HW3_wgeither

Warren Geither

9/23/2020

Problem 3

I already practice alot of the style tips presented. However I can start to always use double quotes to denote character vectors. Since I always used single quotes in python.

Problem 5

a.) I can conclude that they have approximately the same mean, variance, and coorelation

```
# load in file from working directory
hw3_data_df <- readRDS("HW3_data (2).rds")

# create function
summarize_dataframe <- function(x){

  # group up by the first column and calculate means of col 2 & 3
  mean_summary <- x %>%
                    group_by(x[,1]) %>%
                    summarize(mean_1 = mean(x[,2]),mean_2 = mean(x[,3]))

  # same group by first except calc standard deviation for 2&3
  stdev_summary <- x %>%
                    group_by(x[,1]) %>%
                    summarize(std_1 = sd(x[,2]),std_2 = sd(x[,3]))

  # same except calculating covariance
  cor_summary <- x %>%
                  group_by(x[,1]) %>%
                  summarize(cor=cor(x[,2],x[,3]))

  # binding all summaries together be suring not to repeat our first col
  summary <- cbind(mean_summary
                   ,stdev_summary[,2:3]
                   ,cor_summary[2])

  # return final summary
  return(summary)
}

# store result in dataframe
summary_df <- summarize_dataframe(hw3_data_df)
```

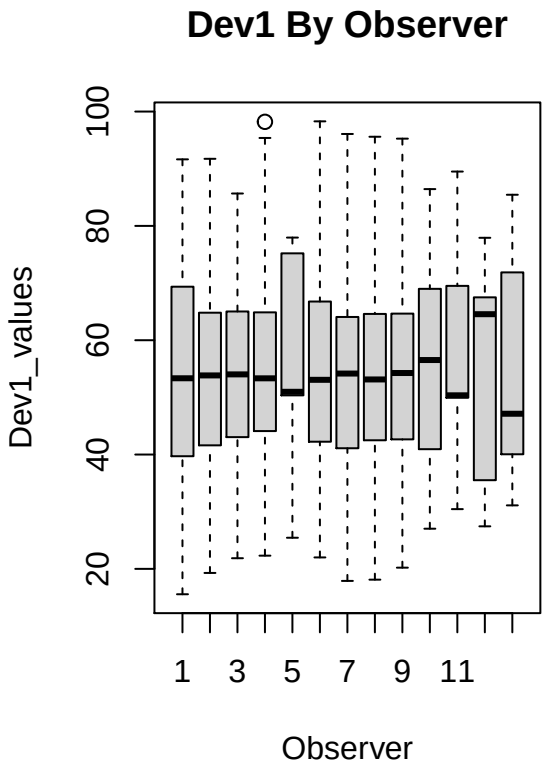
```
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
## `summarise()` ungrouping output (override with `.groups` argument)
```

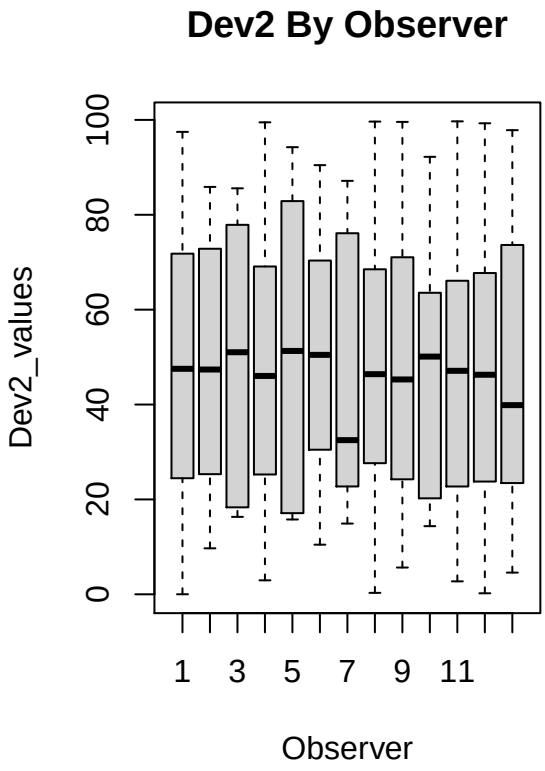
```
# create knitr table, conclude that they are all pretty much the same
knitr::kable(summary_df)
```

x[, 1]	mean_1	mean_2	std_1	std_2	cor
1	54.2657	47.8351	16.713	26.84777	-0.0660189
2	54.2657	47.8351	16.713	26.84777	-0.0660189
3	54.2657	47.8351	16.713	26.84777	-0.0660189
4	54.2657	47.8351	16.713	26.84777	-0.0660189
5	54.2657	47.8351	16.713	26.84777	-0.0660189
6	54.2657	47.8351	16.713	26.84777	-0.0660189
7	54.2657	47.8351	16.713	26.84777	-0.0660189
8	54.2657	47.8351	16.713	26.84777	-0.0660189
9	54.2657	47.8351	16.713	26.84777	-0.0660189
10	54.2657	47.8351	16.713	26.84777	-0.0660189
11	54.2657	47.8351	16.713	26.84777	-0.0660189
12	54.2657	47.8351	16.713	26.84777	-0.0660189
13	54.2657	47.8351	16.713	26.84777	-0.0660189

b.) I can conclude that I'm not insane, the data look very similar in terms of those values.

```
# set window
par(mfrow=c(1,2))

# device 1 boxplot
boxplot(dev1~Observer
        , data=hw3_data_df
        , main="Dev1 By Observer"
        , xlab="Observer"
        , ylab="Dev1_values")

# device 2 boxplot
boxplot(dev2~Observer
        , data=hw3_data_df
        , main="Dev2 By Observer"
        , xlab="Observer",
        ylab="Dev2_values")
```





c.) Looking back and forth at both the violin plots, there seems to definitely be a difference density at certain points

```
# change column to factor column
modified_hw3_data_df <- hw3_data_df
modified_hw3_data_df$Observer <- as.factor(modified_hw3_data_df$Observer)

# graph violin plots
plot_1 <- ggplot(modified_hw3_data_df, aes(x=Observer, y=dev1, color=Observer)) +
            geom_violin() +
            scale_x_discrete(limits=c(1:12)) +
            ggtitle("Observer By Dev1") +
            labs(y="Dev1", x="Observer")
```

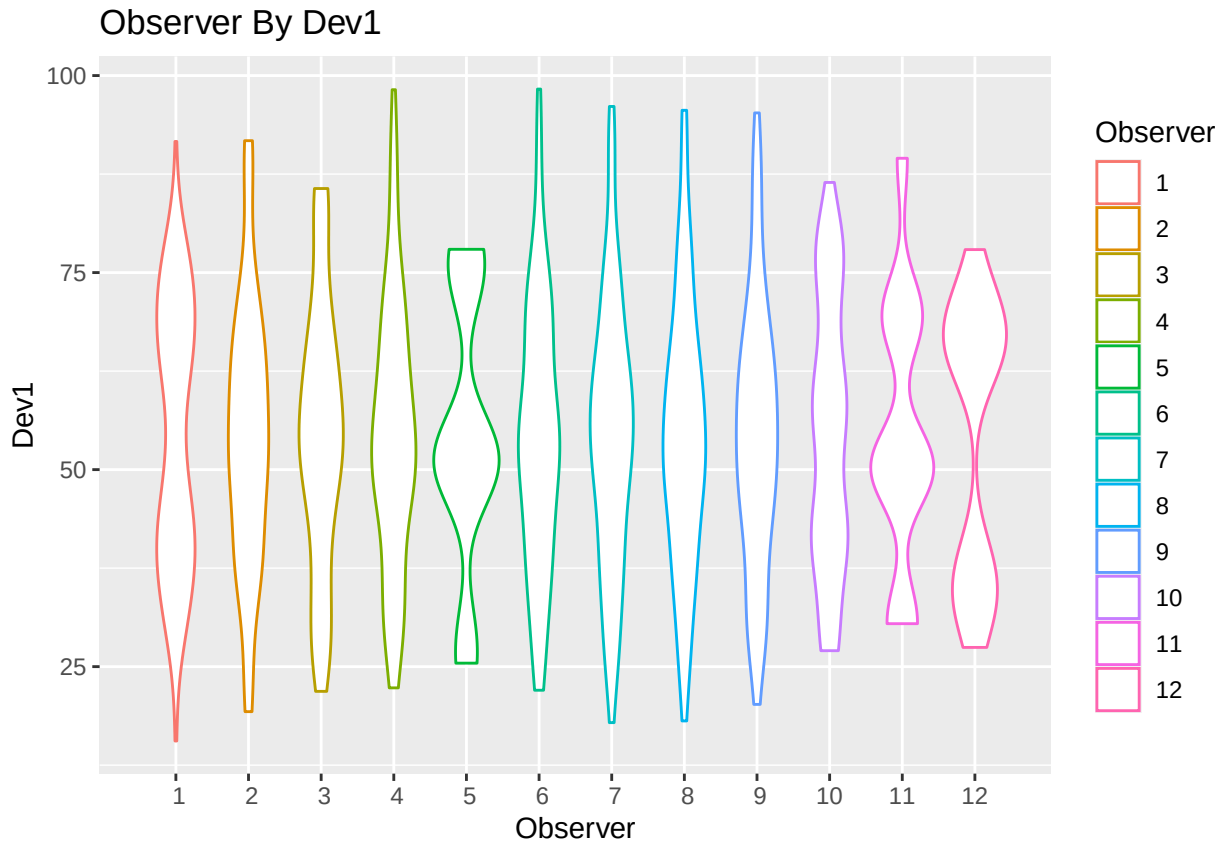
```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean 'limits = factor(...)' or 'scale_*_continuous()'?
```

```
plot_2 <- ggplot(modified_hw3_data_df, aes(x=Observer, y=dev2, color=Observer)) +
            geom_violin() +
            scale_x_discrete(limits=c(1:12)) +
            ggtitle("Observer By Dev2") +
            labs(y="Dev2", x="Observer")
```

```
## Warning: Continuous limits supplied to discrete scale.
## Did you mean 'limits = factor(...)' or 'scale_*_continuous()'?
```

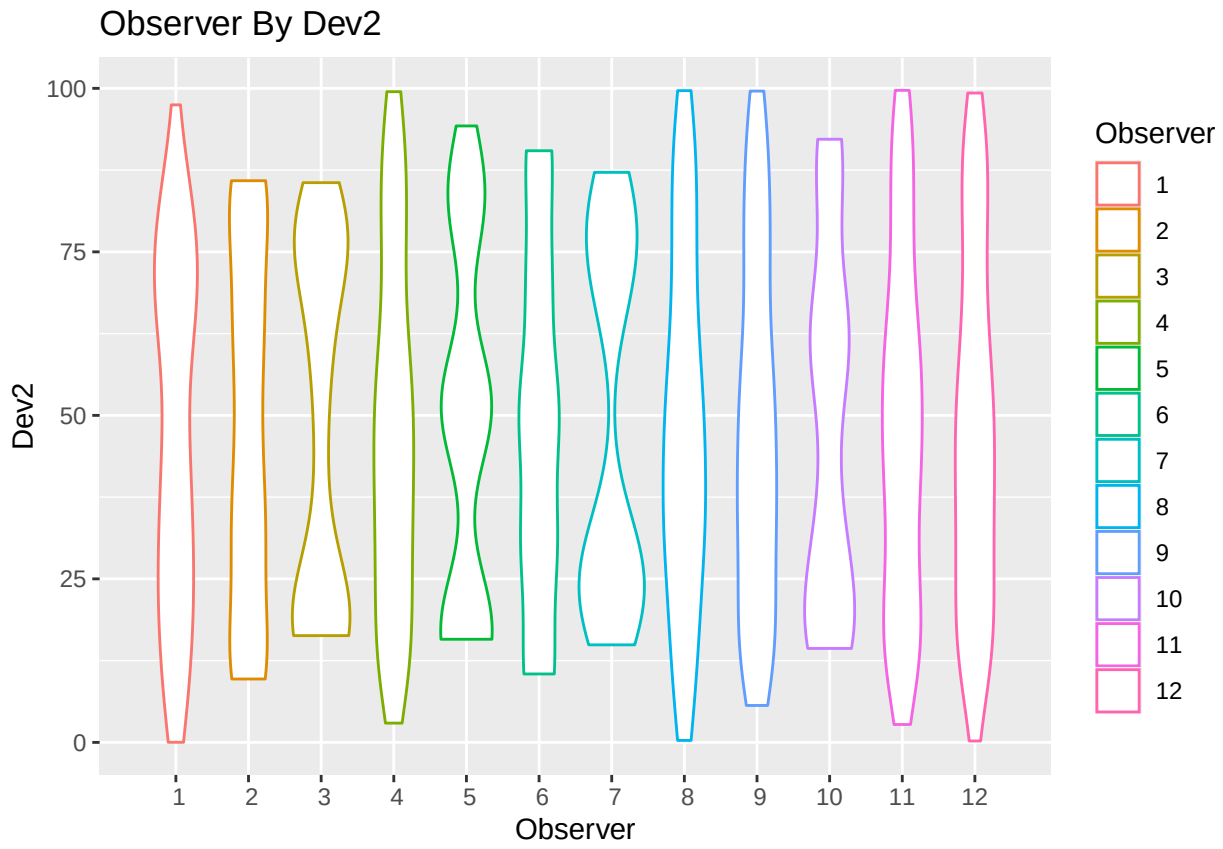
```
# print plot 1
plot_1
```

```
## Warning: Removed 142 rows containing non-finite values (stat_ydensity).
```



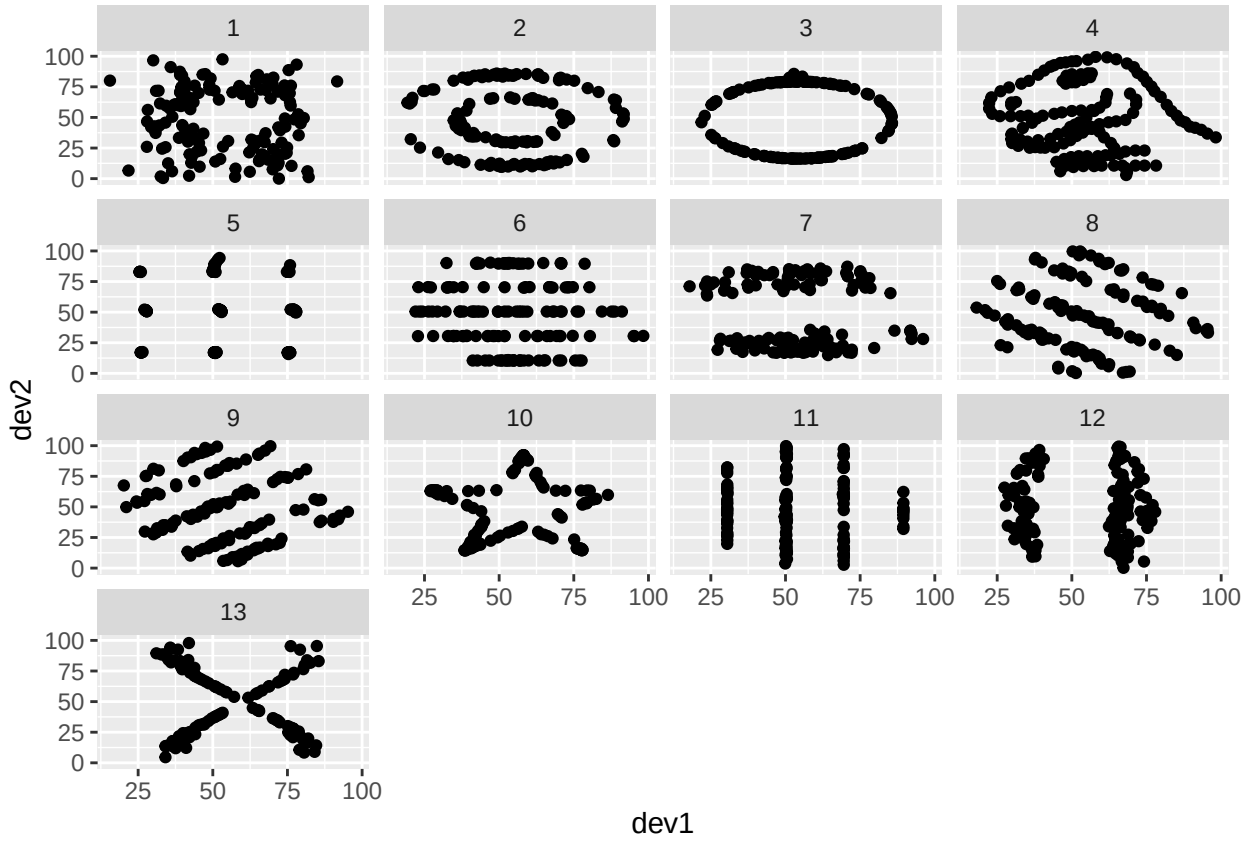

```
# print plot 2
plot_2
```

```
## Warning: Removed 142 rows containing non-finite values (stat_ydensity).
```



d.) LOL! Always make sure to look at the data from all angles.

```
ggplot(hw3_data_df, aes(x=dev1,y=dev2)) + geom_point() + facet_wrap(Observer~.)
```



Problem 6

```
# initialize value for function/loop
sum_val2 = 0

# create function to approximate integral
integral_approx_function <- function(x) {

    # loop to get the left rienamm heights and sum them
    for (i in 0:x-1) {
      # input i into function
      sum_val <- exp(-((1/i)^2)/2)

      # sum the i's together
      sum_val2 <- sum_val + sum_val2
      }

    # muliple by the width of the boxes so we have the total area
    value <- (1/x) * sum_val2
    return(value)
}

# solution is actually sqrt(2pi), just using one though.
numerical_approx <- 1
```

```r
# initialize other variables
rieman_sum <- 0
delta <- numerical_approx - rieman_sum
j <- 0

# loop through possible values
while (delta > 0.000001){
  # plug j into function
  rieman_sum <- integral_approx_function(j)

  # print our values
  print(paste0("Slices being used: ", j))
  print(paste0("Sum calculated: ", rieman_sum))

  # increment j
  j <- j + 100000

  # find the delta
  delta <- abs(numerical_approx - rieman_sum)
}
```

```
## [1] "Slices being used: 0"
## [1] "Sum calculated: Inf"
## [1] "Slices being used: 1e+05"
## [1] "Sum calculated: 0.999989005387281"
## [1] "Slices being used: 2e+05"
## [1] "Sum calculated: 0.999994502679964"
## [1] "Slices being used: 3e+05"
## [1] "Sum calculated: 0.999996335119976"
## [1] "Slices being used: 4e+05"
## [1] "Sum calculated: 0.999997251339982"
## [1] "Slices being used: 5e+05"
## [1] "Sum calculated: 0.999997801071986"
## [1] "Slices being used: 6e+05"
## [1] "Sum calculated: 0.999998167559988"
## [1] "Slices being used: 7e+05"
## [1] "Sum calculated: 0.999998429337133"
## [1] "Slices being used: 8e+05"
## [1] "Sum calculated: 0.999998625669991"
## [1] "Slices being used: 9e+05"
## [1] "Sum calculated: 0.999998778373325"
## [1] "Slices being used: 1e+06"
## [1] "Sum calculated: 0.999998900535993"
## [1] "Slices being used: 1100000"
## [1] "Sum calculated: 0.999999000487266"
```

Problem 7

```r
# create function for newtons method
newtons_method_func <- function(x){
```

```r
  # input the equation
  eq <- (3^x) - sin(x) + cos(5*x)

  # store the derivative
  derivative_eq <- -5*sin(5*x)-cos(x)+log(3)*3^x

  # calculate the next term
  x_n1 <- x - (eq/derivative_eq)

  return(x_n1)
}

# initialize guess value
x <- -1

# while difference from root is greater than 0.1 keep going
while (abs(x - 0) > 0.1) {

  # find the next value given your guess
  x_n1 <- newtons_method_func(x)

  # store that value and continue
  x <- x_n1

  # print x
  print(x)

}
```

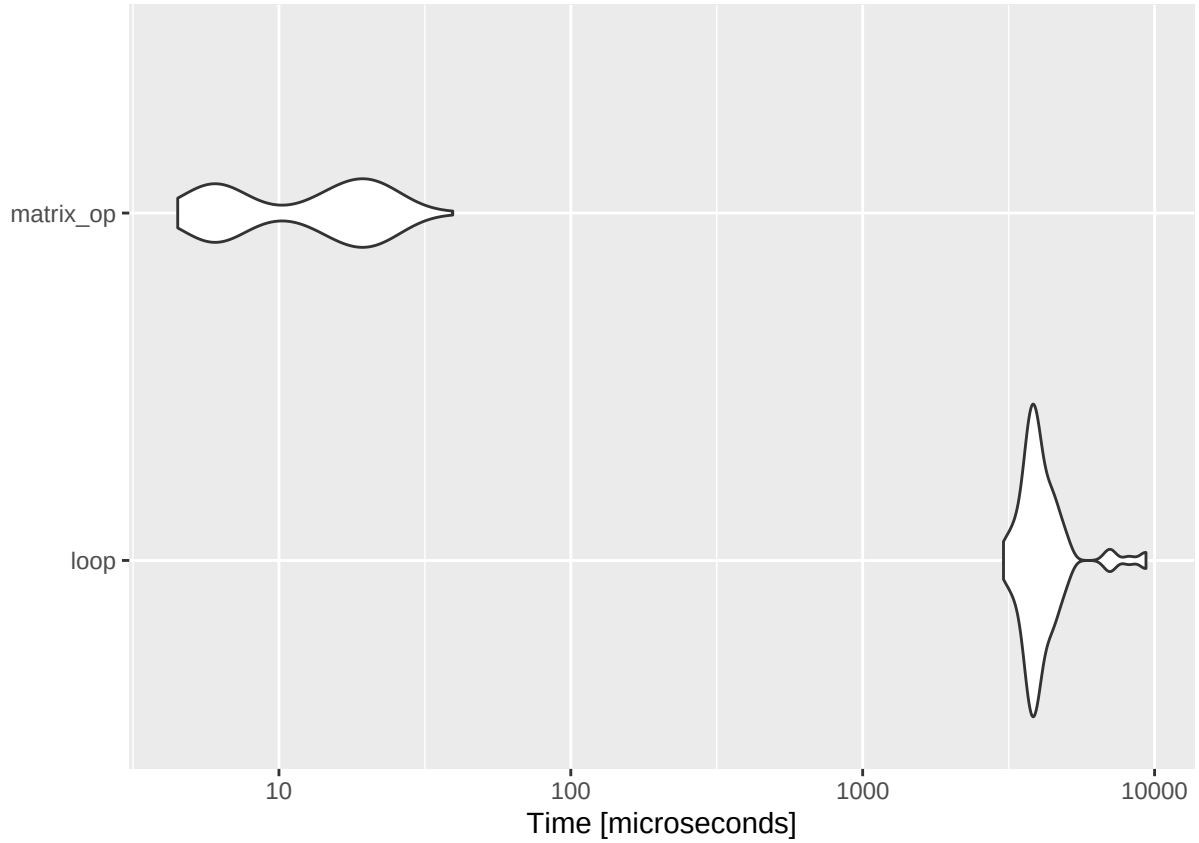
```
## [1] -0.7064704
## [1] -0.6209373
## [1] 0.5767383
## [1] 9.177106
## [1] 8.266759
## [1] 7.356966
## [1] 6.447693
## [1] 5.533814
## [1] 4.617035
## [1] 3.726782
## [1] 2.819543
## [1] 1.741303
## [1] 0.5759834
## [1] 6.424779
## [1] 5.51102
## [1] 4.593032
## [1] 3.702464
## [1] 2.804239
## [1] 1.720811
## [1] 0.4061904
## [1] 0.6026129
## [1] 0.03106367
```

Problem 8

The matrix operations are much quicker.

```
# simulated data
X <- cbind(rep(1,100),rep.int(1:10,time=10))

# create vectore beta
beta <- c(4,5)

# create random vector y
y <- X%*%beta + rnorm(100)

# create mean vector
y_bar <- mean(y)


# initialize values for loop
sum2 <- 0
sum <- 0

# initialize mean matrix
y_bar_vector <- rep(y_bar,100)

# compare finding SST, not sure if theres a better way to wrap the for loop
comparision_table <- microbenchmark(loop = for (i in 1:100) {
  sum <- (y[i] - y_bar)^2
  sum2 <- sum2 + sum
}
, matrix_op = colSums((y- y_bar_vector)^2))

# plot chart
autoplot(comparision_table)
```

```
## Coordinate system already present. Adding new coordinate system, which will replace the existing one
```




```
# print table
knitr::kable(head(comparision_table))
```

expr	time
loop	4609600
matrix_op	26100
matrix_op	6700
loop	4364900
loop	4562600
matrix_op	20300