TOUR SCHEDULER

OCAML SPRING 2019

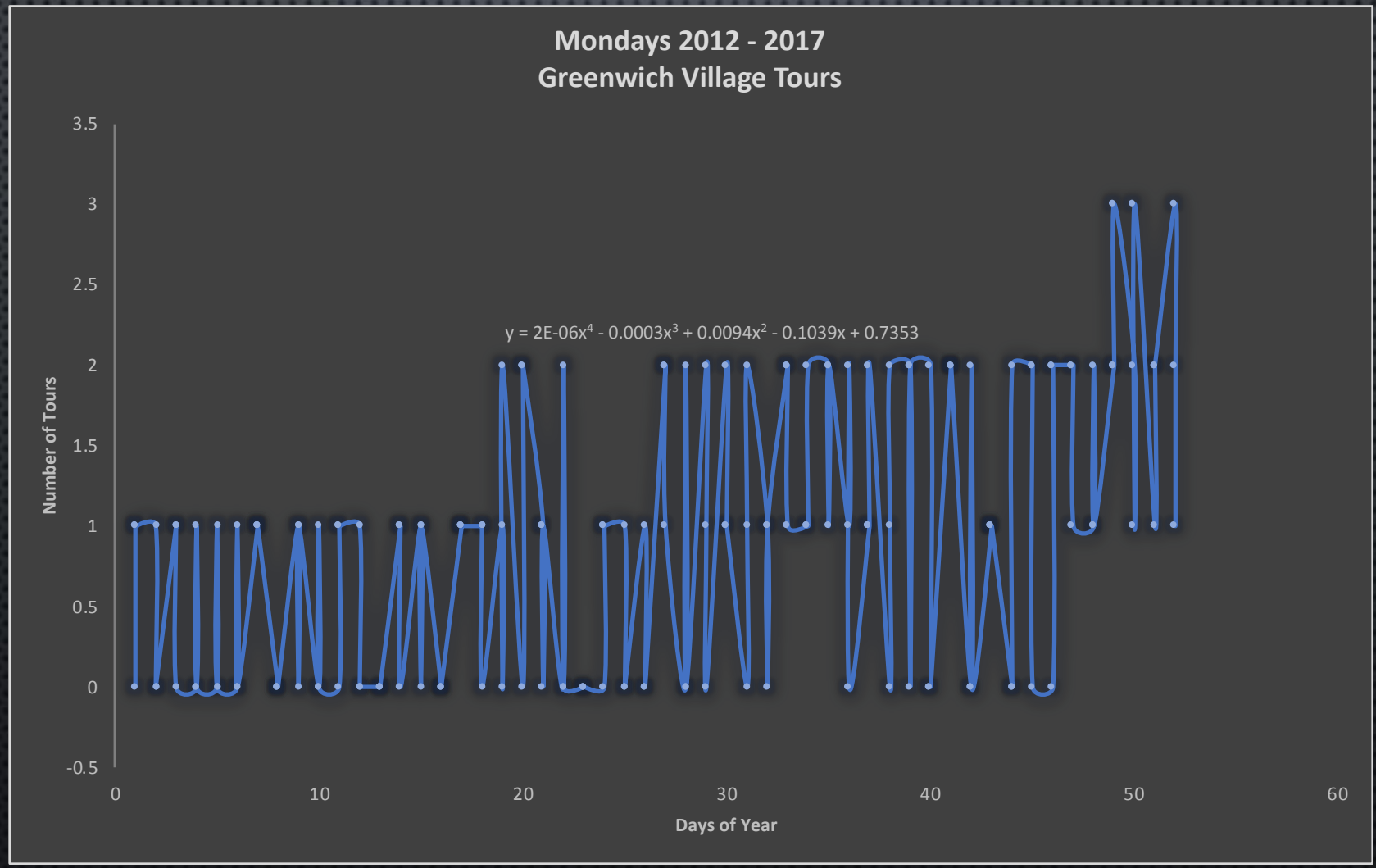
GATHER DATA

MONDAYS IN GREENWICH VILLAGE

- 3 YEARS OF PREVIOUS DATA
- NEIGHBORHOOD
- WEEK OF YEAR
- NUMBER OF TOURS THAT WEEK

Neighborhood	Week of Year	Num Tours
Greenwich	1	0
Greenwich	1	1
Greenwich	1	1
Greenwich	2	1
Greenwich	2	0
Greenwich	2	0
Greenwich	3	1
Greenwich	3	1
Greenwich	3	0
Greenwich	4	0
Greenwich	4	1
Greenwich	4	0
Greenwich	5	0
Greenwich	5	1
Greenwich	5	0
Greenwich	6	0
Greenwich	6	1
Greenwich	6	0
Greenwich	7	1
Greenwich	7	1

CALCULATE TRENDLINE



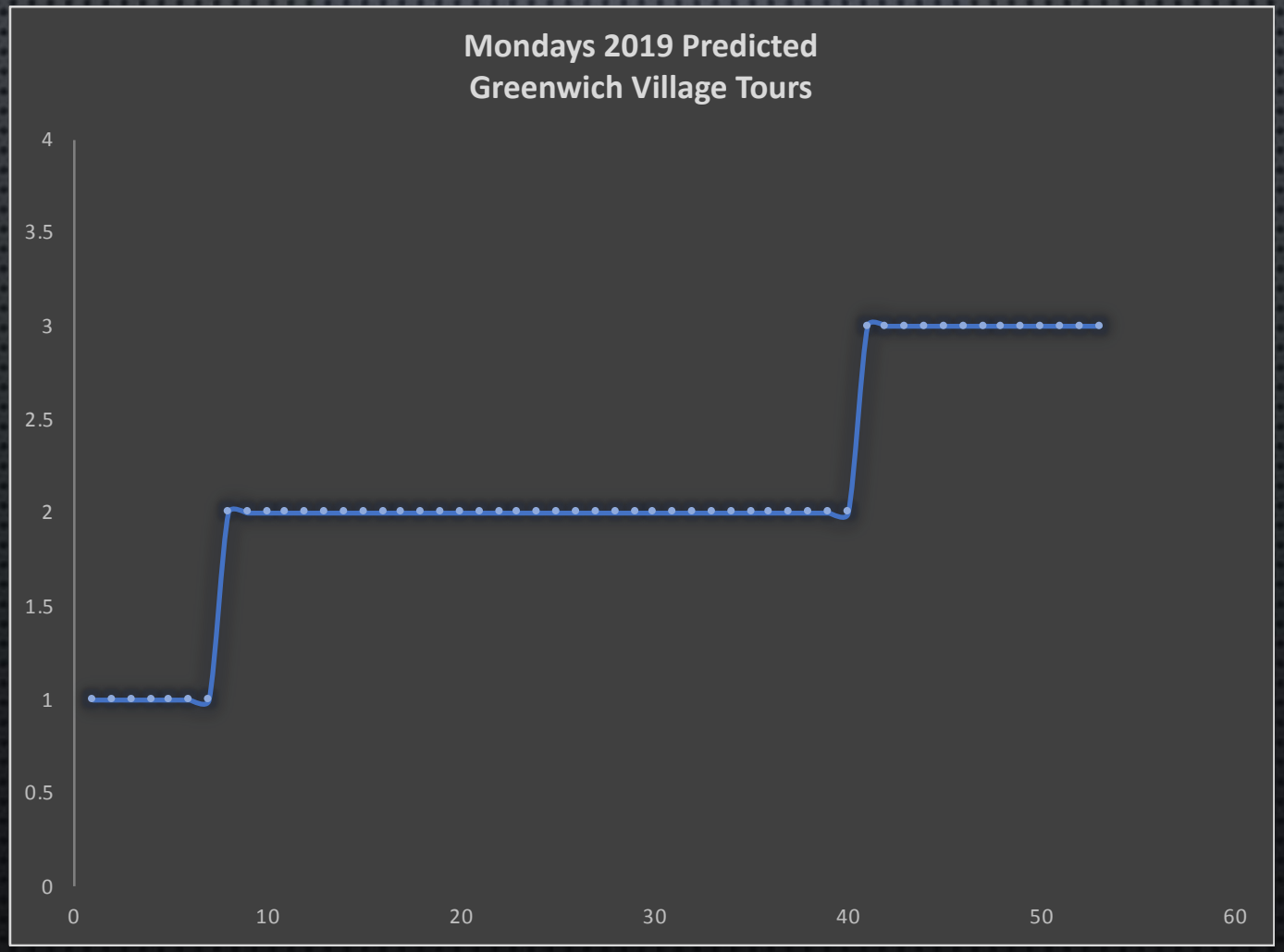
CALCULATE PREDICTED DATA

- AFTER FINDING BEST R^2 VALUE
- USE TRENDLINE TO PREDICT DATA
- CONVERT FROM WEEK NUMBER TO DATE
- CALCULATE NUMBER (ADD 1.5 TO OVERPREDICT)

```
let schedule_helper formula num_weeks =
  let rec aux formula num_weeks curr_week =
  match num_weeks with
  | 0 -> print_endline "finished"
  | _ -> print_int (int_of_float (1.5 +. (formula curr_week)));
      print_endline "";
      aux formula (num_weeks - 1) (curr_week +. 1.0)
  in
  aux formula num_weeks 1.0
```

neighborhood	date	time
Greenwich	1/7/2019	1
Greenwich	1/14/2019	1
Greenwich	1/21/2019	1
Greenwich	1/28/2019	1
Greenwich	2/4/2019	1
Greenwich	2/11/2019	1
Greenwich	2/18/2019	1
Greenwich	2/25/2019	2
Greenwich	3/4/2019	2
Greenwich	3/11/2019	2
Greenwich	3/18/2019	2
Greenwich	3/25/2019	2
Greenwich	4/1/2019	2
Greenwich	4/8/2019	2
Greenwich	4/15/2019	2
Greenwich	4/22/2019	2
Greenwich	4/29/2019	2
Greenwich	5/6/2019	2
Greenwich	5/13/2019	2
Greenwich	5/20/2019	2
Greenwich	5/27/2019	2
Greenwich	6/3/2019	2
Greenwich	6/10/2019	2

PREDICTED VALUES



SCHEDULING PT1

- PARSE LINE FROM SCHEDULE CSV
- PARSE LINE FROM GUIDES CSV

```
(* splits a line by supplied delimiter *)
let parse_to_list delim line =
    Str.split (Str.regexp delim) line

(* returns current line of file *)
let get_line ic =
  try
    Some (input_line ic)
  with
  | End_of_file -> None

(* uses supplied function to parse according to the function *)
let parse_line_to_list start_from my_fun filename =
  let ic = open_in filename in
  let rec read acc i =
    match get_line ic with
    | Some line -> if i >= start_from then
      let ls = (parse_to_list "," (String.trim line)) in
      read ((my_fun ls) :: acc) (succ i)
      else read acc (succ i)
    | None ->
        close_in ic; (* close input channel *)
        List.rev acc (* return parsed list *)
  in
  read [] 0
```

SCHEDULING PT2

- PARSE EACH LINE INTO RECORD

Guide	Priority Number	Days Off	Days Off Time	Set Day/Time	Max tours	Duties
Bert	1	Sunday;1/20/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Stephen	2	1/21/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Cindy	3	1/22/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Curt	4	1/23/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Paulette	5	1/24/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Chip	6	1/25/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Colleen	7	1/26/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Darrel	8	2/20/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
David	9	2/21/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Heather	10	2/22/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Jackie	11	2/23/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Joanna	12	2/24/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Joe	13	2/25/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Jordan	14	2/26/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Marie	15	2/20/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Mason	16	2/21/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Raheem	17	2/22/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
RJ	18	2/23/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Robin	19	2/24/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
RobinS	20	2/25/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Ryan	21	2/26/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita
Ted	22	3/27/2019	ALL	Saturday-1100;Tuesday-1200	2	Greenwich;Chinatown;Nolita

```
(**************************************************************************
   Record setting helpers
**************************************************************************)

let make_schedule_from_list ls = {
  neighborhood = List.nth ls 0;
  date = List.nth ls 1;
  time = (List.nth ls 2);
  day_of_week = List.nth ls 3;
}

let make_employee_from_list ls = {
  name = List.nth ls 0;
  priority = int_of_string (List.nth ls 1);
  days_off = parse_to_list ";" (List.nth ls 2);
  days_off_times = parse_to_list ";" (List.nth ls 3);
  set_schedule = parse_to_list ";" (List.nth ls 4);
  max_tours = (List.nth ls 5);
  duties = parse_to_list ";" (List.nth ls 6);
  weekly_tour_count = 0
}
```

MAIN SCHEDULING ALGORITHM

1. KEEP TRACK WHEN WEEK ENDS
2. GETS NEXT AVAILABLE DAYS SCHEDULE
    1. LOOPS UNTIL THE CURRENT DAY != SCHEDULE DAY
3. GETS REST OF AVAILABLE SCHEDULE
4. MAKES DAILY SCHEDULE
    1. CALCULATES CONFLICTS, IF NONE, PAIR EMP TO DATE
    2. ADD TO WEEKLY SCHEDULE COUNT FOR EMPLOYEE
5. IF STILL IN SAME WEEK CONTINUE SCHEDULE
6. ELSE RESET SCHEDULE COUNT AND UPDATE WEEK END

```
(* takes in an employee record and schedule record and schedules based on availablility *)
let make_daily_schedule employees schedule =
  let rec aux empls sched acc =
    match sched with
    | [] -> employee_sort compare_emp_priority (List.append empls acc)
    | hd::tl ->
      let x = get_next_available_employee_at_head empls hd in
      let emp = {(List.hd x) with weekly_tour_count = (List.hd x).weekly_tour_count + 1} in
      print_schedule_line (List.hd x) hd;
      aux (List.tl x) (List.tl sched) (List.cons emp acc)
  in
  aux employees schedule []

let make_schedule employees schedule =
  let end_of_week = add (date_of_string (List.hd schedule).date) (Period.day 7) in

  let rec aux emps sched eow =
    match sched with
    | [] -> print_endline "All set"
    | _ ->
      let curr_day_sched = get_next_days_schedule sched in
      let rest_of_sched = get_rest_of_schedule sched in
      let curr_day_date = date_of_string (List.hd curr_day_sched).date in
      let emps = make_daily_schedule emps curr_day_sched in
      if (compare curr_day_date eow) >= 0 then
        let eow = add curr_day_date (Period.day 7) in
        let emps = employees in
        aux emps rest_of_sched eow
      else
      aux emps rest_of_sched eow
  in
  aux employees schedule end_of_week
```

FINAL OUTPUT

TODO:

- Check if too many employees at time of year
- Add interface
- Embed regression analysis into program

Name	Date	Time
Bert	1/7/19	1100
Bert	1/8/19	1100
Bert	1/9/19	1100
Bert	1/10/19	1100
Bert	1/11/19	1100
Bert	1/12/19	1100
Chip	1/13/19	1230
Cindy	1/10/19	1200
Cindy	1/11/19	1200
Cindy	1/12/19	1200
Cindy	1/13/19	1100
Cindy	1/14/19	1100
Colleen	1/13/19	1300
Curt	1/11/19	1230
Curt	1/12/19	1230
Curt	1/13/19	1130
Darrel	1/13/19	1330
David	1/13/19	1400
Paulette	1/12/19	1300
Paulette	1/13/19	1200
Stephen	1/7/19	1130
Stephen	1/8/19	1130
Stephen	1/9/19	1130
Stephen	1/10/19	1130
Stephen	1/11/19	1130
Stephen	1/12/19	1130