Laboration 1
Del 2

Kristian Lundkvist
900831

February 22, 2013

Contents

1 Uppgift 1

```
> x = floor(runif(100, -5, 20))
> x
[1]  4  1 18  6  7 13 -4  2 16  1  8 -5  2 13  9 -3  1 11
2 -1 -3 18 -5 -4 18
[26] 19 12  0 14  0 -1  3 16  4 -5  5 -5 14 17  5 -5 14 18
9 -5  8 19  2 -1 -4
[51] 15 17  8 15 15  1  2 16 15 14  7  5 10  9 15 -3 19  1
3  7 12 -3 15  2  5
[76]  0 10 12  9  6 14 19 -2 17 -1  1 11 14 18  3 11  8  0
8 -2  8 -1 -3 -4  3
```

1.1 a

```
> max(x)
[1] 19
```

1.2 b

```
> min(x)
[1] -5
```

1.3 c

```
> length(x)
[1] 100
```

1.4 d

```
> sum(x)
[1] 659
```

1.5 e

```
> cumsum(x)
[1]   4   5  23  29  36  49  45  47  63  64  72  67  69  82
91  88  89 100
[19] 102 101  98 116 111 107 125 144 156 156 170 170 169 172 188 192 187 192
[37] 187 201 218 223 218 232 250 259 254 262 281 283 282 278 293 310 318 333
[55] 348 349 351 367 382 396 403 408 418 427 442 439 458 459 462 469 481 478
[73] 493 495 500 500 510 522 531 537 551 570 568 585 584 585 596 610 628 631
[91] 642 650 650 658 656 664 663 660 656 659
```

cumsum() visar den kulminerande summan. Det vill säga [1] är summan av x[1], [2] är summan av x[1]+x[2], [3] är summan av x[1]+x[2]+x[3] och så vidare.

1.6 f

```
> diff(x)
 [1]  -3  17 -12   1   6 -17   6  14 -15   7 -13   7  11  -4 -12
 4  10  -9  -3
[20]  -2  21 -23   1  22   1  -7 -12  14 -14  -1   4  13 -12
-9  10 -10  19   3
[39] -12 -10  19   4  -9 -14  13  11 -17  -3  -3  19   2
-9   7   0 -14   1  14
[58]  -1  -1  -7  -2   5  -1   6 -18  22 -18   2   4   5 -15
18 -13   3  -5  10
[77]   2  -3  -3   8   5 -21  19 -18   2  10   3   4 -15
 8  -3  -8   8 -10  10
[96]  -9  -2  -1   7
```

1.7 g

```
> y = diff(x)
> diff(y)
 [1]  20 -29  13   5 -23  23   8 -29  22 -20  20   4 -15  -8
16   6 -19   6   1
[20]  23 -44  24  21 -21  -8  -5  26 -28  13   5   9 -25
 3  19 -20  29 -16 -15
[39]   2  29 -15 -13  -5  27  -2 -28  14   0  22 -17 -11
16  -7 -14  15  13 -15
[58]   0  -6   5   7  -6   7 -24  40 -40  20   2   1 -20
33 -31  16  -8  15  -8
[77]  -5   0  11  -3 -26  40 -37  20   8  -7   1 -19  23 -11
-5  16 -18  20 -19
[96]   7   1   8
```

Jag kan inte riktigt se några likheter mellan de två, det har ju dessutom trillat utanför gränserna vi satte när vi skapade mängden x.

1.8 h

```
> prod(x)
[1] 0
```

Detta stämmer bra. prod() multiplicerar alla värde i x och då 0 finns bland elementen i x så kommer resultatet bli 0. Hade 0 inte funnits så hade det blivit ett helt annant resultat.

1.9 i

```
> median(x)
[1] 6.5
```

1.10 j

```
> mean(x)
[1] 6.59
```

1.11 k

```
> quantile(x, type=5)
0%  25%  50%  75% 100%
-5.0  0.0  6.5 14.0 19.0
```

1.12 l

```
> var(x)
[1] 56.68879
```

1.13 m

```
> sd(x)
[1] 7.529196
```

1.14 n

```
> sqrt(x)
 [1] 2.000000 1.000000 4.242641 2.449490 2.645751 3.605551
NaN 1.414214
 [9] 4.000000 1.000000 2.828427      NaN 1.414214 3.605551 3.000000
NaN
[17] 1.000000 3.316625 1.414214      NaN      NaN 4.242641
NaN      NaN
[25] 4.242641 4.358899 3.464102 0.000000 3.741657 0.000000
NaN 1.732051
[33] 4.000000 2.000000      NaN 2.236068      NaN 3.741657 4.123106 2.236068
[41]      NaN 3.741657 4.242641 3.000000      NaN 2.828427 4.358899 1.414214
[49]      NaN      NaN 3.872983 4.123106 2.828427 3.872983 3.872983 1.000000
[57] 1.414214 4.000000 3.872983 3.741657 2.645751 2.236068 3.162278 3.000000
[65] 3.872983      NaN 4.358899 1.000000 1.732051 2.645751 3.464102
NaN
[73] 3.872983 1.414214 2.236068 0.000000 3.162278 3.464102 3.000000 2.449490
[81] 3.741657 4.358899      NaN 4.123106      NaN 1.000000 3.316625 3.741657
[89] 4.242641 1.732051 3.316625 2.828427 0.000000 2.828427
NaN 2.828427
[97]      NaN      NaN      NaN 1.732051
```

NaN är resultatet när man försöker ta kvadratroten ur 0, vilket inte går helt bra.

1.15 o

```
> x^2
 [1]  16   1 324  36  49 169  16   4 256   1  64  25   4 169
81   9   1 121
```

```
[19]   4   1   9 324  25  16 324 361 144   0 196   0   1
9 256  16  25  25
[37]  25 196 289  25  25 196 324  81  25  64 361   4   1
16 225 289  64 225
[55] 225   1   4 256 225 196  49  25 100  81 225   9 361
1   9  49 144   9
[73] 225   4  25   0 100 144  81  36 196 361   4 289   1
1 121 196 324   9
[91] 121  64   0  64   4  64   1   9  16   9
```

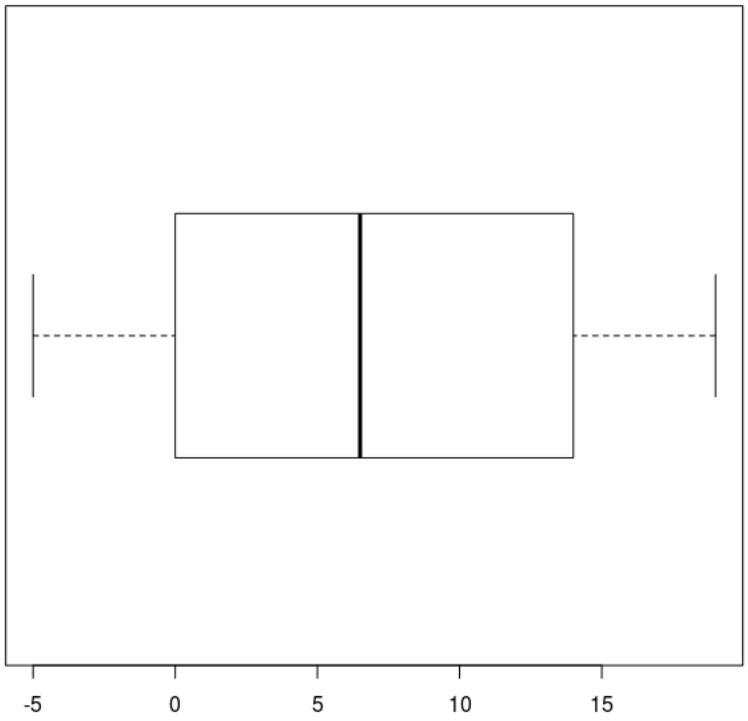
1.16 p

```
> x^(1/2)
 [1] 2.000000 1.000000 4.242641 2.449490 2.645751 3.605551
NaN 1.414214
 [9] 4.000000 1.000000 2.828427      NaN 1.414214 3.605551 3.000000
NaN
[17] 1.000000 3.316625 1.414214      NaN      NaN 4.242641
NaN      NaN
[25] 4.242641 4.358899 3.464102 0.000000 3.741657 0.000000
NaN 1.732051
[33] 4.000000 2.000000      NaN 2.236068      NaN 3.741657 4.123106 2.236068
[41]      NaN 3.741657 4.242641 3.000000      NaN 2.828427 4.358899 1.414214
[49]      NaN      NaN 3.872983 4.123106 2.828427 3.872983 3.872983 1.000000
[57] 1.414214 4.000000 3.872983 3.741657 2.645751 2.236068 3.162278 3.000000
[65] 3.872983      NaN 4.358899 1.000000 1.732051 2.645751 3.464102
NaN
[73] 3.872983 1.414214 2.236068 0.000000 3.162278 3.464102 3.000000 2.449490
[81] 3.741657 4.358899      NaN 4.123106      NaN 1.000000 3.316625 3.741657
[89] 4.242641 1.732051 3.316625 2.828427 0.000000 2.828427
NaN 2.828427
[97]      NaN      NaN      NaN 1.732051
```

1.17 q

```
> sqrt(var(x))
[1] 7.529196
> sqrt(sd(x))
[1] 2.743938
```

1.18 r



1.19 s

Variationsbredden är 24 (skillnaden mellan största och minsta värdet).

1.20 t

```
> max(x)-min(x)
[1] 24
```

1.21 u

```
> unique(x)
[1]  4  1 18  6  7 13 -4  2 16  8 -5  9 -3 11 -1 19 12  0 14
3  5 17 15 10 -2
```

unique visar endast varje specifikt värde en gång till skillnad från x som visar alla värde oavsett hur många gånger de förekommer.

1.22 v

```
> range(x)
[1] -5 19
```

Det minsta och det största värdet.

1.23 w

```
> which.max(x)
[1] 26
> which.min(x)
[1] 12
```

which.max() visar index för det största talet i mängden och which.min() visar index för det minsta talet i mängden.

1.24 x

Notering: min virtuella maskin krashade innan denna uppgiften så jag kommer ha en annan mängd för resterande uppgifter.

```
> rev(x)
[1] -1 13 16 -1  7 -3 19 16 12  4 14  1  7 10 15 16  9 -5
0  5 11  5 13 -1 10
[26] 11 14 19 15 14 -2  4 14  2 -4 16 16 11 18 19 -1  9  8 18
8  4  0 11  8  2
[51] 19  2 -1  0 14  7 13 10 16 -1  1  8 19 -2 14 -5  9 11 14 14 -1 18 15 10
7
[76] 17  6  6 -1  9  7  5 -1 -4  1 14  9 12  4  7 15  8 15
2 -1 11 -1 16 -5 19
```

rev vänder på mängden (första elementet hamnar sist och sista hamnar först osv).

1.25 y

```
> sort(x)
[1] -5 -5 -5 -4 -4 -3 -2 -2 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1
0  0  0  1  1  1
[26]  2  2  2  2  4  4  4  4  5  5  5  6  6  7  7  7  7  7
7  8  8  8  8  8  9
[51]  9  9  9  9 10 10 10 10 11 11 11 11 11 11 12 12 13 13 13 14 14 14 14 14 14
[76] 14 14 14 15 15 15 15 15 16 16 16 16 16 16 16 17 18 18 18 19 19 19 19 19 19
```

1.26 z

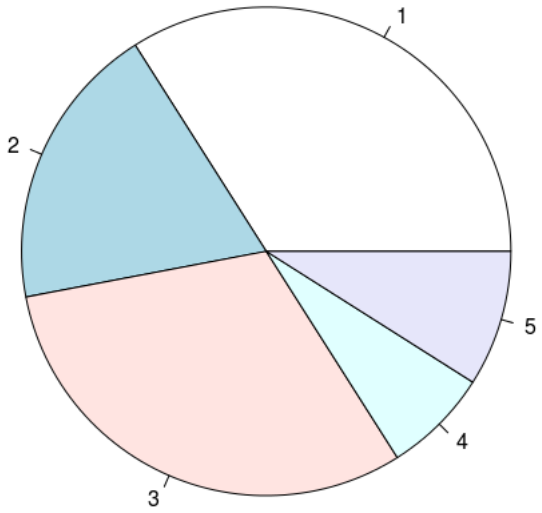
```
> rev(sort(x))
 [1] 19 19 19 19 19 19 18 18 18 17 16 16 16 16 16 16 16 15 15 15 15 15 14 14 14
[26] 14 14 14 14 14 14 13 13 13 12 12 11 11 11 11 11 11 10 10 10 10
 9  9  9  9
[51]  9  8  8  8  8  8  7  7  7  7  7  7  6  6  5  5  5  4
 4  4  4  2  2  2  2
[76]  1  1  1  0  0  0 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -2 -2 -3 -4 -4 -5 -5 -5
```

Det sorterar mängden som i uppgift y men vänder på den samtidigt (största hamnar först och minsta sist).

2 Uppgift 2

```
> p=c(34,19,31,7,9)
> p
[1] 34 19 31  7  9
```

2.1 a



2.2 b