Causal impact study

GB

February 10, 2017

1. Getting data file and setting experiment start, end and rollout times.

```
#myfile<-file.choose()
myfile<-file('client_04-01-2015--11-21-2015.csv', 'r')
startime<-as.Date('2016-04-01')
endtime<-as.Date('2016-06-01')
rolout<-as.Date('2016-09-01')
mydata<-read.csv(myfile, header = TRUE)
close(myfile,'r')
```

checking if there is data prior to experiment- answer into variable data_for_strata

```
mydata$ga.date<-as.Date(mydata$ga.date)
sprintf('start date in file: %s', min(mydata$ga.date))
```

```
## [1] "start date in file: 2015-04-01"
```

```
sprintf('end date in file: %s', max(mydata$ga.date))
```

```
## [1] "end date in file: 2016-11-21"
```

```
if (min(mydata$ga.date) >= startime) { print('Warning: no data prior to experiment start!')
  data_for_strata<-FALSE} else {data_for_strata<-TRUE}
data_for_strata
```

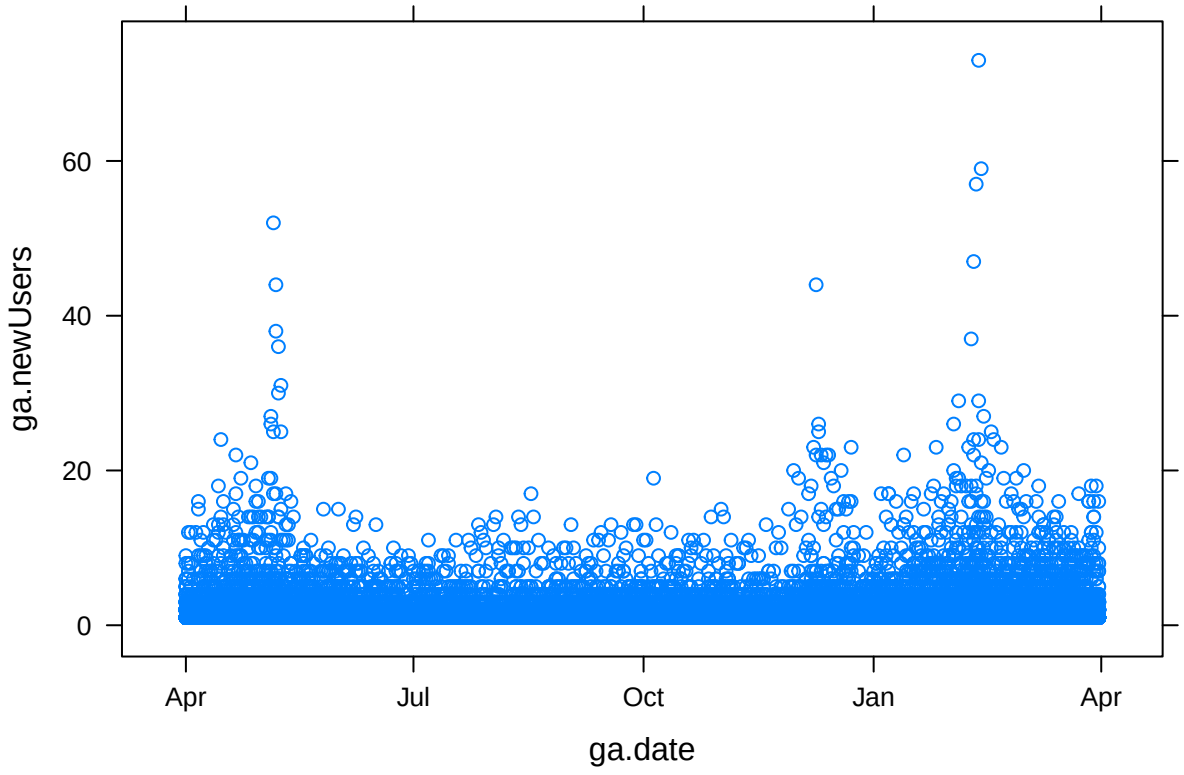
```
## [1] TRUE
```

```
library(CausalImpact)
library(dplyr)
library(lattice)
```

2. Getting to know more about the loaded datafile.

Selecting only data from time period before the experiment start as described in Etsy experiment, and plotting it. After some trials I noticed that largest counts were in the empty landingPagePath. Not sure if that was real path, eliminated it from stratas (change variable eliminate_empty_path to False - to keep it).

```
eliminate_empty_path=TRUE
if (eliminate_empty_path) {mydata<-mydata[!as.character(mydata$ga.landingPagePath)== '/',]}
prior_to_exp<-mydata[mydata$ga.date < startime,]
xyplot(ga.newUsers~ga.date, prior_to_exp)
```



Checking how many unique paths are in landingPagePath column

```
paths<-unique(mydata$ga.landingPagePath)    # selects unique landingPagePath's for the entire data file
sprintf('total different landing pages in the entire data file: %s', length(paths))
```

```
## [1] "total different landing pages in the entire data file: 7225"
```

```
if (data_for_strata) {paths2<-unique(prior_to_exp$ga.landingPagePath) #selects unique landingPagePath's
sprintf('total different landing pages prior to experiment: %s', length(paths2))}
```

```
## [1] "total different landing pages prior to experiment: 6102"
```

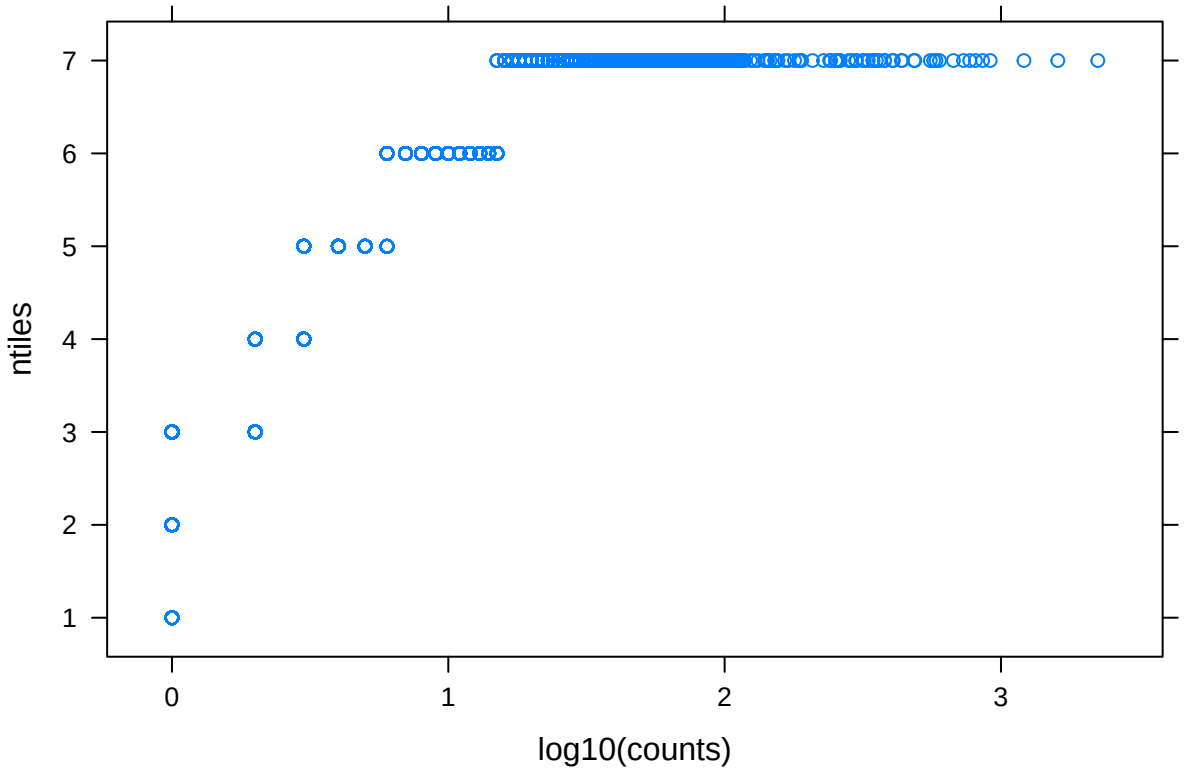
Optional, here default eval=FALSE, chuncks of code can be switched on/off by changing eval=TRUE/FALSE. Ploting some times series for the random NP=10 landing Pages, the number can be changed below, NP variable, user selectable.

```
NP<-10
testpaths<-sample(length(paths), NP)
data2<-mydata[mydata$ga.landingPagePath %in% paths[testpaths], ]
xyplot(ga.newUsers~ga.date|ga.landingPagePath, data2)
rm(data2)
```

3. Building stratas

Generating ntiles and plotting ntiles versus counts. counts here are total NewUsers per page. Number of ntiles is kn=7. can be changed below, user selectable.

```
kn=7 # number of ntiles, arbitrary number, can be changed
sumdata<-group_by(prior_to_exp, ga.landingPagePath) %>% summarise(counts=sum(ga.newUsers)) # getting
sumdata<-sumdata[order(-sumdata$counts), ]   # ordering in decreasing order
sumdata$ntiles<-ntile(sumdata$counts, kn)  # creating ntiles column and assinging ntile number to eac
xyplot(ntiles~log10(counts), sumdata)
```



repackaging data into different kg=5 groups. Number kg can be changed. number of unique landingPagePath's per ntile is printed out below the next script chunk

```
kg=5  # kg - number of test groups, user's choice
  groupeddata<-data.frame()  # new data frame where results will be
  for (i in 1:kn) {   # looping through ntiles
    sampledsumdata<-filter(sumdata, ntiles==i)   # selecting data for specific ntile only
    print(length(sampledsumdata$ga.landingPagePath))  # prints the number of paths per ntile
    set.seed(123) # see coment about it chapter 5.
    sampledsumdata$group<-sample(rep_len(1:kg, length(sampledsumdata$counts)))    # add group column ma
    groupeddata<-rbind(groupeddata, sampledsumdata)  # combine into final table
    rm(sampledsumdata)  # remove extra variables
  }
```

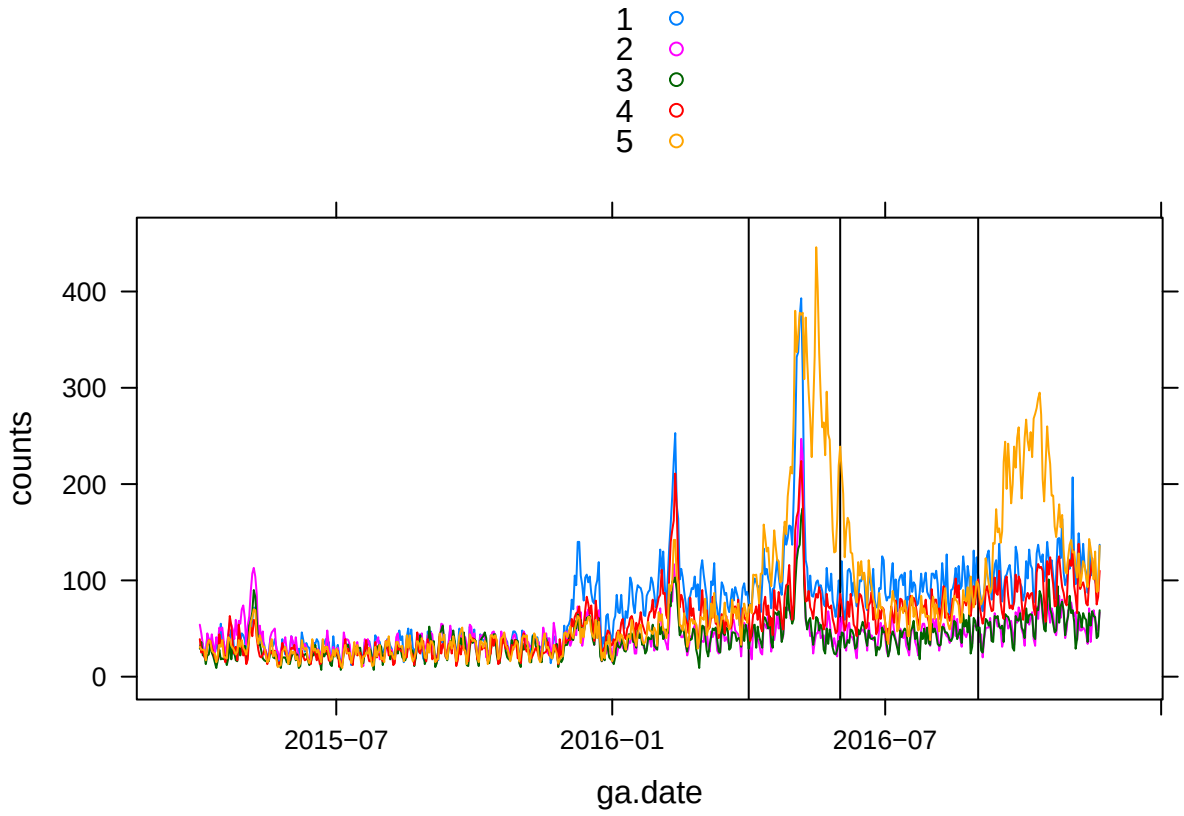
```
## [1] 872
## [1] 872
## [1] 872
## [1] 871
## [1] 872
## [1] 872
## [1] 871
```

here groupeddata contains landingPagePath, total counts per that path, ntile and group for that landingPagePath. this table is used to map landingPagePath to a group in the original data file

```
x<-groupeddata[match(mydata$ga.landingPagePath,groupeddata$ga.landingPagePath),4]
mydata$group<-as.factor(x$group)
rm(x)
```

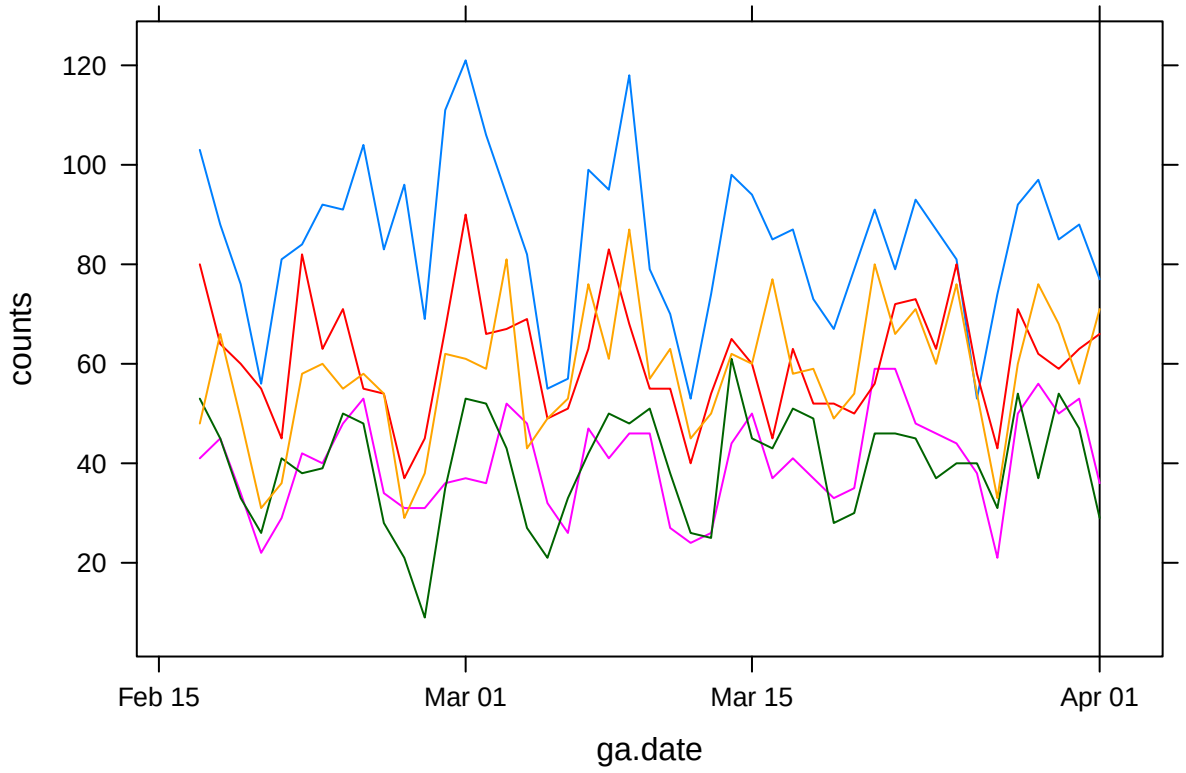
there were some landingPagePath values in data file that were added not existant during 'prior to experiment' time period. That's the time period used to define ntiles and groups for specific landingPagaPath's. Therefore, those extra landingPagePath's do not get assigned a group.

```
mydata_na<-mydata[!is.na(mydata$group),]   #removes NA, puts result into mydata_na
my.lines<-c(startime, endtime, rolout) # marks vertical lines in plot
groupmydata<-group_by(mydata_na, group, ga.date) %>% summarise(counts=sum(ga.newUsers)) # agregates per
xyplot(counts~ga.date, groups=group, groupmydata, type='l',auto.key=T, panel = panel.superpose,
       panel.groups = function(..., group.number) {
           panel.abline(v = my.lines[group.number])
           panel.xyplot(...)})
```



plotting for specific time window, st and en define time period, values can be changed below:

```
st<-as.Date('2016-02-16')
en<-as.Date('2016-04-01')
groupmydata2<-groupmydata[groupmydata$ga.date<=en & groupmydata$ga.date>st, ]
xyplot(counts~ga.date, groups=group, groupmydata2, type='l', panel = panel.superpose,
       panel.groups = function(..., group.number) {
           panel.abline(v = my.lines[group.number])
           panel.xyplot(...) })
```



what group to pick as a control? Assuming the plot above is for the time period that is right before the experiment, group that is very similar to other groups (well correlated with the study group as well), is a good candidate.

4. Basic statistics

```
datastat<-as.data.frame
groupmydata$Month_Yr <- format(groupmydata$ga.date, "%Y-%m")
datastat<-group_by(groupmydata, Month_Yr , group) %>% summarise(avg_values=mean(counts), std_values=sd(c
datastat
```

```
## Source: local data frame [100 x 4]
## Groups: Month_Yr [?]
##
##    Month_Yr  group avg_values std_values
##       <chr> <fctr>      <dbl>      <dbl>
## 1   2015-04      1   35.13333   9.058177
## 2   2015-04      2   43.76667  13.255274
## 3   2015-04      3   22.83333   7.575293
## 4   2015-04      4   31.33333  11.844782
## 5   2015-04      5   29.00000   7.002463
## 6   2015-05      1   36.32258  15.149009
## 7   2015-05      2   46.19355  27.208356
## 8   2015-05      3   29.41935  21.153052
```

```
## 9   2015-05      4   27.25806  12.492578
## 10  2015-05      5   31.25806  15.899618
## # ... with 90 more rows
```

table above shows average and standard deviations per group per month. Below is t-test. P values are printed, choose time periods for which to calculate p values, user adjustable:

```
stt<-as.Date('2016-02-15')  # start t-test at stt
ent<-as.Date('2016-03-31')  # end t-test at ent
ttestdata<-groupmydata[groupmydata$ga.date<=ent & groupmydata$ga.date>stt, ]
get_ttest<-function(group1, group2, stt, ent, ttestdata){
  ttestdata1<-subset(ttestdata, group==group1)
  ttestdata2<-subset(ttestdata, group==group2)
  tt<-t.test(ttestdata1$counts, y=ttestdata2$counts, var.equal = FALSE)
  return(tt)}
pvalues<-matrix(nrow=kg, ncol=kg)
for (group1 in 1:kg){
  for (group2 in 1:kg){
    p<-get_ttest(group1, group2, stt, ent, ttestdata)
    pvalues[group2, group1]<-round(p$p.value,3)}}
colnames(pvalues)<-c(1:kg)
rownames(pvalues)<-c(1:kg)
pvalues
```

```
##   1    2    3     4     5
## 1 1 0.00 0.00 0.000 0.000
## 2 0 1.00 0.92 0.000 0.000
## 3 0 0.92 1.00 0.000 0.000
## 4 0 0.00 0.00 1.000 0.193
## 5 0 0.00 0.00 0.193 1.000
```

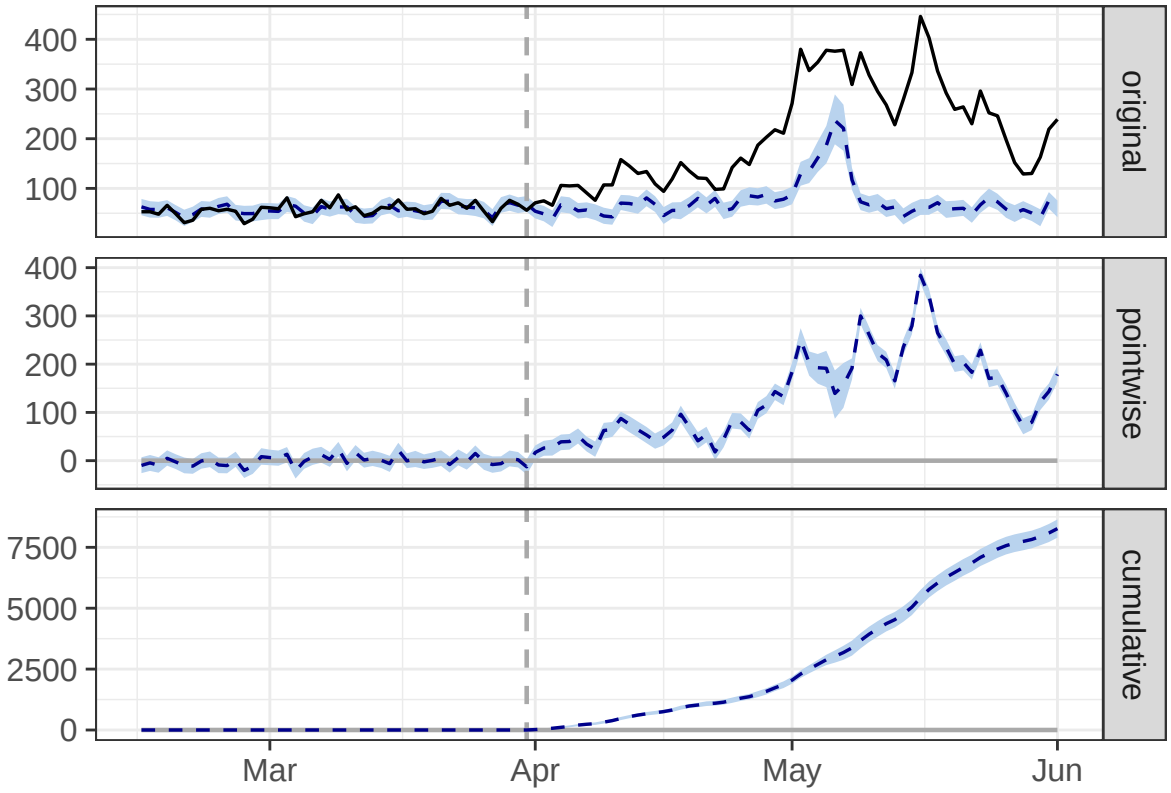
pvalues allow final choices for control and study groups.

4. Causal Impact package. Choose control and study groups.

```
controlgroup<-2
studygroup<-5
pre.period <- c(stt, ent)
post.period <- c(startime, endtime)
groupmydata3<-groupmydata[groupmydata$ga.date<=post.period[2] & groupmydata$ga.date>=pre.period[1], ]
controldata<-groupmydata3[groupmydata3$group==controlgroup,]
studydata<-groupmydata3[groupmydata3$group==studygroup,]
# formatting data to be compatible to the package input
datas<-merge(studydata, controldata, all=TRUE, by.x='ga.date', by.y = 'ga.date')
datas<-zoo(cbind(datas$counts.x,datas$counts.y), datas$ga.date)
# running the package
impact <- CausalImpact(datas, pre.period, post.period)
plot(impact)
```

```
## Warning: Removed 108 rows containing missing values (geom_path).
```

```
## Warning: Removed 216 rows containing missing values (geom_path).
```



5. How to design an experiment?

1. Prior to experiment try building groups, variables are kg and kn. They are for building ntiles and groups. The ntile number really depends on variability in actual data.

2. Critical part is to have large enough number of counts. If the counts number per landingPagePath is relatively low, group size needs to be increased to include more paths into the same group and sum up the counts. #### 3. Identify groups from the set that in the plot prior to experiment appear similar, use p-value.

4. Start experiment by changing tags to all landingPagePath's that are in the same group - the same change to all paths that are within the group. Leave at least one group with the unchanged paths. More than one group can be changed, as long as there is at least one the remaining unchanged.

5. Check logical control eliminate_empty_path. It might need to be set to False, depending on data.

6. On line 95, set.seed() function. This lines ensures repeatability when generating random groups. The repeatability needed for presentation. Not to run that line put # in front of it. Use that line when changing some parameters, but want to maintain the randomized part of program unchanged.