Stat func

Weihao Lu

2018/3/30

EX1: Calculate the value of a polynomial

EX1 Problem1

```
directpoly=function(x, poly.coef){
    ## Calculate the polynomial value of x
    ## x can be a value or a vector
    ## Polynomial coefficients from x^(n-1)'s to constant term
    n=length(poly.coef)
    design.mat=matrix(1, length(x), n)
    for(i in 1:(n-1)){
        design.mat[,n-i]=x*design.mat[,n-i+1]
    }
    return(as.vector(design.mat %*% poly.coef))
}

# x is a value
ans1=directpoly(5, c(2,3,4))
ans1
```

```
## [1] 69
```

```
# x is a vector
ans2=directpoly(5:7, c(2,3,4))
ans2
```

```
## [1]  69  94 123
```

EX1 Problem2

```
hornerpoly=function(x, poly.coef){
    ## Calculate the polynomial value of x
    ## x can be a value or a vector
    ## Polynomial coefficients from x^(n-1)'s to constant term
    n=length(poly.coef)
    horner=rep(poly.coef[1], length(x))
    for(i in 2:n)horner=horner*x+poly.coef[i]
    return(horner)
}

# x is a value
ans1=hornerpoly(5, c(2,3,4))
ans1
```

```
## [1] 69
```

```
# x is a vector
ans2=hornerpoly(5:7, c(2,3,4))
ans2
```

```
## [1]  69  94 123
```

EX1 Problem3

```
# x is a value
system.time({directpoly(2015301000086, 86:2015301)})  #directpoly
```

```
##    user  system elapsed
##    2.74    0.01    2.77
```

```
system.time({hornerpoly(2015301000086, 86:2015301)})  #hornerpoly
```

```
##    user  system elapsed
##    0.22    0.00    0.22
```

```
# x is a vector
rn=rchisq(1000, df=20)
system.time({directpoly(100:1000, rn)})  #directpoly
```

```
##    user  system elapsed
##    0.03    0.00    0.04
```

```
system.time({hornerpoly(100:1000, rn)})  #hornerpoly
```

```
##    user  system elapsed
##       0       0       0
```

EX2: My statistic function

```
mystatistic.f=function(y, na.rm=FALSE){
    ## Calculate some basic statistics of y
    ## y must be a vector
    ## If na.rm=TRUE, all NAs will be removed
    if(!na.rm && sum(is.na(y))){
        na.rm=TRUE
        message("NAs was detected in y but na.rm is set to FALSE, ",
                "so calculate with NAs removed.")
    }
    if(na.rm)y=y[!is.na(y)]
    mymean=mean(y)
    mysd=sd(y)
    n=length(y)
    mysk=sum((y-mymean)^3/mysd^3)/n
    myku=sum((y-mymean)^4/mysd^4)/n-3
    return(cbind(mean=mymean, sd=mysd, skewness=mysk, kurtosis=myku))
}

mystatistic.f(1:5)
```

```
##      mean       sd skewness kurtosis
## [1,]    3 1.581139        0   -1.912
```

```
mystatistic.f(c(1:5, NA))
```

```
## NAs was detected in y but na.rm is set to FALSE, so calculate with NAs removed.
```

```
##      mean       sd skewness kurtosis
## [1,]    3 1.581139        0   -1.912
```

EX3: Another statistic function

```
mysk.f=function(y){
    ## Calculate skewness of y
    ## y must be a numeric vector
    ## All NAs removed
    if(!is.numeric(y)){
        stop("Vector must be number.")
    }
    y=y[!is.na(y)]
    mymean=mean(y)
    mysd=sd(y)
    n=length(y)
    mysk=sum((y-mymean)^3/mysd^3)/n
    if(abs(mysk)<1){
        return(list(skewness=mysk, descstats=cbind(mean=mymean, sd=mysd)))
    }else{
        return(list(skewness=mysk, descstats=quantile(y)))
    }

}

set.seed(086)
```

```
mysk.f(c("Arthur", "Mary", "Rover"))
```

This code will print: *Error in mysk.f(c("Arthur", "Mary", "Rover")) : Vector must be number.*

```
mysk.f(rnorm(100))
```

```
## $skewness
## [1] -0.1402162
##
## $descstats
##           mean        sd
## [1,] 0.1372597 0.9163136
```

```
mysk.f(rexp(100,5))
```

```
## $skewness
## [1] 1.641151
##
## $descstats
##          0%         25%         50%         75%        100%
## 0.000318541 0.059569834 0.144962753 0.268473947 0.961595777
```