Lecture 12: Homework and exam review

STAT598z: Intro. to computing for statistics

Vinayak Rao

Department of Statistics, Purdue University

Homework 2A

In [ ]:

```
trunc_norm <- function(mn, stdv, low, up, num ) {
  trnc <- numeric(num)
  count <- 0
  while(count < num) {
    nrm        <- rnorm(num,mn,stdv)
    valid      <- (nrm > low) & (nrm < up)
    num_valid  <- sum(valid)
    lrange     <- count + 1
    urange     <- count+num_valid

    trnc[lrange:urange] <- nrm[valid]
    count <- urange
  }
  return(trnc[1:num])
}
```

Homework 2B

In [ ]:
```
my_mat <- matrix(rnorm(60000),ncol=6)
my_mat[my_mat < 0] <- 0
prob_mat <- my_mat / rowSums(my_mat)
```

What about rows that are all negative? • Remove NaN s? • Add a small offset? • Remove them?

In [ ]:

```
my_mat <- matrix(rnorm(60000),ncol=6)
my_mat[my_mat < 0] <- 0
my_norm <- rowSums(my_mat)
sel_row <- my_norm > 0
prob_mat <- my_mat[sel_row,] / my_norm[sel_row]
```

In [ ]:
```
ent <- -rowSums(prob_mat * ifelse(prob_mat > 0, log(prob_mat),0))
```

In [ ]:
```
prob_mat[which.max(ent),]
```

Can do it without special functions, but be careful about floats

Homework 3

In [ ]:
```
library('ggplot2')
state_map <- map_data('state'); state_map$InfoValue <- NA
str(state_map)
```

In [ ]:
```
state_info <- as.data.frame(state.x77)
str(state_info)
head(state_info)
```

In [ ]:
```
for(nm in rownames(state_info)) { # Buggy!
  state_map$InfoValue[state_map$region == tolower(nm)] <-
    state_info$Murder[nm]
}
head(state_map)
```

Confusing R quirk: selecting a column with $ loses the rownames!

In [ ]:
```
for(nm in rownames(state_info)) {
  state_map$InfoValue[state_map$region == tolower(nm)] <-
  state_info[nm, 'Murder']
}
```

In [ ]:
```
head(state_map)
```

Were all states considered?

In [ ]:
```
head(state_map[is.na(state_map$InfoValue),])
```

Less efficient, less clean, but still valid

```
state_info <- as.data.frame(state.x77)
rownames(state_info) <-
    tolower(rownames(state_info))
for(rw in 1:nrow(state_map)) {
  rw_state <- state_map$region[rw]
  state_map$InfoValue[rw]
     <- state_info[rw_state, 'Murder' ]
}
```

Were all states covered? (Exercise)

Make functions for repeated operations

In [ ]:
```
AddInfoValue <- function(info_type) {
  state_map <- map_data('state')
  state_map[,c( 'InfoValue', 'InfoType')] <- NA
  for(nm in rownames(state_info)) {
    rw_indx <- state_map$region == tolower(nm)
    state_map$InfoValue[rw_indx] <- state_info[nm,info_type]
    state_map$InfoType[rw_indx]  <- info_type
  }
  state_map <- state_map[!is.na(state_map$InfoValue),]
}
```

In [ ]:
```
info_frame <- rbind(AddInfoValue( 'Life Exp'),
                    AddInfoValue('HS Grad' ))
```

In [ ]:
```
tail(info_frame)
```

In [ ]:

```
ggplot()+geom_map(map = info_frame, map_id=info_frame$region,
        data = info_frame, aes(fill=InfoValue)) +
expand_limits(x = info_frame$long, y = info_frame$lat) +
facet_grid(.~InfoType)
```

We want to add mean latitude and longitude

In [ ]:

```
AddMeanLatLong <- function() {
  state_map  <- map_data('state')
  state_info <- as.data.frame(state.x77)
  state_info[,c('Lat', 'Long')] <- NA
  for(nm in rownames(state_info)) {
    rw_indx <- state_map$region == tolower(nm)
    state_info[nm, 'Lat']  <- mean(state_map$lat[rw_indx])
    state_info[nm, 'Long'] <- mean(state_map$long[rw_indx])
  }
  state_info <- state_info[!is.na(state_info$Lat),]
}
```

In [ ]:

```
p <- ggplot() + geom_map(map = state_map, map_id=state_map$region,
             data = state_map, aes(fill=InfoValue)) +
             expand_limits(x = state_map$long, y = state_map$lat)
info_frame <- AddMeanLatLong()
p1 <- p + geom_point(data = info_frame, aes(x=Long,y=Lat,size=Murder)
)
```

Maybe add one more function?

In [ ]:
```
getMeanLatLong <- function(ip_map, state) {
# Expects lowercase states as input
  rw_indx <- ip_map$region == state
     c(Lat = mean(state_map$lat[rw_indx]),
       Long = mean(state_map$long[rw_indx]))
}
AddMeanLatLong <- function(ip_map) {
  state_info <- as.data.frame(state.x77)
  state_info[,c( 'Lat' , 'Long' )] <- NA
  for(nm in rownames(USArrests))
    state_info[nm, c( 'Lat' , 'Long' )] <-
      getMeanLatLong(ip_map, tolower(nm))
  state_info <- state_info[!is.na(state_info$Lat),]
}
```

apply functions

In the final part I asked you to replace the for loop with sapply

```
my_func <- function(mf_ip) {
# Stuff
}
my_arr <- rnorm(100)
new_arr <- array(0, length(my_arr))
for(i in 1:length(my_arr) {
new_arr[i] <- my_func(my_arr[i]) # For some function my_func
}
```

Cleaner:

```
new_arr <- sapply(my_arr, my_func)
```

apply functions

Note: *apply functions do not vectorize loops

They just hide them for convenience, clarity and to avoid bugs

When possible, vectorize code:

Bad:

```
my_arr <- 1:10
sapply(my_arr, sqrt)
```

Good:

```
my_arr <- 1:10
sqrt(my_arr) # Cleaner and faster
```

`lapply`, `sapply`, `vapply` etc: hard to remember which does what

`plyr` package provides simpler interface