# BIOL 471 Lecture 4

Kirt Onthank

04/02/2019

#What does the 'print' function do?

```
print(23)
```

```
## [1] 23
```

## Basic loop function

```
for (i in 1:20){
  print(i)
}
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] 19
## [1] 20
```

## Does not need to be a sequence. Can be arbitrary

```
for (i in c(-3,10,1001,3,45,0.6)){
  print(i)
}
```

```
## [1] -3
## [1] 10
## [1] 1001
## [1] 3
## [1] 45
## [1] 0.6
```

## Doesn't even need to be numbers

### You can also fill vectors before you call the loop

```
thing=c("people","brown dog","fritos","tank","purple","thirty-six")

for (i in thing){
  print(i)
}
```

```
## [1] "people"
## [1] "brown dog"
## [1] "fritos"
## [1] "tank"
## [1] "purple"
## [1] "thirty-six"
```

## What the 'rnorm' function does

```
rnorm(1,mean=20,sd=7)
```

```
## [1] 27.87431
```

## Using i in a series of functions

```
for (i in 20:40){
  x=rnorm(100,mean=i,sd=7)
  y=mean(x)
  z=sd(x)
  print(paste(i,y,z))
}
```

```
## [1] "20 19.3758465601797 7.45601531338691"
## [1] "21 22.6144855069157 6.6148389994582"
## [1] "22 20.7175971761023 6.7071927216606"
## [1] "23 22.1160281851512 7.23108069120551"
## [1] "24 23.2977465482971 7.81977182284044"
## [1] "25 24.1203191329024 7.08193307405265"
## [1] "26 24.7753901461443 7.41504640122623"
## [1] "27 27.7020575212652 6.94628449599827"
## [1] "28 26.9490101649566 6.23443879312818"
## [1] "29 28.4398599486172 5.94430548999255"
## [1] "30 29.8635255025878 6.95726332400169"
## [1] "31 30.7869818331661 6.97438425524917"
## [1] "32 31.9509922592514 7.50206288425884"
## [1] "33 32.9696714562586 7.04600476159267"
## [1] "34 32.5030335293463 7.46295473640415"
## [1] "35 35.825540519149 7.75205503619291"
## [1] "36 37.0931209924712 8.05593024817335"
## [1] "37 36.1557825859719 6.61484220137226"
## [1] "38 37.8749859266279 6.61304551469594"
## [1] "39 39.345032576721 7.88712865002121"
## [1] "40 40.2251029334091 6.11796357375373"
```

## Initialize a vector to store your output

```
rmean=numeric()
rsd=numeric()

for (i in 1:20){
  x=rnorm(100,mean=i,sd=7)
  rmean[i]=mean(x)
  rsd[i]=sd(x)
}
rmean
```

```
##  [1]  1.416096  3.561537  3.046989  3.365352  4.981460  6.041520  7.304879
##  [8]  7.943072  8.385419  8.783467 10.285554 10.058386 12.902233 15.860003
## [15] 15.061446 15.844968 17.944214 18.747640 18.155181 20.491417
```

```
rsd
```

```
##  [1] 7.087150 7.112024 6.489767 6.633850 7.579745 6.972747 6.254844 7.127132
##  [9] 6.670029 7.184236 6.843520 6.166923 7.762884 7.128892 6.950775 7.440273
## [17] 7.789324 6.892759 7.109202 6.979673
```

## Another way to specify position in the

```
rmean=numeric()
rsd=numeric()
iter=1

for (i in c(-100,100,67,10000,0.5,10)){
  x=rnorm(100,mean=i,sd=7)
  rmean[iter]=mean(x)
  rsd[iter]=sd(x)
  iter=iter+1
}
rmean
```

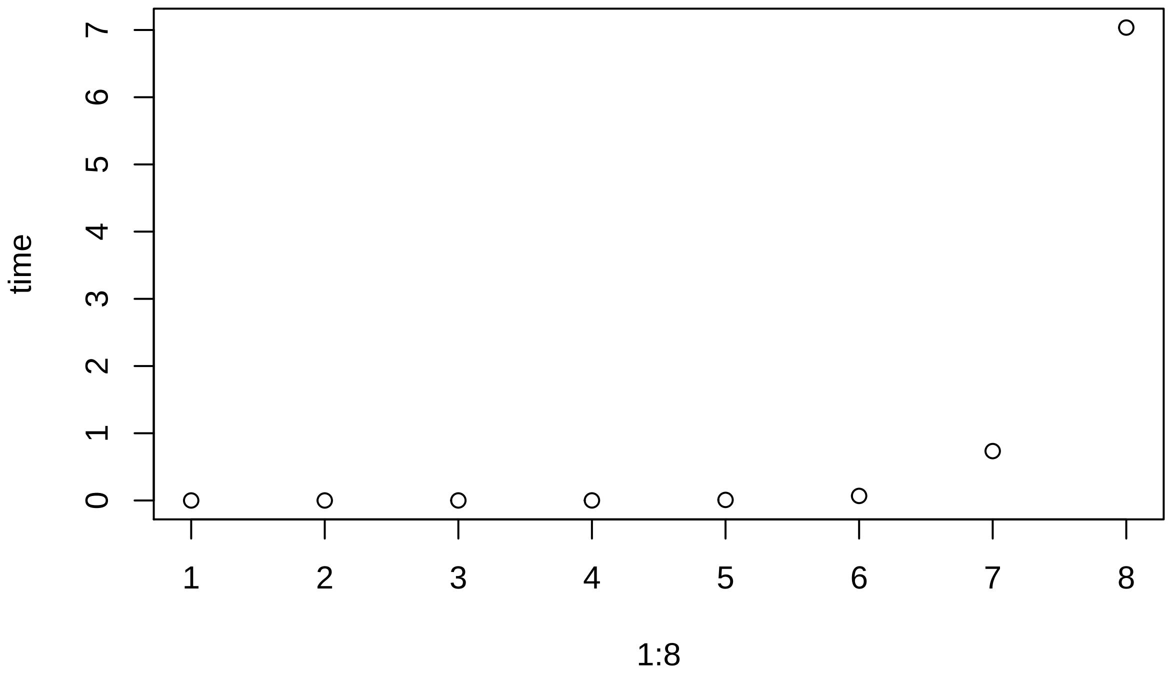
```
## [1] -100.4571789  100.1120211   66.1951975 9999.6909245    0.7950615
## [6]   10.0174912
```

```
rsd
```

```
## [1] 6.675257 7.684439 6.672088 6.486661 7.332698 7.163971
```

## Be careful of how much time something takes

```
time=1
for(i in 1:8){
   start=Sys.time()
   x=rnorm(10^i,mean=100,sd=15)
   mean(x)
   time[i]=Sys.time()-start
}
plot(1:8,time)
```



## What does the 'sample' function do?

```
sample(1:100,1)
```

```
## [1] 25
```

## Simulating 10 dice rolls

```
dice=numeric()
for(i in 1:10){
dice[i]=sample(1:6,1)+sample(1:6,1)
print(dice)
}
```

```
## [1] 5
## [1] 5 8
## [1] 5 8 2
## [1] 5 8 2 5
## [1] 5 8 2 5 7
## [1]  5  8  2  5  7 10
## [1]  5  8  2  5  7 10  9
## [1]  5  8  2  5  7 10  9  7
## [1]  5  8  2  5  7 10  9  7  7
##  [1]  5  8  2  5  7 10  9  7  7  7
```

## Just how fast is this function?

```r
start=Sys.time()
for(i in 1:50000){
dice[i]=sample(1:6,1)+sample(1:6,1)
}
Sys.time()-start
```

```
## Time difference of 0.6691296 secs
```