Homework2-Functions

HuanChen

2017/9/25

Part 1

The function that computes the factorial of an integer greater than or equal to 0 is

```
Factorial<-function(n){
  sum = 1
  if(!abs(n-round(n)) < .Machine$double.eps^0.5){
    stop("n is not an interger!")
  }
  if(n < 0){
    print(NaN)
  }else if(n == 0){
    print(0)
  }else{
    for( i in 1 : n){
      sum = sum * i
    }
    print(sum)
  }
}
```

The outputs of the function are

```
Factorial(4)
```

```
## [1] 24
```

```
Factorial(0)
```

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

```
Factorial(-4)
```

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

Part 2

Design

I divide the main function into three functions `check_pkg_deps`, `CheckError` and `id.Visualize`.

The `check_pkg_deps` function check whether the three packages `readr`, `dplyr` and `ggplot2` have been installed and loaded, if not, install and load them.

The `CheckError` function decides whether the `input` exists in the dataset, if it doesn't, report an error. I acheive this function via the numbers of row of the dataframe after filtered. If numbers of row is 0, then the input doesn't exist, if the number is larger than 0, then the input exists.

I have choosen to implement a function to visualize the data by `id`. My function `id.Visualize` first applies the `check_pkg_deps` function to make sure that all the required packages have been installed. Then, I read the data in with the `read_csv`function and use the `filter` function to choose the data. Next, I apply the `CheckError` function to determine whether the `id` is a valid one. Finally, I visualize the data based on different `visit` or `room`.

Finally, I choose to visualize the data by `id=20`.

Function

The function to check whether the packages have been installed

```
check_pkg_deps <- function() {
  if(!require(readr)) {
    message("installing the 'readr' package")
    install.packages("readr")
    require(readr)
  }
  if(!require(dplyr)){
    message("installing the 'dplyr' package")
    install.packages("dplyr")
    require(dplyr)
  }
  if(!require(ggplot2)){
    message("installing the 'ggplot2' package")
    install.packages("ggplot2")
    require(ggplot2)
  }
}
```

The function to check whether the input is valid

```
CheckError<- function(mie.sub){
  if(nrow(mie.sub) == 0){
    stop("The mie data is not valid for this input")
  }
}
```

The main function to visualize the data

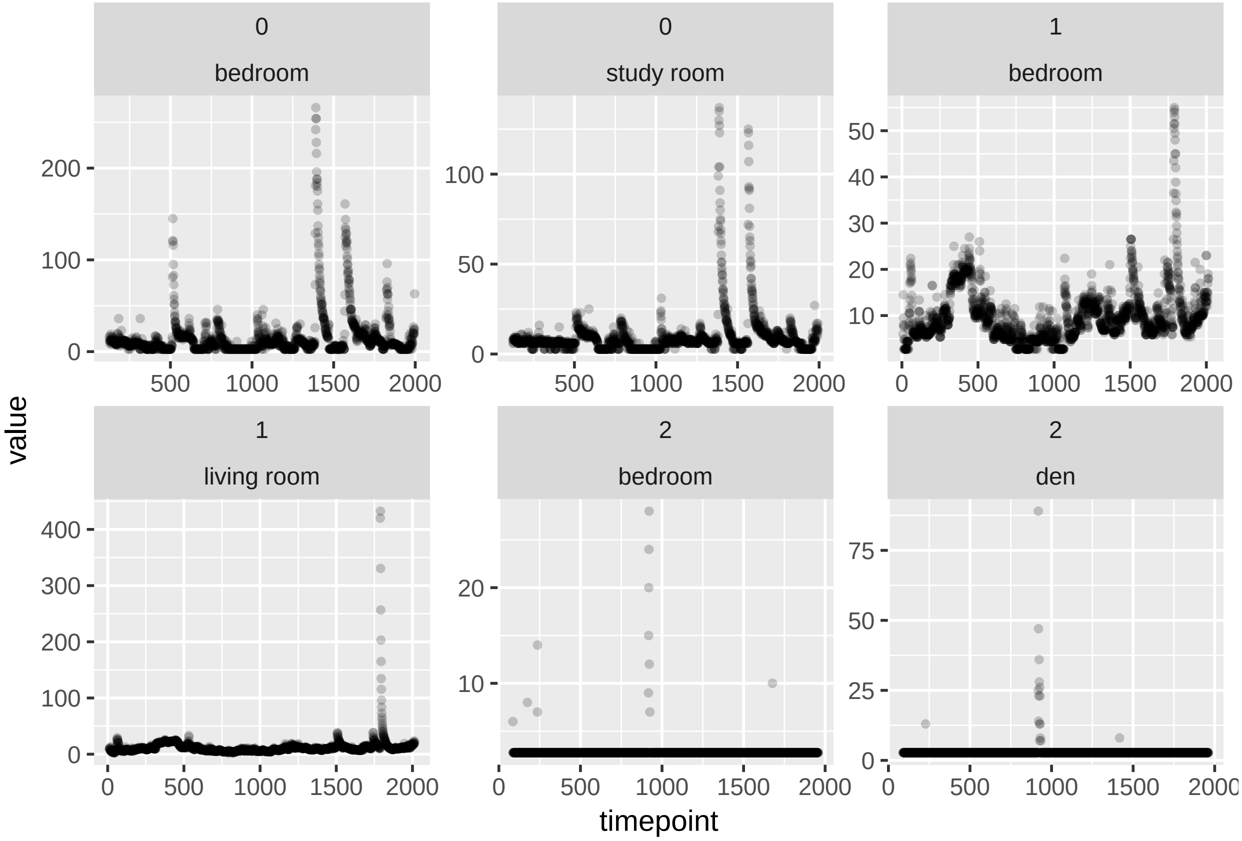
```
id.Visualize <- function(id){
  id.tmp=id
  check_pkg_deps()
  mie <- read_csv("MIE.zip", col_types = "cicdi")
  mie.sub <- mie %>% filter(id == id.tmp)
  CheckError(mie.sub)
  g <- ggplot(mie.sub, aes(timepoint,value))
  g + geom_point(alpha = 1/5 , size = 1) + facet_wrap(visit ~ room, scales='free')

}
```

Examples

The examples to visualize the data

```r
id.Visualize(20)
```



Part 3

Specification

My function `MedianCompute` takes the **two** inputs: `x` and `times`. `x` is the vector that will be entered to compute its 95% confidence interval for the median. `times` is the number of times to do the resampling, and it has a default value of 10000. Inside the function, when the input has missing values, I just omit them with the `na.omit` function. Then, I create a variable called `Store_Median`, which will store every median of the resampling data. With the `Store_Median`, I compute its 2.5% and 97.5% quantiles and obtain the final results. For the output, I first output the length of the vector, then output the 95% confidence interval of the median of that vector.

Function

The function that computes the 95% confidence interval for the median of a vector goes as follows

```r
MedianCompute <- function(x,times = 10000){
  x.tmp = na.omit(x)
  Store_Median = numeric(times)
  for( i in 1:times){
    Store_Median[i] = median(sample(x.tmp,replace = TRUE))
  }
  Low.quantile = quantile(Store_Median,0.025)
  High.quantile = quantile(Store_Median,0.975)
  print(paste("The length of the vector is", length(x)))
  print(paste("The 95% confidence interval is [",Low.quantile,",",High.quantile,"]"))
}
```

Read data throuth the `source` function

```r
source("median_testdata.R")
```

Then, based on the functions above, the median of the vectors in the `median_testdata.R` dataset is

```r
MedianCompute(x1)
```

```
## [1] "The length of the vector is 100"
## [1] "The 95% confidence interval is [ -0.377334790260323 , 0.0588250154502225 ]"
```

```r
MedianCompute(x2)
```

```
## [1] "The length of the vector is 1000"
## [1] "The 95% confidence interval is [ 13.4605026524514 , 15.8779056825996 ]"
```

```r
MedianCompute(x3)
```

```
## [1] "The length of the vector is 749"
## [1] "The 95% confidence interval is [ -1.10826348548744 , 1.16882975792226 ]"
```

```r
MedianCompute(x4)
```

```
## [1] "The length of the vector is 85"
## [1] "The 95% confidence interval is [ 0.791661172636908 , 0.863630334269703 ]"
```

```r
MedianCompute(x5)
```

```
## [1] "The length of the vector is 5"
## [1] "The 95% confidence interval is [ 5 , 15 ]"
```