Problem 1

a) This is a hypergeometric distribution with following parameters

1. N=36 (Number of cards)
2. k=18 (Number of success states among the cards)
3. We draw n=6 units without replacement

For n=6, we get the expectation

$$E(X) = \frac{nk}{N}$$
$$\Rightarrow \frac{6 \cdot 18}{36} = 3$$

and probability that at least 2 cards are red is

$$f(x) = \frac{\binom{k}{x}\binom{N-k}{n-x}}{\binom{N}{n}}$$
$$\Rightarrow P(X \geqslant 2) = 1 - (P(x=0) + P(x=1))$$
$$\Rightarrow P(X \geqslant 2) = 1 - (\frac{\binom{18}{0}\binom{18}{6}}{\binom{36}{6}} + \frac{\binom{18}{1}\binom{18}{5}}{\binom{36}{6}})$$
$$\Rightarrow P(X \geqslant 2) = 1 - \frac{11}{124} = 0.9113$$

b) Simulation algorithm for Y_n

1. Create a vector of size N, with k "success" and N-k "not success" items
2. Sample n items without replacement from the uniform distribution on [0,1]
3. Count the number of times the "success" item occur in the sample vector
4. Store the counted value in a result vector, where each index represents the number of "success"(red cards) per simulation
5. Run the simulation a large number of times, e.g. 10 000 times. Then use the result vector of size 10 000 to compute expectation and probabilities of x. This could be done by predefined vector functions.

To be (approximately) 95% certain that an error of no more than $e = 0.01$ occur in the estimated probability:

$$n > \frac{1}{e^2}$$

$$\Rightarrow n > \frac{1}{0.01^2} = 10000$$

A more precise formula could also be used

$$n > \frac{Z^2_{\alpha/2} p(1-p)}{e^2}$$

$$\Rightarrow n > \frac{1.96^2 \cdot 0.5(1-0.5)}{0.01^2} = 9604$$

d) Simulation algorithm

1. Create a vector of size N=36, with 18 red values, 18 black values and 1 yellow value
2. Draw n cards every other time, where modulo is used inside a loop to choose which player is to sample from the deck
3. For condition 1., ensure that Player 1 draws a red card the 3 first times
4. After the players have drawn their cards, do a coin toss where p determines the probability of head. Store the correct player as a current winner of the game.
5. Compare the amount of red cards. Change the current winner if needed
6. If a player has a yellow card, set that player as final winner
7. Store the winner for each simulation in a vector. Count the number of times Player 2 won after n simulations, and divide by n to get the probability

Problem 2

a) The mean of a *compound Poisson process*:

$$E(Y(t)) = E(D_1 + ... + D_{N(t)}) = E(N(t))E(D_1) = E(N(t))(E(D) = \lambda t E(D)$$

We have that $\lambda = 0.2$, t=24, and

$$E(D) = 1(0.4) + 2(0.3) + 3(0.2) + 4(0.1) = 2$$

$$\Rightarrow E(Y(t)) = 0.2 \cdot 24 \cdot 2 = 9.6$$

In this case, the number of people injured in an accident represents the i.i.d. sequence of rvs that are independent of the Poisson process N_t. The Poisson process itself is given with a parameter λ=0.2. The process of injuries could be simulated by first generating this Poisson process, which represents the arrival times of accidents within a time period t. For each of the accidents, we must compute the number of people injured based on the rv independent of arrival times. Now we have a process of injuries within time period t, where arrival times is considered on the x-axis, and number of people injured on the y-axis. A similar process could be generated by applying the formula of a compound Poisson process from the description.

b) Simulation algorithm

1. Sample from the given probability distribution to determine how many people got injured for each simulation
2. For each injured person, sample from the exponential distribution with mean 15. This will simulate the ambulance time
3. Sum up the ambulance times for this accident and store it in a result vector
4. Run the simulation many times. The result vector is then the distribution of T.

d) Simulation algorithm

1. Start by simulating the Poisson process with $\lambda = 0.2$ and stoptime = t. The inter arrival times is simulated from an exponential distribution with parameter λ. Be sure to produce enough values to exceed the stoptime. The cumulative sum of the inter arrival times can be considered as arrival times. Filter out the values above stoptime. This simulation is inspired by plotHPP in stochastic_processes_examples.R
2. Loop through each index in the arrival time vector (representing an accident at arrival T_n)
3. Generate the ambulance time for each accident (same way as in 2b)
4. Compare the ambulance time with the time until next accident in order to see if it will be busy for the next accident
5. If it will be busy, stop the simulation and store the result
6. Run the simulation a large number of times. Compute the probability by counting the number of times it was busy in the result vector, then divide by nsim.

Problem 3

a) As seen in figure 1, the degree distributions resembles a binomial distribution where X $\sim$ Bin(n,p).

b) The degree vs age for the preferential attachment model in figure 2 shows notable differences based on the p value. As p=P(head) decrease, the probability that a new vertex choose a random one decrease. The vertex with most edge connections will dominate the probability to be chosen, and will be considered as the hub of the network.

The probability distribution for the preferential attachment model can be seen in figure 3. As mentioned, the decrease of p leads to a few vertices behaving as hubs for the network. The probability that a vertex has very few connections will therefore dominate the distribution. As shown in the figure, for p=0.1, maximum degree is 99, meaning that every new vertex got connected to the hub vertex. The distribution does not resemble a binomial distribution, such as the random graph in a).

Problem 4

a) P(X=k) is stated in question 4a)

1. Simulation algorithm 1 (*Inverse transfer method*)
   Since Zipf's law is a discrete distribution, we look at the *discrete inverse-transform method*[1]. Consider the discrete random variable X with a probability mass function p(k) = P(X = k), k$\geqslant$1. Now use the fact that X=F^{-1}(U) is given by:

$$X = k, if \sum_{i=1}^{k-1} p(i) < U \leq \sum_{i=1}^{k} p(i) k \geqslant 1, \tag{1}$$

   Where U $\sim$ unif(0,1). The simulation algorithm starts of with a generation of U. Then we determine the index k based on equation 1. Return k.

2. Simulation algorithm 2 (*Acceptance-rejection method*)[2]
   We want to generate a rv X with pmf p(k) = P(X=k) in this case as well. Suppose we can generate another discrete rv Y with pmf q(k) = P(Y = k) s.t.

$$sup\{\frac{p(k)}{q(k)}\} \leq c < \infty,$$

   The simulation algorithm for X is then:

   (a) Generate U $\sim$ unif(0,1)

   (b) Generate Y, distributed as q(k) independent from U

   (c) Check if U $\leq \frac{p(Y)}{cq(Y)}$. If thats the case, set X=Y. If not, go back to the first step of this algorithm.

Plots

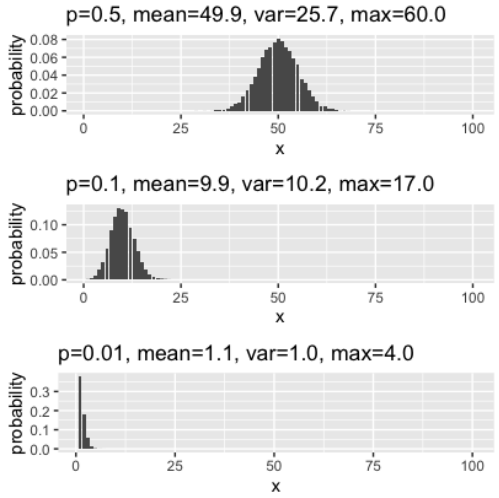

Figure 1: Bar plot of estimated degree distributions for 3a

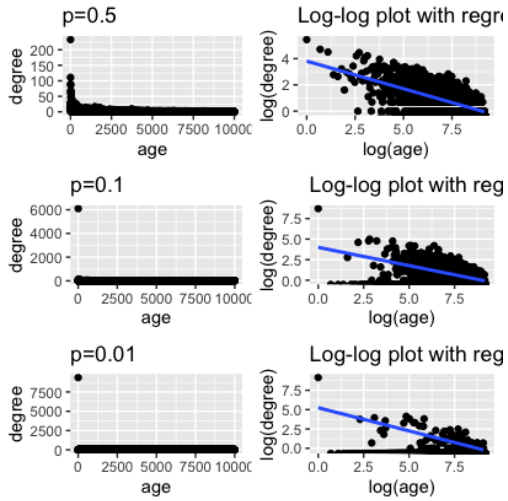

Figure 2: Degree vs age for the preferential attachment model

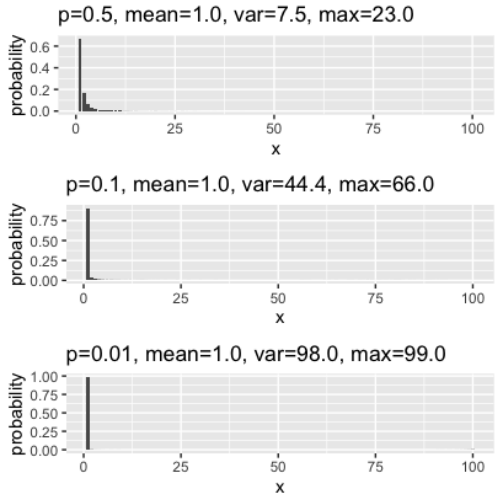

Figure 3: Degree distribution for the preferential attachment model

Bibliography

These notes where used in order to describe the simulation algorithms for 4a
1 http://www.columbia.edu/ ks20/4404-Sigman/4404-Notes-ITM.pdf
2 http://www.columbia.edu/ ks20/4703-Sigman/4703-07-Notes-ARM.pdf