Missing values in R*

```
> n = 15
> x = rpois(n,5)
> y = rpois(n,5);   d = x-y
> cbind(x,y,d)
      x y  d
 [1,] 9 3  6
 [2,] 1 5 -4
 [3,] 7 2  5
 [4,] 2 7 -5
 [5,] 6 3  3
 [6,] 4 3  1
 [7,] 1 5 -4
 [8,] 6 5  1
 [9,] 5 7 -2
[10,] 3 1  2
[11,] 6 5  1
[12,] 7 5  2
[13,] 6 6  0
[14,] 3 3  0
[15,] 5 6 -1
>
> # How many ties?
> length(d[x==y])
[1] 2
> # Which ones are they?
> (1:n)[x==y]
[1] 13 14
>

> # Now introduce some missing values, and re-calculate d
> x[4] = NA;  x[8] = NA
> y[3] = NA ; y[8] = NA;   d = x-y
>
```

* Copyright information is on the last page.

```
> cbind(x,y,d)
       x  y  d
 [1,]  9  3  6
 [2,]  1  5 -4
 [3,]  7 NA NA
 [4,] NA  7 NA
 [5,]  6  3  3
 [6,]  4  3  1
 [7,]  1  5 -4
 [8,] NA NA NA
 [9,]  5  7 -2
[10,]  3  1  2
[11,]  6  5  1
[12,]  7  5  2
[13,]  6  6  0
[14,]  3  3  0
[15,]  5  6 -1
>
> # How many ties? The answer should be two, or maybe three. Two is better.
> length(d[x==y])
[1] 5
> # Which ones are they?
> (1:n)[x==y]
[1] NA NA NA 13 14
> # This should work.
> d[d==0]
[1] NA NA NA  0  0
>
> # Just to show you that != is "not equal to"
> index = 1:5; index[index != 4]
[1] 1 2 3 5
>
> d[d != 0]
 [1]  6 -4 NA NA  3  1 -4 NA -2  2  1  2 -1
>
> # We see that NA matches BOTH logical conditions, maybe because when
> # R tries to do logic on NA, it can't, and so the result is NA.
>
> # This may explain the following:
> length(d[d != NA])
[1] 15
> length(d[d == NA])
[1] 15
```

```
>
> 3 == NA
[1] NA
> is.na(3)
[1] FALSE
> is.na(3) == F
[1] TRUE
>
>
> # Try this:

> length(d[x==y && is.na(x)==F && is.na(y)==F])
[1] 0
```

No doubt an expert can find a way around this, but also an honest, careful user can easily produce garbage.

Be particularly careful when sub-setting data, especially when sub-setting on a variable that might have missing values, like the crime for which a prisoner was originally arrested.

Or better, avoid using R when there are missing data.
