Ising Model

Bartosz Banasik (229760)

April 30, 2019

1. Below are located 3 pairs of graphs. Each of them shows the configuration of the spins for a given size of the system at a fixed temperature. For the left panel, the size of the lattice equal `20x20` and for the right equal `100x100`.For figure 1 and 2 $T < T_c (T = 1)$, for figure 3 and 4 $T \simeq T_c (T = 2.26)$, for figure 5 and 6 $T > T_c (T = 5)$.

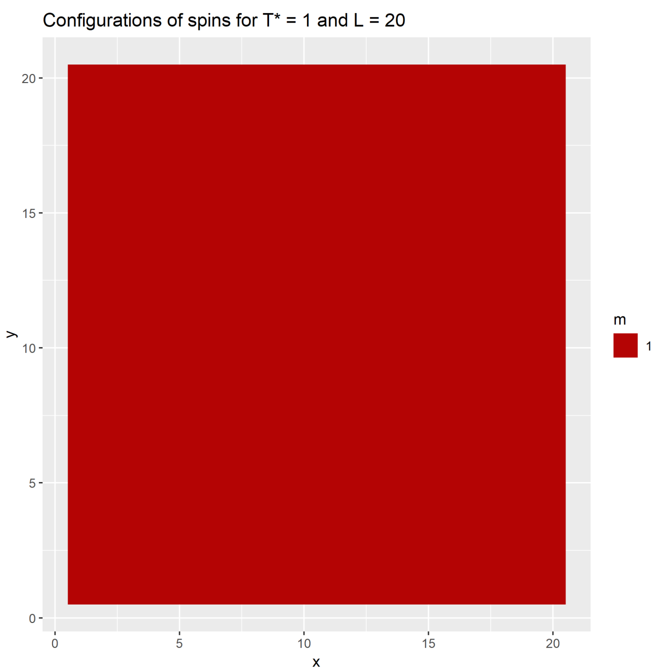

Figure 1

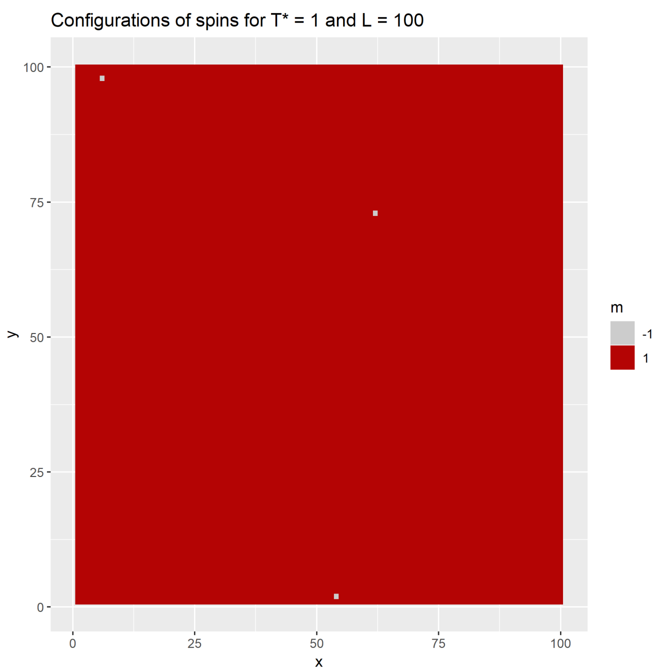

Figure 2

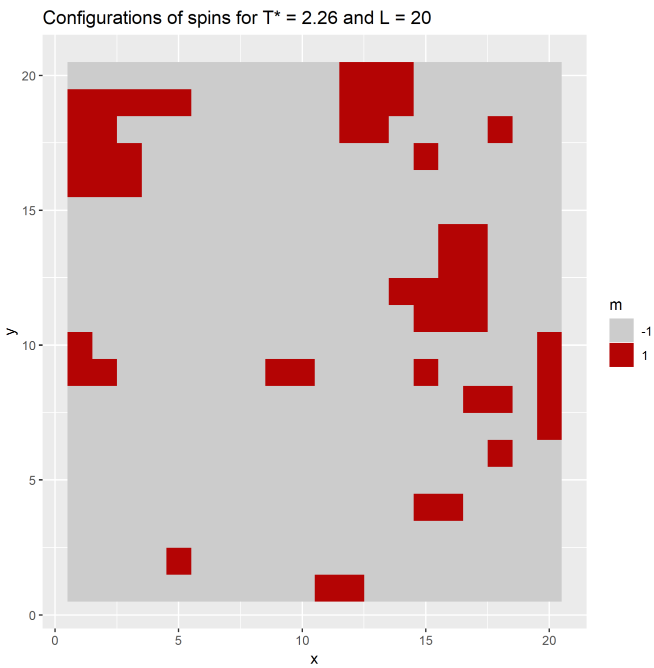

Figure 3

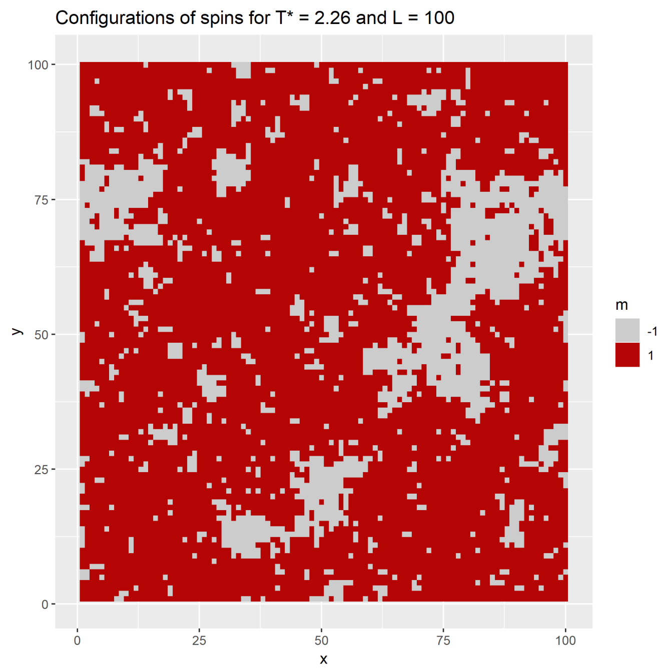

Figure 4

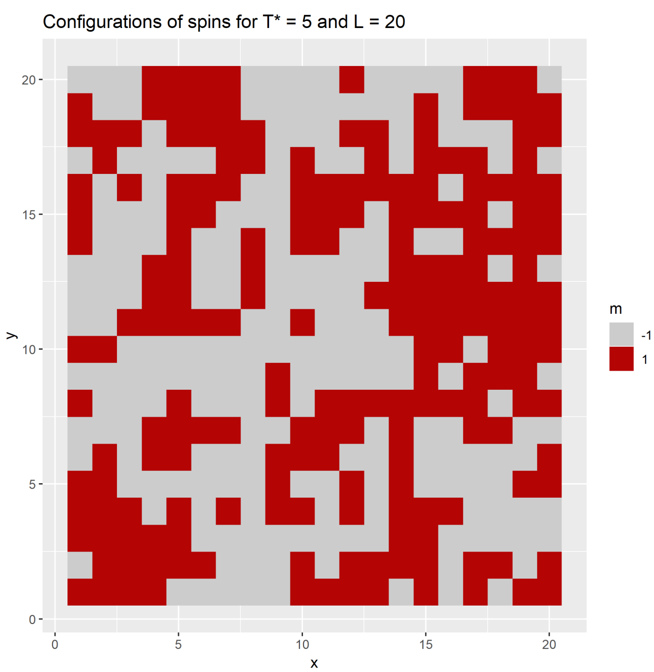

Figure 5

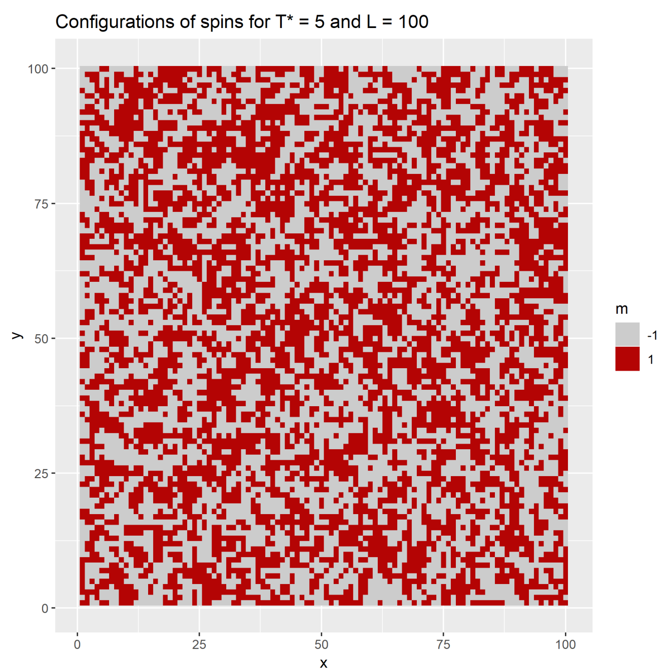

Figure 6

2. Observation of the flips between states $m = +1$ and $m = -1$ at a low temperature $(T < Tc) T = 1.7$. For the lattice of size `10x10`

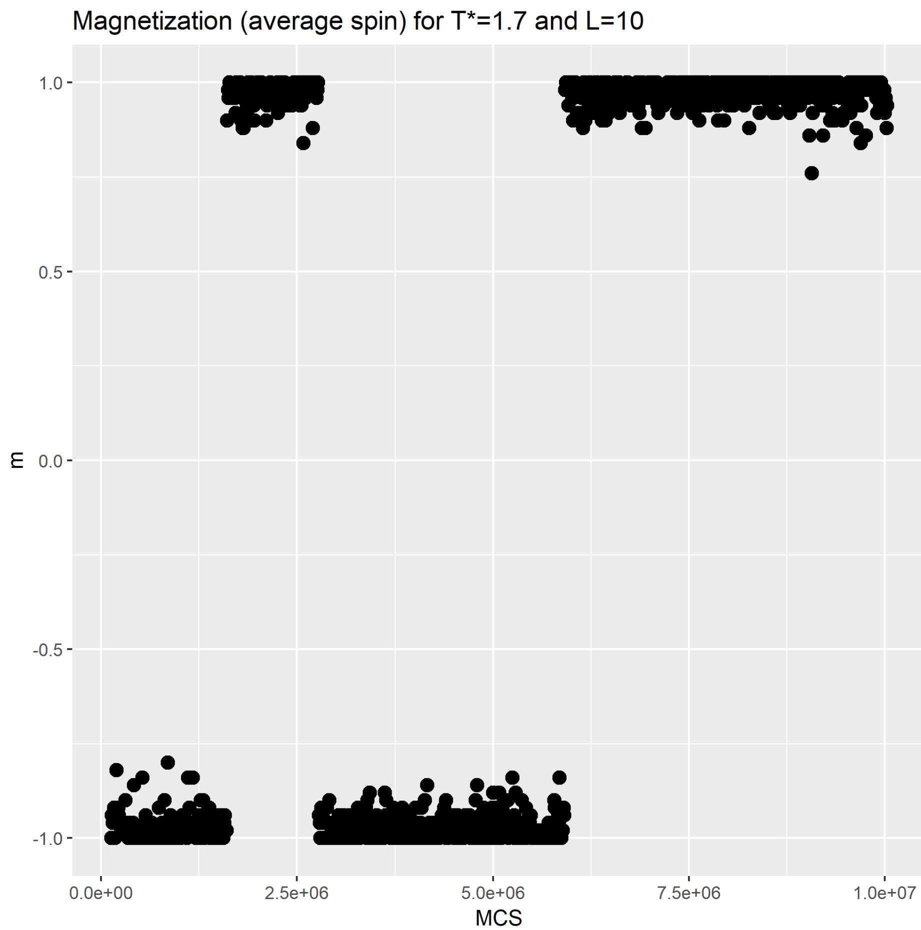

Figure 7

3. On the graphs below there are located respectively the temperature dependence of the mean value of magnetization 8, energy 9, susceptibility 10 and specific heat 11 for different size of the system (L=10,50,100). Figures for $K = 530000MCS$, $k_0 = 30000MCS$ and $k = \frac{K-k_0}{100} = 5000$ - equilibrium configurations that have been analyzed.

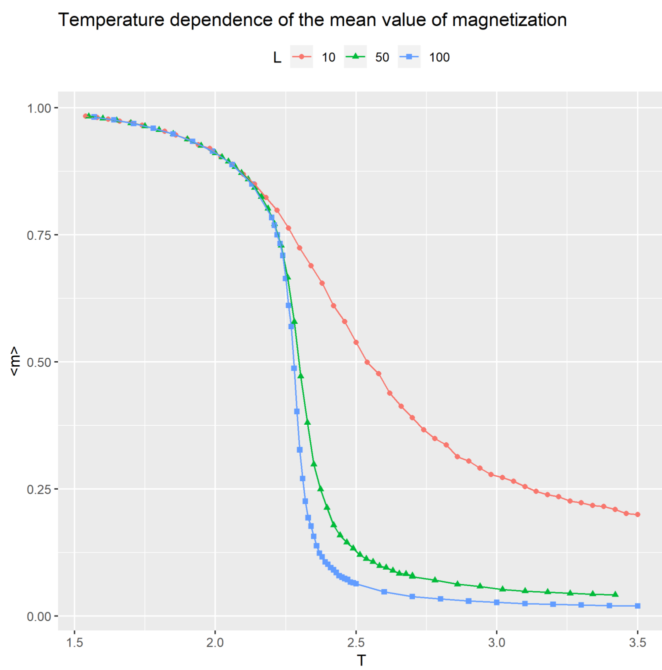

Figure 8

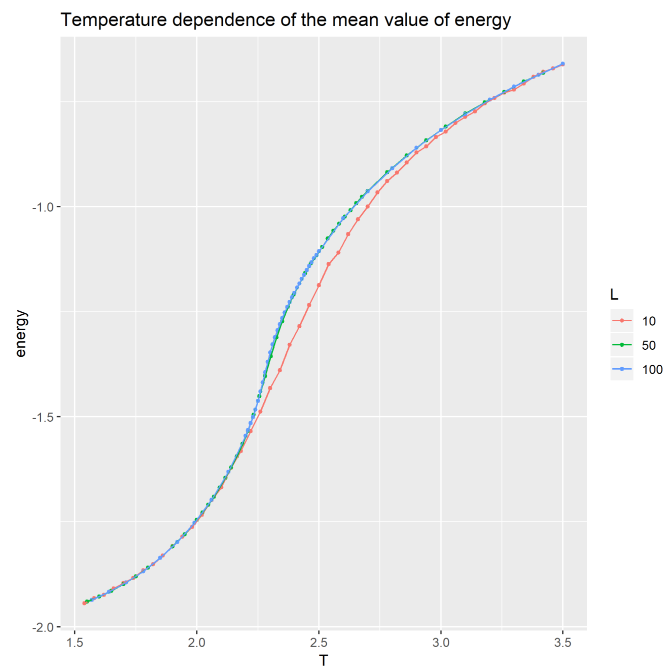

Figure 9

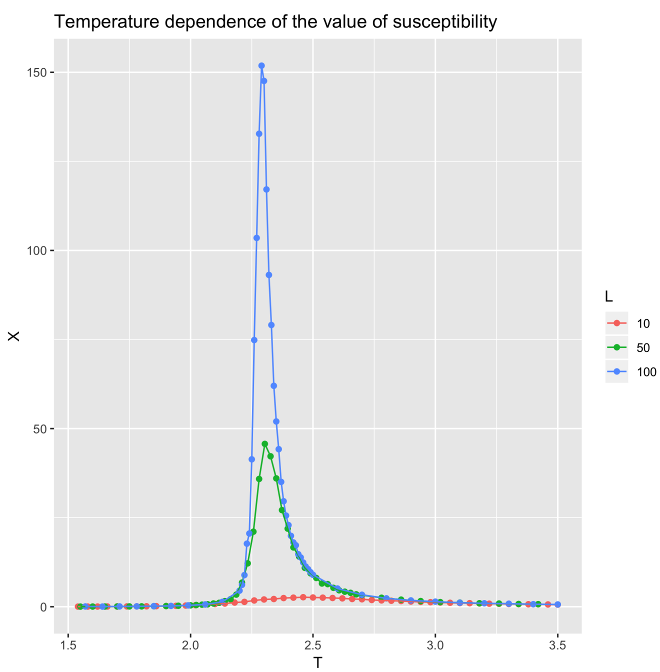

Figure 10

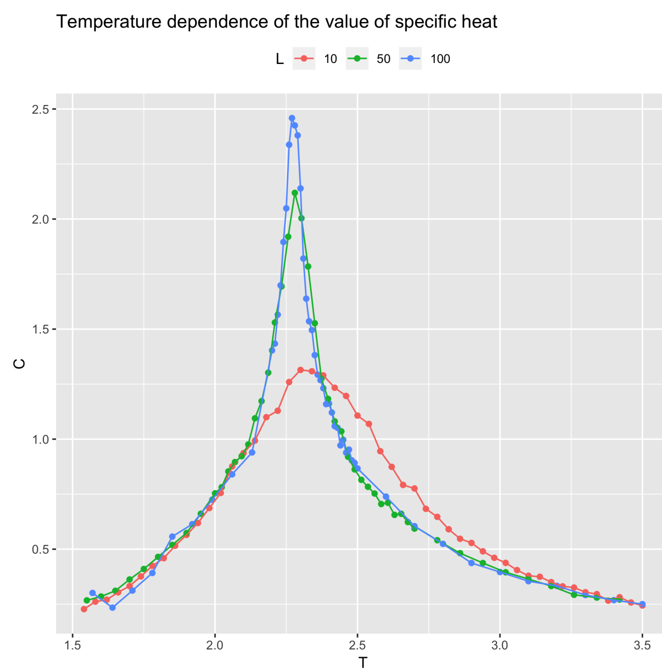

Figure 11

4. The graph below 12 presents the temperature dependence of the mean value of Binders cumulant for a few values of the linear size of the system (L=10,50,100). Based on the temperature dependence of Binders cumulant for different sizes of the system. From graphs, we see that cross point is $T_c \approx 2.269$.

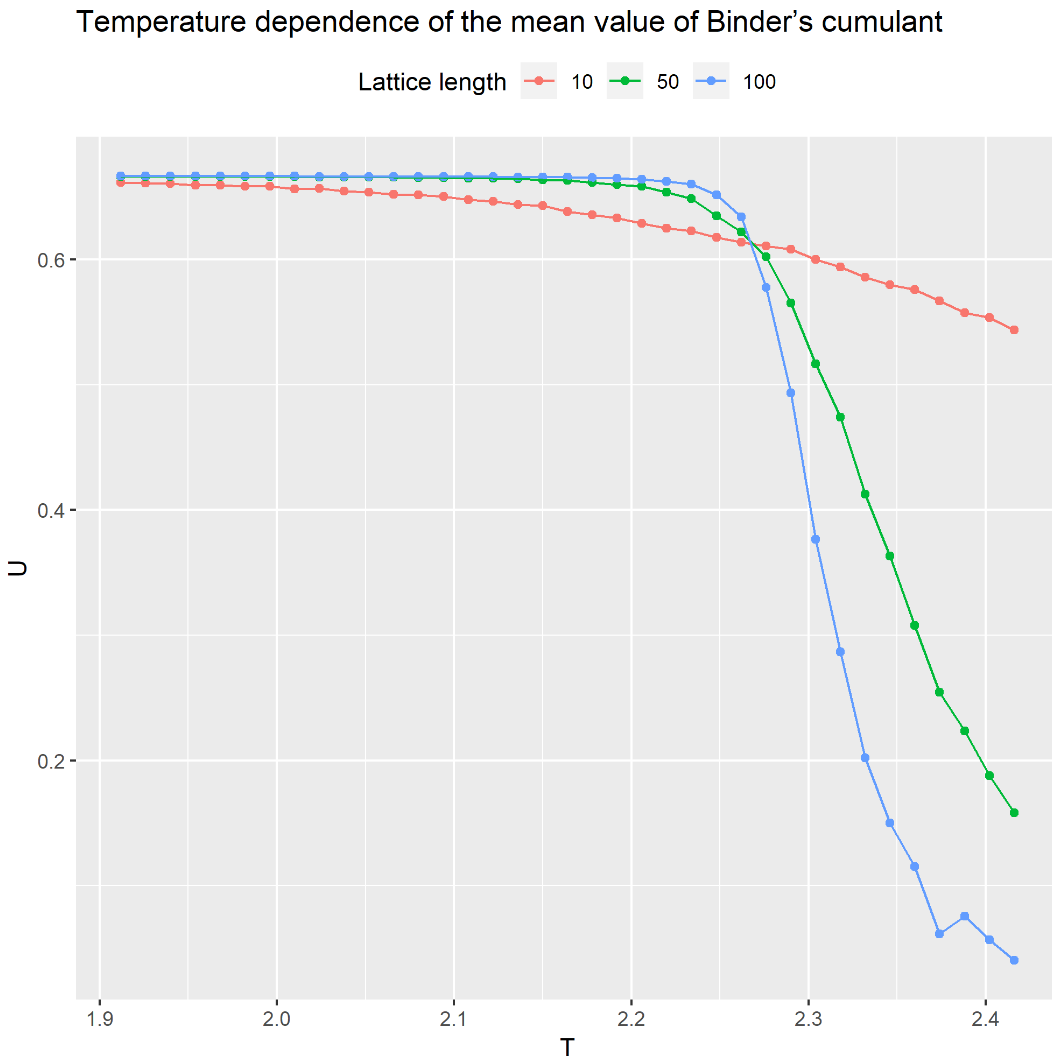

Figure 12

5. Finite-size scaling close to the critical point using theoretical values of critical exponents $\beta = \frac{1}{8}$, $\nu = 1$ for the 2D Ising model.

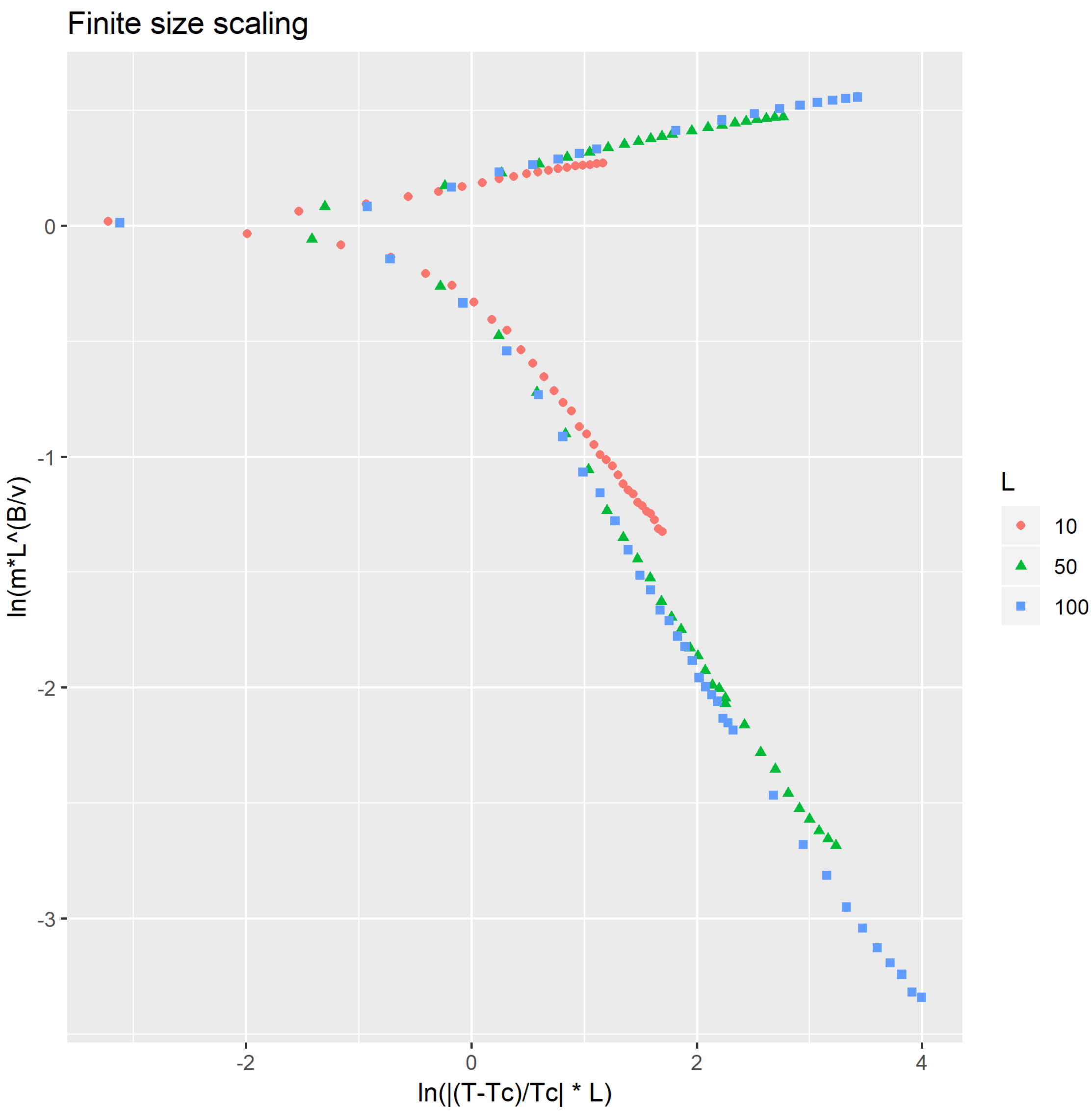

Figure 13

```fortran
PROGRAM ising
  IMPLICIT NONE
  integer, parameter :: L = 10
  integer, dimension(L,L) :: S
  ! PBC
  integer, dimension(L) :: NI, PI
  ! number of MCS
  integer :: k = 530000
  real :: T = 0
  ! array to store prob acceptance for
  ! given T
  real, dimension(-8:8) :: Tex
  ! T sampling
  real, dimension(50) :: Ta
  real :: ma, ea, avgm, avgm2
  real :: sus, avge, avge2, heatc
  real :: bind_m4, bind_m2
  real :: bind

  integer :: i, j, counter

  DO I=1,L
      NI(I)=I+1
      PI(I)=I-1
  ENDDO
  NI(L)=1
  PI(1)=L

  call init_Ta_reg(Ta)

  call initS(S, L)

  ! open(unit=2, file="sus100.csv")
  ! write(2, *) "T,m,L"
  ! open(unit=3, file="heat100.csv")
  ! write(3, *) "T,e,L"
  open(unit=4, file="finite10.csv")
  write(4, *) "T,m,L"

  do j=1,50
  print *, 2*j, "%"
  T = ta(j)
  ! initialize exp(-de/T) array
  call initTex(tex,T)
  ma = 0
  ea = 0
  avge = 0
  avge2 = 0
  avgm = 0
  avgm2 = 0
  counter = 0
  bind_m4 = 0
  bind_m2 = 0
  do i=1,k
      call mcs(t)
      if ((i >= 30000) .and. (mod(i,100) == 0)) then
          ! write(2, *) i, ",", calc_m()
          avgm = avgm + abs(calc_m())
          avgm2 = avgm2 + (calc_m()**2)
          ma = ma + abs(calc_m())
          ea = ea + calc_ener()
          avge = avge + calc_ener2()
          avge2 = avge2 + (calc_ener2()**2)
          bind_m4 = bind_m4 + (calc_m2()**4)
          bind_m2 = bind_m2 + (calc_m2()**2)
          counter = counter + 1
      end if
  enddo
  ma = ma/float(counter)
  ea = ea/float(counter)
  avge = avge/float(counter)
  avge2 = avge2/float(counter)
  avgm = avgm/float(counter)
  avgm2 = avgm2/float(counter)
  bind_m4 = bind_m4/float(counter)
  bind_m2 = bind_m2/float(counter)
  sus = ((L*L)/T)*(avgm2 - (avgm**2))
  heatc = (1/(float(L**2)*(T**2)))*(avge2 - (avge**2))
  bind = 1 - (bind_m4/(3* (bind_m2**2)))
  ! write(3, *) T, ",", ea, ",", L
  ! write(2, *) T, ",", sus, ",", L
  ! write(3, *) T, ",", heatc, ",", L
  ! write(4, *) T, ",", bind, ",", L
  write(4, *) T, ",", ma, ",", L
  enddo
  ! close(2)
  ! close(3)
  close(4)

  contains
      function calc_m() result(m)
          real :: m
          integer :: i,j
          m = 0
          do i=1,L
              do j=1,L
                  m = m + S(i,j)
              enddo
          enddo
          m = m/(L*L)
      end function calc_m

      function calc_m2() result(m)
          real :: m
          integer :: i,j
          m = 0
          do i=1,L
              do j=1,L
                  m = m + S(i,j)
              enddo
          enddo
      end function calc_m2

      function calc_ener() result(en)
          real :: en
          integer :: i, j, su
          en = 0
          do i=1,L
              do j=1,L
                  su = S(NI(i), j) + &
                      S(PI(i), j) + &
                      S(i, NI(j))+ &
                      S(i, PI(j))
                  en = en + ((-su) * S(i,j))
              enddo
          enddo
          en = en/((L*L)*2.0)
      end function calc_ener

      function calc_ener2() result(en)
          real :: en
          integer :: i, j, su
          en = 0
          do i=1,L
              do j=1,L
                  su = S(NI(i), j) + &
                      S(PI(i), j) + &
                      S(i, NI(j))+ &
                      S(i, PI(j))
                  en = en + ((-su) * S(i,j))
              enddo
          enddo
          en = en/(2.0)
      end function calc_ener2

      function calc_du(x, y) result(du)
          integer :: du
          integer, intent(in) :: x,y
          integer :: su ! suma

          su = S(NI(x), y) + &
              S(PI(x), y) + &
              S(x, NI(y))+ &
              S(x, PI(y))
          du = 2 * S(x,y) * su
      end function calc_du
```

```
      subroutine trial(i, j, t)
          integer, intent(in) :: i, j
          real, intent(in) :: t
          integer :: du
          real :: paccept, ptrial

          du = calc_du(i, j)
          if (du < 0) then
              S(i,j) = -S(i,j)
          else
              call random_number(paccept)
              ptrial = tex(du)
              if (paccept <= ptrial) then
                  s(i,j) = -s(i,j)
              end if
          endif
      end subroutine trial

      subroutine mcs(t)
          real, intent(in) :: t
          integer i, j

          do i=1,L
              do j=1,L
                  call trial(i,j,t)
              enddo
          enddo
      end subroutine mcs

      subroutine print_S_out()
          integer :: i, j
          do i=1,L
              print *, S(i,:)
          enddo
      end subroutine print_S_out

      subroutine print_S()
          integer :: i, j
          open(unit=2, file="lattice5-100.csv")
          write(2, *) "x,y,m"
          do i=1,L
              do j=1,L
                  write(2, *) i,",", j, ",",S(i,j)
              enddo
          enddo
          close(2)
      end subroutine print_S

END PROGRAM

subroutine initTex(tex,tmp)
    real, dimension(-8:8), intent(inout) :: tex
    real, intent(inout) :: tmp
    integer :: i
    do i=-8,8,4
      tex(i) = exp((-i)/tmp)
    enddo
end subroutine initTex

subroutine initS(s, l)
    integer, intent(in) :: L
    integer, dimension(l,l) :: S
    integer :: i,j
    do i=1,L
      do j=1,L
          S(i,j) = 1
      enddo
    enddo
end subroutine initS

subroutine init_Ta(ta)
    real, intent(inout), dimension(50) :: ta
    integer :: i, j = 0
    do i=1,10
      ta(i) = 1.5 + i * ((2.0-1.5)/10)
    enddo

    j=1
    do i=11,40
      ta(i) = 2.0 + j * ((2.7-2.0)/30)
      j = j+1
    enddo

    j = 0
    do i=41,50
      ta(i) = 2.7 + j * ((3.5-2.7)/10)
      j = j+1
    enddo
end subroutine init_Ta
```

```
subroutine init_Ta_100(ta)
    real, intent(inout), dimension(50) :: ta
    integer :: i, j = 0
    do i=1,10
      ta(i) = 1.5 + i * ((2.2-1.5)/10)
    enddo

    j=1
    do i=11,40
      ta(i) = 2.2 + j * ((2.5-2.2)/30)
      j = j+1
    enddo

    j = 1
    do i=41,50
      ta(i) = 2.5 + j * ((3.5-2.5)/10)
      j = j+1
    enddo
end subroutine init_Ta_100

subroutine init_Ta_reg(ta)
  real, intent(inout), dimension(50) :: ta
  integer :: i
  do i=1,50
      ta(i) = 1.5 + i * ((3.5-1.5)/50)
  enddo
end subroutine init_Ta_reg

subroutine init_Ta_bind(ta)
  real, intent(inout), dimension(50) :: ta
  integer :: i
  do i=1,50
      ta(i) = 1.8 + i * ((2.5-1.8)/50)
  enddo
end subroutine init_Ta_bind

subroutine init_Ta_flip(ta)
  real, intent(inout), dimension(50) :: ta
  integer :: i
  do i=1,50
      ta(i) = 1.8 + i * ((2.5-1.8)/50)
  enddo
end subroutine init_Ta_flip
```