Branching Process

Aymen Rumi

Functions For Generating Branching Process & Probability Generating Functions

```r
# Function to generate next generation given probability values for offspring
next_gen = function(pars,z_nmin1,n){

  prob  = pars[[1]]
  if(z_nmin1>0){
    tibble(Generation=rep(n,z_nmin1), Label=1:z_nmin1,
             offspring=sample(0:(length(prob)-1), z_nmin1, replace=T, prob=prob))
  }
  else{
    tibble(Generation=n, Label=1,
             offspring=0)
  }

}

# Function to generate branching process up to certain generation
branch = function(n,sample_func,pars) {

    z = tibble(Generation=0,offspring=1,Label=1)

    for(i in 1:n){
    zmin1 = z %>% filter(Generation==(i-1)) %>% summarise(sum(offspring)) %>% as.numeric(.)
      z = bind_rows(z,sample_func(pars,z_nmin1=zmin1,n=i))
     }
  z

}


# Probability generating function for our branching process to generate probability values from poisson
poisson_dist<-function(n,lambda){
  l<-c()
  i<-0
  while(i<n)
  {
    sample<-dpois(i,lambda)
    l<-c(l,sample)
    i=i+1
  }
  return (l)
}
```

Supercritical Case: μ of Probability Generating Function > 1

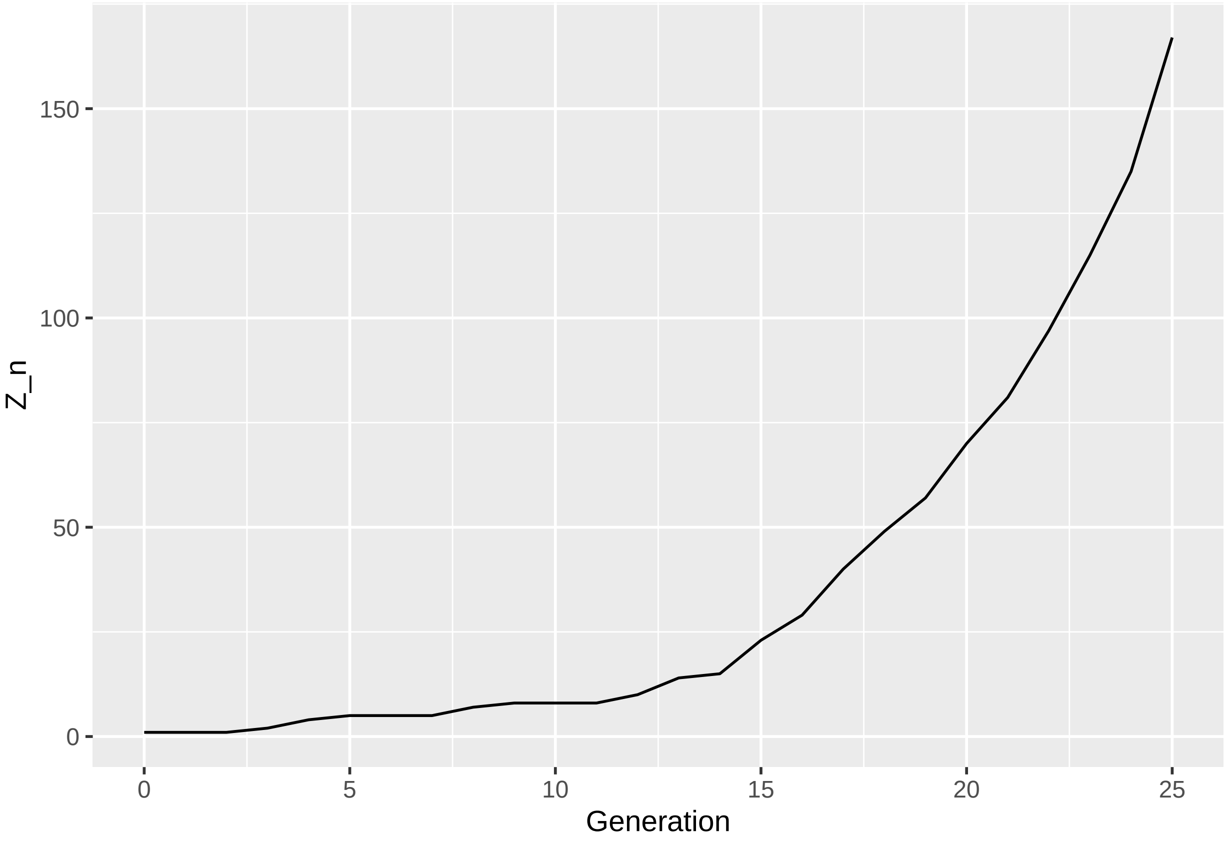
```
# Max generations: 25
MaxN = 25

# Offspring Distribution: with mean greater than 1
process_super= branch(MaxN,next_gen,list(c(1/6,1/2,1/3)))

# Simulation of Branching Processes for Supercritical Case
process_super = process_super %>% group_by(Generation) %>% mutate(Position = (Label - mean(Label))/(max

p0_super = ggplot(process_super %>% group_by(Generation) %>% summarise(Z_n = n()), aes(y=Z_n,x=Generati

p0_super
```



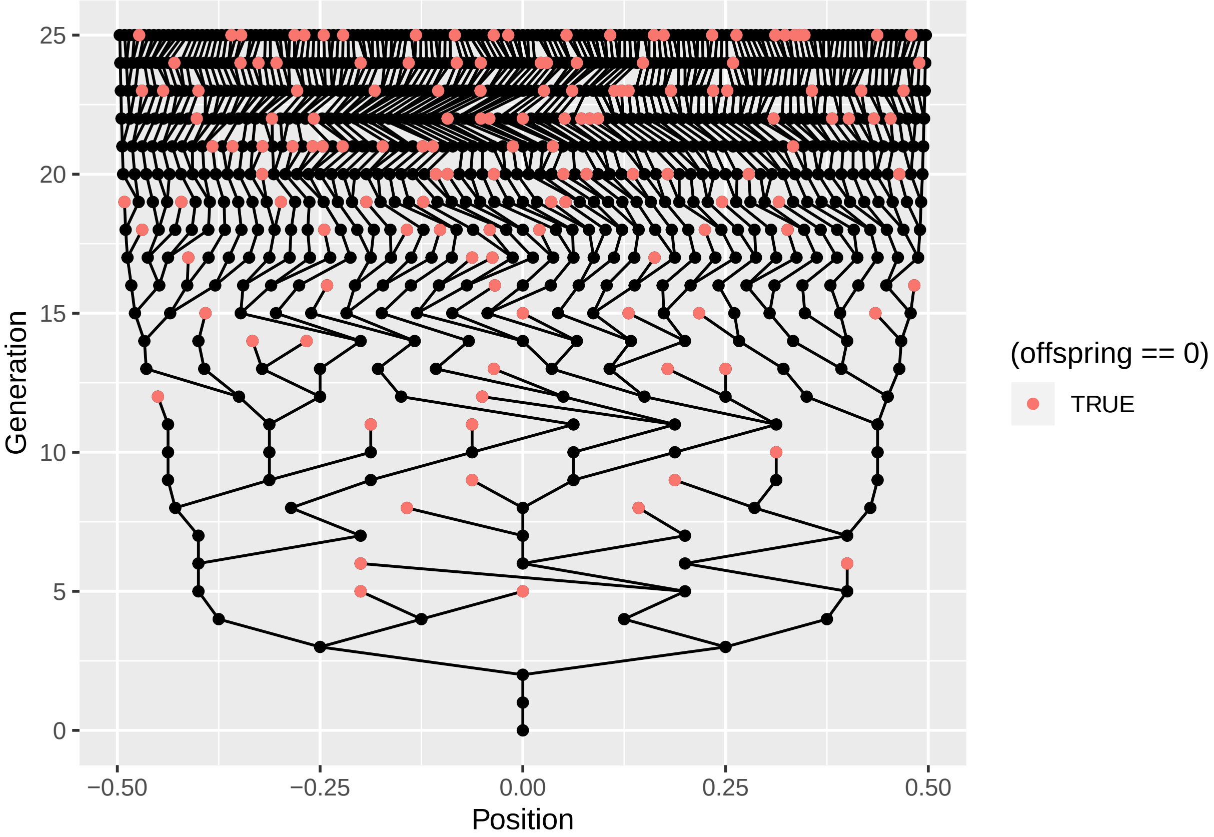
```
p1_super = ggplot(process_super, aes(y=Generation,x=Position)) + geom_point()
segment_frame=NULL

for(i in 0:(MaxN-1)){
  current_gen = process_super %>% filter(Generation==i) %>% filter(offspring>0) %>% mutate(EndOff = cum
  if(nrow(current_gen)==0) break;
  next_gen = process_super %>% filter(Generation == i+1) %>% ungroup(.)
  for(j in 1:nrow(current_gen)){
      if(current_gen$offspring[j] > 0){
        for(k in current_gen$StartOff[j]:current_gen$EndOff[j]){
          segment_frame=bind_rows(segment_frame,tibble(x=current_gen$Position[j],xend=next_gen$Position
        }
      }
  }

}

p1_super = p1_super + geom_segment(data=as.data.frame(segment_frame),aes(x=x,xend=xend,y=y,yend=yend))

p1_super
```



Critical Case: μ of Probability Generating Function = 1

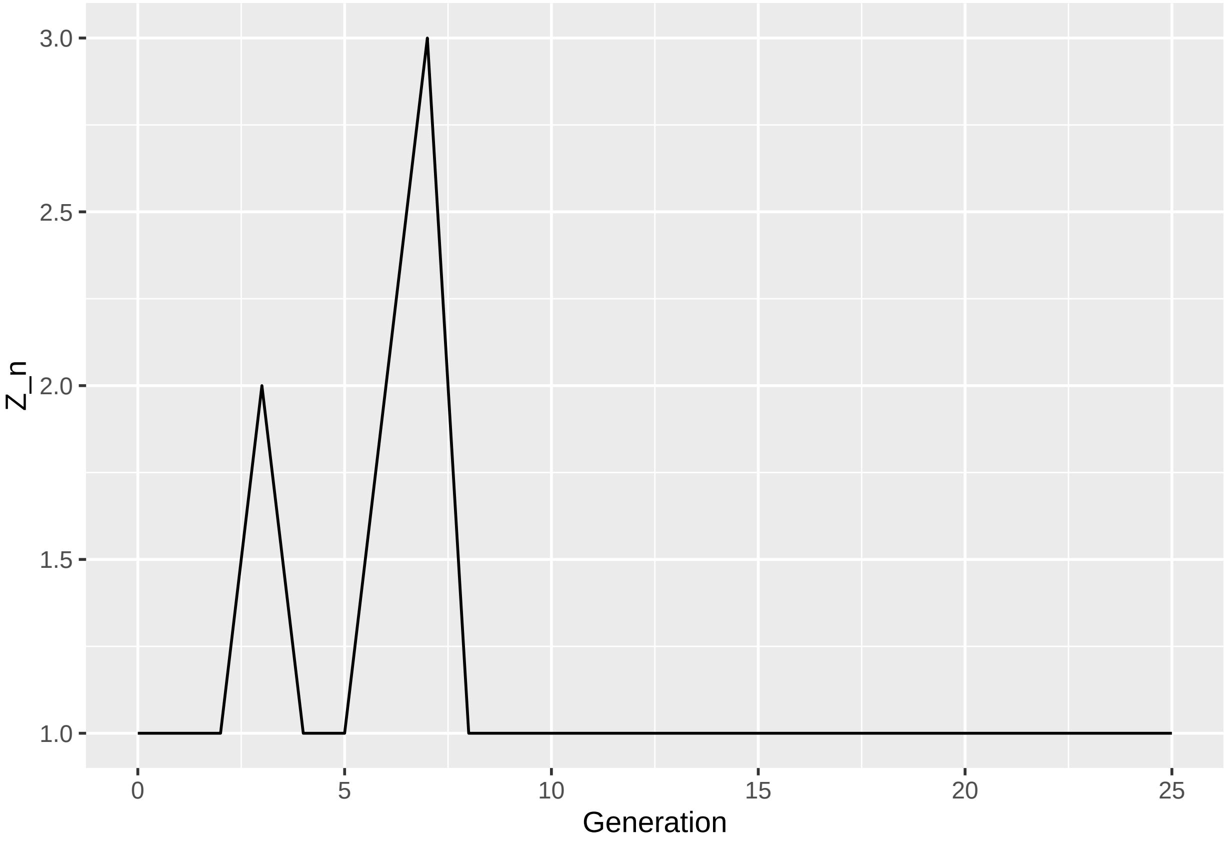
```
# Max generations: 25
MaxN = 25

# Offspring Distribution: with mean greater equal to 1
process= branch(MaxN,next_gen,list(c(1/3,1/3,1/3)))
```

```
process = process %>% group_by(Generation) %>% mutate(Position = (Label - mean(Label))/(max(Label)-min(


# Simulation of Branching Processes for Critical Case
p0 = ggplot(process %>% group_by(Generation) %>% summarise(Z_n = n()), aes(y=Z_n,x=Generation)) + geom_

p0
```

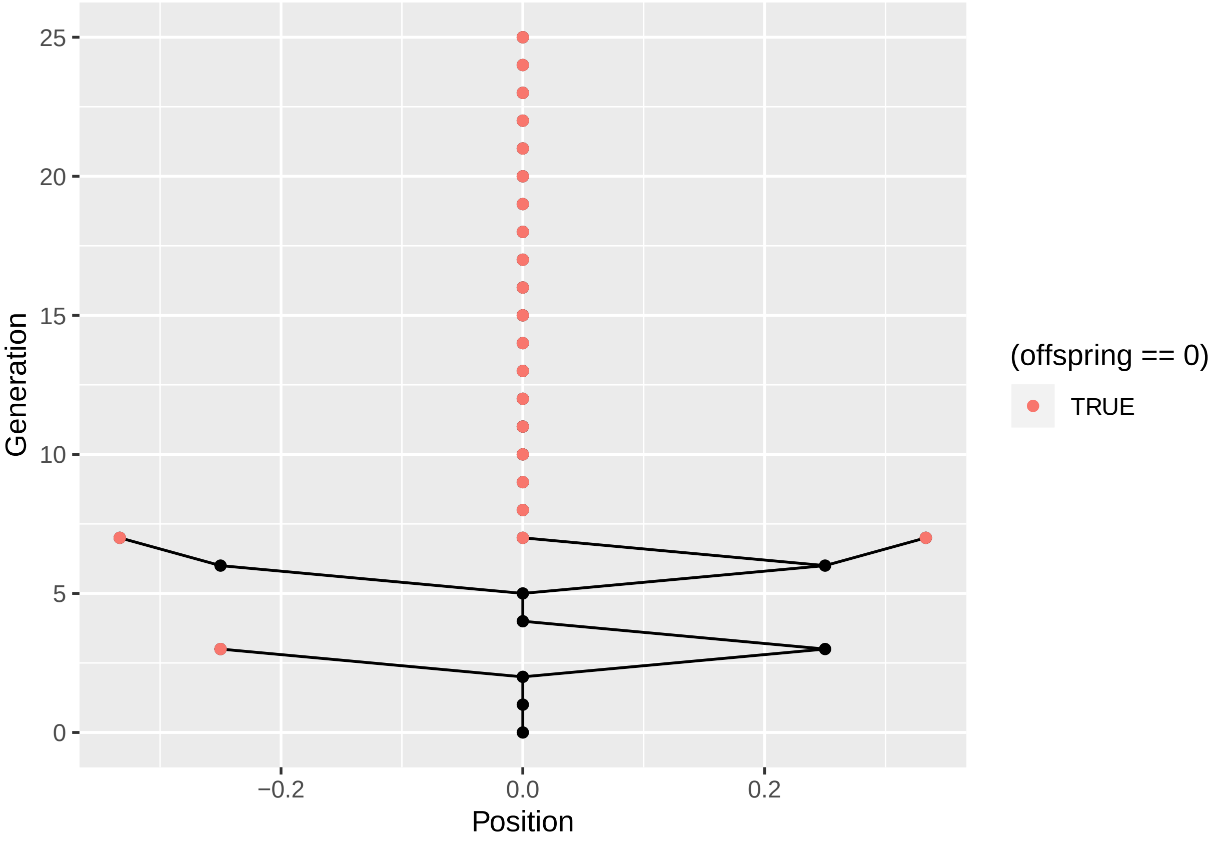


```
p1 = ggplot(process, aes(y=Generation,x=Position)) + geom_point()
segment_frame=NULL

for(i in 0:(MaxN-1)){
  current_gen = process %>% filter(Generation==i) %>% filter(offspring>0) %>% mutate(EndOff = cumsum(of
  if(nrow(current_gen)==0) break;
  next_gen = process %>% filter(Generation == i+1) %>% ungroup(.)
  for(j in 1:nrow(current_gen)){
      if(current_gen$offspring[j] > 0){
        for(k in current_gen$StartOff[j]:current_gen$EndOff[j]){
          segment_frame=bind_rows(segment_frame,tibble(x=current_gen$Position[j],xend=next_gen$Position
        }
      }
  }

}

p1= p1 + geom_segment(data=as.data.frame(segment_frame),aes(x=x,xend=xend,y=y,yend=yend)) + geom_point(
p1
```



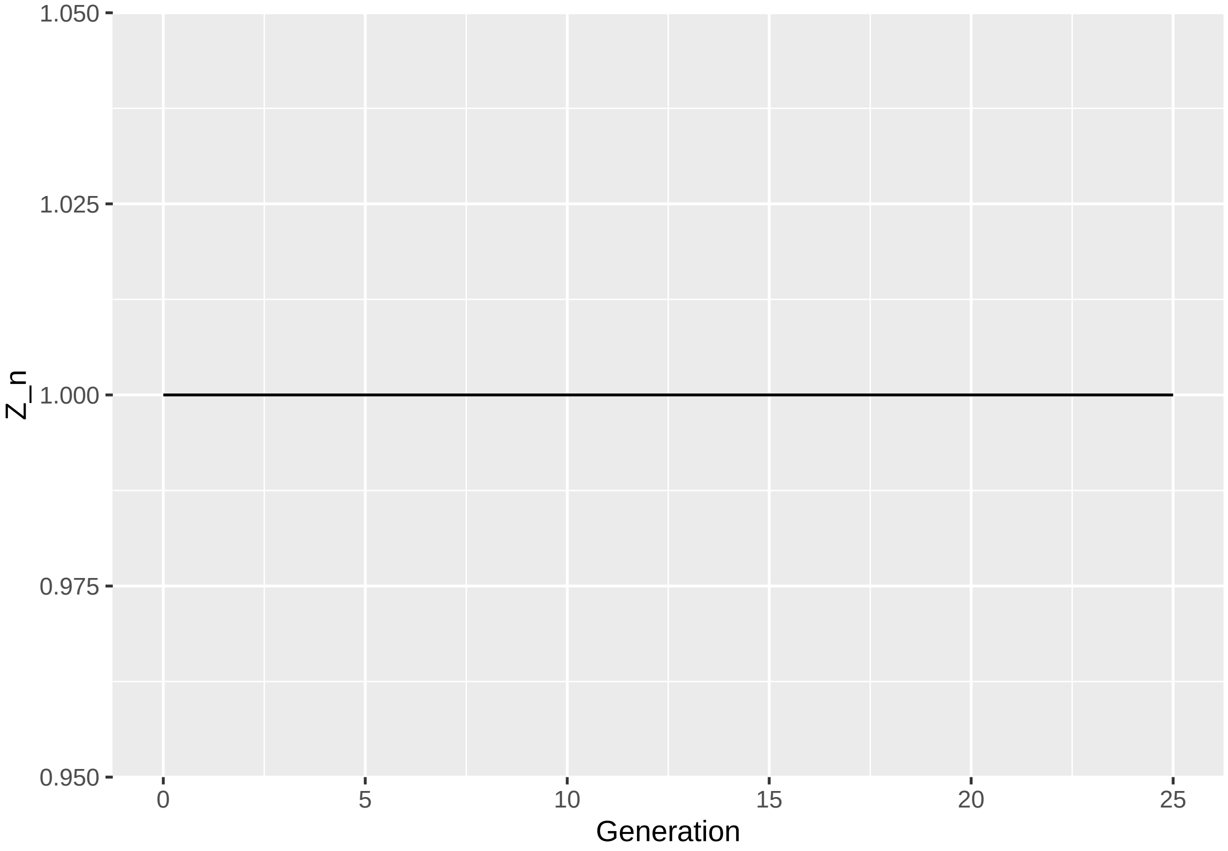
Subcritical Case: μ of Probability Generating Function < 1

```
MaxN = 25

# Offspring Distribution: with mean greater less than 1
process_sub= branch(MaxN,next_gen,list(c(1/2,1/4,1/4)))
process_sub = process_sub %>% group_by(Generation) %>% mutate(Position = (Label - mean(Label))/(max(Labe


# Simulation of Branching Processes for Subcritical Case
p0_sub = ggplot(process_sub %>% group_by(Generation) %>% summarise(Z_n = n()), aes(y=Z_n,x=Generation))

p0_sub
```



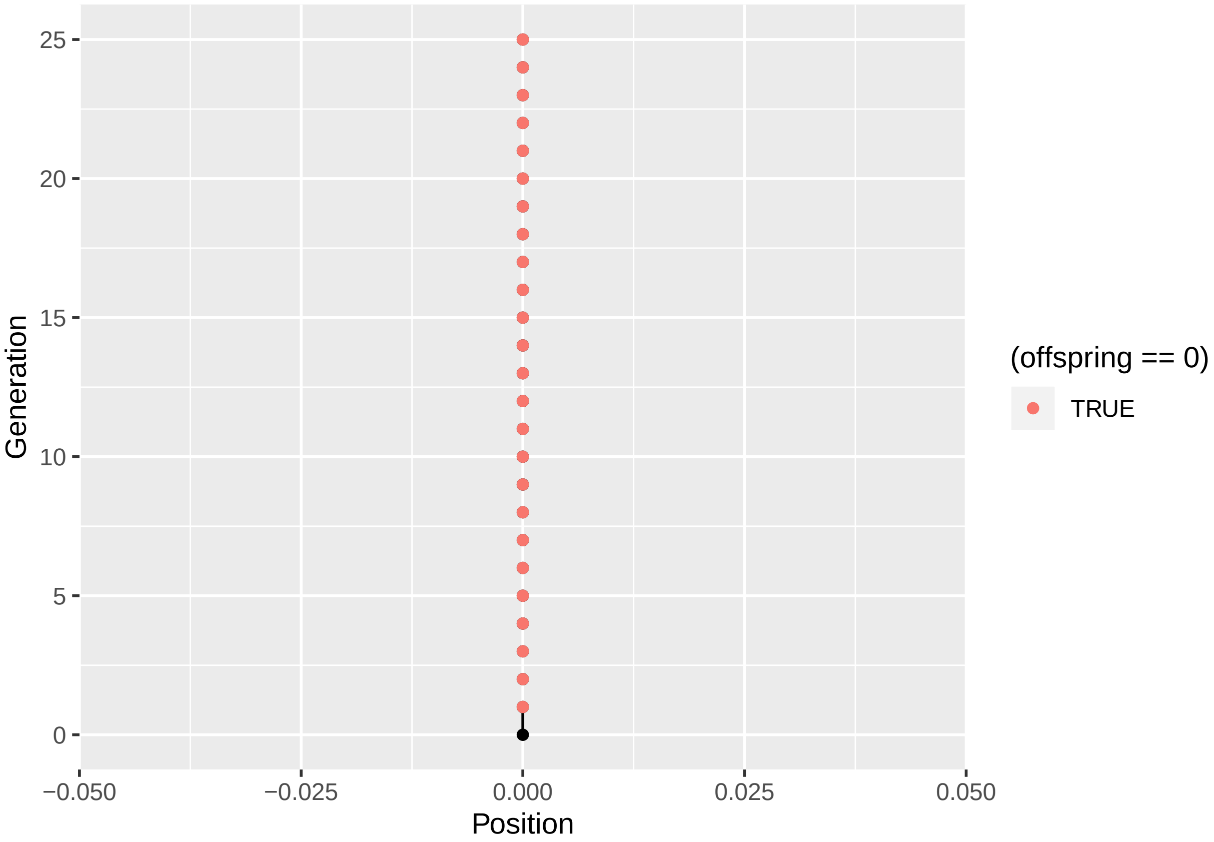
```r
p1_sub = ggplot(process_sub, aes(y=Generation,x=Position)) + geom_point()
segment_frame=NULL

for(i in 0:(MaxN-1)){
  current_gen = process_sub %>% filter(Generation==i) %>% filter(offspring>0) %>% mutate(EndOff = cumsu
  if(nrow(current_gen)==0) break;
  next_gen = process_sub %>% filter(Generation == i+1) %>% ungroup(.)
  for(j in 1:nrow(current_gen)){
      if(current_gen$offspring[j] > 0){
        for(k in current_gen$StartOff[j]:current_gen$EndOff[j]){
          segment_frame=bind_rows(segment_frame,tibble(x=current_gen$Position[j],xend=next_gen$Position
        }
      }
  }

}

p1_sub= p1_sub + geom_segment(data=as.data.frame(segment_frame),aes(x=x,xend=xend,y=y,yend=yend)) + geo

p1_sub
```



Poisson Probability Generating Function with $\lambda = 5$, up to n=10

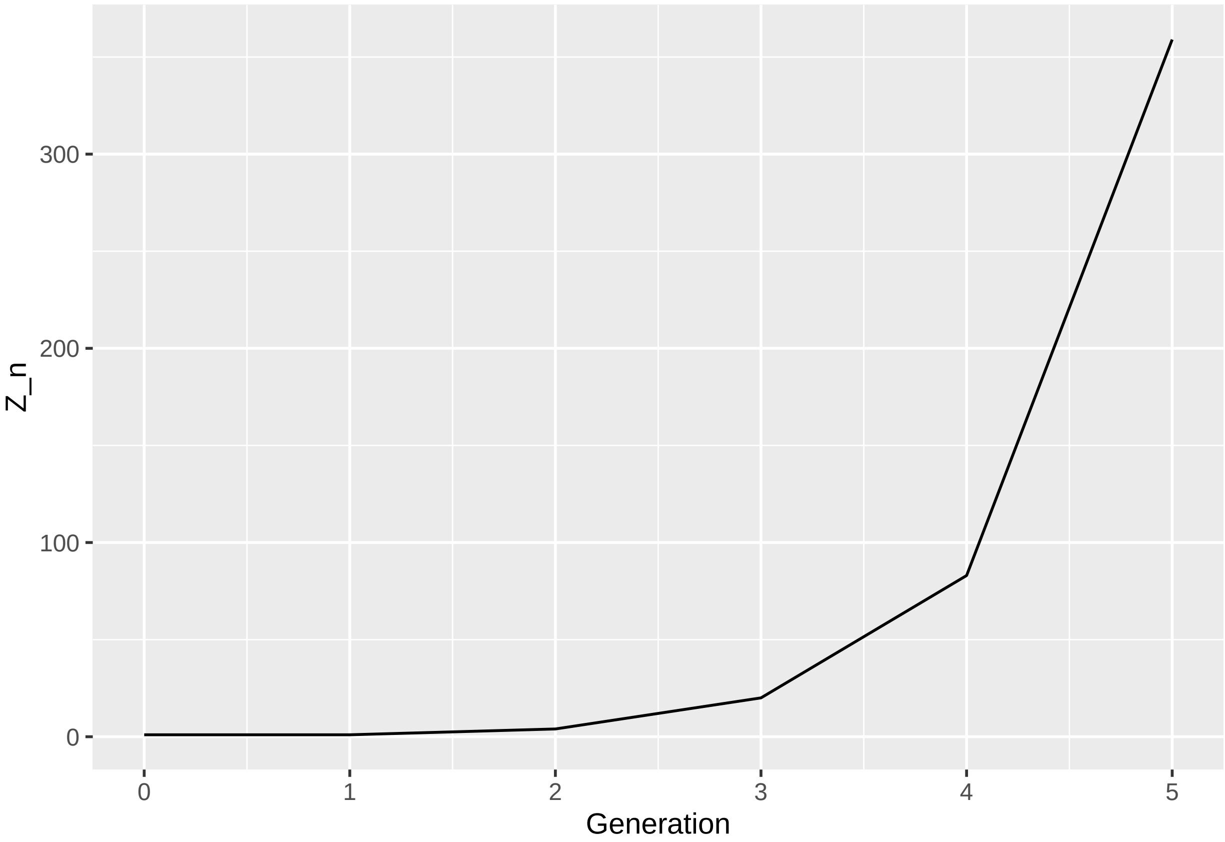
```
# Max generations: 5
MaxN = 5


# Offspring Distribution: Poisson Generating Function up to 10 children with lambda=5

process_pois= branch(MaxN,next_gen,list(poisson_dist(10,5)))
process_pois = process_pois %>% group_by(Generation) %>% mutate(Position = (Label - mean(Label))/(max(La

p0_pois = ggplot(process_pois %>% group_by(Generation) %>% summarise(Z_n = n()), aes(y=Z_n,x=Generation)

p0_pois
```



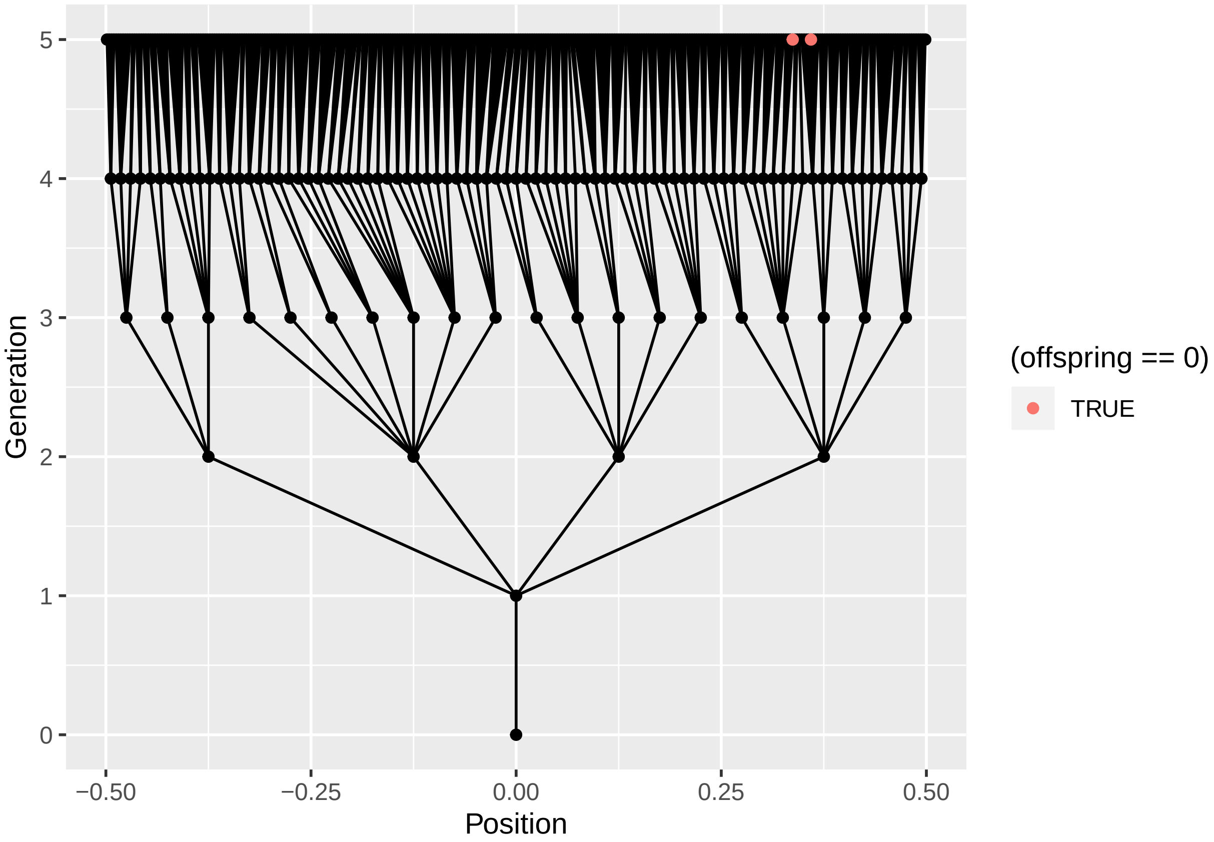
```
p1_pois = ggplot(process_pois, aes(y=Generation,x=Position)) + geom_point()
segment_frame=NULL

for(i in 0:(MaxN-1)){
  current_gen = process_pois %>% filter(Generation==i) %>% filter(offspring>0) %>% mutate(EndOff = cumsu
  if(nrow(current_gen)==0) break;
  next_gen = process_pois %>% filter(Generation == i+1) %>% ungroup(.)
  for(j in 1:nrow(current_gen)){
      if(current_gen$offspring[j] > 0){
        for(k in current_gen$StartOff[j]:current_gen$EndOff[j]){
          segment_frame=bind_rows(segment_frame,tibble(x=current_gen$Position[j],xend=next_gen$Position
        }
      }
  }

}

p1_pois= p1_pois + geom_segment(data=as.data.frame(segment_frame),aes(x=x,xend=xend,y=y,yend=yend)) + g

p1_pois
```


Generation
Position
0
1
2
3
4
5
−0.50
−0.25
0.00
0.25
0.50
(offspring == 0)
● TRUE