Simulazioni stocastiche

```
## Title: Stochastic simulations
## Author: Luca La Rocca
## Date: 1 October 2024

## Sys.setLanguage("en", unset="it") # uncomment to disable message
translation
rm(list=ls(all=TRUE)) # clean the workspace

ls(all=TRUE) # empty workspace

sample(1:6,1) # 1d6 or single extraction from an urn with six numbered balls
sample(1:6,3) # three extractions without replacement from that urn
sample(1:6,6) # random permutation of the balls in that urn
try(sample(1:6,7)) # impossible
sample(1:6,7, replace=TRUE) # possible
sample(1:6,3, replace=TRUE) # 3d6 or three extractions with replacement

ls(all=TRUE) # hidden RNG state

d6size <- 1000 # sample size for simulation
d6samp <- sample(1:6, d6size, replace=TRUE)
d6six <- d6samp==6
plot(1:d6size, cumsum(d6six)/1:d6size, type="l", ylim=c(0,1),
     xlab = expression(n), ylab=expression(p[n]), main="Bernoulli's
theorem")
abline(h=1/6, lty="dashed")

bloodtypes <- c("A", "AB", "B", "O") # in alphabetical order
bloodprobs <- c(0.40, 0.04, 0.11, 0.45) # need not sum to one
bloodsize <- 1000 # sample size for simulation
bloodsamp <- sample(bloodtypes, bloodsize, prob=bloodprobs,
                    replace=TRUE) # order of named arguments doesn't matter
bloodprop <- proportions(table(bloodsamp))
cat("When comparing a sample proportion to a population probability",
fill=TRUE)
cat("based on a sample of size n =", bloodsize, fill=TRUE)
cat("we are confident that the difference will not exceed 1/sqrt(n) =",
    1/sqrt(bloodsize), fill=TRUE)
cat("and in the present case (blood types in the United States)", fill=TRUE)
print(all.equal(bloodprobs, as.vector(bloodprop), scale=1))

birthdayStudents <- 22 # number of students in the classroom (plus one
teacher)
cat("The probability of at least two coinciding birthdays", fill=TRUE)
cat("among", birthdayStudents,"students plus one teacher is", fill=TRUE)
birthdayCoincidence <- replicate(10000,{
  birthdays <- sample(1:365, birthdayStudents+1, replace=TRUE)
```

```r
  anyDuplicated(birthdays)>0
}) # end of replicate
cat("approximately", mean(birthdayCoincidence),
    "and exactly", 1-prod(365:(365-
birthdayStudents))/365^(birthdayStudents+1),
    fill=TRUE)
cat("The probability of at least a birthday coinciding with", fill=TRUE)
cat("the teacher's birthday among", birthdayStudents,"students is",
fill=TRUE)
birthdayTeacher <- replicate(10000,{
  teacherday <- sample(1:365, 1)
  birthdays <- sample(1:365, birthdayStudents, replace=TRUE)
  any(birthdays==teacherday)
}) # end of replicate
cat("approximately", mean(birthdayTeacher),
    "and exactly", 1-(364/365)^birthdayStudents,
    fill=TRUE)

LottoNumeri <- c(9,81) # to fix ideas
cat("The probability of \"ambo secco\" is", fill=TRUE)
LottoAmbo <- replicate(100000,{
  estratti <- sample(1:90, 5)
  all(LottoNumeri %in% estratti)
}) # end of replicate
LottoEmpirP <- mean(LottoAmbo)
LottoTheorP <- prod(c(5,4,88,87,86))/prod(90:86)
cat("approximately", LottoEmpirP, "and exactly p =", LottoTheorP, fill=TRUE)
cat("so that a fair payment for a 1 euro bet would be 1/p =",
    1/LottoTheorP, "euro", fill=TRUE)
cat("whereas the actual payment is 250 euro (minus taxes)", fill=TRUE)

seme <- function(carta) # an integer between 1 and 40
{c("Coppe", "Denari", "Bastoni", "Spade")[(carta-1)%/%10+1]}
valore <- function(carta) # an integer between 1 and 40
{(carta-1)%%10+1}
cat("The probability of two Italian cards having the same suit or value is",
    fill=TRUE)
carteEvento <- replicate(40000,{
  mano <- sample(1:40, 2)
  anyDuplicated(seme(mano))>0 | anyDuplicated(valore(mano))>0
}) # end of replicate
cat("approximately", mean(carteEvento),
    "and exactly", (40*9)/(40*39)+(40*3)/(40*39),
    fill=TRUE)

cat("The probability of forming a triangle when breaking a stick",
fill=TRUE)
cat("in two (uniformly chosen) random points is ")
stickTriangle <- replicate(10000,{
  stickbreaks <- sort(runif(2))
  sticklengths <- sort(diff(c(0, stickbreaks, 1)))
  sum(sticklengths[1:2]) > sticklengths[3]
}) # end of replicate
```

```r
cat("approximately", mean(stickTriangle), fill=TRUE)

RisikoAsize <- 5
RisikoDsize <- 3
cat("The probability of winning a no-tomorrow Risiko attack", fill=TRUE)
cat("using", RisikoAsize, "armies against", RisikoDsize, "defending armies
is ")
RisikoAwins <- replicate(10000,{
  Attackers <- RisikoAsize
  Defenders <- RisikoDsize
  while(Attackers > 0 & Defenders > 0)
  {
    a <- min(Attackers, 3)
    d <- min(Defenders, 3)
    attack <- sort(sample(1:6, a, rep = TRUE), decreasing = TRUE)
    defend <- sort(sample(1:6, d, rep = TRUE), decreasing = TRUE)
    m <- min(a, d)
    losses <- sum(attack[1:m]<=defend[1:m])
    Attackers <- Attackers - losses
    Defenders <- Defenders - (m - losses)
  }
  Defenders == 0
}) # end of replicate
cat("approximately", mean(RisikoAwins), fill=TRUE)
```

Conteggi

```r
## Title: Counting
## Author: Luca La Rocca
## Date: 15 October 2024

## Sys.setLanguage("en", unset="it") # uncomment to disable message
translation
rm(list=ls(all=TRUE)) # clean the workspace

prodCart <- function(spaces) # list of vectors
{ # begin function
  S <- expand.grid(rev(spaces))
  if(ncol(S)>1) S <- S[, ncol(S):1]
  colnames(S) <- paste("Pos", 1:ncol(S), sep="")
  return(S)
} # end function

dispRep <- function(S0, # vector of items available for selection
                    k) # number of items to be selected
  return(prodCart(rep(list(S0), k)))

dispSemp <- function(S0, # vector of items available for selection
                     k) # number of items to be selected
{ # begin function
  if(k==1){ # base case
```

```r
    S <- as.data.frame(S0)
    colnames(S) <- "Pos1"
  }else{ # recursive case
    S <- Recall(S0[-1], k-1)
    recsize <- nrow(S)
    S <- cbind(rep(S0[1], recsize), S)
    colnames(S) <- paste("Pos", 1:ncol(S), sep="")
    for(i in 2:length(S0)){
      Schunk <- cbind(rep(S0[i], recsize), Recall(S0[-i], k-1))
      colnames(Schunk) <- colnames(S)
      S <- rbind(S, Schunk)
    } # end for
  } # end if-else
  return(S)
} # end function

perm <- function(S0) # vector of items to be permuted
  return(dispSemp(S0,length(S0)))

combSemp <- function(S0, # vector of items available for selection
                      k) # number of items to be selected
{ # begin function
  S <- as.data.frame(t(combn(S0, k)))
  colnames(S) <- paste("Item", 1:ncol(S), sep="")
  return(S)
} # end function

## SPAZI CAMPIONARI ELEMENTARI
cat("Sample space for a coin (head?):", fill=TRUE)
Scoin <- c(FALSE, TRUE)
print(Scoin)
cat("total outcomes =", length(Scoin), fill=TRUE)

cat("Sample space for a French card seed:", fill=TRUE)
Scard <- c("heart", "diamond", "club", "spade")
print(Scard)
cat("total outcomes =", length(Scard), fill=TRUE)

cat("Sample space for a die:", fill=TRUE)
Sdie <- 1:6
print(Sdie)
cat("total outcomes =", length(Sdie), fill=TRUE)

## PRODOTTO CARTESIANO DI SPAZI FINITI
cat("Sample space for a coin (head?) and a French card seed:", fill=TRUE)
Scoincard <- prodCart(list(Scoin, Scard))
print(Scoincard)
cat("total outcomes =", length(Scoin)*length(Scard), fill=TRUE)

cat("Sample space for a coin (head?) a French card seed and a die:",
fill=TRUE)
Scoincardie <- prodCart(list(Scoin, Scard, Sdie))
print(Scoincardie)
```

```r
cat("total outcomes =", length(Scoin)*length(Scard)*length(Sdie), fill=TRUE)

## SPAZI DI DISPOSIZIONI CON RIPETIZIONE
cat("Sample space for two dice:", fill=TRUE)
Stwodice <- dispRep(Sdie, 2)
print(Stwodice)
cat("total outcomes =", length(Sdie)^2, fill=TRUE)

cat("Sample space for three coins (head?):", fill=TRUE)
Sthreecoins <- dispRep(Scoin, 3)
print(Sthreecoins)
cat("total outcomes =", length(Scoin)^3, fill=TRUE)

## SPAZI DI DISPOSIZIONI SEMPLICI
cat("Sample space for an ordered hand of two French queens:", fill=TRUE)
Stwocards <- dispSemp(Scard, 2)
print(Stwocards)
cat("total outcomes =", prod(length(Scard):(length(Scard)-2+1)), fill=TRUE)

cat("Sample space for an ordered hand of three French queens:", fill=TRUE)
Sthreecards <- dispSemp(Scard, 3)
print(Sthreecards)
cat("total outcomes =", prod(length(Scard):(length(Scard)-3+1)), fill=TRUE)

## SPAZI DI PERMUTAZIONI
nlet <- 4 # number of letters
AnagramSimSize <- 2500 # number of simulations
cat("Sample space for a random anagram of", nlet, "distinct letters:",
    fill=TRUE)
AnagramS <- perm(LETTERS[1:nlet])
print(AnagramS)
cat("total outcomes =", factorial(length(AnagramS)), fill=TRUE)
AnagramD <- apply(AnagramS, 1, function(row) !any(row==LETTERS[1:nlet]))
cat("derangement event", fill=TRUE)
print(AnagramS[AnagramD,])
cat("favorable outcomes =", sum(AnagramD), fill=TRUE)
AnagramP <- sum(AnagramD)/nrow(AnagramS)
cat("exact probability of derangement =", AnagramP, fill=TRUE)
cat("how far is it from the limit value?",fill=TRUE)
print(all.equal(exp(-1), AnagramP))
AnagramDtrials <- replicate(AnagramSimSize,{
  permutation <- AnagramS[sample(1:nrow(AnagramS), 1),]
  !any(permutation==LETTERS[1:nlet])
})
cat("empirical proportion of derangement =", mean(AnagramDtrials),
    "with margin of error =", 1/sqrt(AnagramSimSize), fill=TRUE)

## SPAZI DI COMBINAZIONI SEMPLICI
cat("Sample space for an unordered hand of two French queens:", fill=TRUE)
SunordHand <- combSemp(Scard, 2)
print(SunordHand)
cat("total outcomes =", choose(length(Scard), 2), fill=TRUE)
cat("Sample space for an unordered hand of three French queens:", fill=TRUE)
SunordHand <- combSemp(Scard, 3)
print(SunordHand)
cat("total outcomes =", choose(length(Scard), 3), fill=TRUE)
```