Fix my mistakes - sequentially.shortened.sequence function

David Zeleny

I wrote a bit silly R function `sequentially.shortened.sequence`, which creates a sequentially shortened sentence, and prints it on screen. However, I was a bit in a hurry and did not really tidy up my R-code, so it may not work as expected. Would you please fix it up so that it works?

Note that I ask you to fix only the mistakes/mistypes, not to considerably rewrite and modify the script.

```
# define the function:
sequentially.shortened.sentence (sentence == "No need to have R depresssion today, things go
 perfectly smoothly!')
{
  for (i inn nchar (Sentence):1)
    print (substr (sentence, start = 1. stop = i))
}}

# Run the function with default values:
sequential1y.shortened,sentence {}

# Run the function with another sentence:
sequentiallly.shOrtened.sentenCe (sentencece = 'Here we go! Go - where?'#)
```

When the code is correct, the first run of the function (with default values) will return this:

```
[1] "No need to have R depresssion today, things go perfectly smoothly!"
[1] "No need to have R depresssion today, things go perfectly smoothly"
[1] "No need to have R depresssion today, things go perfectly smoothl"
[1] "No need to have R depresssion today, things go perfectly smooth"
[1] "No need to have R depresssion today, things go perfectly smoot"
[1] "No need to have R depresssion today, things go perfectly smoo"
[1] "No need to have R depresssion today, things go perfectly smo"
[1] "No need to have R depresssion today, things go perfectly sm"
[1] "No need to have R depresssion today, things go perfectly s"
[1] "No need to have R depresssion today, things go perfectly "
[1] "No need to have R depresssion today, things go perfectly"
[1] "No need to have R depresssion today, things go perfectl"
[1] "No need to have R depresssion today, things go perfect"
[1] "No need to have R depresssion today, things go perfec"
[1] "No need to have R depresssion today, things go perfe"
[1] "No need to have R depresssion today, things go perf"
[1] "No need to have R depresssion today, things go per"
[1] "No need to have R depresssion today, things go pe"
[1] "No need to have R depresssion today, things go p"
[1] "No need to have R depresssion today, things go "
[1] "No need to have R depresssion today, things go"
[1] "No need to have R depresssion today, things g"
[1] "No need to have R depresssion today, things "
[1] "No need to have R depresssion today, things"
[1] "No need to have R depresssion today, thing"
[1] "No need to have R depresssion today, thin"
[1] "No need to have R depresssion today, thi"
[1] "No need to have R depresssion today, th"
```

```
[1] "No need to have R depresssion today, t"
[1] "No need to have R depresssion today, "
[1] "No need to have R depresssion today,"
[1] "No need to have R depresssion today"
[1] "No need to have R depresssion toda"
[1] "No need to have R depresssion tod"
[1] "No need to have R depresssion to"
[1] "No need to have R depresssion t"
[1] "No need to have R depresssion "
[1] "No need to have R depresssion"
[1] "No need to have R depresssio"
[1] "No need to have R depresssi"
[1] "No need to have R depresss"
[1] "No need to have R depress"
[1] "No need to have R depres"
[1] "No need to have R depre"
[1] "No need to have R depr"
[1] "No need to have R dep"
[1] "No need to have R de"
[1] "No need to have R d"
[1] "No need to have R "
[1] "No need to have R"
[1] "No need to have "
[1] "No need to have"
[1] "No need to hav"
[1] "No need to ha"
[1] "No need to h"
[1] "No need to "
[1] "No need to"
[1] "No need t"
[1] "No need "
[1] "No need"
[1] "No nee"
[1] "No ne"
[1] "No n"
[1] "No "
[1] "No"
[1] "N"
```

The second run, with alternative text, will return:

```
[1] "Here we go! Go - where?"
[1] "Here we go! Go - where"
[1] "Here we go! Go - wher"
[1] "Here we go! Go - whe"
[1] "Here we go! Go - wh"
[1] "Here we go! Go - w"
[1] "Here we go! Go - "
[1] "Here we go! Go -"
[1] "Here we go! Go "
[1] "Here we go! Go"
[1] "Here we go! G"
[1] "Here we go! "
[1] "Here we go!"
```

```
[1] "Here we go"
[1] "Here we g"
[1] "Here we "
[1] "Here we"
[1] "Here w"
[1] "Here "
[1] "Here"
[1] "Her"
[1] "He"
[1] "H"
```

Hints:

1. Check the script first - is it really a definition of the function which should do all the job? And does it have too many or too few parentheses?
2. The function `nchar` counts the number of characters in the character string (e.g., `nchar ("Hello Doly!")` will return `11`.
3. The function `substr` subsets the character string (e.g., `substr ("Hello Doly!", start = 4, stop = 8)` will return `lo Do`).