



Practicum in Statistical Computing

2021 Fall / APSTA-GE.2352

Lab week 7

Kwan Bo Shim

Oct 26, 2021

P A R T 0 1

Week 7

Week 7

1. Poll

2. Attendance

3. Homework due today

Week 7

Density estimate / review Lab_report 2

- Density estimate with mode
- Lab_report2 -> moving average

P A R T 0 2

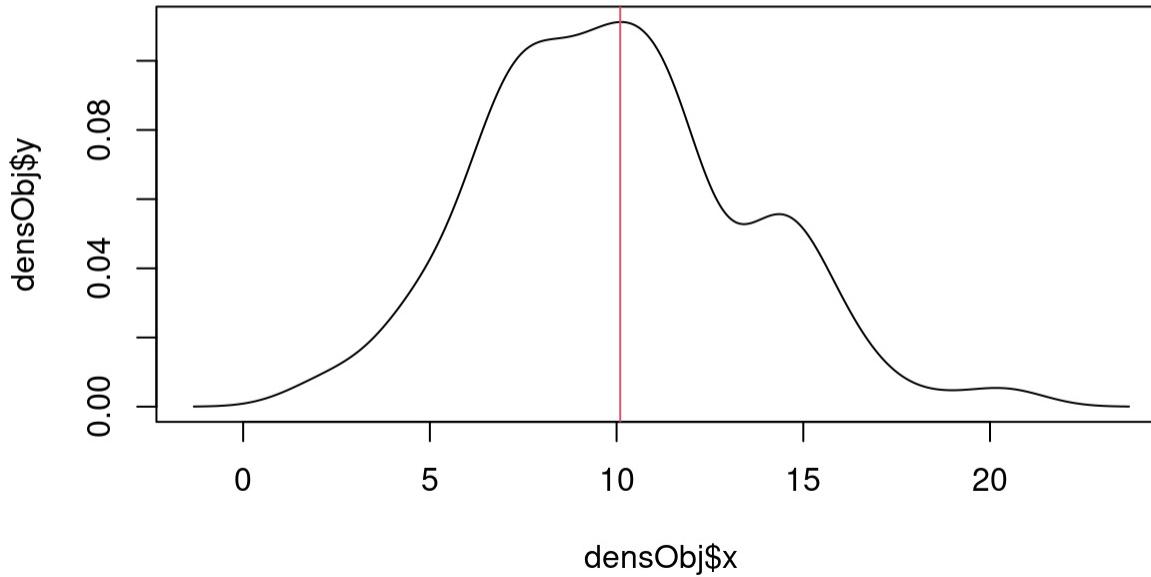


Density

Density function

1. Density function -> density curve, “smooth estimate” -> so it allows us to find **a maximum value**
 - (=the mode)
 - (=most frequent value)
 - (=most likely value)

Density estimate & max value



P A R T 0 3



function

Make a function

```
# Function signature:  
  
findMode <- function(x) {  
  #call density on x; save result  
  densObj <- density( )  
  #find maximum y location  
  loc.max.y <-  
  #return the x value at that location  
  return( )  
}
```

Make a function

```
# Function signature:  
  
findMode <- function(x) {  
  #call density on x; save result  
  densObj <- density( x )  
  #find maximum y location  
  loc.max.y <- which.max( )  
  #return the x value at that location  
  return( densObj$x[ ] )  
}
```

Make a function

```
# Function signature:  
  
findMode <- function(x) {  
  #call density on x; save result  
  densObj <- density( x )  
  #find maximum y location  
  loc.max.y <- which.max(densObj$y)  
  #return the x value at that location  
  return( densObj$x[loc.max.y] )  
}
```

P A R T 0 4



practice

```
findMode <- function(x) {  
  #call density on x; save result  
  densObj <- density( x )  
  #find maximum y location  
  loc.max.y <- which.max(densObj$y)  
  #return the x value at that location  
  return( densObj$x[loc.max.y] )  
}  
  
# try run it  
findMode( airquality$Wind )  
-> error!  
  
# why??
```

Fix the function

```
findMode <- function(x) {  
  Maybe use this area?  
  #call density on x; save result  
  densObj <- density( x )  
  #find maximum y location  
  loc.max.y <- which.max(densObj$y)  
  #return the x value at that location  
  return( densObj$x[loc.max.y] )  
}
```



for loop? Conditional? Direct imputation?

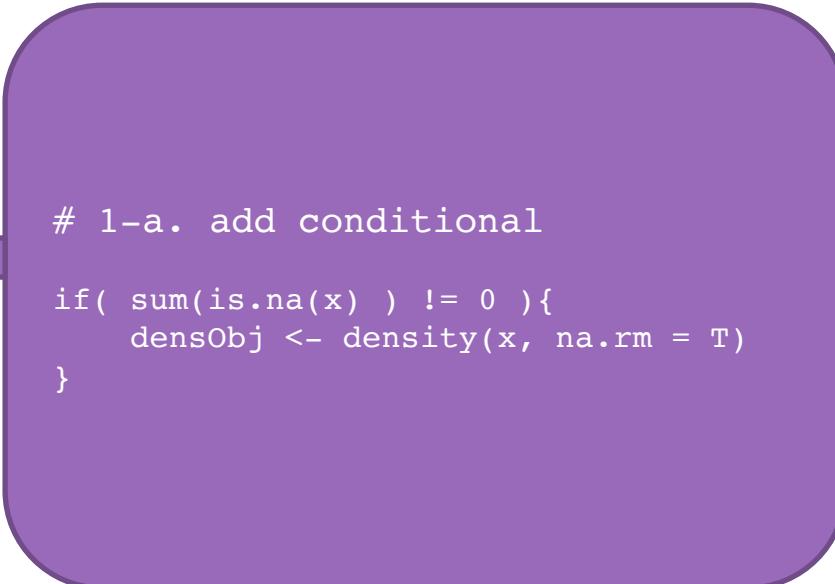
P A R T 05



Some ways

Fix the function

```
findMode <- function(x) {  
  #call density on x; save result  
  densObj <- density( x )  
  
  #find maximum y location  
  loc.max.y <- which.max(densObj$y)  
  #return the x value at that location  
  return( densObj$x[loc.max.y] )  
}
```



```
# 1-a. add conditional  
  
if( sum(is.na(x) ) != 0 ){  
  densObj <- density(x, na.rm = T)  
}
```

Fix the function

```
findMode <- function(x) {  
  #call density on x; save result  
  densObj <- density( x )  
  #find maximum y location  
  loc.max.y <- which.max(densObj$y)  
  #return the x value at that location  
  return( densObj$x[loc.max.y] )  
}
```



```
# 1-b. Complete conditional  
  
if( sum(is.na(x) ) != 0 ){  
  densObj <- density(x, na.rm = T)  
} else {  
  densObj <- density(x)  
}
```

Fix the function

```
findMode <- function(x) {  
  #call density on x; save result  
  densObj <- density( x )  
  #find maximum y location  
  loc.max.y <- which.max(densObj$y)  
  #return the x value at that location  
  return( densObj$x[loc.max.y] )  
}
```



```
# 2. direct imputation  
  
densObj <- density(x, na.rm = T)
```

P A R T 0 6



Lab_report 2

Moving average function

1. Recommendation : Review R Shiny App: Week5
2. Smoothing using small moving window batches: In this lab we practice moving from a scripted solution to building a more **general function in the context of smoothing.**

Moving average function

```
moving_average <- function(x,y,k,na.rm=T){  
  ord <- order(x)  
  x <- x[ord]  
  y <- y[ord]  
  #establish the number of evaluation points from the data itself.  
  n <- length(x)  
  
  # loop to create a moving window. Calculate median inside  
  moving_average <- rep(NA, n) #placeholder for results of the median call  
  for (i in 1:n) {  
    if (i - k < 1) { #boundary condition  
      lower <- 1  
    } else {  
      lower <- i - k #  
    }  
    upper <- min(n, i + k) #prevents going past the end of the series  
    moving_average[i] <- mean(y[lower:upper], na.rm = na.rm)  
  }  
  return(list(x=x,y=moving_average))  
}
```

1. Preprocessing

Moving average function

```
moving_average <- function(x,y,k,na.rm=T){  
  ord <- order(x)  
  x <- x[ord]  
  y <- y[ord]  
  #establish the number of evaluation points from the data itself.  
  n <- length(x)  
  # loop to create a moving window. Calculate median inside  
  moving_average <- rep(NA, n) #placeholder for results of the median call  
  for (i in 1:n) {  
    if (i - k < 1) { #boundary condition  
      lower <- 1  
    } else {  
      lower <- i - k #  
    }  
    upper <- min(n, i + k) #prevents going past the end of the series  
    moving_average[i] <- mean(y[lower:upper], na.rm = na.rm)  
  }  
  return(list(x=x,y=moving_average))  
}
```

2. Loop for lower /
upper bounds

Moving average function

```
moving_average <- function(x,y,k,na.rm=T){  
  ord <- order(x)  
  x <- x[ord]  
  y <- y[ord]  
  #establish the number of evaluation points from the data itself.  
  n <- length(x)  
  # loop to create a moving window. Calculate median inside  
  moving_average <- rep(NA, n) #placeholder for results of the median call  
  for (i in 1:n) {  
    if (i - k < 1) { #boundary condition  
      lower <- 1  
    } else {  
      lower <- i - k #  
    }  
    upper <- min(n, i + k) #prevents going past the end of the series  
    moving_average[i] <- mean(y[lower:upper], na.rm = na.rm)  
  }  
  return(list(x=x, y=moving_average))  
}
```

3. Compute
moving average

Scatter smoothing function

1. Recommendation : Review [R Shiny App Week4 Lab](#) section for boxcar weight + creating grid blocks: “The purpose of this lab is to create our own “rough” density estimation routine using boxcar weights.”
2. **n: the number of grid points** (THE NUMBER OF GRID POINTS DETERMINES THE LEVEL OF PRECISION OF THE ESTIMATE).
Example: if the data range from 4 to 10, you want to evaluate the density at n points in that range, so if n = 1000, then seq(4, 10, length.out = 1000) is the set of points being evaluated.
3. **frac:** the fraction of available data **on either side of evaluation point**. That is, if the data range from 4 to 10, then they span $10 - 4 = 6$ units. The fraction of that range that is used in the boxcar is $\text{frac} * 6$. If $\text{frac} < 0.5$, then the boxcar total width would be 3, with 1.5 on either side of each evaluation point.

Scatter smoothing function

- Recall that **scatterplot smoothing** is different from **density estimation**.
- You are smoothing y values based on a moving window through x values.
- Think of it as "looking up" from the x to see the values of y that are associated with those x values.

Scatter smoothing function

I am gridblock

4

10

Scatter smoothing function

I am gridblock

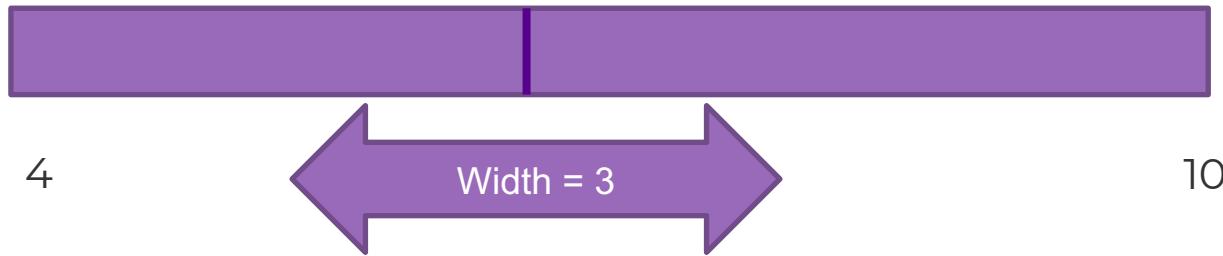
4

10



- Span = $10 - 4 = 6$
(difference between min, max)

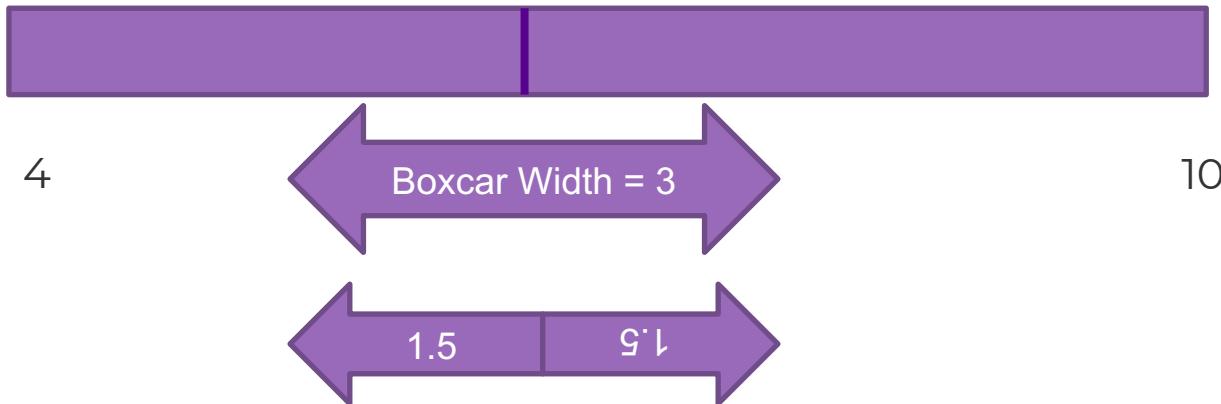
Scatter smoothing function



If $\text{frac}=0.5$

- Boxcar width = $6 * 0.5 = 3$

Scatter smoothing function



If frac=0.5

- Boxcar width = $6 * 0.5 = 3$
- Left=right= $3/2 = 1.5$ each

Think of it as choosing # neighbors (like choosing k value before)

scatterplot smoother function

```
moving_average <- function(x,y,frac=0.1,n=100,na.rm=T){  
  rng <- range(x)  
  gridLength <- diff(rng)  
  boxcarLength <- frac*gridLength  
  x.grid <- seq(rng[1],rng[2],length=n)  
  
  # loop to create a moving window. Calculate mean inside  
  movingAverage <- rep(NA, n) #placeholder for results of the mean calls  
  for (i in 1:n) {  
    #define a vector of `selected` points centered at x.grid[i]  
    ###selPoints <- ...  
    selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2  
    # calculate and assign mean of y[selPoints]:  
    ###movingAverage[i] <- ... #remember to average the y associated with x.  
    movingAverage[i] <- mean( y[selPoints], na.rm = na.rm)  
  }  
  return(list(x=x.grid,y=movingAverage))  
}
```

1. Preprocessing making grid block

- What range(x) do?

scatterplot smoother function

```
moving_average <- function(x,y,frac=0.1,n=100,na.rm=T){  
  rng <- range(x)  
  gridLength <- ceiling(rng)  
  boxcarLength <- frac * gridLength  
  x.grid <- seq(rng[1],rng[2],length=n)  
  
  # loop to create a moving window. Calculate mean inside  
  movingAverage <- rep(NA, n) #placeholder for results of the mean calls  
  for (i in 1:n) {  
    #define a vector of `selected` points centered at x.grid[i]  
    ###selPoints <- ...  
    selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2  
    # calculate and assign mean of y[selPoints]:  
    ###movingAverage[i] <- ... #remember to average the y associated with x.  
    movingAverage[i] <- mean( y[selPoints], na.rm = na.rm)  
  }  
  return(list(x=x.grid,y=movingAverage))  
}
```

1. Preprocessing
making grid block

scatterplot smoother function

```
moving_average <- function(x,y,frac=0.1,n=100,na.rm=T){  
  rng <- range(x)  
  gridLength <- diff(rng)  
  boxcarLength <- frac*gridLength  
  x.grid <- seq(rng[1],rng[2],length=n)  
  
  # loop to create a moving window. Calculate mean inside  
  movingAverage <- rep(NA, n) #placeholder for results of the mean calls  
  for (i in 1:n) {  
    #define a vector of `selected` points centered at x.grid[i]  
    ##selPoints <- ...  
    selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2  
    # calculate and assign mean of y[selPoints]:  
    ##movingAverage[i] <- ... #remember to average the y associated with x.  
    movingAverage[i] <- mean( y[selPoints], na.rm = na.rm)  
  }  
  return(list(x=x.grid,y=movingAverage))  
}
```

1. Preprocessing
making grid block

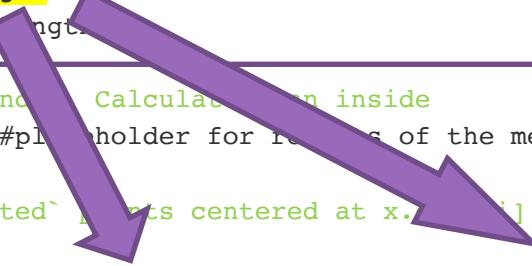
- boxcarLength?

scatterplot smoother function

```

moving_average <- function(x,y,frac=0.1,n=100,na.rm=T){
  rng <- range(x)
  gridLength <- diff(rng)
  boxcarLength <- frac*gridLength
  x.grid <- seq(rng[1],rng[2],length=n)
  # loop to create a moving window
  # calculate mean inside
  movingAverage <- rep(NA, n) #placeholder for results of the mean calls
  for (i in 1:n) {
    #define a vector of `selected` points centered at x[i]
    ##selPoints <- ...
    selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2
    # calculate and assign mean of y[selPoints]:
    ##movingAverage[i] <- ... #remember to average the y associated with x.
    movingAverage[i] <- mean( y[selPoints], na.rm = na.rm)
  }
  return(list(x=x.grid,y=movingAverage))
}

```



The diagram consists of two purple arrows. One arrow points from the line **boxcarLength <- frac*gridLength** to the explanatory text "1. Preprocessing making grid block". Another arrow points from the line **selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2** to the explanatory text "- boxcarLength? Creating range for boxcar width".

1. Preprocessing making grid block

- boxcarLength?
- Creating range for boxcar width

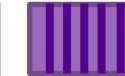
scatterplot smoother function

```

moving_average <- function(x,y,frac=0.1,n=100,na.rm=T){
  rng <- range(x)
  gridLength <- diff(rng)
  boxcarLength <- frac*gridLength
  x.grid <- seq(rng[1],rng[2],length=n)

# loop to create a moving window. Calculate mean inside
movingAverage <- rep(NA, n) #placeholder for results of the mean calls
for (i in 1:n) {
  #define a vector of `selected` points centered at x.grid[i]
  ##selPoints <- ...
  selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2
  # calculate and assign mean of y[selPoints]:
  ##movingAverage[i] <- ... #remember to average the y associated with x.
  movingAverage[i] <- mean( y[selPoints], na.rm = na.rm)
}
return(list(x=x.grid,y=movingAverage))
}

```



I am gridblock

Min(x)

Max(x)

of blocks = n

- Define how many room (=grid point) you want to build
- Benefit of having more room?

scatterplot smoother function

```
moving_average <- function(x,y,frac=0.1,n=100,na.rm=T){  
  rng <- range(x)  
  gridLength <- diff(rng)  
  boxcarLength <- frac*gridLength  
  x.grid <- seq(rng[1],rng[2],length=n)  
  
  # loop to create a moving window. Calculate mean inside  
  movingAverage <- rep(NA, n) #placeholder for results of the mean calls  
  for (i in 1:n) {  
    #define a vector of `selected` points centered at x.grid[i]  
    ###selPoints <- ...  
    selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2  
    # calculate and assign mean of y[selPoints]:  
    ###movingAverage[i] <- ... #remember to average the y associated with x.  
    movingAverage[i] <- mean( y[selPoints], na.rm = na.rm)  
  }  
  return(list(x=x.grid,y=movingAverage))  
}
```

selpoint ?
Who are u ?

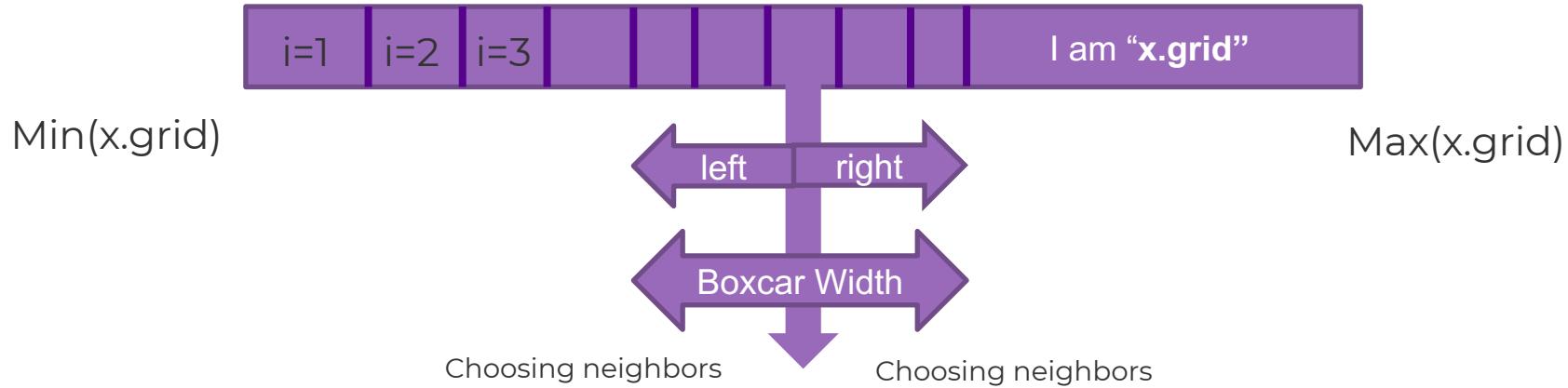
scatterplot smoother function

```
moving_average <- function(x,y,frac=0.1,n=100,na.rm=T){  
  rng <- range(x)  
  gridLength <- diff(rng)  
  boxcarLength <- frac*gridLength  
  x.grid <- seq(rng[1],rng[2],length=n)  
  
  # loop to create a moving window. Calculate mean inside  
  movingAverage <- rep(NA, n) #placeholder for results of the mean calls  
  for (i in 1:n) {  
    #define a vector of `selected` points centered at x.grid[i]  
    ###selPoints <- ...  
    selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2  
    # calculate and assign mean of y[selPoints]:  
    ###movingAverage[i] <- ... #remember to average the y associated with x.  
    movingAverage[i] <- mean( y[selPoints], na.rm = na.rm)  
  }  
  return(list(x=x.grid,y=movingAverage))  
}
```

“selpoint”

- all selected points of **x**
- vector format

Scatterplot smoother



- For some room $i = 1, 2, 3, \dots, n$, (because we made n rooms -> we defined already)
select all points located inside this room!
Where each **length = boxcar width = left + right**
So **grid length != boxcar width**

scatterplot smoother function

```
moving_average <- function(x,y,frac=0.1,n=100,na.rm=T){  
  rng <- range(x)  
  gridLength <- diff(rng)  
  boxcarLength <- frac*gridLength  
  x.grid <- seq(rng[1],rng[2],length=n)  
  
  # loop to create a moving window. Calculate mean inside  
  movingAverage <- rep(NA, n) #placeholder for results of the mean calls  
  for (i in 1:n) {  
    #define a vector of `selected` points centered at x.grid[i]  
    ###selPoints <- ...  
    selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2  
    # calculate and assign mean of y[selPoints]:  
    movingAverage[i] <- ... #remember to average the y associated with x.  
    movingAverage[i] <- mean( y[selPoints], na.rm = na.rm)  
  }  
  return(list(x=x.grid, y=movingAverage))  
}
```

Since we found all x values located in each room (box width that we defined), what we have to do? -> **average**

scatterplot smoother function

```

moving_average <- function(x,y,frac=0.1,n=100,na.rm=T){
  rng <- range(x)
  gridLength <- diff(rng)
  boxcarLength <- frac*gridLength
  x.grid <- seq(rng[1],rng[2],length=n)

  # loop to create a moving window. Calculate mean inside
  movingAverage <- rep(NA, n) #placeholder for results of the mean calls
  for (i in 1:n) {
    #define a vector of `selected` points centered at x.grid[i]
    ###selPoints <- ...
    selPoints <- x>=x.grid[i]-boxcarLength/2 & x <= x.grid[i]+boxcarLength/2
    # calculate and assign mean of y[selPoints]:
    ###movingAverage[i] <- ... #remember to average the y associated with x.
    movingAverage[i] <- mean( y[selPoints], na.rm = na.rm)
  }
  return(list(x=x.grid,y=movingAverage))
}

```

Print out the result as list form

X = each grid point

Y = mean of all y value with x value's location in each grid point

scatterplot smoother function

Grid = 1	X=1,2,3	Mean(c(y[1], y[2] , y[3]))
Grid = 2	X=3,4,5,6	Mean(c(y[3], y[4] , y[5], y[6]))
Grid = 3	X=6,7,8,9,10	Guess ?

See you next week!

Please review R Shiny App
Week 4 and Week 5 materials