POL 345 Precept Week 9: Some Useful Commands

Jing Qian

2020/04/07

Welcome! In this note we will look into several commands that (hopefully) would be helpful when you are working on Problem set 6 and the precept exercise. Specifically, we will talk about three topics:

1. Merge two datasets together with `merge()`
2. Use `sign()` to get the sign of numerical values
3. Add texts to a plot

1. Merge two datasets

Sometimes we have two different datasets that share a same column, but each contain different information of the observation. For example, say we have two datasets as below:

First, we have the `info` dataset which contains some basic information for four people: Amy, Bob, Carol, and David.

```
info
```

```
##    name id class
## 1   Amy 12  2020
## 2   Bob 33  2022
## 3 Carol 40  2021
## 4 David 19  2023
```

In addition, we have another dataset called `home`, that contains the home state for each of the four people.

```
home
```

```
##    name state
## 1   Bob    CA
## 2 David    NY
## 3   Amy    NJ
## 4 Carol    MA
```

What if we want to combine the information of home state from `home` to other information in `info`? If the order of the `name` variable in both `info` and `home` are the same, we can just use `cbind` to combine the second column in `home` to `info`. However, as in the case here, the order of the names are different in two datasets. In such cases, we want R to combine the two datasets together by the information in the `name` variable, and we can do so by using the `merge()` command:

```
info_name = merge(info, home,
                  by = "name")
info_name
```

```
##    name id class state
## 1   Amy 12  2020    NJ
## 2   Bob 33  2022    CA
## 3 Carol 40  2021    MA
## 4 David 19  2023    NY
```

The command above asks R to combine any column in `home` to the dataset `info`, according to the value of `name` in both datasets, and the combined dataset is stored in the `info_name` object.

What if the column of the names is not called `name`? For example, say we have another dataset `major`, which contains the major of each person. However, this time the names are stored in the column called `x`.

```
major
```

```
##       x            major
## 1   Amy        Economics
## 2 Carol           Polics
## 3   Bob             Math
## 4 David Computer Science
```

In such case, we can use additional options in the `merge()` function to specify the column names from the two datasets that contain the identifier. For example:

```
info_name_major = merge(info_name,
                        major,
                        by.x = "name",
                        by.y = "x")
info_name_major
```

```
##    name id class state            major
## 1   Amy 12  2020    NJ        Economics
## 2   Bob 33  2022    CA             Math
## 3 Carol 40  2021    MA           Polics
## 4 David 19  2023    NY Computer Science
```

As the codes above show, we are now combining the dataset `info_name` with `major`, and we tell R that the names are stored in the `name` column in the first dataset (`info_name`), and stored in the `x` column in the second dataset (`major`).

What if you have repeated `names` in one dataset? For example, the dataset `phone` contains the screentime for each person across three different days:

```
phone
```

```
##     name day screentime
## 1    Amy   1         10
## 2    Amy   2          5
## 3    Amy   3          2
## 4    Bob   1          3
## 5    Bob   2          1
## 6    Bob   3          9
## 7  Carol   1          4
## 8  Carol   2          8
## 9  Carol   3         12
## 10 David   1          7
## 11 David   2         11
## 12 David   3          6
```

And we can still use the `merge()` function to combine information stored in `info` to the `phone` dataset, it won't be a problem that the name "Amy" or "Bob" appears for three times in `phone`. R will just assign the respective value from `info` to each of the observation.

```
merge(phone, info,
      by = "name")
```

```
##     name day screentime id class
## 1    Amy   1         10 12  2020
## 2    Amy   2          5 12  2020
## 3    Amy   3          2 12  2020
## 4    Bob   1          3 33  2022
## 5    Bob   2          1 33  2022
## 6    Bob   3          9 33  2022
## 7  Carol   1          4 40  2021
## 8  Carol   2          8 40  2021
## 9  Carol   3         12 40  2021
## 10 David   1          7 19  2023
## 11 David   2         11 19  2023
## 12 David   3          6 19  2023
```

2. Check the sign of numerical values

In cases where you simply want to know the sign of a number (i.e., whether it's greater than, equal to, or less than zero), you can use the `sign()` function to do that. The `sign()` function will return one of the three numbers: {-1, 0, 1}, as in the example below:

```r
sign(5)
```

```
## [1] 1
```

```r
sign(-3)
```

```
## [1] -1
```

```r
sign(0)
```

```
## [1] 0
```

And you can also use `sign()` to check whether two numbers are of the same sign:

```r
sign(3) == sign(4)
```

```
## [1] TRUE
```

```r
sign(-5) == sign(111)
```

```
## [1] FALSE
```

Of course, `sign()` can also be applied to a vector of numbers, in which case it will return a vector of results:

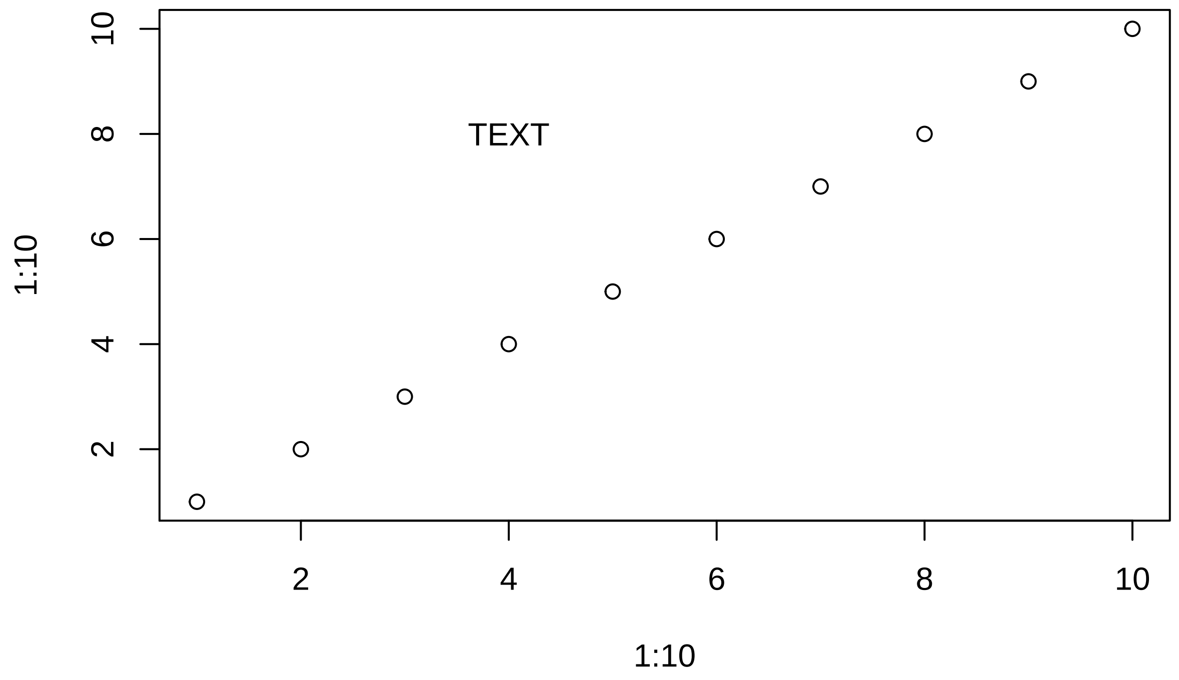
```r
x = c(3, 0, 11, -4, 1)
sign(x)
```

```
## [1]  1  0  1 -1  1
```

3. Add texts to a plot

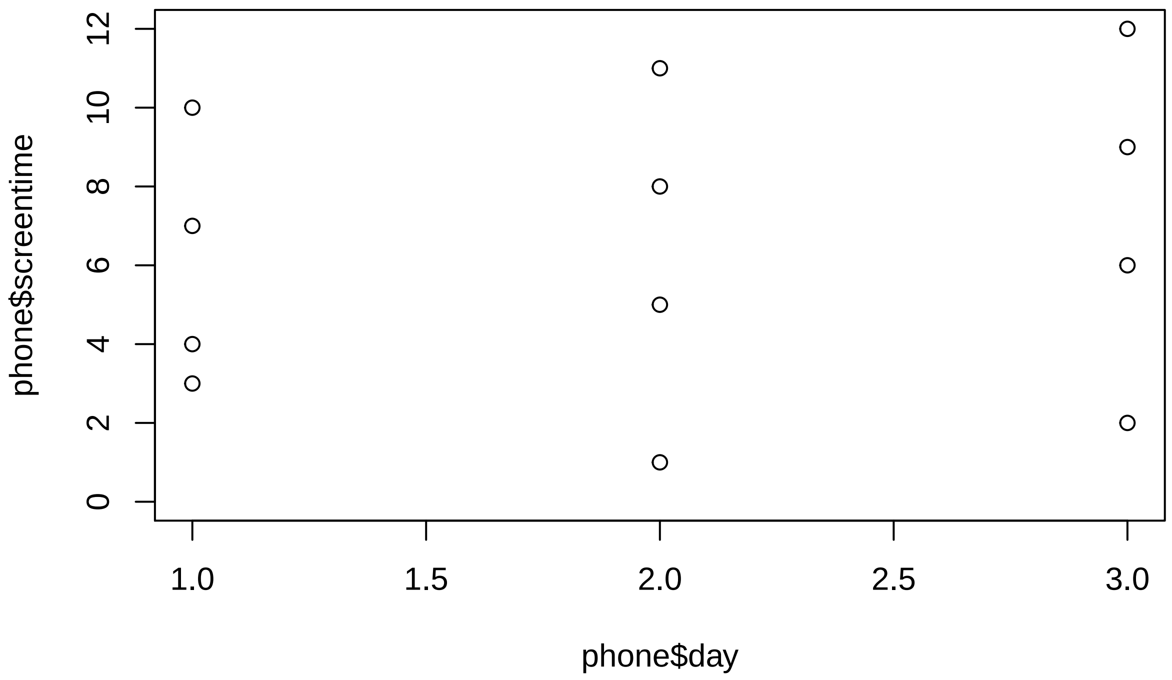
Previously, we have learned how to add text to a specific location of an existing plot. As in the example below, we add "TEXT" at the pont (4, 8) on the plot:

```r
plot(1:10,
     1:10)
text(x = 4, y = 8, labels = "TEXT")
```



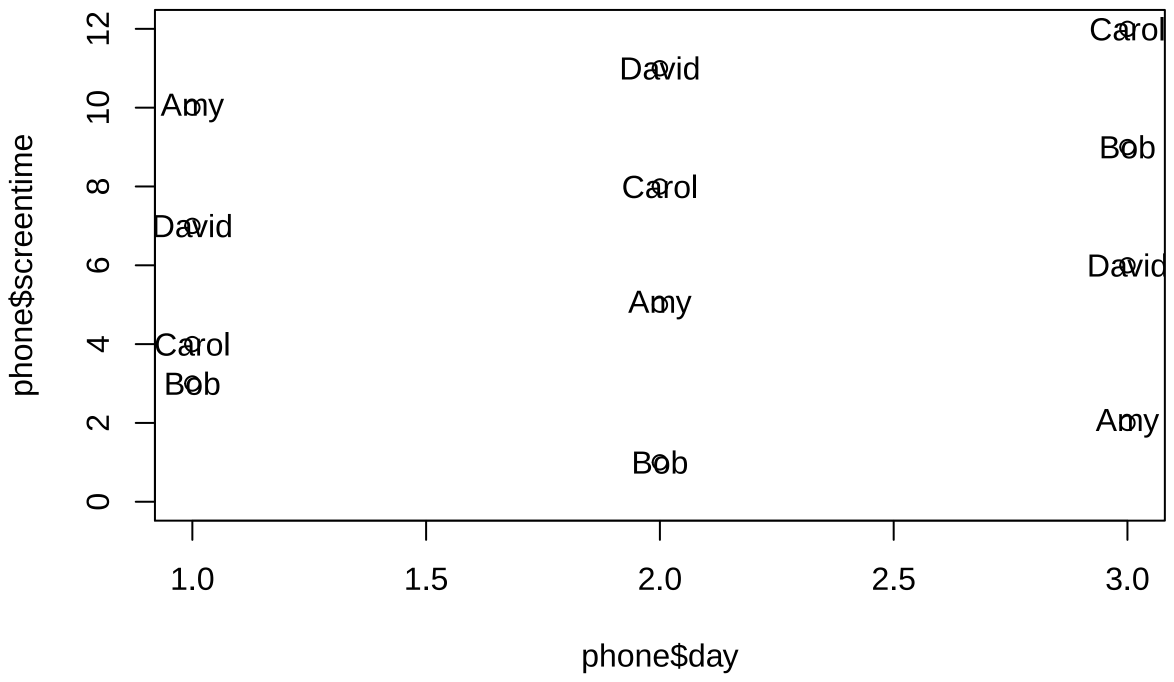
However, sometimes you might want to use texts to replace the points, as in the case of a scatter plot. For example, using the `phone` dataset, we can plot the relationship between `day` and `screentime`:

```
plot(phone$day,
     phone$screentime,
     ylim = c(0, 12))
```



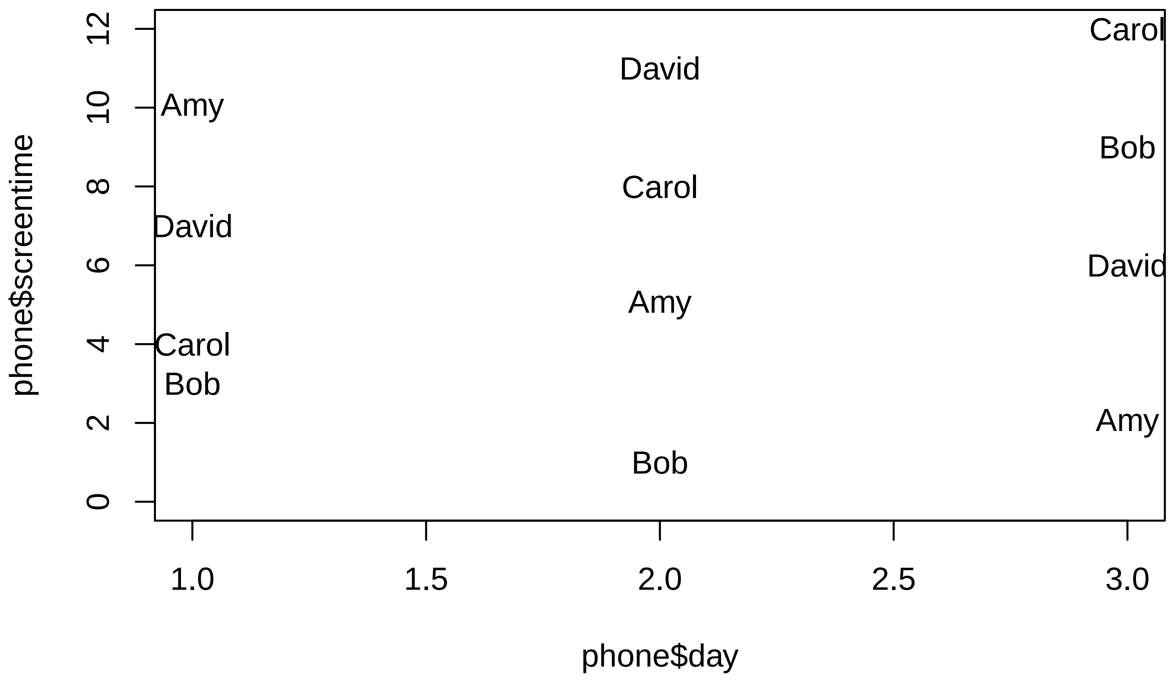
What if we want to add the name of each person to the plot? We can still do that by using `text()` after `plot()`. The same as adding any additional elements to an existing plot, we can feed the `text()` command with a vector of values, each for one text:

```
plot(phone$day,
     phone$screentime,
     ylim = c(0, 12))
#Add names
text(x = phone$day,
     y = phone$screentime,
     labels = phone$name)
```



As shown in the plot above, we have added the name of each person to the corresponding point. However, you might also want to get rid of the original points, which are now overlapped with our texts. You can do that by setting the `type` option in `plot` to "n", which means the symbol of the point will simply be empty.

```
plot(phone$day,
     phone$screentime,
     type = "n", #hide points
     ylim = c(0, 12))
#Add names
text(x = phone$day,
     y = phone$screentime,
     labels = phone$name)
```


phone$screentime
phone$day
0
2
4
6
8
10
12
1.0
1.5
2.0
2.5
3.0
Amy
David
Carol
Bob
David
Carol
Amy
Bob
Carol
Bob
David
Amy