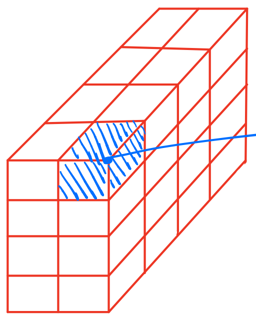
Arrays



value in a cell
all in same modes.

The difference between
matrix vs. dataframe
↓
same mode everywhere

IMPORTANT

Arrays

If a value is empty, put NA

[ ] => [ ]

1704x1 12x142

```
lifeExp_array<-with(gapminder,tapply(lifeExp,
                                     data.frame(year,country),
                                     c))

lifeExp_array[1:5,1:3]
```

create vector (same mode)

```
      country
year   Afghanistan Albania Algeria
  1952      28.801   55.23  43.077
  1957      30.332   59.28  45.685
  1962      31.997   64.82  48.303
  1967      34.020   66.22  51.407
  1972      36.088   67.69  54.518
```

not column → attribut

dimnames

Orderd by factor

```
dim(lifeExp_array)
```

```
[1]  12 142
```

Arrays

```
alltogether<-array(NA,dim=c(12,142,3),
                   dimnames=list(year=dimnames(lifeExp_array)$year,
                                 country=dimnames(lifeExp_array)$country,
                                 var=c("lifeExp","gdpPercap","pop")))
alltogether[,,"lifeExp"]<-
        with(gapminder,tapply(lifeExp,data.frame(year,country),c))
alltogether[,,"gdpPercap"]<-
        with(gapminder,tapply(gdpPercap,data.frame(year,country),c))
alltogether[,,"pop"]<-
        with(gapminder,tapply(pop,data.frame(year,country),c))
dim(alltogether)
```

```
[1]  12 142   3
```

Arrays

```
alltogether[1:2,1:2,c(1,3)]
```

layers

```
, , var = lifeExp

      country
year   Afghanistan Albania
  1952      28.801   55.23
  1957      30.332   59.28

, , var = pop

      country
year   Afghanistan Albania
  1952     8425333 1282697
  1957     9240934 1476505
```

Arrays

for each year, var, compute mean

12 x 142 x 3 → 12 x 3

```r
apply(alltogether,c(1,3), mean)
```

dims that are fixed.

```
      var
year    lifeExp gdpPercap      pop
  1952   49.058    3725.3 16950402
  1957   51.507    4299.4 18763413
  1962   53.609    4725.8 20421007
  1967   55.678    5483.7 22658298
  1972   57.647    6770.1 25189980
  1977   59.570    7313.2 27676379
  1982   61.533    7518.9 30207302
  1987   63.213    7900.9 33038573
  1992   64.160    8158.6 35990917
  1997   65.015    9090.2 38839468
  2002   65.695    9917.8 41457589
  2007   67.007   11680.1 44021220
```

c(3) average over year & country.

↓

vector of length 3

Arrays

```
totalGDP<-alltogether[,,"pop"]*alltogether[,,"gdpPercap"]
totalGDP[1:3,2:4]
```

element-wise

```
      country
year      Albania     Algeria      Angola
  1952 2053669902 2.2726e+10 1.4900e+10
  1957 2867792398 3.0956e+10 1.7461e+10
  1962 3996988985 2.8061e+10 2.0604e+10
```

totalGDP preserve "country"

```
sort(apply(alltogether[,,"pop"]*alltogether[,,"gdpPercap"],2,sum)/1e12,
     dec=TRUE)[1:4]
```

```
United States         Japan         China       Germany
       76.762        25.435        20.395        19.497
```

Arrays: permuting dimensions

```
aperm(alltogether,c(2,3,1))[1:5,1:3,1:2]
```

```
, , year = 1952

             var
country         lifeExp gdpPercap       pop
  Afghanistan    28.801    779.45   8425333
  Albania        55.230   1601.06   1282697
  Algeria        43.077   2449.01   9279525
  Angola         30.015   3520.61   4232095
  Argentina      62.485   5911.32  17876956

, , year = 1957

             var
country         lifeExp gdpPercap       pop
  Afghanistan    30.332    820.85   9240934
  Albania        59.280   1942.28   1476505
  Algeria        45.685   3013.98  10270856
  Angola         31.999   3827.94   4561361
```

Arrays

hold country, var

```
min_max_array<-apply(alltogether,c(2,3),
      function(x){
        y<-c(min(x), max(x));
        names(y)<-c("Min","Max");
        y})
```

Arrays

```
min_max_array[,1:3,c(1:2)]
```

12 x 142 x 5 → 2 x 142 x 5

```
, , var = lifeExp

     country
      Afghanistan Albania Algeria
  Min      28.801  55.230  43.077
  Max      43.828  76.423  72.301

, , var = gdpPercap

     country
      Afghanistan Albania Algeria
  Min      635.34  1601.1  2449.0
  Max      978.01  5937.0  6223.4
```

Arrays

```
aperm(min_max_array,c(2,3,1))[1:3,,]
```

```
, ,  = Min

             var
country        lifeExp gdpPercap     pop
  Afghanistan   28.801    635.34 8425333
  Albania       55.230   1601.06 1282697
  Algeria       43.077   2449.01 9279525

, ,  = Max

             var
country        lifeExp gdpPercap      pop
  Afghanistan   43.828    978.01 31889923
  Albania       76.423   5937.03  3600523
  Algeria       72.301   6223.37 33333216
```

Arrays: contingency tables

```r
syph_data<-read_csv(here("Documents/syphilis89d.csv"))
# Random sample of 8 subjects
syph_data[sample(1:nrow(syph_data),8),]  %>% kable(.)
```

Sex	Race	Age
Male	White	30-44
Male	Other	20-29
Female	Black	30-44
Female	White	30-44
Female	Other	<=19
Male	Black	30-44
Female	White	20-29
Male	Other	20-29

Arrays: contingency tables

n=n()

```
syph_counts<-syph_data %>% group_by(Sex, Race, Age) %>% count()
dim(syph_counts)
```

by default n=count()

```
[1] 24  4
```

```
syph_counts%>%arrange(desc(n)) %>% head()
```

```
# A tibble: 6 x 4
# Groups:   Sex, Race, Age [6]
  Sex    Race  Age       n
  <chr>  <chr> <chr> <int>
1 Male   Black 30-44  8311
2 Male   Black 20-29  8180
3 Female Black 20-29  8093
4 Female Black 30-44  4133
5 Male   Black 45+    2442
6 Female Black <=19   2422
```

Arrays: contingency tables

```
syph_counts%>% group_by(Race) %>% summarise(sum(n))
```

```
# A tibble: 3 x 2
  Race  `sum(n)`
  <chr>    <int>
1 Black    35508
2 Other     3956
3 White     4617
```

```
syph_counts%>% group_by(Age) %>% summarise(sum(n))
```

```
# A tibble: 4 x 2
  Age   `sum(n)`
  <chr>    <int>
1 <=19      4608
2 20-29    20015
3 30-44    15549
4 45+       3909
```

Arrays: contingency tables

cross tab

```
syph_counts_array<-xtabs(~Sex+Race+Age, data=syph_data)
dim(syph_counts_array)
```

formula

```
[1] 2 3 4
```

Sex Race Age

$\boxed{\text{LHS}} \sim \boxed{\text{RHS}}$

- nothing
  count rows
- variable
  take sum of that
  variable by row

factor variable
add up

xtabs(n~Sex+Race+Age, data=syph_counts)

⇒ Total number for Sex, Race, Age.

Arrays: contingency tables

```
syph_counts_array[1:2,1:3,c(2,4)]
```

```
, , Age = 20-29

        Race
Sex       Black Other White
  Female   8093   590   908
  Male     8180  1287   957

, , Age = 45+

        Race
Sex       Black Other White
  Female    484    55    79
  Male     2442   310   539
```

Arrays: contingency tables

```
apply(syph_counts_array, c(2),sum)
```

```
Black Other White
35508  3956  4617
```

could also use dimension name.

```
apply(syph_counts_array, c("Race"),sum)
```

```
Black Other White
35508  3956  4617
```

Arrays: contingency tables

```
apply(syph_counts_array, c("Race","Age"),sum)
```

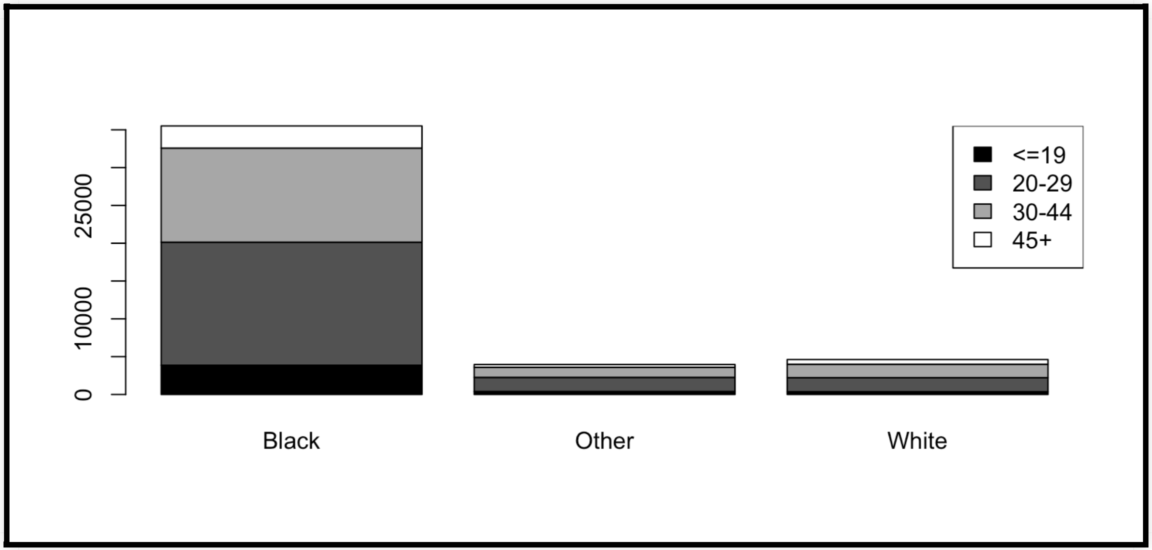
```
       Age
Race     <=19 20-29 30-44  45+
  Black 3865 16273 12444 2926
  Other  386  1877  1328  365
  White  357  1865  1777  618
```

Arrays: contingency tables

```
myplot<-barplot(apply(syph_counts_array, c("Age","Race"),sum),
                    col=grey(seq(0,1,length=4)))
legend("topright",fill=grey(seq(0,1,length=4)),
       legend=levels(factor(syph_data$Age)))
```

4 levels of shading
$(0, \frac{1}{3}, \frac{2}{3}, 1)$

?



Arrays: contingency tables

```
column_props <-apply(syph_counts_array, c("Age","Race"),sum) %>%
                 prop.table(.,c(2))
column_props
```

what comes to the pipe ⇒ array

→ fixed : proportion of different "Age" group in a "Race".

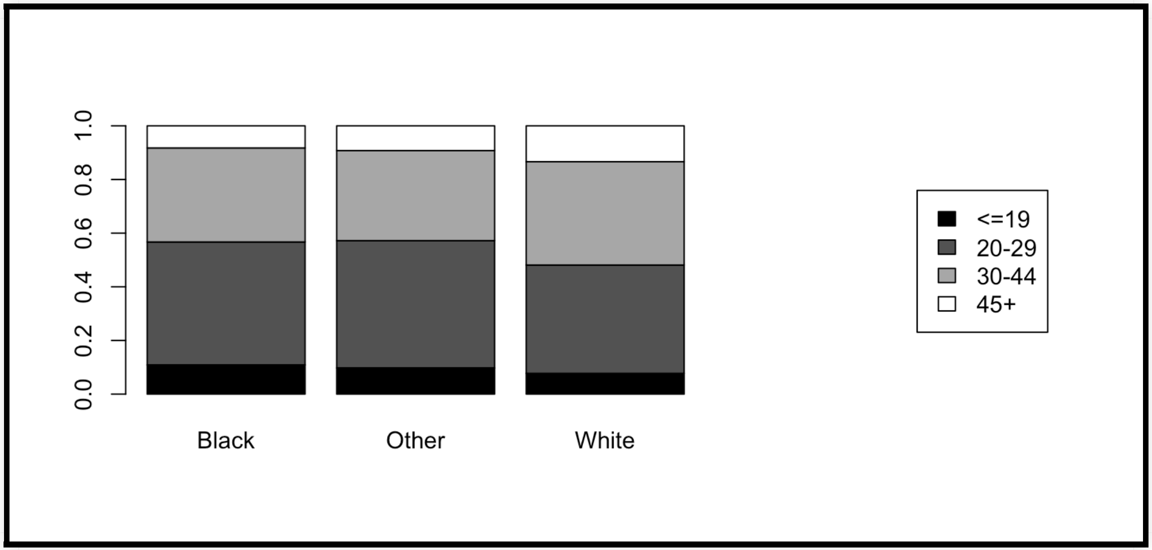
```
       Race
Age         Black    Other    White
  <=19  0.108849 0.097573 0.077323
  20-29 0.458291 0.474469 0.403942
  30-44 0.350456 0.335693 0.384882
  45+   0.082404 0.092265 0.133853
```

=1 =1 =1

- could use c(3) ?
- c() or nothing ⇒ total.

Arrays: contingency tables

```
layout(matrix(c(1,2),nrow=1), c(2,1)) ## For multiple plots
myplot<-barplot(column_props,col=grey(seq(0,1,length=4))) # First plot
# Need empty plot for legend
plot(c(0,0),type="n",axes=FALSE,ylab="",xlab="")
legend("center",fill=grey(seq(0,1,length=4)),
       legend=levels(factor(syph_data$Age)))
```



Speed: an example

Convolution of two vectors

$\vec{a}_{n \times 1}$

$\vec{b}_{m \times 1}$

$$\vec{a} \cdot \vec{b} = \sum_{u} a[u]\, b[x-u]$$

$(x+1)(3x^2+2x+3)$

Convolution of two vectors

```
convolve_loop_one<-function(vec_1,vec_2){
  ## Compute lengths of both vectors
  n<-length(vec_1)
  m<-length(vec_2)
  ## Pad vectors to avoid boundary issues
  vec_1_star <- c(vec_1,rep(0,n+m-1-n))
  vec_2_star <- c(vec_2,rep(0,n+m-1-m))
  k<-n + m - 1
  new_vec<-rep(0,k)
  for(i in 1:k){
    for(u in seq(max(1,i-k+1),min(i,k),by=1)){
      new_vec[i]<-new_vec[i]+vec_1_star[u]*vec_2_star[i-u+1]
    }
  }
  new_vec
}
```

padding

maximum polynomial length

Convolution of two vectors

```
convolve_loop_one(c(1,2,3),c(0.5,0.25))
```

$(0.25x+0.5)(3x^2+2x+1)$

```
[1] 0.50 1.25 2.00 0.75
```

```
# A similar function from pracma package
library(pracma)
conv(c(1,2,3),c(0.5,0.25))
```

```
[1] 0.50 1.25 2.00 0.75
```

Benchmarking the time

```r
library(microbenchmark)
res<-microbenchmark(Loop_1=convolve_loop_one(c(1,2,3),c(0.5,0.25)),
                    pracma_conv=conv(c(1,2,3),c(0.5,0.25)),
                    times=1000L)
res
```

myFunction

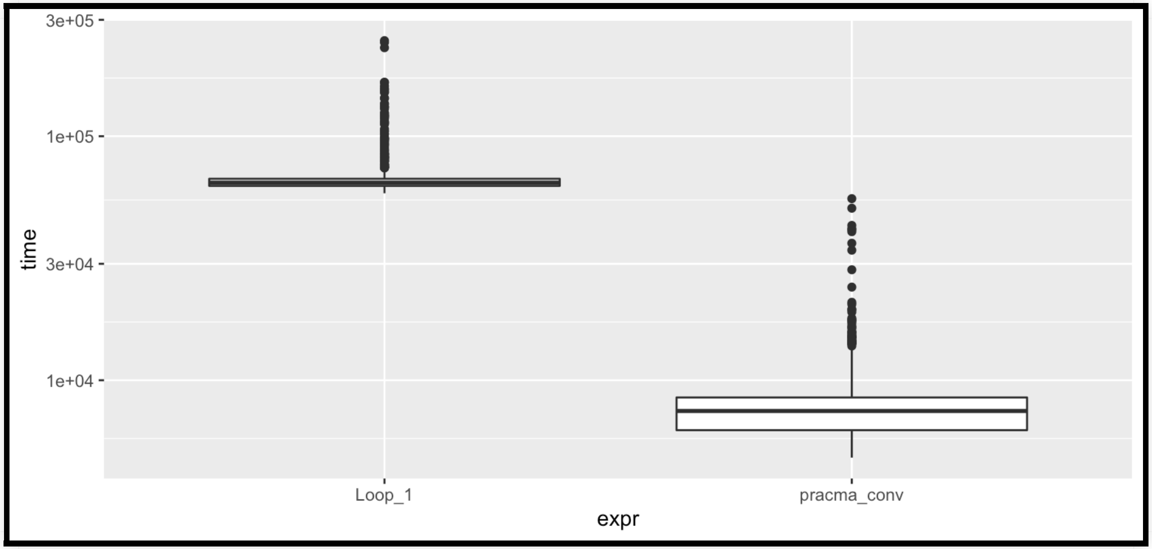
compare run time

benchmark function

```
Unit: microseconds
        expr    min     lq    mean median      uq     max neval cld
      Loop_1 58.367 62.582 68.3741 64.525 66.9390 246.490  1000   b
 pracma_conv  4.816  6.242  7.9762  7.482  8.4995  55.488  1000  a
```

Benchmarking the time

```
ggplot(res,aes(x=expr,y=time)) + geom_boxplot() + scale_y_log10()
```



Vectorizing the inner loop

```
convolve_loop_two<-function(vec_1,vec_2){
  ## Compute lengths of both vectors
  n<-length(vec_1)
  m<-length(vec_2)
  ## Pad vectors to avoid boundary issues
  vec_1_star <- c(vec_1,rep(0,n+m-1-n))
  vec_2_star <- c(vec_2,rep(0,n+m-1-m))
  k<-n + m - 1
  new_vec<-rep(0,k)
  for(i in 1:k){
    u<-seq(max(1,i-k+1),min(i,k),by=1)
    new_vec[i]<-sum(vec_1_star[u]*vec_2_star[i-u+1])
  }
  new_vec
}
```

remove the inner loop, vectorize, get sum

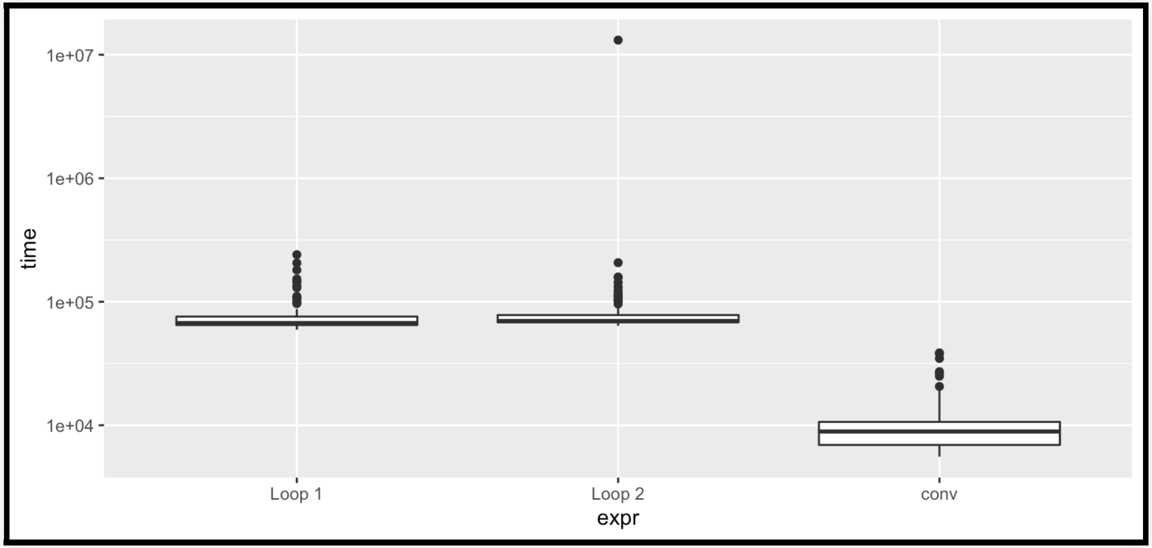
Vectorizing the inner loop

```
res<-microbenchmark(Loop_1=convolve_loop_one(c(1,2,3),c(0.5,0.25)),
                    Loop_2=convolve_loop_two(c(1,2,3),c(0.5,0.25)),
                    pracma_conv=conv(c(1,2,3),c(0.5,0.25)), times=100L)
res
```

```
Unit: microseconds
        expr    min     lq    mean median     uq       max neval cld
      Loop_1 59.449 64.752  78.090 67.171 75.942   240.615   100   a
      Loop_2 63.721 68.199 209.738 69.777 78.108 13142.798   100   a
 pracma_conv  5.557  6.922  10.726  8.896 10.668    38.537   100   a
```

Vectorizing the inner loop

```
ggplot(res,aes(x=expr,y=time)) + geom_boxplot() + scale_y_log10() +
  scale_x_discrete(labels=c("Loop 1","Loop 2", "conv"))
```



Vectorizing the inner loop

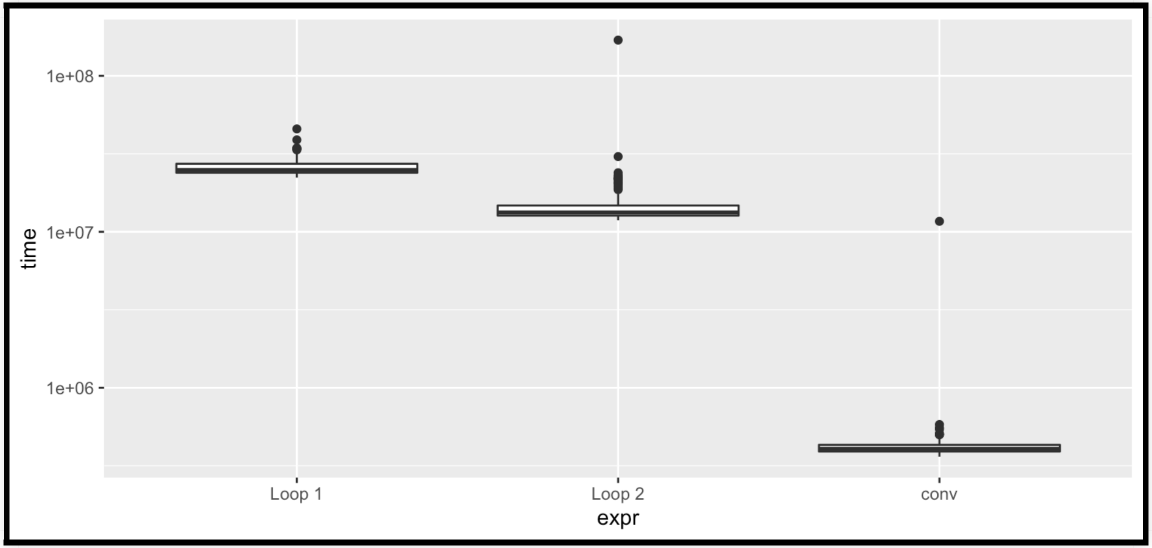
```
vec_1<-rep(c(1,2,3),100)
vec_2<-rep(c(0.5,0.25),100)

res<-microbenchmark(Loop_1=convolve_loop_one(vec_1,vec_2),
                    Loop_2=convolve_loop_two(vec_1,vec_2),
                    pracma_conv=conv(vec_1,vec_2),
                    times=100L)
res
```

```
Unit: microseconds
        expr     min       lq     mean   median      uq    max neval clc
      Loop_1 22238.1 23918.32 26263.81 24881.55 27334.3  45665   100   c
      Loop_2 11840.8 12695.64 16163.61 13287.97 14782.4 169440   100  b
 pracma_conv   361.3   390.42   530.34   406.14   431.4  11670   100 a
```

better

Vectorizing the inner loop

```r
ggplot(res,aes(x=expr,y=time)) + geom_boxplot() + scale_y_log10() +
  scale_x_discrete(labels=c("Loop 1","Loop 2", "conv"))
```



lapply the outer loop

```
convolve_loop_three<-function(vec_1,vec_2){
  ## Compute lengths of both vectors
  n<-length(vec_1)
  m<-length(vec_2)
  ## Pad vectors to avoid boundary issues
  vec_1_star <- c(vec_1,rep(0,n+m-1-n))
  vec_2_star <- c(vec_2,rep(0,n+m-1-m))
  k<-n + m - 1                 indices for the vector
  new_vec_list<-lapply(1:k, function(x){
    u<-seq(max(1,x-k+1),min(x,k),by=1)
    sum(vec_1_star[u]*vec_2_star[x-u+1])
}) }
```

lapply the outer loop

```
vec_1<-rep(c(1,2,3),100)
vec_2<-rep(c(0.5,0.25),100)

res<-microbenchmark(Loop_1=convolve_loop_one(vec_1,vec_2),
                    Loop_2=convolve_loop_two(vec_1,vec_2),
                    Loop_3=convolve_loop_three(vec_1,vec_2),
                    pracma_conv=conv(vec_1,vec_2),
                    times=100L)
res
```

DFT

```
Unit: microseconds
        expr      min       lq     mean   median       uq       max neva
      Loop_1 22539.64 23553.13 24868.68 24125.05 25081.62  38209.50   10
      Loop_2 11340.99 12485.10 14230.95 13050.24 13797.26  25332.50   10
      Loop_3 12171.85 12861.41 16370.17 13277.77 14634.61 167110.63   10
 pracma_conv   359.05   387.82   414.23   403.82   426.29    623.61   10
 cld
   c
  b
  b
 a
```

speed up: convert to matrix multiplication