Parallelization

Breezing through the tidyverse

Introduction

- This module goes over some advanced topics that may help with simulation.
- You will only master with much practice.
- This module should provide a foundation to know when you may want to attempt these methods but you will not master it by the end of this module.

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

What is Parallelization?

Parallelization is the process of splitting a task into multiple smaller tasks that can be processed simultaneously.

- Exploits multiple CPU cores
- Ideal for repetitive tasks like simulations

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Libraries to Know

For parallel execution in R:

- doParallel: backend for foreach
- foreach: loop construct for parallelization

```
library(tidyverse)
# install.packages(c("doParallel", "foreach"))
library(doParallel)
library(foreach)
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Sequential vs Parallel

Sequential (*%do%*): - Processes one task after the other.

- Like *for()* but returns a list at the end.

Parallel (*%dopar%*): - Processes multiple tasks at once.

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization

Breezing through the Tidyverse

Simulating the NBA Season

Given game win probabilities, we'll simulate each game to determine a winner.

We will use the same data from last module's quiz: the 2022 NBA Season

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization

Breezing through the tidyverse

Simulating the NBA Season

We simulate the season as we did before, not in parallel. We do 1,000 simulations (for time and brevity).

```
nba_schedule <- read_rds("nba_2022_season_schedule.rds")

# Number of simulations
n.iter <- 1000

# Number of games per season
n.games <- nrow(nba_schedule)/2

# Calculate who won each game
nba_schedule <- nba_schedule %>%
  mutate(team_win = ifelse(team_score > opp_score, 1, 0) )

# Create simulations

nba_schedule_simulations <- nba_schedule %>%
  slice(rep(1:n(), each = n.iter)) %>%
  mutate(sim = rep(1:n.iter, times = n.games*2) ) %>%
  ungroup()
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization

Breezing through the tidyverse

Simulating the NBA Season

We simulate the season as we did before, not in parallel. We do 1,000 simulations (for time and brevity).

```
# simulate out the season
set.seed(100)
nba_schedule_simulations <- nba_schedule_simulations %>%
  mutate(sim_team_win = rbinom(n(), size = 1, team_wp))

#since we had two rows per game, we only take the sim of the home team..
## then force the other one to match
nba_schedule_simulations <- nba_schedule_simulations %>%
  group_by(sim,
           game_id) %>%
  mutate(sim_team_win = ifelse(team_home == 1,
                               sim_team_win,
                               1 - sim_team_win[team_home == 1])
         ) %>%
  ungroup()
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Playoff Seeding

- After simulating, teams are ranked based on wins:
  - Top 8 from each conference go to the playoffs (now play-in round).
  - Two-way ties are broken with head-to-head records.

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Tiebreakers

If two teams have equal wins:

1. Head-to-head record
2. Division winner
3. Conference record … (follow NBA's tiebreaker rules)

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Playoff Seeding

We can get playoff seeding before breaking ties with *min_rank()*.

```
nba_sim_records <- nba_schedule_simulations %>%
  group_by(sim,
           team_display_name,
           team_conference) %>%
  summarize(wins = sum(sim_team_win),
            losses = sum(1-sim_team_win),
            games = n()
            ) %>%
  mutate(win_pct = wins / games) %>%
  ungroup()
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Playoff Seeding

We can get playoff seeding before breaking ties with *min_rank()*.

```
nba_sim_records <- nba_sim_records %>%
  group_by(sim, team_conference) %>%
  mutate(playoff_seed = min_rank(desc(win_pct))) %>%
  ungroup() %>%
  arrange(sim,
          team_conference,
          playoff_seed)

nba_sim_records %>% head(10)
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization

Breezing through the tidyverse

Playoff Seeding

We can get playoff seeding before breaking ties with *min_rank()*.

```
# A tibble: 10 × 8
     sim team_display_name   team_conference  wins losses games win_pct
   <int> <glue>              <chr>           <dbl>  <dbl> <int>   <dbl>
 1     1 Miami Heat          East               60     23    83   0.723
 2     1 Milwaukee Bucks     East               58     24    82   0.707
 3     1 Atlanta Hawks       East               53     30    83   0.639
 4     1 Boston Celtics      East               52     30    82   0.634
 5     1 Toronto Raptors     East               50     35    85   0.588
 6     1 Brooklyn Nets       East               49     36    85   0.576
 7     1 Cleveland Cavaliers East               46     37    83   0.554
 8     1 New York Knicks     East               44     38    82   0.537
 9     1 Philadelphia 76ers  East               44     39    83   0.530
10     1 Charlotte Hornets   East               43     39    82   0.524
   playoff_seed
          <int>
 1            1
 2            2
 3            3
 4            4
 5            5
 6            6
 7            7
 8            8
 9            9
10           10
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Playoff Seeding

- Tiebreakers are first determined by head-to-head record (if the tie involves 2 teams).
- This has to be done individually in each simulation.
- We will look at just 1 simulation and build a function that breaks head-to-head ties and then randomly breaks the rest after that.
- You can extend the code to follow the rest of the NBA's tiebreakers.

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Tiebreaker Function

We need both the current standings & the simulation standings.

```
sim_1_records <- nba_sim_records %>%
  filter(sim == 2,
         team_conference == "West")
sim_1_games <- nba_schedule_simulations %>%
  filter(sim == 2,
         team_conference == "West")
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization

Breezing through the tidyverse

Tiebreaker Function

```
#extract out tied teams
tied_teams <- sim_1_records %>%
  group_by(sim,
           team_conference,
           playoff_seed) %>%
  filter(n() > 1) %>%
  ungroup()

tied_teams
```

```
# A tibble: 2 × 8
    sim team_display_name team_conference  wins losses games win_pct
  <int> <glue>           <chr>           <dbl>  <dbl> <int>   <dbl>
1     2 Dallas Mavericks West               57     25    82   0.695
2     2 Phoenix Suns     West               57     25    82   0.695
  playoff_seed
         <int>
1            1
2            1
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization

Breezing through the tidyverse

Tiebreaker Function

```
# create unique key of variables that defines each tie
tied_teams_key <- tied_teams %>%
  group_by(sim,
           team_conference,
           playoff_seed) %>%
  summarise(teams = stringr::str_c(team_display_name, collapse = ", ")) %>%
  ungroup()

tied_teams_key
```

```
# A tibble: 1 × 4
    sim team_conference playoff_seed teams
  <int> <chr>                  <int> <chr>
1     2 West                       1 Dallas Mavericks, Phoenix Suns
```

Paul Sabin
Senior Fellow, Wharton
ex ESPN & SumerSports

Now we pull out all games between the teams. If you don't follow all the code, that isn't as neccessary for this exercise.

```
n_tiebreaks <- nrow(tied_teams_key)

head_2_head_results <- NULL
for(i in 1:length(n_tiebreaks)){
  head_2_head_results <- sim_1_games %>%
    filter(sim == tied_teams_key$sim[i],
           team_conference == tied_teams_key$team_conference[i],
           #if any team is in the list of teams, return those games
           str_detect(tied_teams_key$teams[i], team_display_name),
           str_detect(tied_teams_key$teams[i], opp_display_name)

           ) %>%
    ## now return the number of wins for each of them
    group_by(team_display_name,
             opp_display_name,
             sim,
             team_conference) %>%
    summarize(wins = sum(sim_team_win),
              losses = sum(1-sim_team_win),
              win_pct = sum(sim_team_win) / n() )%>%
    ungroup() %>%
    mutate(playoff_seed = tied_teams_key$playoff_seed[i]) %>%
    # stack values with previous tied teams
    bind_rows(head_2_head_results, .)

}


head_2_head_results
```

Parallelization

Breezing through the tidyverse

Tiebreaker Function

Now we pull out all games between the teams. If you don't follow all the code, that isn't as necessary for this exercise.

```
# A tibble: 2 × 8
  team_display_name opp_display_name   sim team_conference  wins losses win_pct
  <glue>            <glue>           <int> <chr>           <dbl>  <dbl>   <dbl>
1 Dallas Mavericks  Phoenix Suns         2 West                3      0       1
2 Phoenix Suns      Dallas Mavericks     2 West                0      3       0
  playoff_seed
         <int>
1            1
2            1
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Tiebreaker Function

Now we break the ties

```
head_2_head_results %>%
  group_by(sim,
           team_conference,
           playoff_seed) %>%
  mutate(h2h_rank = min_rank(desc(win_pct)),
         new_playoff_seed = playoff_seed + (h2h_rank - 1) )
```

```
# A tibble: 2 × 10
# Groups:   sim, team_conference, playoff_seed [1]
  team_display_name opp_display_name   sim team_conference  wins losses win_pct
  <glue>            <glue>           <int> <chr>           <dbl>  <dbl>   <dbl>
1 Dallas Mavericks  Phoenix Suns         2 West                3      0       1
2 Phoenix Suns      Dallas Mavericks     2 West                0      3       0
  playoff_seed h2h_rank new_playoff_seed
         <int>    <int>            <dbl>
1            1        1                1
2            1        2                2
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
BREEZING THROUGH THE TIDYVERSE

Tiebreaker Function

Now we break the ties

```
head_2_head_results
```

```
# A tibble: 2 × 8
  team_display_name opp_display_name   sim team_conference  wins losses win_pct
  <glue>            <glue>           <int> <chr>           <dbl>  <dbl>   <dbl>
1 Dallas Mavericks  Phoenix Suns         2 West                3      0       1
2 Phoenix Suns      Dallas Mavericks     2 West                0      3       0
  playoff_seed
         <int>
1            1
2            1
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing Through the Tidyverse

Tiebreaker Function

- For our purposes we will then break any further ties by a "coin flip."
- We will then add the new playoff seeds back and create a function that does all this.
- See code file for complete function code.

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing Through the Tidyverse

Tiebreaker Function

```
break_ties <- function(game_results,
                       season_standings){⟷}
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization

Breezing through the tidyverse

Parallel Simulation

Parallel simulation can be helpful if:

- the processes don't depend on one another (sequentially or otherwise),
- you can't vectorize it, or
- the savings of running multiple process at once outweighs the overhead.

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing Through the Tidyverse

Setting Up Parallel Backend

Before using %dopar%, you need to set up a parallel backend:

```
#how many cores your computer has
n_cores <- detectCores()
#register that many cores (or pick smaller number)
cl <- makeCluster(n_cores)
registerDoParallel(cl)
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Parallel Simulation

Using %dopar% for parallelized simulation.

- You need to pass through each package (via .packages) you need to run the code.
- You may also need to export which objects need to be passed into parallel.

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization

Breezing through the tidyverse

Parallel Simulation

```
tic()
sim_standings <- foreach(i = 1:n.iter, #i = icount(n.iter),
                         .inorder = TRUE,
                  .combine = 'rbind',
                  .packages = c("tidyverse"),
                  .export = c("nba_schedule_simulations",
                              "nba_sim_records")
                  ) %dopar% {

    # select season game simulation
    this_season_sim_games <- nba_schedule_simulations %>% filter(sim == i)
    this_season_sim_records <- nba_sim_records %>% filter(sim == i)

    #now apply function to break ties in this simulation
    new_this_season_sim_records <- break_ties(game_results = this_season_sim_games,
                                              season_standings = this_season_sim_records)

    #now leave the dataframe we want to combine at the end to get the simulated results
    new_this_season_sim_records
                  }

toc()
```

```
52.211 sec elapsed
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the Tidyverse

Stopping the Cluster

After parallel operations, stop the cluster:

```
stopCluster(cl)
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Stopping the Cluster

Now we can look at the results of the tiebreaking in the simulation.

```
head(sim_standings, n = 10)
```

```
# A tibble: 10 × 8
     sim team_display_name   team_conference  wins losses games win_pct
   <int> <glue>             <chr>           <dbl>  <dbl> <int>   <dbl>
 1     1 Miami Heat         East               60     23    83   0.723
 2     1 Milwaukee Bucks    East               58     24    82   0.707
 3     1 Atlanta Hawks      East               53     30    83   0.639
 4     1 Boston Celtics     East               52     30    82   0.634
 5     1 Toronto Raptors    East               50     35    85   0.588
 6     1 Brooklyn Nets      East               49     36    85   0.576
 7     1 Cleveland Cavaliers East              46     37    83   0.554
 8     1 New York Knicks    East               44     38    82   0.537
 9     1 Philadelphia 76ers East               44     39    83   0.530
10     1 Charlotte Hornets  East               43     39    82   0.524
   playoff_seed
          <dbl>
 1            1
 2            2
 3            3
 4            4
 5            5
 6            6
 7            7
 8            8
 9            9
10           10
```

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Benefits of Parallelization

- Speeds up computation time
- Essential for large-scale simulations
- Utilizes computer resources efficiently

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports

Parallelization
Breezing through the tidyverse

Recap and Further Reading

- Parallelization for efficient simulations
- foreach, %dopar%, and %do% for loops
- Always test parallel code outside parallelization for correctness

Further reading: Parallel computing in R guide

Paul Sabin
Senior Fellow, Wharton
Ex ESPN & SumerSports