STAT GR5206 HW4_mjs2364

Mathieu Sauterey - UNI: MJS2364

10 November 2017

Part1.

We continue working with the World Top Incomes Database, and the Pareto distribution, as in previous labs and homework. Recall that for most countries in most time periods, the upper end of the income distribution roughly follows a Pareto distribution, with probability density function

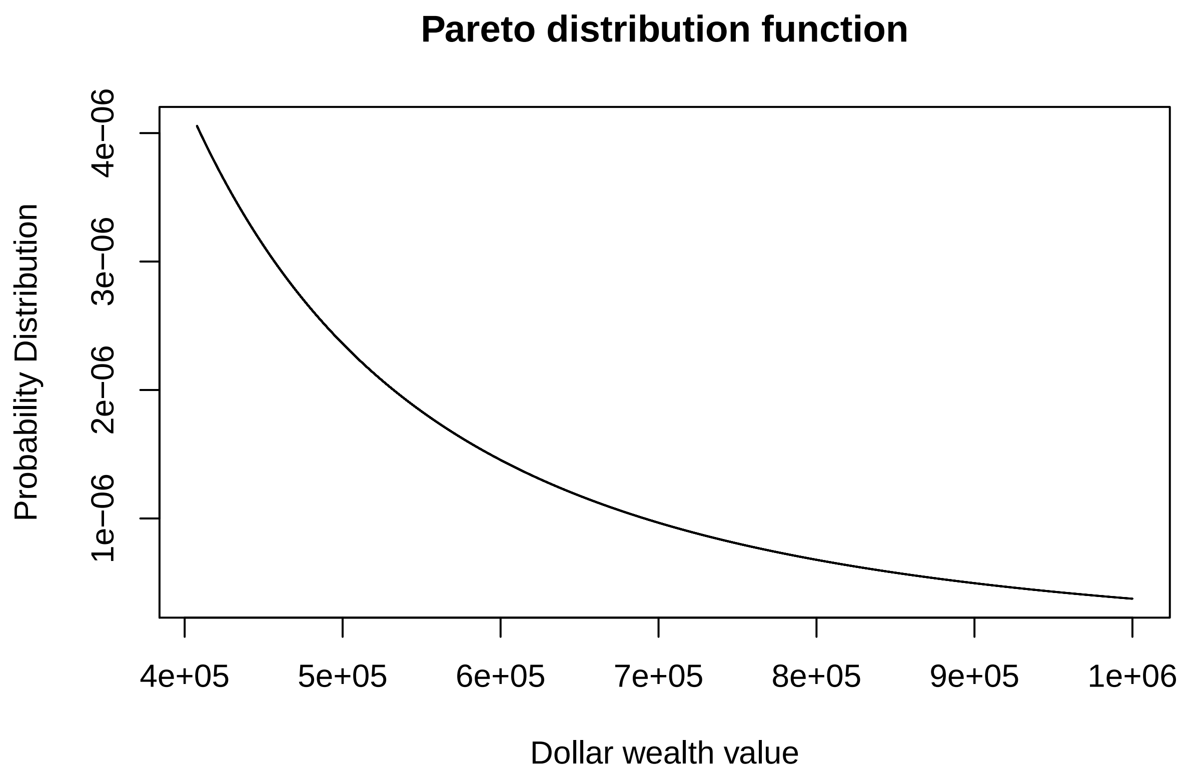
$$f(x) = (\frac{a-1}{x_{min}})(\frac{x}{x_{min}})^{-a}$$

Equation (1)

i. Define a function f which takes three inputs x, a vector, and scalars a and xmin having default values of a = a_hat and xmin = $407,760. The function should output f(x) for a given input x > xmin. Plot the function between xmin and 1,000,000. Make sure your plot is labeled appropriately.

```r
# Pareto Function in equation (1)
f <- function(x, a = 2.654, xmin = 407760){
  return((a-1)*((x/xmin)^-a)/xmin)
}

# Plots the function between xmin and 1,000,000
x <- seq(407760,10^6)
plot(x, f(x), xlab = "Dollar wealth value", ylab = "Probability Distribution",
     main="Pareto distribution function", type="l")
```



ii. Find the inverse function F^-1(u) and define a function 'upper.income'. The function should have three inputs u, a vector, and scalars a and xmin taking default values of a = a_hat and xmin = $407,760. The function should output F^-1(u) for a given input u in (0,1). Make sure upper.income(0.5) returns 620020.2.

```r
# Inverse Paretto Function
upper.income <- function(u, a = 2.654, xmin = 407760){
  return(xmin*(1-u)^(1/(-a+1)))
}

# Test for the function. Should return 620020.2.
upper.income(0.5)
```

```
## [1] 620020.2
```

As expected, upper.income(0.5) returns 620020.2

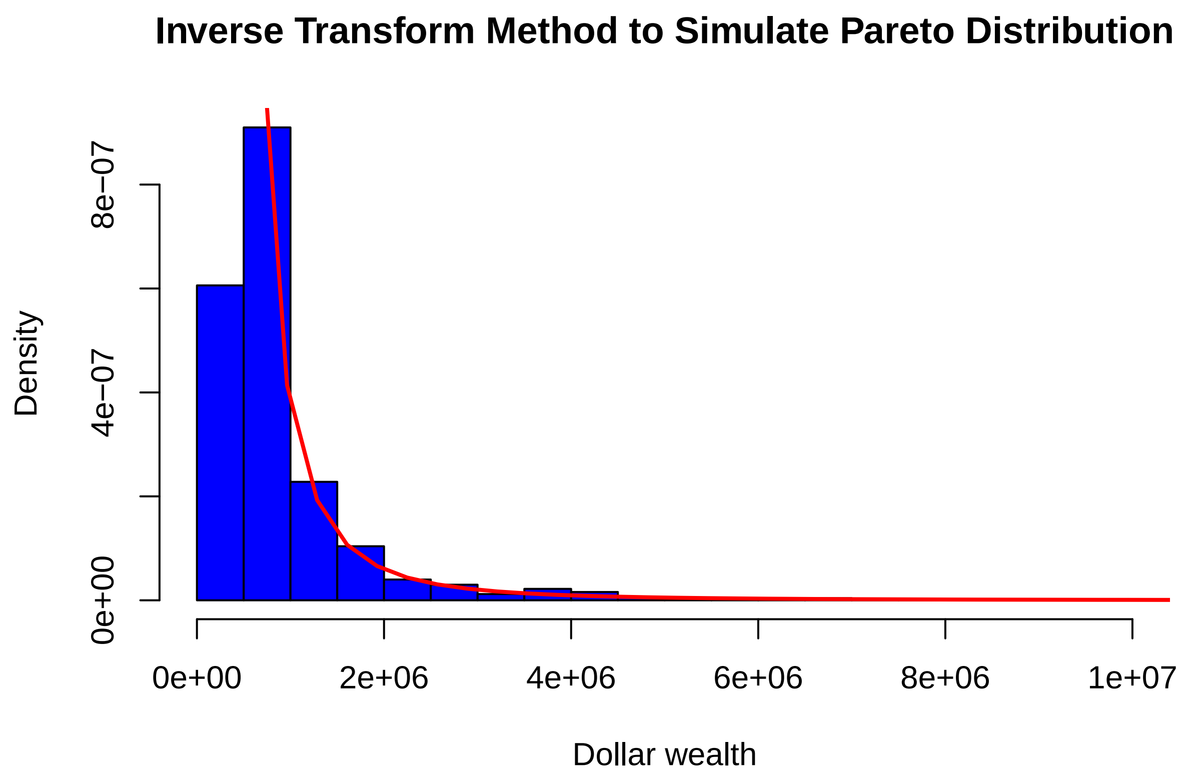
iii. Using the Inverse Transform Method, simulate 1000 draws from the Pareto distribution (1) and plot a histogram of your values. Overlay the simulated distribution with the Pareto density (1). Make sure to label the histogram appropriately.

```r
set.seed(2)

# Simulates 1,000 draws from the Inverse Pareto function
n <- 1000
u <- runif(n)
x_sim <- upper.income(u)

# Loads the graphic visualization package
library(ggplot2)

# Plots a histogram of the Paretto distribution overlaid with the Pareto density
hist(x_sim,breaks=100, probability = TRUE, xlab = "Dollar wealth", xlim = range(1:1e+07),
     main = "Inverse Transform Method to Simulate Pareto Distribution", col="blue")
curve(f,from = 0, to = max(x_sim), col="red", lwd=2, add = TRUE)
```



iv. Using your simulated set, estimate the median income for the richest 1% of the world. Compare your estimated 50th percentile to the actual 50th percentile of the Pareto distribution.

```
# Calculates the estimated median from the simulated set
quantile (x_sim,0.5)
```

```
##      50%
## 616793.7
```

```
# Calculates the actual median of the Pareto distribution
upper.income(0.5)
```

```
## [1] 620020.2
```

The estimated median is equal to $616,793 and it is slightly larger than the actual median of the Pareto distribution which is equal to $620,020. These two figures are still significantly close to each other.

Part 2

The file moretti.csv contains data compiled by the literary scholar Franco Moretti on the history of genres of novels in Britain between 1740 and 1900 (Gothic romances, mystery stories, stories, science fiction, etc.). Each record shows the name of the genre, the year it first appeared, and the year it died out.

It has been conjectured that that genres tend to appear together in bursts, bunches, or clusters. We want to know if this is right. We will simulate what we would expect to see if genres really did appear randomly, at a constant rate - a Poisson process. Under the assumption, the number of genres which appear in a given year should follow a Poisson distribution with some mean lambda, and every year should be independent of every other.

i. Assume the variables x1, x2, . . . , xn are independent and Poisson-distributed with mean lambda. Write a function poisLoglik, which takes as inputs a single number lambda and a vector data and returns the log-likelihood of that parameter value on that data. Return the value your function calculates when data = c(1, 0, 0, 1, 1) and lambda = 1.

```
# Function returns the log-likelihood value of input lambda on input data.
poisLoglik <- function(lambda, data){
    Sigma=log((lambda^data)*exp(-lambda)/factorial(data))
    return(sum(Sigma))
}

# Test for the function.
poisLoglik(1,c(1, 0, 0, 1, 1))
```

```
## [1] -5
```

When data = c(1, 0, 0, 1, 1) and lambda = 1, the function returns the value -5.

ii. Write a function count_new_genres which takes in a year, and returns the number of new genres which appeared in that year: 0 if there were no new genres that year, 1 if there was one, 3 if there were three, etc. Return the value your function calculates for 1803 and 1850.

```r
#Load Moretti's data
moretti <- read.csv("moretti.csv", as.is = TRUE)

# Function returns number of new genres that appeared in the given input year
count_new_genres <- function(year){
  return(sum(moretti$Begin==year))
}

# Test for the function in years 1803 and 1850.
count_new_genres(1803)
```

```
## [1] 0
```

```r
count_new_genres(1850)
```

```
## [1] 3
```

For 1803 the function returns 0 and for 1850 it returns 3, meaning that respectively 0 and 3 new genres appeared in these years.

iii. Create a vector, new genres, which counts the number of new genres which appeared in each year of the data, from 1740 to 1900. What positions in the vector correspond to the years 1803 and 1850? What should those values be? Is that what your vector new genres has for those years?

```r
# Creates a vector new_genre that counts the # of new genres per year of data
new_genres <- NULL
for (i in 1740:1900){
  new_genres <- c(new_genres,count_new_genres(i))
}

# Finds the position of year 1803 and 1850 in vector new_genre
which(1740:1900==1803)
```

```
## [1] 64
```

```r
which(1740:1900==1850)
```

```
## [1] 111
```

```r
# Returns the number of new genres appeared in 1803 and 1850
new_genres[64]
```

```
## [1] 0
```

```r
new_genres[111]
```

```
## [1] 3
```

Year 1803 corresponds to position 64 in the vector and year 1850 corresponds to position 111. Those values should be respectively 0 and 3 and this is excatly what my vector new_genres has for those years.

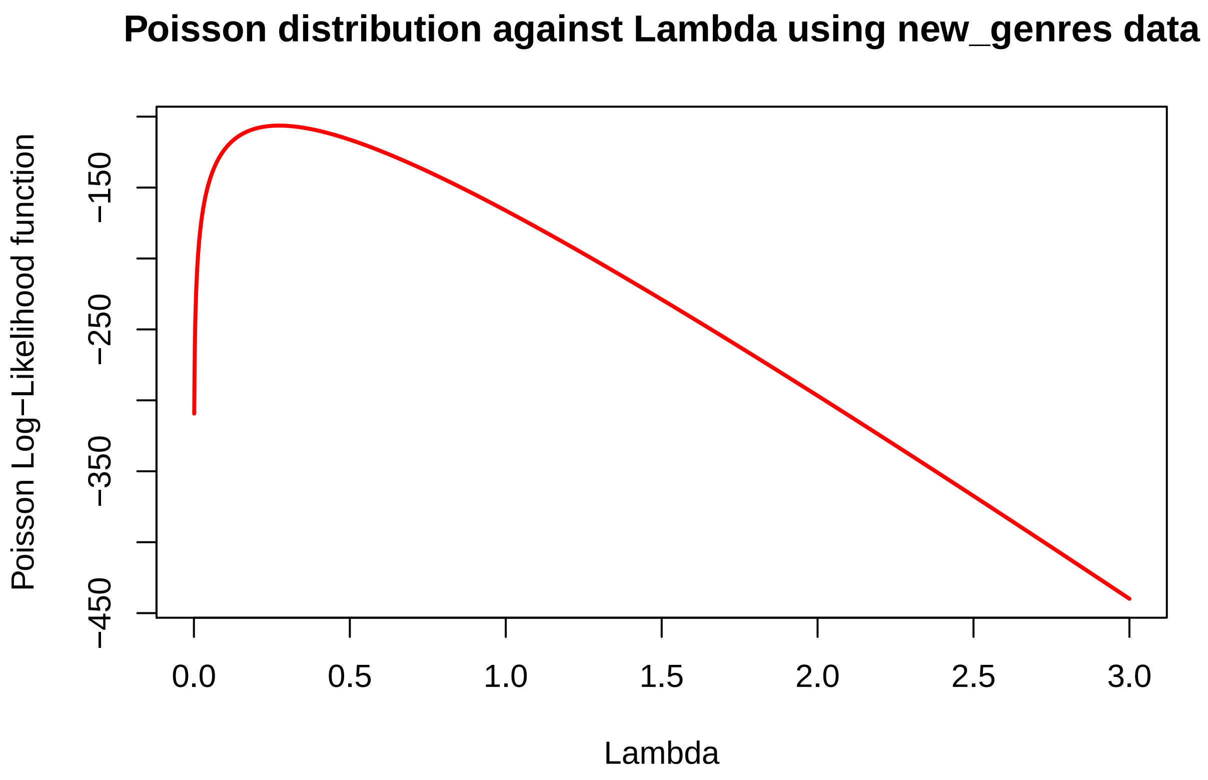
iv. Plot poisLoglik as a function of lambda using the new_genres data. I plotted lambda ranging from 0 to 3 (note that lambda = 0 has a log-likelihood of -Inf, but it shouldn't cause problems). The maximum should be lambda = 0.273.

```r
# Creates a vector of preimage lambda
x = seq(0,3,0.001)

# Creates an empty vector of function image
results <- rep(NA,length(x))

# Calculates the Poisson Log-Likelihood function images
for (lambda in 1:length(x)){
  results[lambda] <- poisLoglik(x[lambda],new_genres)
}

# Plots the Poisson Log-Likelihood function against lambda vector
plot(x, results, col="red", lwd=2, type="l", xlab = "Lambda",
     ylab = "Poisson Log-Likelihood function",
     main="Poisson distribution against Lambda using new_genres data")
```



```r
# Finds the lambda value lambda where the function maximum is found
x[which(results==max(results))]
```

```
## [1] 0.273
```

The maximum is indeed found at lambda = 0.273

v. Use nlm() to maximize the log likelihood to check the lambda = 0.273 value suggested in the previous question.

```r
# Calculates the opposite of the Poisson Log-likelihood function
cost_poisLoglik <- function(lambda, data){
    Sigma=log((lambda^data)*exp(-lambda)/factorial(data))
    return(-sum(Sigma))
}

# Finds the value lambda where cost_poisLoglik is minimal (poisLoglik is maximal)
est_lambda <- 0.25
nlm(cost_poisLoglik, est_lambda, data = new_genres)$estimate
```

```
## [1] 0.2732914
```

Using nlm() function, the maximum is correctly found at lambda = 0.273

vi. Create a vector, intergenre_intervals, which shows how many years elapsed between new genres appearing.

```r
#Creates a vector showing how many years elapsed between new genres appearing
intergenre_intervals <- diff(sort(moretti$Begin))

#Calculates the vector's mean, standard deviation and coefficient of variation
moretti_mean <- mean(intergenre_intervals)
moretti_sd   <- sd(intergenre_intervals)
moretti_coeff_var <- moretti_sd/moretti_mean

#Prints these results
moretti_mean
```

```
## [1] 3.44186
```

```r
moretti_sd
```

```
## [1] 3.705224
```

```r
moretti_coeff_var
```

```
## [1] 1.076518
```

The mean of the time intervals between genre appearances is 3.44 years, and the standard deviation is 3.70 years. The coefficient of variation is 1.076.

vii.

a. Write a line of code that generates 161 random draws from a Poisson distribution with lambda = 0.273.

```r
#Generates 161 random draws a Poisson Distributio with lambda = 0.273
rpois(161,lambda=0.273)
```

```
##   [1] 0 0 0 0 0 0 0 1 0 0 1 0 0 0 1 0 0 0 1 0 0 1 0 0 0 1 2 0 0 1 0 0 1 1 0
##  [36] 0 0 0 1 0 0 0 1 0 0 0 0 1 0 3 0 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0 0 0
##  [71] 0 1 1 0 0 1 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 2 0 0 0 0 2 0 0 0 0 0 1
## [106] 0 1 2 0 0 1 1 0 0 2 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 1 1 2 0 1 2 1 0 0 0
## [141] 1 1 1 1 0 1 0 1 1 1 0 0 1 0 0 0 1 0 1 1 0
```

b. Write function that takes as input a vector of numbers representing how many new genres appear in each year and returns the vector of the intervals between appearances.

```
# Function takes as vector input the # of new genres per year and returns intervals vector
new_to_intervals <- function(new){

  years <- NULL

  for (i in 1:length(new)){
    if (new[i]>0){
      years <- c(years,rep((1740:1900)[i],new[i]))
    }
  }

  intervals <- diff(sort(years))
  return(intervals)
}

# Checks if the function returns vector intergenre_intervals if given new_genres as input
identical(new_to_intervals(new_genres),intergenre_intervals)
```

```
## [1] TRUE
```

Given the vector *new_genres*, the function returns a vector identical to *new_to_intervals* so the function works correctly.

c. Write a function to simulate a Poisson process and calculate the coefficient of variation of its inter-appearance intervals. It should return a list, one component of which is the vector of inter-appearance intervals of the simulated data, and the other the coefficient of variation of the data.

```
#Function simulates a poisson process and returns coeff of variation and intervals vector
SimulGenres <- function(num.years,mean.genres){

  simPois<- rpois(num.years, mean.genres)
  interval_sim <- new_to_intervals(simPois)
  inter_coeff <- sd(interval_sim)/mean(interval_sim)

  list("Inter-appearance intervals" = interval_sim,
       "Coefficient of variation"= inter_coeff)
}

# Simulates for 161 years and mean = 0.273
SimulGenres(161,0.273)
```

```
## $`Inter-appearance intervals`
##  [1]  3  5  7  7  2  2  0  3 16  2  1  2  1  1  1  5  2  2  2  4  1  3  7
## [24]  3  3  3  4  1  3  2  1  3  0  1  3  3  0  6 12  3  2  4  1  2  3  3
## [47]  1  6  6
```

```
##
## $`Coefficient of variation`
## [1] 0.9069663
```

viii. Run your simulation 10,000 times, taking the coefficient of variation (only) from each. What fraction of simulations runs have a higher coefficient of variation than Moretti's data?

```r
set.seed(1)

# Pre-allocates a vector of length 10000
coeff_var_sim <- rep(NA, 10000)

# Simulates 10000 poisson processes and saves the coeff of variation for each one
for(i in 1:10000){
  coeff_var_sim[i] <- SimulGenres(161,0.273)[[2]]
}

# Computes the proportion of simulated processes with coeff of var > than Moretti's
mean(coeff_var_sim > moretti_coeff_var)
```

```
## [1] 0.2222
```

The fraction of simulations runs that have a higher coefficient of variation than Moretti's data is equal to 0.2222.