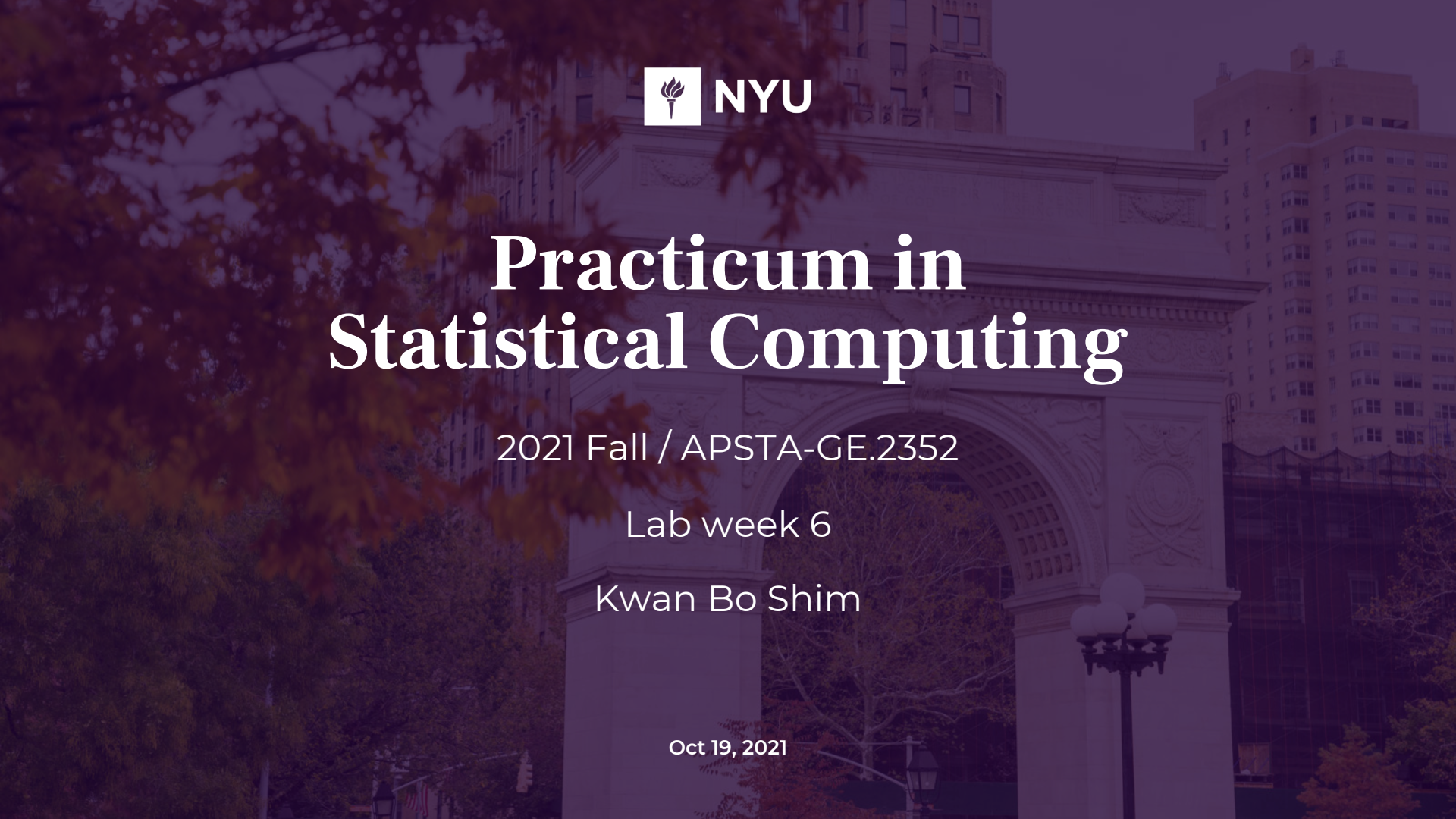

NYU

Practicum in Statistical Computing

2021 Fall / APSTA-GE.2352

Lab week 6

Kwan Bo Shim

Oct 19, 2021

PART 01

Week 6

PART 01

Week 6

1. Poll
2. Attendance
3. Homework (due today)

NYU

Week 6

Apply family

- Apply()
- lappy(), sapply()
- Tapply
- exercise

PART 02

Apply family

Why apply family?

1. We have learned various types of loop.
2. Using loops may take long time to run as they are not optimized for vector operations.
3. we can use “apply” functions, so called “apply family”, which allows more faster and efficient operations, because the actually looping is “internal” to the function and uses compiled code.

PART 03

Apply

1. Apply()

apply(X=data, MARGIN = 1, FUN= mean)

1. X=data : is an array or a matrix
2. MARGIN = 1 : is a type of application (1=row, 2=column)
3. FUN= mean : is a function that we want to apply

PART 04

lapply, sapply

2. lapply() and sapply()

lapply(data, FUN= mean)

1. data : is a matrix or dataframe
2. FUN=mean : is a function that we want to apply

sapply(data, FUN= mean)

1. data : is a matrix or dataframe
2. FUN=mean : is a function that we want to apply

Results

lapply(data, FUN= mean)

```
$winter
[1] 4.142857
$spring
[1] 14.28571
$summer
[1] 27.28571
$autumn
[1] 14.42857
```

sapply(data, FUN= mean)

```
  winter   spring   summer   autumn
4.142857 14.285714 27.285714 14.428571
```

You can use lapply to textual data!

Goal: I want to manipulate this data -> 3 columns (name, email, school)


```
[1] "Harry Potter/harry.p@nyu.edu"
[2] "Hermione Granger/hermione.g@nyu.edu"
[3] "Draco Malfoy/draco.m@nyu.edu"
[4] "Dobby/dobby.d@nyu.edu"
[5] "Fleur Delacour/fleur.d@beauxbatons.edu"
[6] "Luna Lovegood/luna.l@nyu.edu"
[7] "Viktor Krum/viktor.k@dumstrang.edu"
```

You can use lapply to textual data!

We will use **str_replace_all()** function from **stringr** library with some other functions as follows:

```
strsplit(), sub(), substr(), nchar()
```

Install and load library

```
Install.package("stringr")

library(stringr)
```

You can use lapply to textual data!

If we want to simply replace “/” with space

lapply(student_list, FUN = function(x) **str_replace_all**(x, "/", " "))

```
[1] "Harry Potter/harry.p@nyu.edu"
```

```
[1] "Harry Potter harry.p@nyu.edu"
```

You can use lapply to textual data!

Extract name and email address using strsplit() function

name <- lapply(**strsplit**(student_list, split = "/"), FUN = function(x) x[1])
email <- lapply(**strsplit**(student_list, split = "/"), FUN = function(x) x[2])

```
[1] "Harry Potter harry.p@nyu.edu"

-> name
[1] "Harry Potter”
-> email
[1] "harry.p@nyu.edu"
```

NYU

You can use lapply to textual data!

Now, clean up -> extract school's name!

domain <- **sub**(".*@" , "" , email)
-> just removing(=substituting) characters in front of @
school <- **substr**(domain, 1 , **nchar**(domain)-4)
-> removing 4 letters (from the right end) in email section

```
[1] "harry.p@nyu.edu"

-> domain
[1] "nyu.edu"
-> school
[1] "nyu"
```

NYU

You can use lapply to textual data!

Finally, combine all outcomes

cbind(name, email, school)

```
      name            email              school
[1,] "Harry Potter" "harry.p@nyu.edu" "nyu"
[2,] "Hermione Granger" "hermione.g@nyu.edu" "nyu"
[3,] "Draco Malfoy" "draco.m@nyu.edu" "nyu"
[4,] "Dobby" "dobby.d@nyu.edu" "nyu"
[5,] "Fleur Delacour" "fleur.d@beauxbatons.edu" "beauxbatons"
[6,] "Luna Lovegood" "luna.l@nyu.edu" "nyu"
[7,] "Viktor Krum" "viktor.k@dumstrang.edu" "dumstrang"
```

PART 05

tapply

3. tapply()

with(bill, tapply(X = price, INDEX = category, FUN = mean))

- tapply() function performs the operation on each subsets

1. x= is a vector
2. INDEX= is a factor or a list
3. simplify= is an argument to simplify or not.

category <chr>	price <dbl>	location <chr>
snacks	4.0	CVS
drinks	30.0	bar
cleaning	45.0	apt
rent	1800.0	apt
food	15.0	Trader
food	20.0	Trader
food	17.0	Trader
snacks	2.8	Walgreens
snacks	4.0	CVS
drinks	35.0	bar

category <chr>	price <dbl>	location <chr>
snacks	4.0	CVS
drinks	30.0	bar
cleaning	45.0	apt
rent	1800.0	apt
food	15.0	Trader
food	20.0	Trader
food	17.0	Trader
snacks	2.8	Walgreens
snacks	4.0	CVS
drinks	35.0	bar

```
with(bill, tapply(X = price, INDEX = category, FUN = mean) )
```

Output:

```
cleaning drinks food rent snacks
45.00    32.50  29.4 1800 4.95
```

category <chr>	price <dbl>	location <chr>
snacks	4.0	CVS
drinks	30.0	bar
cleaning	45.0	apt
rent	1800.0	apt
food	15.0	Trader
food	20.0	Trader
food	17.0	Trader
snacks	2.8	Walgreens
snacks	4.0	CVS
drinks	35.0	bar

```
with(bill, tapply(X = price, INDEX = location, FUN = mean) )
```

Output:

```
apt   bar  CVS Trader Walgreens
922.5 32.5 4.0 29.4   5.9
```

Comparison (for-loop & apply)

```
# using for loop
months <- unique(airquality$Month)
for (month in months) {
  temp_for_month <- airquality$Temp[airquality$Month == month]
  print( mean(temp_for_month) )
}

# using tapply()
tapply(X = airquality$Temp,
       INDEX = airquality$Month,
       FUN = mean)
```

PART 06

exercise

Consulting project

- You will help client's data analysis project
- Need to send out a small gift to participants (worldwide) depending on their geolocation.
- Company has their own consumer price index (CPI) algorithm

1. Correct(recode) the country's name (remove all NA)
    1. Remove all NA
    2. Hint: USA and Russia are miscoded multiple times -> change all in one name
2. Create column that calculates average CPI value (row-wise)
    1. Average CPI = mean of all listed items
    2. use **apply()** function
3. New Information indicates that there should be regional multipliers
    1. If country is in Europe, 50% increase. (but not higher than 2sd range)
    2. If country is North America, 35% increase (but not higher than 2sd range)
    3. If country is South America, 15% decrease (but not lower than 2sd range)
    4. If country is in Africa, 30% decrease (but not lower than 2sd range)
4. Calculate average expense per student by COUNTRY using **tapply()**
5. Calculate median expense per student by REGION
6. Calculate total expense by REGION (One way is create such as 'expense' which computes total expense per country. Then, sum all those values grouped by region using **tapply)**