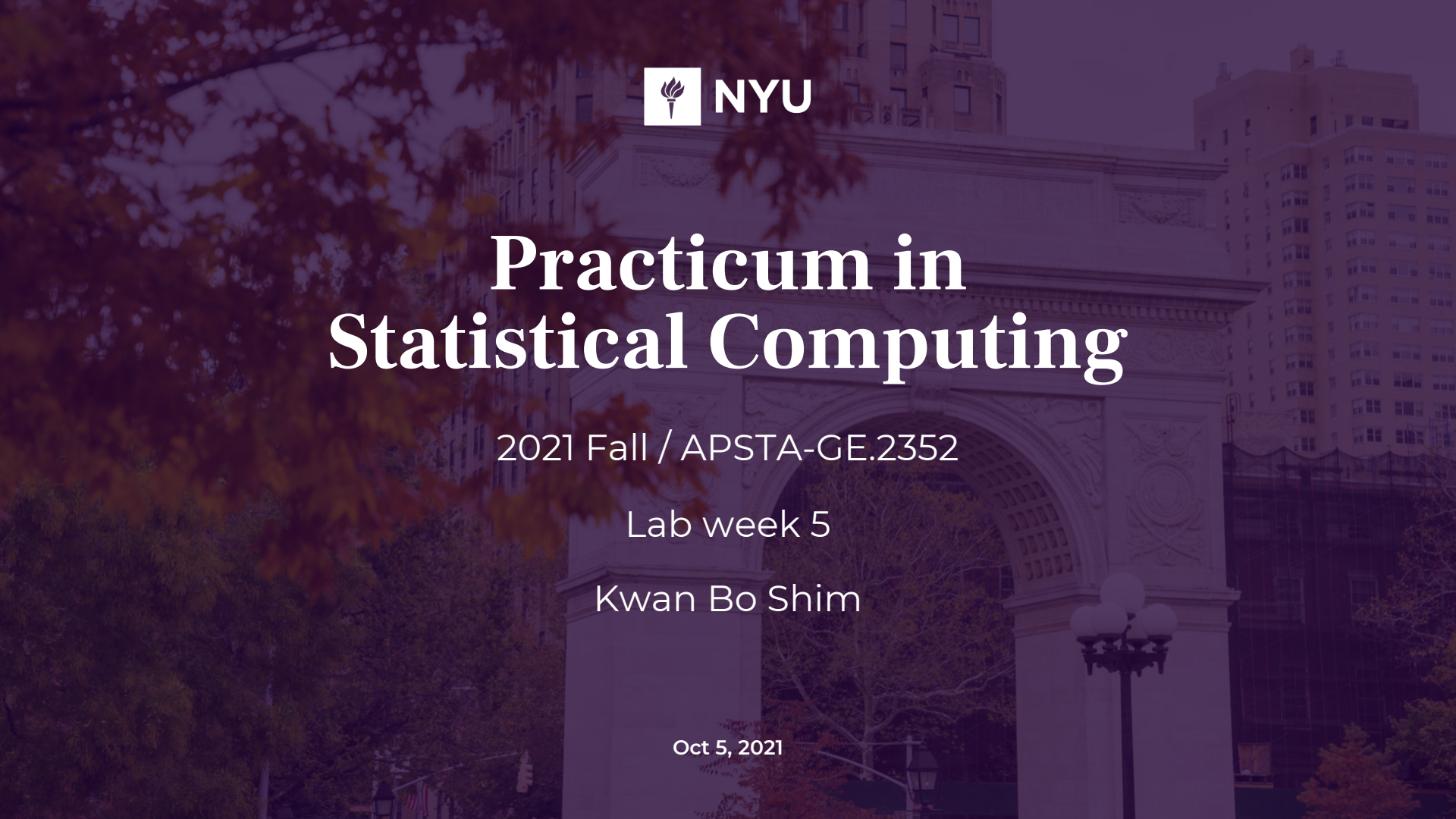


NYU

Practicum in Statistical Computing

2021 Fall / APSTA-GE.2352

Lab week 5

Kwan Bo Shim

Oct 5, 2021

PART 01

Week 5

PART 01

Week 5

1. Poll
2. Attendance
3. Homework / template

NYU

Week 5

- **Smoothing concept**
  - Running medians/means
  - Kernel Smoothing (weighted mean)

Density estimation

1. Nonparametric **density estimation** is an approach to estimating the distribution of the population from which the sample was drawn. The density estimation is nonparametric in the sense that the sample data suggest the shape, rather than imposing the shape of a known population distribution with particular values of its parameters.
2. Nonparametric **kernel density estimation** can be thought of as a method of averaging and smoothing the density estimate that would be provided by a histogram.
3. Changing either the **kernel function or the smoothing parameter** would affect the overall density that is estimated. The former by changing the relative weights and the latter by changing the range of data used in the estimate.

PART 02

Date

Airquality

Ozone <int>	Solar.R <int>	Wind <dbl>	Temp <int>	Month <int>	Day <int>
41	190	7.4	67	5	1
36	118	8.0	72	5	2
12	149	12.6	74	5	3
18	313	11.5	62	5	4
NA	*NA*	14.3	56	5	5
28	*NA*	14.9	66	5	6
23	299	8.6	65	5	7
19	99	13.8	59	5	8
8	19	20.1	61	5	9
NA	194	8.6	69	5	10

Airquality

Ozone <int>	Solar.R <int>	Wind <dbl>	Temp <int>	Month <int>	Day <int>	Date <date>
41	190	7.4	67	5	1	1973-05-01
36	118	8.0	72	5	2	1973-05-02
12	149	12.6	74	5	3	1973-05-03
18	313	11.5	62	5	4	1973-05-04
NA	*NA*	14.3	56	5	5	1973-05-05
28	*NA*	14.9	66	5	6	1973-05-06
23	299	8.6	65	5	7	1973-05-07
19	99	13.8	59	5	8	1973-05-08
8	19	20.1	61	5	9	1973-05-09
NA	194	8.6	69	5	10	1973-05-10

Why?

Airquality

lubridate library

- Lubridate is an R package that makes it easier to work with dates and times.
- Lubridate was created by Garrett Grolemund and Hadley Wickham, and is now maintained by Vitalie Spinu.

ymd("20110604")
#> [1] "2011-06-04"

mdy("06-04-2011")
#> [1] "2011-06-04"

dmy("04/06/2011")
#> [1] "2011-06-04"

```
# download and import lubridate package
install.packages("lubridate")
library(lubridate)
```

Change format of a column as date

```
# 1.
airquality <- within(airquality, {
    Date <- as.Date (paste0("1973-", Month, "-", Day))
})
```

```
# 2.
Airquality$Date <- as.Date (paste0("1973-", Month, "-", Day))
```

PART 03

Smoothing

Airquality

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

How to do **smoothing our data**?
-> moving / rolling mean! (or median)

Moving mean

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

If I want to make an average of 1 neighbors (1 left and 1 right) each ; **k=1**

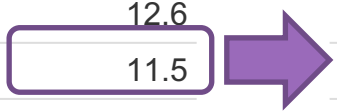
```
for (i in 1:100) {

        mean(data[ (i-1) : (i+1) ] )

}
```

Mean of Neighbors

(1 left and 1 right)

Moving mean

Wind <dbl>	Wind <dbl>
7.4	7.4
8.0	8.0
12.6	12.6
11.5	11.5
14.3	14.3
14.9	14.9
8.6	8.6
13.8	13.8
20.1	20.1
8.6	8.6



When I want to make average of 1 neighbors (1 left and 1 right) each ; **k=1**

```
for (i in 1:100) {

        mean(data[ (i-1) : (i+1) ] )

}
```

Moving mean

Wind <dbl>	Wind <dbl>
7.4	7.4
8.0	8.0
12.6	12.6
11.5	11.5
14.3	14.3
14.9	14.9
8.6	8.6
13.8	13.8
20.1	20.1
8.6	8.6

When I want to make average of 1 neighbors (1 left and 1 right) each ; **k=1**

```
for (i in 1:100) {

        mean(data[ (i-1) : (i+1) ] )

}
```

Moving mean

Wind <dbl>	Wind <dbl>
7.4	7.4
8.0	8.0
12.6	12.6
11.5	11.5
14.3	14.3
14.9	14.9
8.6	8.6
13.8	13.8
20.1	20.1
8.6	8.6

When I want to make average of 2 neighbors (2 left and 2 right) each ; **k=2**

```
for (i in 1:100) {

        mean(data[ (i-2) : (i+2) ] )

}
```

Moving mean

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

When I want to make average of 2 neighbors (2 left and 2 right) each ; **k=3**

```
for (i in 1:100) {

        mean(data[ (i-3) : (i+3) ] )

}
```

PART 04

Ghost neighbors

Moving mean

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

But what if im here?

-> Only right neighbors available

-> getting “ghost” neighbors

e.g.) data[-1], data[-5]... etc

-> **ERROR**

Moving mean

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Again, what if im here?

-> Only left neighbors available

-> getting "ghost" neighbors

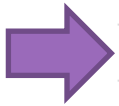
e.g.) data[n+1] = data[n+2] = NA

-> **ERROR**

NYU

Moving mean

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6



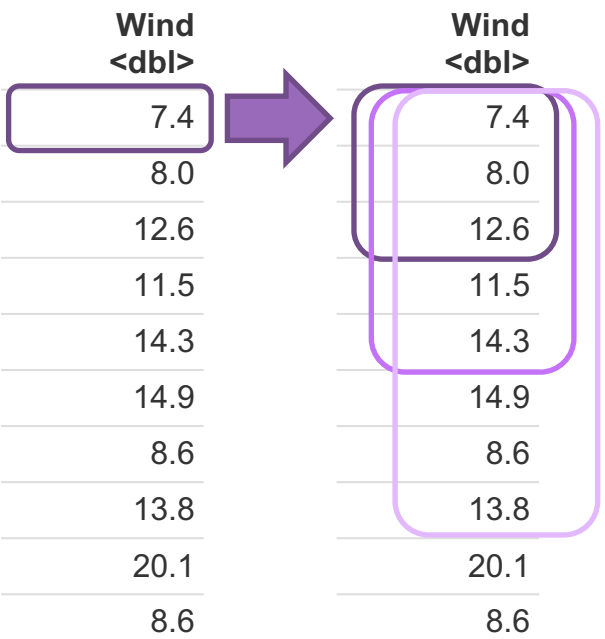
Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Solution: for loop + conditional

```
for (i   in 1 : n) {

  if (i == 1) {

     median( data[(i - 1) : (i + 2)] , na.rm = TRUE)

  } else {

     median( data[(i - 2) : (i + 2)] , na.rm = TRUE)

  }

}
```

NYU

Moving mean



Wind <dbl>	Wind <dbl>
7.4	7.4
8.0	8.0
12.6	12.6
11.5	11.5
14.3	14.3
14.9	14.9
8.6	8.6
13.8	13.8
20.1	20.1
8.6	8.6

What if I want to change k?

Should I change this all the time?

-> generalize the method!

PART 05

Generalization

Moving mean

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Solution: generalization (k=3)

```
for (i in 1:n) {

  if (i - k < 1) {

       lower <- 1

       } else {

       lower <- i – k

     }

  upper <- min(n, i + k)

  saving <- median(data[lower:upper], na.rm = TRUE)

}
```

Moving mean

Solution: generalization (k=3)

```
for (i in 1:n) {
  if (i - k < 1) {
      lower <- 1
      } else {
      lower <- i – k
    }
  upper <- min(n, i + k)
  saving <- median(data[lower:upper], na.rm = TRUE)
}
```

Define **"lower"** -> make sure that it must be <u>at least 1</u> so that no one gets "ghost" neighbors

i=1 -> 1-3 = -2 ☹

i=2 -> 2-3 = -1 ☹

i=3 -> 3-3 = 0 ☹

i=4 -> 4-3 = 1 ☺

Moving mean

Solution: generalization (k=3)

```
for (i in 1:n) {

  if (i - k < 1) {

      lower <- 1

      } else {

      lower <- i – k

    }

  upper <- min(n, i + k)

  saving <- median(data[lower:upper], na.rm = TRUE)

}
```

Define **"Upper"** -> make sure that it must be n at maximum so that no one gets "ghost" neighbor

i=47 -> 47+3 = 50 

i=48 -> 48+3 = 51 

i=49 -> 49+3 = 52 

i=50 -> 50 +3 = 53 

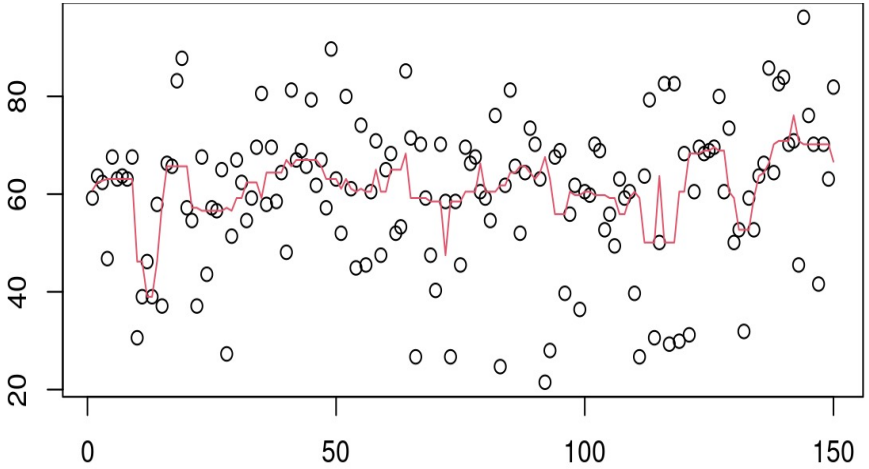
when n=50

Moving mean

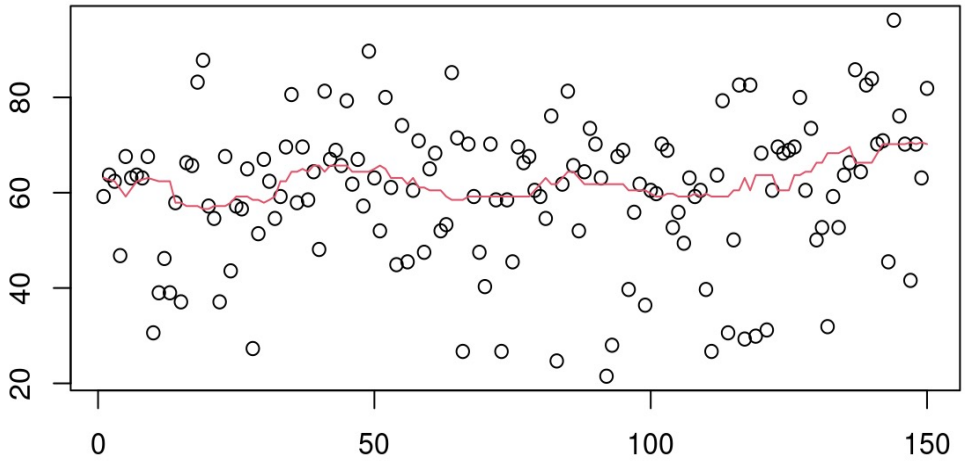
Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Wind <dbl>
7.4
8.0
12.6
11.5
14.3
14.9
8.6
13.8
20.1
8.6

Moving mean



K=3



K=10

NYU

PART 06

Homework 3

Homework 3 review

- Make sure to install sm.package!

- **Reference Band**: A reference band shades the area from and to a certain range, making it easier to identify and understand the values falling into the band.

sm.density(x, h, model = "none", weights = NA, group=NA, ...)

“Model=“ This argument applies only with one-dimensional data. Its default value is "none". If it is set to "Normal” (or indeed any value other than "none") then a reference band, indicating where a density estimate is likely to lie when the data are normally distributed, will be superimposed on any plot.

Reference band with normal distribution assumption looks like ..