p", "-p-->p") %>% mutate(Algorithm = "p\u27f6p")

finVtoV <- finDF %>% group_by(Algorithm, Config) %>%
  filter(Algorithm == "v-->p" || Algorithm == "v-->v") %>%
  spread(Algorithm, Mean) %>% mutate(MeanRatio = (`v-->p` / `v-->v`)) %>%
  select("-v-->v", "-v-->p") %>% mutate(Algorithm = "v\u27f6v")

finPtoV <- finDF %>% group_by(Algorithm, Config) %>%
  filter(Algorithm == "v-->p" || Algorithm == "p-->v") %>%
  spread(Algorithm, Mean) %>% mutate(MeanRatio = (`v-->p` / `p-->v`)) %>%
  select("-v-->p", "-p-->v") %>% mutate(Algorithm = "p\u27f6v")

## construct fin data frame
finSharedDF <- rbind(finPtoP, finVtoV, finPtoV)

```

Now the data frames are combined and the plot can be generated:

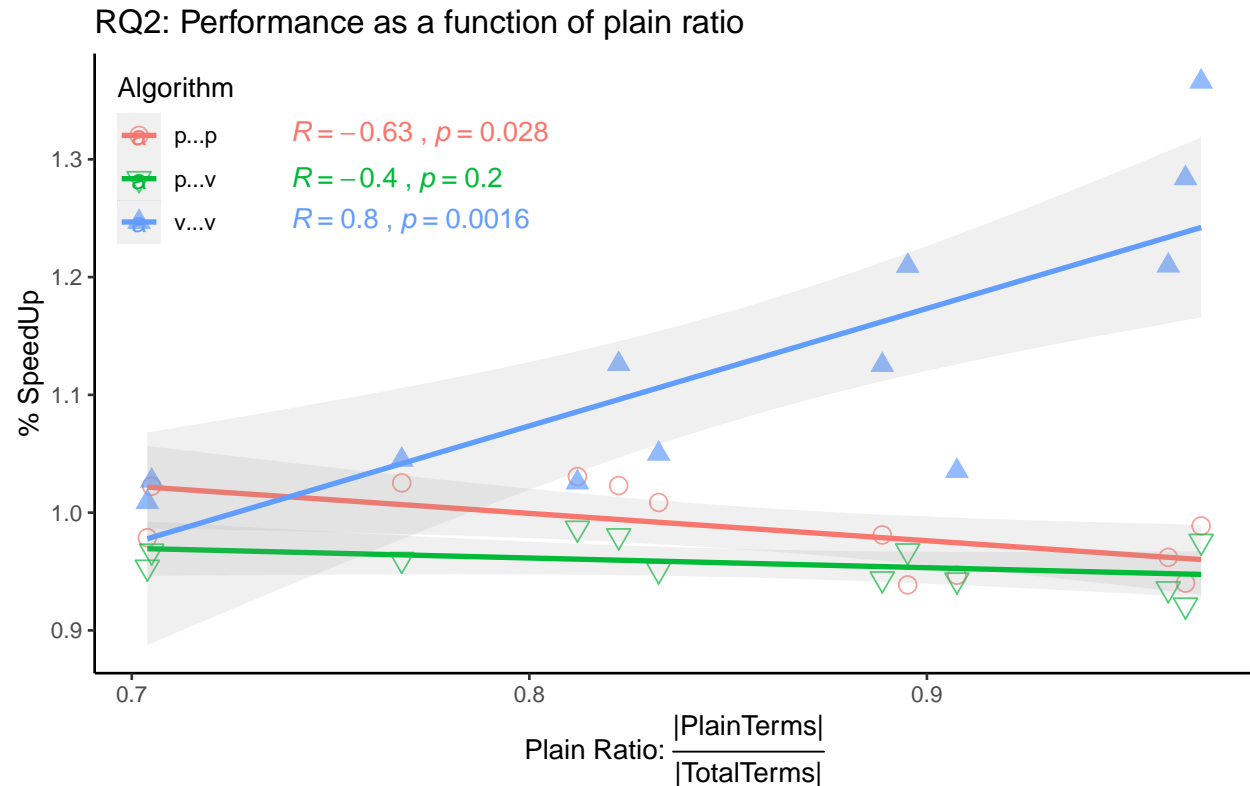
```

## make the data frame
df <- rbind(autoSharedDF, finSharedDF) %>%
  ## calculate the ratio of plain terms to total terms, i.e., the sharing ratio
  mutate(PlainRatio = PlainCount / (ChcCount + PlainCount),
  ## get the converse ratio at the same time
  ChcRatio = ChcCount / (ChcCount + PlainCount))

## make the plot as a function of plain ratio to mean ratio
ggplot(df, mapping = aes(x=PlainRatio, y=MeanRatio, colour = Algorithm, shape = Algorithm)) +
  ## it is a scatter plot
  geom_point(size=3, alpha=0.7) +
  ylab("% SpeedUp") +
  ## insert LaTeX into the x-axis labels
  xlab(TeX("Plain Ratio:  $\frac{|Plain Terms|}{|Total Terms|}$ ")) +
  ## manually scale shapes so the algorithms have the same shape over different plots
  scale_shape_manual(values = c(1,6,17)) +
  ## add the confidence intervals, using a linear regression, alpha makes these more
  ## transparent, se adds the shadings for the 95% confidence intervals
  geom_smooth(method=lm, formula = y ~ x, se=TRUE, alpha=0.15) +

```

```
ggtitle("RQ2: Performance as a function of plain ratio") +
theme_classic() +
## stat_cor calculates the R^2 values, that are placed next to the legend
stat_cor(aes(color=Algorithm),
         label.x=0.74, label.y.npc=c(0.91, 0.89, 0.88)) +
theme(legend.position = c(0.07, 0.83))
```



## Statistical Significance

We perform the same two way ANOVA to ensure fair comparisons:

```
### Perform the anova
res.aov <- aov(MeanRatio ~ PlainRatio * Algorithm, data = df)
res.aov
```

```
## Call:
## aov(formula = MeanRatio ~ PlainRatio * Algorithm, data = df)
##
## Terms:
##          PlainRatio  Algorithm PlainRatio:Algorithm  Residuals
## Sum of Squares  0.01522695  0.19480856          0.08822280  0.06465027
## Deg. of Freedom           1           2              2          30
##
## Residual standard error: 0.04642207
## Estimated effects may be unbalanced
```

```
### Check the summary to see what is significant, all of it is as expected
summary(res.aov)
```

```
##              Df Sum Sq Mean Sq F value    Pr(>F)
```

```
## PlainRatio          1 0.01523 0.01523   7.066   0.0125 *
## Algorithm           2 0.19481 0.09740  45.199 8.86e-10 ***
## PlainRatio:Algorithm 2 0.08822 0.04411  20.469 2.48e-06 ***
## Residuals          30 0.06465 0.00216
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

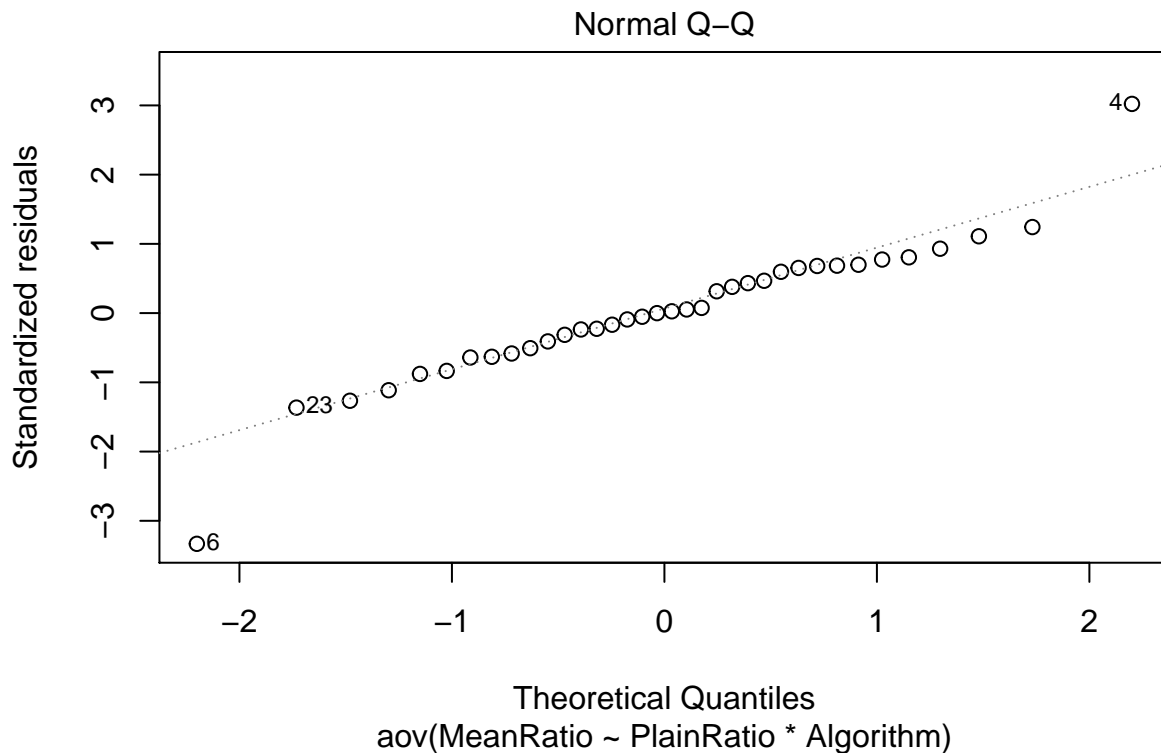
We see that Algorithm, PlainRatio, and their interaction are significant. This makes sense given the nature of the tool and analysis. We perform a Tukey pair-wise comparison to check the significantly different comparisons in the dataset.

```
### perform the pair-wise Tukey comparison to test the difference
### between groups
TukeyHSD(res.aov) %>% tidy %>% mutate(pVal = scientific(adj.p.value, 3))
```

```
## # A tibble: 3 x 7
##   term      comparison estimate conf.low conf.high adj.p.value pVal
##   <chr>      <chr>      <dbl>   <dbl>   <dbl>     <dbl> <chr>
## 1 Algorithm p v-p p    -0.0300 -0.0768   0.0167 0.268      2.68e-01
## 2 Algorithm v v-p p     0.139    0.0921   0.186 0.000000109 1.09e-07
## 3 Algorithm v v-p v     0.169    0.122    0.216 0.00000000185 1.85e-09
```

We observe that v-->v differs from the other algorithms, and the difference is statistically significant. Next, we check the residuals to assess normality.

```
res.ass <- plot(res.aov, 2)
```



Residuals look good and there are three outliers, which agrees with other analyses. To verify, we perform the Shapiro-Wilks test, identically to `rq3`.

```
aov.resids <- residuals(object=res.aov)
```

```
## Shapiro-Wilk normality test
```

```
shapiro.test(x = aov.resids)
```

```
##  
## Shapiro-Wilk normality test  
##  
## data: aov.resids  
## W = 0.92064, p-value = 0.01315
```

Unfortunately, We see that again the dataset violates the assumptions of the ANOVA test, so we perform the Kruskal-Wallis test instead:

```
## Algorithms are significant  
kruskal.test(MeanRatio ~ Algorithm, df)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: MeanRatio by Algorithm  
## Kruskal-Wallis chi-squared = 23.73, df = 2, p-value = 7.033e-06
```

Algorithms are found to be statistically different as expected.

```
## Plain ratio is not significant!!  
kruskal.test(MeanRatio ~ PlainRatio, df)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: MeanRatio by PlainRatio  
## Kruskal-Wallis chi-squared = 4.2733, df = 11, p-value = 0.9612
```

Plain ratio, interestingly is found not to be statistically different, which agrees with the ANOVA analysis.

```
## Interaction is not significant surprisingly  
rq2.inters <- interaction(df$Algorithm, df$PlainRatio)  
kruskal.test(MeanRatio ~ rq2.inters, df)
```

```
##  
## Kruskal-Wallis rank sum test  
##  
## data: MeanRatio by rq2.inters  
## Kruskal-Wallis chi-squared = 35, df = 35, p-value = 0.4682
```

However, when accounting for Algorithms, i.e., when analyzing the interaction between the two, we find the interaction is just barely significant. We perform the Pairwise Wilcoxon test to check what exactly is significant:

```
## Show the pairs which are significant  
pairwise.wilcox.test(df$MeanRatio, df$Algorithm,  
                     p.adj="bonf" , exact=TRUE, paired=FALSE) %>%  
  tidy %>% arrange(p.value)
```

```
## # A tibble: 3 x 3  
##   group1 group2 p.value  
##   <chr> <chr>    <dbl>  
## 1 v v    p v    0.00000222  
## 2 v v    p p     0.0000666  
## 3 p v    p p     0.116
```

We find that  $v \rightarrow v$  is statistically different from the other algorithms, confirming the hypothesis for **rq2**