Survival Analysis

## Simulation

library(ggplot2)

library(gganimate)

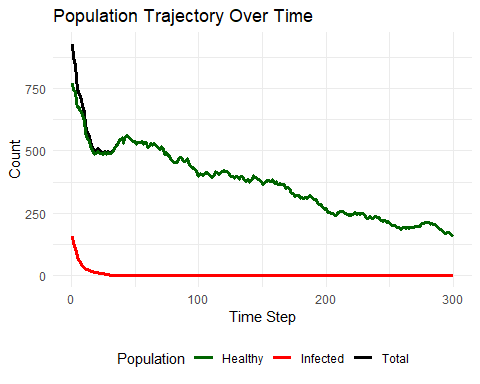
library(dplyr)

# Parameters  
n\_animals <- 1000  
n\_steps <- 300  
forest\_size <- 500  
food\_prob <- 0.20 # prob of food at each location  
starve\_limit <- 10 # steps without food before starving  
repro\_chance <- 0.6 # chance of reproducing when eating  
trans\_prob <- 0.50 # transmission probability when sharing a cell  
disease\_death\_prob <- 0.2 # base death prob per step if infected  
prop\_infected <- 0.2 # 5% of the initial populations starts with infected  
rage <- 30 # The reproduction reduction age  
  
# Initialize animals  
animals <- data.frame(  
 id = 1:n\_animals,  
 age\_enter = sample(1:50, n\_animals, replace = TRUE),  
 speed = runif(n\_animals, 1, 10),  
 strength = runif(n\_animals, 0.5, 2),  
 resistance = runif(n\_animals, 0, 1),  
 hunger = 0, # time since last food  
 alive = TRUE,  
 time = 0,  
 status = 0,  
 infected = FALSE, # disease status  
 x = sample(1:forest\_size, n\_animals, replace = TRUE), # Randomly Spawnpoint  
 y = sample(1:forest\_size, n\_animals, replace = TRUE),  
 born = 0,  
 step = 0  
)  
  
# Set age after age\_enter is created  
animals$age <- animals$age\_enter  
  
# Seed infection in a few animals  
animals$infected[sample(1:n\_animals, n\_animals\*prop\_infected)] <- TRUE   
  
death\_log <- data.frame(id=integer(), time=integer(), cause=character())  
next\_id <- n\_animals + 1  
  
# Store snapshots for animation  
snapshots <- list()  
pop\_counts <- data.frame(time = integer(), alive = integer(),   
 infected = integer(), healthy = integer())  
  
  
for (t in 1:n\_steps) {  
 animals$time[animals$alive] <- animals$time[animals$alive] + 1  
   
 # If still alive then the age plus 1 after each loop  
 animals$age[animals$alive] <- animals$age[animals$alive] + 1  
   
 animals$step <- animals$step + 1  
   
 # Old age mortality  
 for (i in which(animals$alive)) {  
 if (runif(1) < 0.003 \* animals$age[i]) {  
 animals$alive[i] <- FALSE  
 animals$status[i] <- 1  
 death\_log <- rbind(death\_log,  
 data.frame(id=animals$id[i], time=t, cause="old\_age"))  
 }  
 }  
   
 # Generate food  
 food <- matrix(0, nrow = forest\_size, ncol = forest\_size)  
 food[runif(forest\_size\*forest\_size) < food\_prob] <- 1 # good food  
 food[runif(forest\_size\*forest\_size) < 0.01] <- -1 # poisonous  
   
 for (i in which(animals$alive)) {  
 # Random movement  
 move\_x <- sample(c(-1,0,1), 1) \* round(animals$speed[i])  
 move\_y <- sample(c(-1,0,1), 1) \* round(animals$speed[i])  
 animals$x[i] <- max(1, min(forest\_size, animals$x[i] + move\_x))  
 animals$y[i] <- max(1, min(forest\_size, animals$y[i] + move\_y))  
   
 # Hunger increases  
 animals$hunger[i] <- animals$hunger[i] + 1  
   
 # Food check  
 cell <- food[animals$x[i], animals$y[i]]  
 if (cell == 1) {  
 animals$hunger[i] <- 0  
   
 # Reduced Reproduction chance if above a threshold age  
 if (animals$age[i] > 10 && animals$age[i] < 50) {  
 repro\_chance\_i <- ifelse(animals$age[i] < rage, repro\_chance, repro\_chance/3)  
 if (runif(1) < repro\_chance\_i) {  
 baby <- animals[i,]  
 baby$id <- next\_id; next\_id <- next\_id + 1  
 baby$age <- 0  
 baby$age\_enter <- 0  
 baby$x <- animals$x[i]  
 baby$y <- animals$y[i]  
 baby$time <- 0  
 baby$hunger <- 0  
 baby$infected <- FALSE # babies start healthy  
 baby$alive <- TRUE  
 baby$status <- 0  
 baby$born <- t  
 animals <- rbind(animals, baby)  
 }  
 }  
 } else if (cell == -1) {  
 if (runif(1) > animals$resistance[i]) {  
 animals$alive[i] <- FALSE  
 animals$status[i] <- 1  
 death\_log <- rbind(death\_log, data.frame(id=animals$id[i],   
 time=t, cause="poison"))  
 }  
 }  
   
 # Disease mortality (if infected)  
 if (animals$infected[i] && animals$alive[i]) {  
 if (runif(1) < disease\_death\_prob \* (1 - animals$resistance[i])  
 \* (1 + animals$age[i]/50)) {  
 animals$alive[i] <- FALSE  
 animals$status[i] <- 1  
 death\_log <- rbind(death\_log, data.frame(id=animals$id[i],   
 time=t, cause="disease"))  
 }  
 }  
 }  
   
 # Starvation deaths  
 starved <- which(animals$alive & animals$hunger > starve\_limit)  
 if (length(starved) > 0) {  
 animals$alive[starved] <- FALSE  
 animals$status[starved] <- 1  
 death\_log <- rbind(death\_log,  
 data.frame(id=animals$id[starved], time=t, cause="starve"))  
 }  
   
 # Fighting  
 living\_animals <- which(animals$alive)  
 if (length(living\_animals) > 1) {  
 coords <- paste(animals$x[living\_animals], animals$y[living\_animals])  
 coord\_table <- table(coords)  
 contested\_coords <- names(coord\_table)[coord\_table > 1]  
   
 for (coord in contested\_coords) {  
 occupants <- living\_animals[coords == coord]  
 if (length(occupants) > 1) {  
   
 # Separate adults and juveniles - adults don't fight babies  
 adults <- occupants[animals$age[occupants] >= 10]  
 juveniles <- occupants[animals$age[occupants] < 10]  
   
 # Only adults fight each other, juveniles are ignored  
 if (length(adults) > 1) {  
   
 # Probabilistic combat (more realistic)  
 # Stronger animals have higher chance to win but not guaranteed  
 strengths <- animals$strength[adults]  
 ages <- animals$age[adults]  
   
 # Combat score = strength \* age\_factor \* random\_factor  
 age\_factors <- pmin(2, ages / 20) # Peak fighting at age 40  
 combat\_scores <- strengths \* age\_factors \* runif(length(adults),0.5, 1.5)  
   
 # Winner based on combat score  
 winner\_idx <- which.max(combat\_scores)  
 winner <- adults[winner\_idx]  
   
 # Not all losers die - some might flee  
 potential\_casualties <- setdiff(adults, winner)  
 strength\_diff <- strengths[winner\_idx] - strengths[-winner\_idx]  
   
 # Higher strength difference = higher casualty rate  
 casualty\_prob <- pmin(0.8, pmax(0.1, 0.3 + strength\_diff \* 0.3))  
 casualties <- potential\_casualties[  
 runif(length(potential\_casualties)) < casualty\_prob]  
   
 if (length(casualties) > 0) {  
 animals$alive[casualties] <- FALSE  
 animals$status[casualties] <- 1  
 death\_log <- rbind(death\_log,  
 data.frame(id=animals$id[casualties],   
 time=t, cause="combat"))  
 }  
   
 } else if (length(adults) == 1 && length(juveniles) > 1) {  
 # Single adult with multiple juveniles - carrying capacity  
 # Adult might kill some juveniles if overcrowded (>5 total)  
 if (length(occupants) > 5) {  
 excess\_juveniles <- length(juveniles) - 3 # Keep max 3 juveniles  
 if (excess\_juveniles > 0) {  
 # Kill weakest juveniles  
 juv\_strengths <- animals$strength[juveniles]  
 casualties <- juveniles[order(juv\_strengths)[1:excess\_juveniles]]  
   
 animals$alive[casualties] <- FALSE  
 animals$status[casualties] <- 1  
 death\_log <- rbind(death\_log,  
 data.frame(id=animals$id[casualties],   
 time=t, cause="overcrowding"))  
 }  
 }  
 }  
 }  
 }  
 }  
   
 # Disease spread  
 coords <- paste(animals$x, animals$y)  
 for (cell in unique(coords)) {  
 occupants <- which(coords == cell & animals$alive)  
 if (length(occupants) > 1) {  
 if (any(animals$infected[occupants])) {  
 new\_infected <- occupants[!animals$infected[occupants]]  
 infected\_now <- new\_infected[runif(length(new\_infected)) < trans\_prob]  
 animals$infected[infected\_now] <- TRUE  
 }  
 }  
 }  
  
 # Save snapshot  
 alive\_count <- sum(animals$alive)  
 infected\_count <- sum(animals$alive & animals$infected)  
 healthy\_count <- alive\_count - infected\_count  
   
 # Store population data  
 pop\_counts <- rbind(pop\_counts, data.frame(  
 time = t,   
 alive = alive\_count,   
 infected = infected\_count,  
 healthy = healthy\_count  
 ))  
   
 # Save snapshot with population info  
 current\_animals <- animals %>%  
 mutate(step = t)  
   
 snapshots[[t]] <- current\_animals  
}

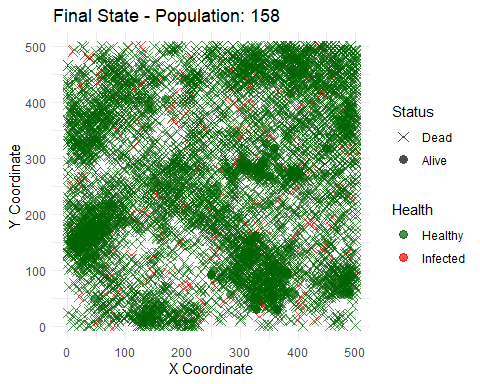
## Plot

library(ggplot2)  
  
ggplot(pop\_counts, aes(x=time)) +  
 geom\_line(aes(y=alive, color="Total"), size=1.2) +  
 geom\_line(aes(y=infected, color="Infected"), size=1.2) +  
 geom\_line(aes(y=healthy, color="Healthy"), size=1.2) +  
 scale\_color\_manual(values=c("Total"="black", "Infected"="red",   
 "Healthy"="darkgreen")) +  
 theme\_minimal() +  
 labs(title="Population Trajectory Over Time",   
 x="Time Step", y="Count", color="Population") +  
 theme(legend.position = "bottom")

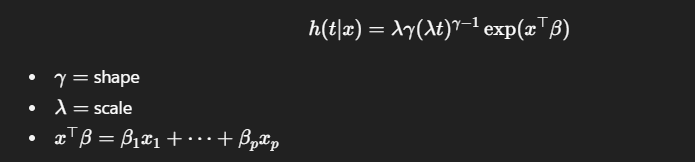
Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
ℹ Please use `linewidth` instead.



# Final state snapshot  
ggplot(animals, aes(x=x, y=y, color=infected, shape=alive)) +  
 geom\_point(size=3, alpha=0.7) +  
 scale\_shape\_manual(values=c(4, 16), name="Status", labels=c("Dead", "Alive")) +  
 scale\_color\_manual(values=c("darkgreen","red"), name="Health",   
 labels=c("Healthy", "Infected")) +  
 theme\_minimal() +  
 labs(title=paste("Final State - Population:", sum(animals$alive)),  
 x="X Coordinate", y="Y Coordinate")



## Model



library(survival)  
library(flexsurv)

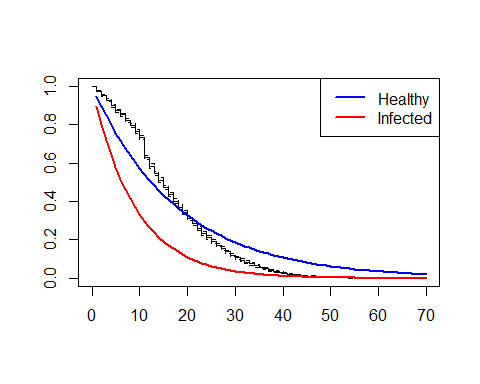
library(dplyr)  
  
animals <- animals %>%  
 filter(time > 0)  
  
# Fit model using actual time at risk  
fit <- flexsurvreg(  
 Surv(time, status) ~   
 age\_enter + speed + strength + resistance + infected,  
 data = animals,  
 dist = "exponential"  
)  
  
fit\_cox <- coxph(  
 Surv(time, status) ~   
 age\_enter + speed + strength + resistance + infected,  
 data = animals  
)  
  
fit

Call:  
flexsurvreg(formula = Surv(time, status) ~ age\_enter + speed +   
 strength + resistance + infected, data = animals, dist = "exponential")  
  
Estimates:   
 data mean est L95% U95% se exp(est)   
rate NA 0.061346 0.055027 0.068391 0.003403 NA  
age\_enter 3.857544 0.024061 0.021727 0.026395 0.001191 1.024352  
speed 5.099044 -0.000998 -0.011117 0.009120 0.005163 0.999002  
strength 1.245795 -0.006118 -0.065220 0.052984 0.030154 0.993901  
resistance 0.571318 -0.177351 -0.260333 -0.094369 0.042339 0.837486  
infectedTRUE 0.033693 0.679967 0.538974 0.820961 0.071937 1.973813  
 L95% U95%   
rate NA NA  
age\_enter 1.021965 1.026746  
speed 0.988945 1.009162  
strength 0.936862 1.054413  
resistance 0.770795 0.909947  
infectedTRUE 1.714247 2.272682  
  
N = 6767, Events: 6619, Censored: 148  
Total time at risk: 111986  
Log-likelihood = -25061.03, df = 6  
AIC = 50134.06

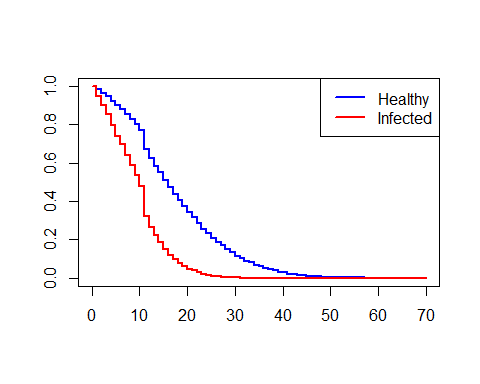
fit\_cox

Call:  
coxph(formula = Surv(time, status) ~ age\_enter + speed + strength +   
 resistance + infected, data = animals)  
  
 coef exp(coef) se(coef) z p  
age\_enter 0.037406 1.038115 0.001218 30.714 < 2e-16  
speed -0.002989 0.997015 0.005146 -0.581 0.561  
strength -0.012026 0.988046 0.030194 -0.398 0.690  
resistance -0.261452 0.769933 0.042385 -6.168 6.89e-10  
infectedTRUE 1.035945 2.817768 0.072995 14.192 < 2e-16  
  
Likelihood ratio test=1120 on 5 df, p=< 2.2e-16  
n= 6767, number of events= 6619

# flexsurvreg  
plot(fit, type="survival",   
 newdata=data.frame(age\_enter=0, speed=1, strength=1, resistance=0.5, infected=FALSE),  
 col="blue", lwd=2, ci=FALSE)  
  
plot(fit, type="survival",   
 newdata=data.frame(age\_enter=0, speed=1, strength=1, resistance=0.5, infected=TRUE),  
 col="red", lwd=2, ci=FALSE, add=TRUE)  
  
legend("topright", legend=c("Healthy","Infected"), col=c("blue","red"), lwd=2)



# coxph  
sf <- survfit(fit\_cox, newdata = data.frame(  
 age\_enter = 0, speed = 1, strength = 1, resistance = 0.5,   
 infected = c(FALSE, TRUE)  
))  
  
plot(sf, col = c("blue", "red"), lwd = 2, conf.int = FALSE)  
legend("topright", legend = c("Healthy", "Infected"), col = c("blue", "red"), lwd = 2)



times <- 1:100  
  
surv\_healthy <- summary(fit, t=times, type="survival",   
 newdata=data.frame(age\_enter=0, speed=1,   
 strength=1, resistance=0.5,   
 infected=FALSE))  
  
surv\_infected <- summary(fit, t=times, type="survival",   
 newdata=data.frame(age\_enter=0, speed=1,  
 strength=1, resistance=0.5,  
 infected=TRUE))  
  
mort\_table <- data.frame(  
 time = times,  
 surv\_healthy = round(surv\_healthy[[1]]$est,3),  
 surv\_infected = round(surv\_infected[[1]]$est,3)  
)  
  
print(mort\_table)

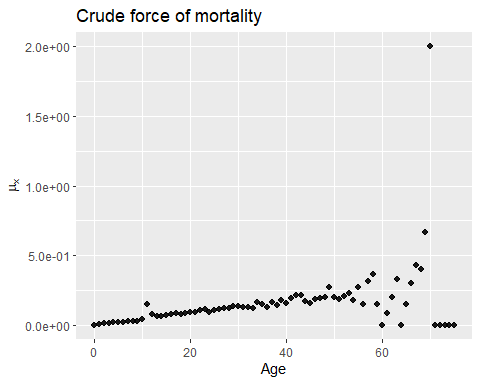
time surv\_healthy surv\_infected  
1 1 0.946 0.896  
2 2 0.895 0.802  
3 3 0.846 0.719  
4 4 0.800 0.644  
5 5 0.757 0.577  
6 6 0.716 0.517  
7 7 0.677 0.463  
8 8 0.640 0.415  
9 9 0.606 0.371  
10 10 0.573 0.333  
11 11 0.542 0.298  
12 12 0.512 0.267  
13 13 0.484 0.239  
14 14 0.458 0.214  
15 15 0.433 0.192  
16 16 0.410 0.172  
17 17 0.388 0.154  
18 18 0.367 0.138  
19 19 0.347 0.124  
20 20 0.328 0.111  
21 21 0.310 0.099  
22 22 0.293 0.089  
23 23 0.277 0.080  
24 24 0.262 0.071  
25 25 0.248 0.064  
26 26 0.235 0.057  
27 27 0.222 0.051  
28 28 0.210 0.046  
29 29 0.199 0.041  
30 30 0.188 0.037  
31 31 0.178 0.033  
32 32 0.168 0.030  
33 33 0.159 0.026  
34 34 0.150 0.024  
35 35 0.142 0.021  
36 36 0.134 0.019  
37 37 0.127 0.017  
38 38 0.120 0.015  
39 39 0.114 0.014  
40 40 0.108 0.012  
41 41 0.102 0.011  
42 42 0.096 0.010  
43 43 0.091 0.009  
44 44 0.086 0.008  
45 45 0.081 0.007  
46 46 0.077 0.006  
47 47 0.073 0.006  
48 48 0.069 0.005  
49 49 0.065 0.005  
50 50 0.062 0.004  
51 51 0.058 0.004  
52 52 0.055 0.003  
53 53 0.052 0.003  
54 54 0.049 0.003  
55 55 0.047 0.002  
56 56 0.044 0.002  
57 57 0.042 0.002  
58 58 0.039 0.002  
59 59 0.037 0.002  
60 60 0.035 0.001  
61 61 0.033 0.001  
62 62 0.032 0.001  
63 63 0.030 0.001  
64 64 0.028 0.001  
65 65 0.027 0.001  
66 66 0.025 0.001  
67 67 0.024 0.001  
68 68 0.023 0.001  
69 69 0.021 0.001  
70 70 0.020 0.000  
71 71 0.019 0.000  
72 72 0.018 0.000  
73 73 0.017 0.000  
74 74 0.016 0.000  
75 75 0.015 0.000  
76 76 0.014 0.000  
77 77 0.014 0.000  
78 78 0.013 0.000  
79 79 0.012 0.000  
80 80 0.012 0.000  
81 81 0.011 0.000  
82 82 0.010 0.000  
83 83 0.010 0.000  
84 84 0.009 0.000  
85 85 0.009 0.000  
86 86 0.008 0.000  
87 87 0.008 0.000  
88 88 0.007 0.000  
89 89 0.007 0.000  
90 90 0.007 0.000  
91 91 0.006 0.000  
92 92 0.006 0.000  
93 93 0.006 0.000  
94 94 0.005 0.000  
95 95 0.005 0.000  
96 96 0.005 0.000  
97 97 0.004 0.000  
98 98 0.004 0.000  
99 99 0.004 0.000  
100 100 0.004 0.000

## Life Table

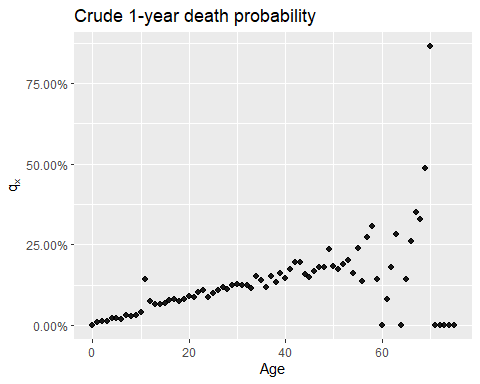
# P\_xt: counts of animals alive by (age, time)  
# D\_x: deaths by age (nearest birthday)  
# E^c\_x: central exposed-to-risk by trapezium rule across time steps  
  
library(dplyr)  
library(tidyverse)

library(scales)

# 1) Build P\_xt: counts alive by (age, step)  
P\_xt <- bind\_rows(snapshots) %>%  
 filter(!is.na(age)) %>%  
 group\_by(step, age) %>%  
 summarise(P = sum(alive, na.rm = TRUE), .groups = "drop") %>%  
 arrange(step, age)  
  
time\_points <- sort(unique(P\_xt$step))  
  
# 2) Build D\_x: deaths by age  
# Extract (id, step, age) for all individuals from snapshots  
age\_lookup <- bind\_rows(snapshots) %>%  
 select(id, step, age)  
  
# Join death\_log with ages at time of death  
death\_ages <- death\_log %>%  
 rename(step = time) %>%   
 left\_join(age\_lookup, by = c("id", "step")) # Append the age\_lookup to death\_ages  
  
# Summarise deaths by age  
D\_x\_df <- death\_ages %>%  
 filter(!is.na(age)) %>%  
 group\_by(age) %>%  
 summarise(D\_x = n(), .groups = "drop")  
  
# 3) Compute exposures E^c\_x via trapezium rule  
# Ensure full age grid  
ages <- seq(min(P\_xt$age), max(P\_xt$age), 1)  
P\_full <- expand.grid(step = time\_points, age = ages) %>%  
 left\_join(P\_xt, by = c("step", "age")) %>%  
 mutate(P = replace\_na(P, 0)) %>%  
 arrange(age, step)  
  
# Trapezium sum per age  
Ecx <- P\_full %>%  
 group\_by(age) %>%  
 summarise(E\_cx = sum((P[-length(P)] + P[-1]) / 2), .groups = "drop") %>%  
 left\_join(D\_x\_df, by = "age") %>%  
 mutate(D\_x = replace\_na(D\_x, 0)) %>%  
 filter(E\_cx > 0)  
  
# 4) Crude estimators  
Ecx <- Ecx %>%  
 mutate(  
 mu\_hat = round(D\_x / E\_cx,3),  
 q\_hat = round(1 - exp(-mu\_hat),3),  
 p\_hat = round(1 - q\_hat,3)  
 )  
  
# 6) Results  
res <- Ecx %>% select(age, D\_x, E\_cx, mu\_hat, q\_hat, p\_hat)  
  
# Plots  
p1 <- ggplot(res, aes(x = age)) +  
 geom\_point(aes(y = mu\_hat), size = 1.8, alpha = 0.9) +  
 labs(title = "Crude force of mortality",  
 y = expression(mu[x]), x = "Age") +  
 scale\_y\_continuous(labels = scales::scientific)  
  
p2 <- ggplot(res, aes(x = age)) +  
 geom\_point(aes(y = q\_hat), size = 1.8, alpha = 0.9) +  
 labs(title = "Crude 1-year death probability",  
 y = expression(q[x]), x = "Age") +  
 scale\_y\_continuous(labels = scales::percent\_format(accuracy = 0.01))  
  
print(p1)



print(p2)



head(res, 20)

# A tibble: 20 × 6  
 age D\_x E\_cx mu\_hat q\_hat p\_hat  
 <dbl> <int> <dbl> <dbl> <dbl> <dbl>  
 1 0 0 5748 0 0 1   
 2 1 50 5714. 0.009 0.009 0.991  
 3 2 61 5659 0.011 0.011 0.989  
 4 3 72 5602. 0.013 0.013 0.987  
 5 4 115 5501 0.021 0.021 0.979  
 6 5 118 5394 0.022 0.022 0.978  
 7 6 101 5304. 0.019 0.019 0.981  
 8 7 164 5146 0.032 0.031 0.969  
 9 8 144 5011 0.029 0.029 0.971  
10 9 152 4871 0.031 0.031 0.969  
11 10 198 4686. 0.042 0.041 0.959  
12 11 623 4080. 0.153 0.142 0.858  
13 12 291 3799 0.077 0.074 0.926  
14 13 239 3574. 0.067 0.065 0.935  
15 14 223 3367 0.066 0.064 0.936  
16 15 226 3154 0.072 0.069 0.931  
17 16 241 2926. 0.082 0.079 0.921  
18 17 231 2710. 0.085 0.081 0.919  
19 18 196 2524 0.078 0.075 0.925  
20 19 194 2344 0.083 0.08 0.92