Marco Carega - 20018128

Relazione Computational Statistics

Introduzione

L'obiettivo è lo svolgimento degli esercizi assegnati attraverso l'ausilio del linguaggio di programmazione R, un linguaggio adatto per effettuare computazioni di tipo statistico.
I dati utilizzati per lo svolgimento degli esercizi indicano le misurazioni riguardanti il mese di settembre 2018.
Ogni file contiene le seguenti colonne nel seguente ordine:

- Data di misurazione;
- ora di misurazione;
- Radiazione incidente;
- Radiazione riflessa;
- Umidità relativa;
- Temperatura dell'aria;
- Velocità del vento;
- Precipitazioni;
- Pressione;
- Temperatura del radiometro superiore;
- Temperatura del radiometro inferiore;
- Radiazione OL incidente;
- Radiazione OL riflessa.

Nel progetto sono state implementate diverse funzione che permettono lo lettura di questi dati in base al giorno o al mese e alla colonna che si intende prendere in considerazione:

- getDataOfDay(i,type) raccoglie tutti i dati relativi al giorno i e del tipo considerato;
- getDatamonth(type) raccoglie tutti i dati di un mese del tipo considerato.

```
getDataOfDay=function(i,type)
{
  return(readFile(i)[,type])
}

getDataMonth=function(type)
{
  data=c();
  for(i in 1:30)
  {
    data=c(data,getDataOfDay(i,type))
  }
  return(data) ^getDataMonth
}
```

```
readFile=function(day)
{
  fileName="";
  if(day<=9) fileName=paste("settembre/2018-09-0",day,".dat",sep="")
  else fileName=paste("settembre/2018-09-",day,".dat",sep="");
  return(read.table(fileName)) ^readFile
}
```

Durante la spiegazione dei vari esercizi e del loro svolgimento, ogni formula sarà seguita da una funzione indicante la sua implementazione in R.

Esercizio 1

Il primo esercizio consiste inizialmente nel calcolo dei momenti e dei momenti centrati e in seguito ricavare i primi dei secondi e viceversa.
Il momento di ordine n è una media dei dati elevati per n:

$$m_n = \frac{1}{N}\sum_{i=1}^{N} \hat{x}_i^{\,n}$$

```
moment=function(x,n)
{
  return(sum(x^n)/length(x))
}
```

La formula per il calcolo del momento centrato di ordine n è molto simile alla precedente, la differenza rispetto a prima è che si sottrae la media prima di elevare per n:

$$\mu_n = \frac{1}{N}\sum_{i=1}^{N}(\hat{x}_i - \eta)^n \; dove \; \eta = m_1$$

```
centralMoment=function(x,n)
{
  avg=moment(x, n: 1);
  return(sum((x-avg)^n)/length(x))
}
```

I due momenti sono correlati tra loro, esistono infatti due formule che permettono di calcolare il momento a partire da quello centrato e viceversa.
Per ottenere il momento di ordine n, per ogni k minore di n si calcola il coefficiente binomiale $\binom{n}{k}$ tramite la funzione choose(n,k), dopo si deve moltiplicare il valore ottenuto col momento centrato di ordine n e con la media η elevata per n-k.
In base alla spiegazione precedente, il momento di ordine n si ottiene con la seguente formula:

$$m_n = \sum_{k=1}^{n} \binom{n}{k} * \mu_k * (\eta)^{n-k}$$

```
centralToNormal=function(x,n)
{
  avg=moment(x, n: 1);
  acc=0;
  for(k in 0:n)
  {
    coeff=choose(n,k);
    mk=centralMoment(x,k)
    expAvg=avg^(n-k)
    acc=acc+(coeff*mk*expAvg)
  }
  return(acc); ^centralToNormal
}
```

Il ragionamento per il calcolo del momento centrato da quello normale segue la stessa logica ma con qualche differenza: la media η è infatti negativa quando elevata per n-k.
La formula sottostante permette il calcolo del momento centrato di ordine n da quello normale e, confrontandola con quella precedente, sono molto evidenti le similitudini:

$$\mu_n = \sum_{k=1}^{N} \binom{n}{k} * m_k * (-\eta)^{n-k}$$

```
normalToCentral=function(x,n)
{
  avg=moment(x, n: 1);
  acc=0;
  for(k in 0:n)
    {
    coeff=choose(n,k);
    mk=moment(x,k)
    expAvg=(-avg)^(n-k)
    acc=acc+(coeff*mk*expAvg)
  }
  return(acc); ^normalToCentral
}
```

Per quanto riguarda la fase di testing, sono stati calcolati i momenti (centrati e non) di ordine 0, 1 e 2.
Tutti i risultati coincidono a eccezione di u1 che, quando calcolato con la formula, risente dell'errore di approssimazione. Questo errore è comunque trascurabile e quindi si può approssimare il risultato a 0.
E' stata inoltre testata la correlazione tra i vari momenti calcolando momento e momento centrato di ordine 3, come visibile nell'immagine a destra:

```
[1] "Moments:"
[1] 1
[1] 18.33798
[1] 337.9974
[1] "Central Moments:"
[1] 1
[1] 3.391789e-16
[1] 1.715916
[1] "Central to normal:"
[1] 1
[1] 18.33798
[1] 337.9974
[1] "Normal to central:"
[1] 1
[1] 0
[1] 1.715916
[1] "Stats:"
[1] "Mean:  18.3379790940767"
[1] "Var:  1.71711306790294"
[1] "m3 =u3 + 3*m1*u2 + m1^3 ==> 6262.81692682927 6262.81692682927"
[1] "u3 = m3 - 3*m1*m2 + 2*m1^3 ==> 1.69490595481716 1.69490595481693"
```

Come si può ben vedere dall'immagine sopra, il risultato del momento di ordine 1 equivale alla media mentre il momento centrato di ordine 2 equivale alla varianza.

Esercizio 2

Il secondo esercizio richiede il calcolo del giorno tipo: la media di un mese per ogni minuto del giorno.
Per questo esercizio, si è seguito il seguente ragionamento:

- è stato estratto il dato relativo all'i-esimo minuto di ogni giorno;
- tutti i valori estratti sono stati sommati;
- si divide la somma ottenuta per il numero di minuti estratti.

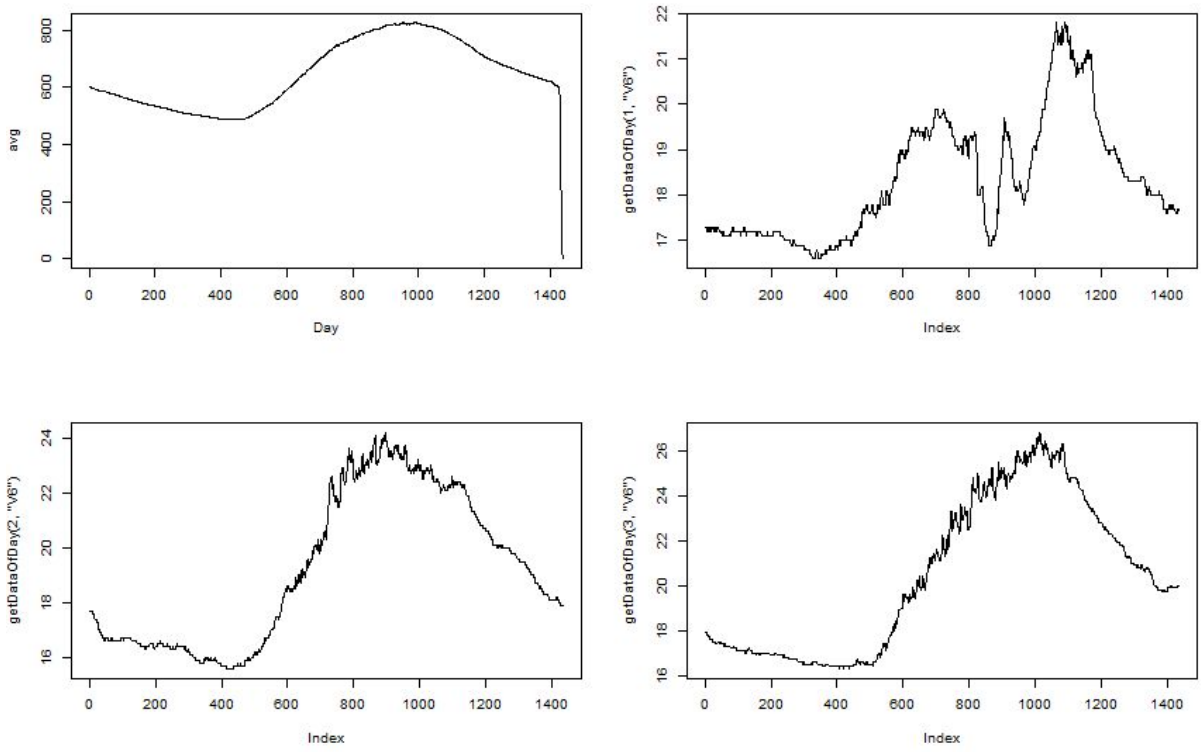
Da questo ragionamento è stata implementata la seguente funzione:

```
typicalDay=function(type)
{
  means=array(data=0,dim=24*60)
 counts=array(data=0,dim=24*60)
  for(i in 1:30)
  {
    x=getDataOfDay(i,type)
    for(j in 1:length(x))
    {

      if(!is.na(x[j]))
      {
        means[j]=means[j]+x[j]
        counts[i]=counts[j]+1
      }
    }
  }
  for(i in 1:30)
  {
    if(counts[i]!=0) means[i]=means[i]/counts[i]
  }
  return(means) ^typicalDay
}
```

In cui means è un array contenente le medie mentre count è un array contenente il numero di valori estratti ogni giorno.

Con i dati a disposizione, è stato ottenuto il seguente risultato:



Confrontando il giorno tipo (in alto a sinistra) con i primi tre giorni, è possibile notare che l'andamento del primo è abbastanza simile a quelli dei singoli giorni.

Esercizio 3

Il terzo esercizio richiede il calcolo della densità di probabilità (PDF) delle fluttuazioni di velocità del vento.
Una fluttuazione indica la differenza dei dati rispetto alla loro media:

$$\widehat{u} = \widehat{x} - \overline{\widehat{x}}$$

```
fluctuations=function (x)
{
  avg=mean(x,na.rm=TRUE)
  return(x-avg) ^fluctuations
}
```

La densità di probabilità descrive appunto la densità di una variabile aleatoria in ogni punto dello spazio campionario, essa viene calcolata attraverso la funzione density().
Per l'esercizio sono stati presi in considerazione i seguenti intervalli:

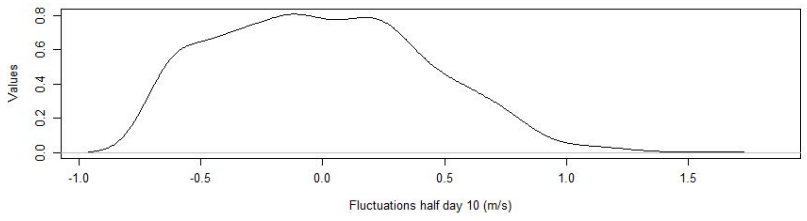
- tutto il primo giorno di settembre;
- tutto il mese di settembre;
- metà giorno dei primi dieci giorni di settembre;
- metà notte dei primi dieci giorni di settembre.

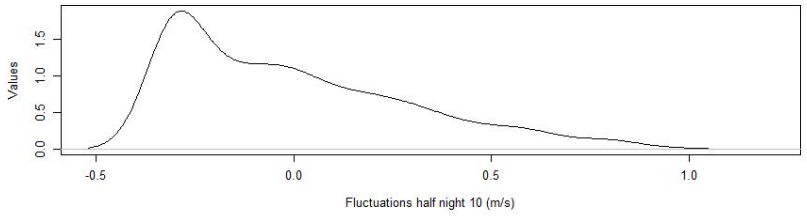
I dati relativi ai primi due punti sono stati presi rispettivamente con le funzione getDataOfDay e getDataMonth indicate all'inizio della relazione.

Per quanto riguarda invece l'alternanza giorno/notte, si è utilizzata la seguente funzione:

```
getDataHalf=function(i,type)
  {
  mat=matrix( data: 0, nrow: 2, ncol: 720)
  data=getDataOfDay(i,type);
  d=1;
  n=1;
  for(h in 1:1440)
    {
    if(!is.na(data[h]))
      {
      if(h>360 && h<=1080)
        {
        mat[1,d]=data[h]
        d=d+1
      }
      else
        {
        mat[2,n]=data[h]
        n=n+1
      }
    }
  }
  return(mat) ^getDataHalf
}
```

La funzione restituisce una matrice in cui nella prima riga vi sono i dati giornalieri (dalle 6 alle 18), nella seconda vi sono invece quelli notturni.
Utilizzando dei plot, si ottengono i grafici sottostanti e, com'è possibile notare, tutti hanno un andamento tendente alla distribuzione gaussiana.





Marco Carega - 20018128

Esercizio 4

Il quarto esercizio consiste nell'effettuare il test del chi quadro sulle densità delle fluttuazioni.
Il test del chi quadro è un test in cui si verifica se i dati in input hanno un andamento tendente alla distribuzione gaussiana.
Il test del chi quadro si effettua nel seguente modo:

$$\chi^2 \leq d$$

dove d è il numero di gradi di libertà, equivalente al numero di intervalli meno il numero di parametri:

$$d = \#intervalli - \#parametri$$

Il valore del chi quadro si calcola attraverso la seguente formula:

$$\chi^2 = \sum_k \frac{o_k - e_k}{e_k}$$

```
standard=function(x)
{
  return((x-mean(x))/sd(x))
}

chisquare=function (x,numOfIntervals)
{
  o=standard(frequencies(x,numOfIntervals))
  e=expected(o,numOfIntervals)
  acc=0;
  for(k in 2:numOfIntervals)
  {
    value=o[k] - e[k]
    value=value^2;
    value=value/e[k]
    acc=acc+value;
  }
  return(acc) ^chisquare
}
```

In cui:

- o_k indica i valori osservati, equivalente alla densità delle fluttuazioni standardizzata;
- e_k indica i valori attesi, calcolati a partire da quelli osservati.

I valori attesi si calcolano con la seguente formula:

$$e = \rho * \#valori$$

dove ρ è l'area della distribuzione gaussiana compresa tra il k-1-esimo e il k-esimo valore osservato:

$$\rho = pnorm(o_k) - pnorm(o_{k-1})$$

```
expected=function(x,numOfIntervals)
{
  stdX=standard(x)
  e=c(pnorm(stdX[1])*length(x))
  for(i in 2:numOfIntervals)
  {
    down=pnorm(q=stdX[i-1])
    up=pnorm(q=stdX[i])
    e=c(e,(up-down)*length(x))
  }
  return(e) ^expected
}
```

La sottrazione nella formula precedente è dovuta al fatto che pnorm calcola l'area da meno infinito all'estremo dato in input, quindi in questo modo si può calcolare l'area tra due estremi.
I risultati sono stati calcolati prendendo in considerazioni un numero di intervalli pari a 7 sugli stessi periodi utilizzati nel precedente esercizio.
Inoltre sono stati utilizzati due metodi per calcolare le densità:

- Sulla base degli intervalli, ricavare gli intervalli e quindi le frequenze;
- Utilizzare la funzione hist prendendo il campo $counts.

In entrambi i casi, la maggior parte dei valori passa il test, confermando di conseguenza l'andamento gaussiano dei dati.

Esercizio 5

Nel quinto esercizio lo scopo è quello di calcolare il tempo lagrangiano, utilizzando come mezzo la regressione esponenziale sull'autocorrelazione.
Una regressione è un'approssimazione di una serie di dati a una funzione, in questo caso l'approssimazione calcolata avrà un andamento esponenziale.
Per effettuare la regressione esponenziale, occorre calcolare i coefficienti A e B della seguente formula:

$$y = A + e^{Bx}$$

Da questa formula è possibile effettuare una semplificazione: ponendo z = ln(y) è infatti possibile calcolare i due coefficienti come nella regressione lineare.
La formula diventa quindi la seguente:

$$z = ln(y) = ln(A) + Bx$$

Il tempo lagrangiano dipende dal coefficiente di correlazione, quest'ultimo si calcola utilizzando la seguente formula:

$$r = e^{-\frac{\tau}{Lt}}$$

Da questa si può ricavare la formula del tempo lagrangiano:

$$ln(r) = \frac{\tau}{Lt} \rightarrow \frac{ln(r)}{\tau} = \frac{1}{Lt} \rightarrow \frac{\tau}{ln(r)} = Lt$$

Seguendo il ragionamento fatto prima, il tempo lagrangiano ha bisogno del coefficiente B per essere calcolato, la formula diventa quindi la seguente:

$$Lt = -\frac{1}{B} = -\frac{1}{line(ln(r))}$$

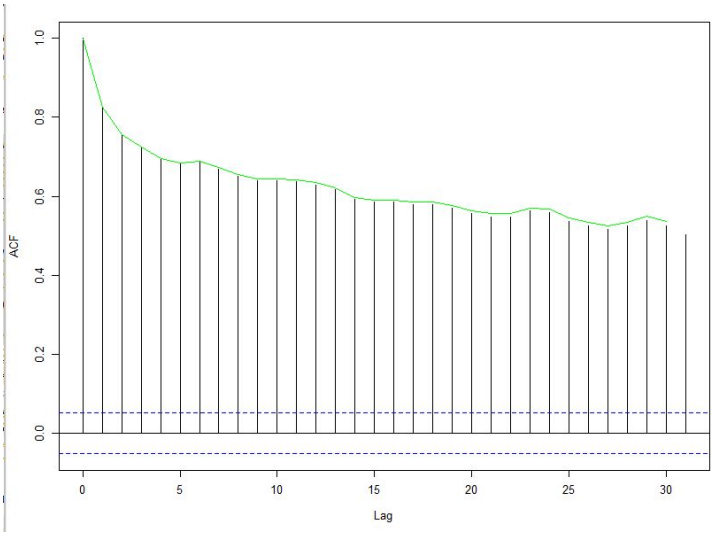
```
lagrangianTime=function(x)
  {
  r=c()
  for(dt in 0:length(x)-1)
    r[dt]=correlation(x,dt)
  #r=acf(x,plot=FALSE,type=c("correlation"))$acf
  ln=log(abs(r))
  firstCoeff=line(ln)$coefficients[1]
  return(-(1/firstCoeff)) ^lagrangianTime
}
```

Il coefficiente di correlazione r si calcola attraverso la seguente formula:

$$r(\tau) = \frac{\overline{u(t) * u(t+\tau)}}{\sigma^2}$$

```
correlation=function(x,dt)
  {
  variance=var(x)
  size=length(x)/(dt);
  dtX=c();
  t=1;
  for(t in 1:length(x))
    {
    if(t+dt <=length(x))
      {
      dtX[t]=x[t]*x[t+dt]
      t=t+1;
    }
  }
  return(mean(dtX)/variance)
}
```

L'esercizio chiedeva inoltre di confrontare la correlazione calcolata implementando la formula (linea verde) con quella ottenuta dalla funzione acf (linee verticali nere).
Come si può notare nell'immagine sottostante, vi sono somigliante nell'andamento delle due funzioni.



Marco Carega - 20018128

Esercizio 6

Nell'ultimo esercizio bisogna implementare un modello di Langevin in cui si disegnano le traiettorie di ogni particella in un piano (t,x).
L'implementazione di questo modello avviene attraverso la seguente formula:

$$du_i = -\frac{u_i}{Lt} * dt + \sqrt{Co * \varepsilon} * dW$$

dove du indica la variazione di velocità, dt è l'intervallo di tempo, dW è un processo di rumore bianco, Lt è il tempo lagrangiano, C0 è una costante (di solito uguale a 2) e ε è il rapporto di dissipazione dell'energia cinetica.
La formula precedente si può scrivere nel seguente modo:

$$u(t+\Delta t) - u(t) = -\frac{u(t)}{Lt} * dt + \sqrt{Co * \varepsilon} * dW$$

Sapendo che $\varepsilon = \frac{2*\sigma^2}{Lt*Co}$, si può effettuare la seguente semplificazione:

$$u(t+\Delta t) - u(t) = -\frac{u(t)}{Lt} * dt + \sqrt{\frac{2*\sigma^2}{Lt}} * dW$$

Per un funzionamento corretto del modello, è opportuno scegliere un Δt sufficientemente piccolo rispetto al tempo lagrangiano, quindi ponendo $\Delta t = \frac{Lt}{10}$ la formula si semplifica in questo modo:

$$u(t+\Delta t) = \frac{9}{10} * u(t) + \sqrt{\frac{2*\sigma^2}{Lt}} * dW$$

che diventa:

$$u_{i+1} = \frac{9}{10} * u_i + \sqrt{\frac{2*\sigma^2}{Lt}} * dW$$

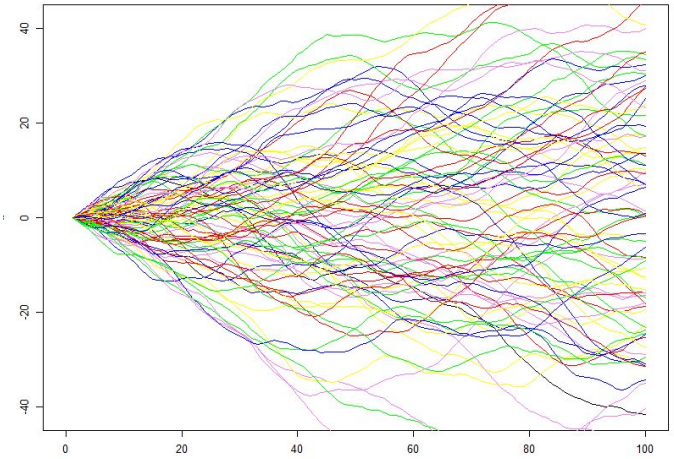
L'ultima formula è quella che è stata effettivamente implementata, ottenendo quindi la seguente funzione:

```
langevinModel=function(type,Co,numOfParticles,numOfTimes)
{

    for(day in 1:30)
    {
      particlesForInterations=array(data=0, dim=numOfParticles*numOfTimes)
      x=fluctuations(getDataOfDay(day,type))

      Lt=lagrangianTime(x)
      dt=Lt/10;
      for(p in 1:numOfParticles)
        {
        speed=0;
        positions=array(data=0,dim=numOfTimes)
        for(t in 1:numOfTimes)
          {
          speed=0.9*speed+sqrt(2*var(x)/Lt)*whiteNoise(t,dt)
          if(t>1)
          {
            positions[t]=positions[t-1]+speed*dt;
            particlesForInterations[p*t]=positions[t];
          }
        }
        plotline(positions,p,numOfParticles,numOfTimes)
      }
      differences(x,particlesForInterations)
    } <n>^langevinModel
}
```

Considerando i dati relativi al mese di settembre, al trentesimo giorno si è ottenuto il seguente risultato:



Ultimo ma non momento importante è il calcolo delle statistiche delle velocità con l'obiettivo di confrontarle con media e varianza, calcolate attraverso la seguente funzione:

```
differences=function(x,array)
{
  meanArray=mean(array,na.rm=TRUE);
  meanX=mean(x,na.rm=TRUE);
  sdArray=var(array,na.rm = TRUE)
  sdX=var(x,na.rm = TRUE)
  print(paste("Mean differences: ",abs(abs(meanArray) - abs(meanX))))
  print(paste("Deviation differences: ",abs(abs(sdArray) - abs(sdX))))
  print(" ") ^differences
}
```

In base ai dati relativi al giorno 30 settembre 2018, sono stati ottenuti i seguenti dati:

```
"Mean differences:  0.0415366376645302"
"Deviation differences:  0.302650273348315"
```