HW02 - Solutions

Stat 133, Spring 2018, Prof. Sanchez

Slef-grade due date: Tue Mar-13 (before midnight)

General Self-Grading Instructions

- Please read this document (answer key).
- You will have to enter your score and comments in the *Assignments Comments* section of the correspoing assignment on bCourses.
- Enter your own scores and comments for (every part) of every problem in the homework on a simple coarse scale:
  - **0** = Didn't attempt or very very wrong,
  - **2** = Got started and made some progress, but went off in the wrong direction or with no clear direction,
  - **5** = Right direction and got half-way there,
  - **8** = Mostly right but a minor thing missing or wrong,
  - **10** = 100% correct.
- Also, enter your total score (out of an overall score of 110 points).
- If there is a cascading error, use a single deduction (don't double or triple or multiple penalize).
- Note: You must justify every self-grade score with a comment. If you are really confused about how to grade a particular problem, you should post on Piazza. This is not supposed to be a stressful process.
- Your self-grades will be due four days after the homework deadline at 11:59 PM sharp (i.e Tue Mar-13).
- We will accept late self-grades up to a week after the original homework deadline for half credit on the associated homework assignment.
- If you don't enter a proper grade by this deadline, you are giving yourself a zero on that assignment.
- Merely doing the homework is not enough, you must do the homework; turn it in on time; read the solutions; do the self-grade; and turn it in on time. Unless all of these steps are done, you will get a zero for that assignment.

File Structure (10 pts)

After completing this assignment, the file structure of your project should look like this. Notice that `first` and `last` correspond to your first and last names):

```
hw02/
  README.md
  data/
    andre-iguodala.csv
    draymond-green.csv
    kevin-durant.csv
    klay-thompson.csv
    stephen-curry.csv
    shots-data.csv
    data-dictionary.md
  code/
    make-shots-data-script.R
    make-shot-charts-script.R
  output/
    shots-data-summary.txt
    ... # other summary.txt files
  images/
    nba-court.jpg
    ... # pdf images of shot charts
    ... # png images (of knitted Rmd)
  report/
    hw02-first-last.Rmd
    hw02-first-last.md
```

It is possible that your directory `hw02/` contains hidden files such as `.Rhistory`, `.RData`, or `.DS_Store`. No need to worry about these extra files.

The `README.md` file should have a description of what HW02 is about.

In genereal, you should write content in an `README.md` in such a way that when you (or another reader) look at the folder `hw02`, it helps you remember what the assignment is about. Simply put: think of `README.md` as a *post-it* note to your future self.

2.2) Data Dictionary (10 pts)

Create a data dictionary—using markdown syntax—in a separate text file: `data-dictionary.md`. Include names of the variables, and a short description.

Your `data/` folder shoud include a data dictionary file containing the name and description of the variables of the original CSV files.

Optionally, your dictionary could also include names of additional variables added to the `shots-data.csv`.

3) Data Preparation

All the R code to complete the data preparation stage must be written in an `.R` script file. Name the `R` script file as `make-shots-data-script.R` and save it inside the `code/` folder.

Data Prep R script (10 pts)

- The code for the data preparation must be in an R script file (NOT in an Rmd file).
- The script must conatin a header with at least the following items:
  - title
  - description
  - input(s)
  - outpus(s)
- The five original CSV data files have to imported with **relative filepaths**; you can use either `read.csv()` or `read_csv()`
- All sinking operations must be done with **relative paths**.

If you specified a working directory with an absolute path (see example below), you must deduct 2 points. Recall that **absolute paths break computational reproducibility**!

```
# example of absolute path
setwd("~/Desktop/hw-stat133/hw02/report")
```

Exported Data Summaries (10 pts)

- Your `output/` folder should have text files with the output of `summary()` from each original data table, as well as the assembled table:
  - `andre-iguodala-summary.txt`
  - `draymond-green-summary.txt`
  - `kevin-durant-summary.txt`
  - `klay-thompson-summary.txt`
  - `stephen-curry-summary.txt`
  - `shots-data-summary.txt`

4) Shot Charts

This part of the assignment has to do with the creation of shot charts. Write the code in an R script called `make-shot-charts-script.R`, and save it in the `code/` folder. Make sure you include a header with fields about title, description, inputs, and outputs.

- The code for creation of shot charts must be in an R script file (NOT in an Rmd file).
- The script `make-shot-charts-script.R` must conatin a header with items such as:
  - title
  - description
  - input(s)
  - outpus(s)
- The creation of the images in PDF format must be done using **relative filepaths**; you can use either `ggsave()` or `pdf()` and `dev.off()`.

4.1) Shot charts of each player (10 pts)

Create shot charts (with court backgrounds) for each player, and save the plots in PDF format, with dimensions `width = 6.5` and `height = 5` inches, inside the folder `images/`:

- `andre-iguodala-shot-chart.pdf`
- `draymond-green-shot-chart.pdf`
- `kevin-durant-shot-chart.pdf`
- `klay-thompson-shot-chart.pdf`
- `stephen-curry-shot-chart.pdf`

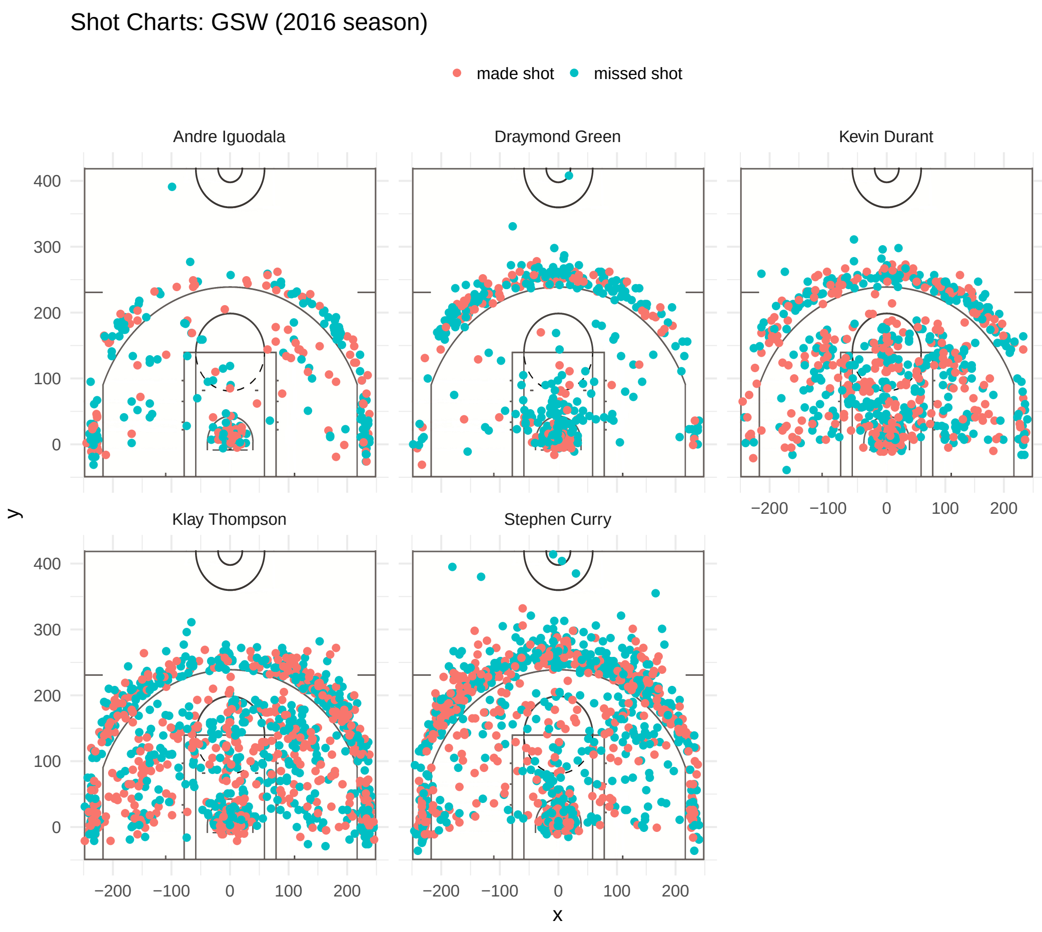
Here's a sample code for the shot chart of Stephen Curry. Notice that an object of class `"ggplot"` is being created. And then we can refer to this object in the exporting call to `ggsave()`. Alternatively, you could also use `pdf()` and `dev.off()`.

```r
# ===============================================
# Stephen Curry's shot chart
# ===============================================
stephen_shot_chart <- ggplot(data = filter(dat, name == "Stephen Curry")) +
  annotation_custom(court_image, -250, 250, -50, 420) +
  geom_point(aes(x = x, y = y, color = shot_made_flag)) +
  ylim(-50, 420) +
  ggtitle('Shot Chart: Stephen Curry (2016 season)') +
  theme_minimal()

# export plot as PDF image
ggsave(
  filename = '../images/stephen-curry-shot-chart.pdf',
  plot = stephen_shot_chart,
  width = 6.5,
  height = 5)
```

4.2) Facetted Shot Chart (10 pts)

Create one graph, using facetting, to show all the shot charts in one image, similar to the one below. Save this image in PDF format as `gsw-shot-charts.pdf`, inside the folder `images/`. Specify image dimensions `width = 8` and `height = 7` inches.

```r
# shot-charts of all five players using facetting
gsw_shot_chart <- ggplot(data = dat) +
  annotation_custom(court_image, -250, 250, -50, 420) +
  geom_point(aes(x = x, y = y, color = shot_made_flag)) +
  ylim(-50, 420) +
  ggtitle('Shot Charts: GSW (2016 season)') +
  facet_wrap(~ name) +
  theme_minimal() +
  theme(legend.position = "top",
        legend.title = element_blank())

ggsave(
  filename = 'gsw-shot-facets.pdf',
  plot = gsw_shot_chart,
  width = 8,
  height = 7)
```



5) Summary Tables (20 pts)

This and subsequent parts of the assignment should be included in your `Rmd` file (in the `report/` subdirectory). In your `Rmd` file include a global chunk option to specify the location of plots and graphics.

5.1) Total Shots by Player (10 pts)

Total number of shots (2PT and 3PT, both made and missed) by player, arranged in descending order.

The code below is one way to obtain the required table. Of course, your solution will probably be different. The important part is to have the same output, and to use `"dplyr"` functions. By the way, `merge()` is NOT a `dplyr` function; the dplyr merge is `join()`; so deduct 2 pts if you used `merge()` or any other non-dplyr functions.

```r
# Total number of shots (2PT & 3PT, made & missed) by player
# (in descending order)
total_shots_by_player <- dat %>%
  group_by(name) %>%
  select(name) %>%
  summarise(total = n()) %>%
  arrange(desc(total))

total_shots_by_player
```

```
## # A tibble: 5 x 2
##   name            total
##   <fct>           <int>
## 1 Stephen Curry    1250
## 2 Klay Thompson    1220
## 3 Kevin Durant      915
## 4 Draymond Green    578
## 5 Andre Iguodala    371
```

5.2) Effective Shooting Percentage (10 pts)

Effective Shooting % by Player: Overall (i.e. including 2PT and 3PT Field Goals) effective shooting percentage by player, arranged in descending order by percentage.

Same previous comments: the code below is one way to obtain the required table. Your solution may be different. The important part is to have the same output, and to use `"dplyr"` functions. By the way, `merge()` is NOT a `dplyr` function; the dplyr merge is `join()`; so deduct 2 pts if you used `merge()` or any other non-dplyr functions.

```r
# overall effective shooting percentage by player (in descending order)
# (includes 2PT and 3PT Field Goals)
dat %>%
  group_by(name) %>%
  select(name, shot_made_flag) %>%
  summarise(
    total = n(),
    made = sum(shot_made_flag == 'made shot')
  ) %>%
  mutate(perc_made = 100 * made / total) %>%
  arrange(desc(perc_made))
```

```
## # A tibble: 5 x 4
##   name            total  made perc_made
##   <fct>           <int> <int>     <dbl>
## 1 Kevin Durant      915   495      54.1
## 2 Andre Iguodala    371   192      51.8
## 3 Klay Thompson    1220   575      47.1
## 4 Stephen Curry    1250   584      46.7
## 5 Draymond Green    578   245      42.4
```

2PT Effective Shooting % by Player: 2 PT Field Goal effective shooting percentage by player, arranged in descending order by percentage.

```
# 2PT Field Goal effective shooting percentage by player
# (in descending order)
dat %>%
  filter(shot_type == '2PT Field Goal') %>%
  group_by(name) %>%
  select(name, shot_made_flag) %>%
  summarise(
    total = n(),
    made = sum(shot_made_flag == 'made shot')
  ) %>%
  mutate(perc_made = 100 * made / total) %>%
  arrange(desc(perc_made))
```

```
## # A tibble: 5 x 4
##   name            total  made perc_made
##   <fct>           <int> <int>     <dbl>
## 1 Andre Iguodala    210   134      63.8
## 2 Kevin Durant      643   390      60.7
## 3 Stephen Curry     563   304      54.0
## 4 Klay Thompson     640   329      51.4
## 5 Draymond Green    346   171      49.4
```

3PT Effective Shooting % by Player: 3 PT Field Goal effective shooting percentage by player, arranged in descending order by percentage.

```
# 3PT Field Goal effective shooting percentage
# (in descending order)
dat %>%
  filter(shot_type == '3PT Field Goal') %>%
  group_by(name) %>%
  select(name, shot_made_flag) %>%
  summarise(
    total = n(),
    made = sum(shot_made_flag == 'made shot')
  ) %>%
  mutate(perc_made = 100 * made / total) %>%
  arrange(desc(perc_made))
```

```
## # A tibble: 5 x 4
##   name           total  made perc_made
##   <fct>          <int> <int>     <dbl>
## 1 Klay Thompson    580   246      42.4
## 2 Stephen Curry    687   280      40.8
## 3 Kevin Durant     272   105      38.6
## 4 Andre Iguodala   161    58      36.0
## 5 Draymond Green   232    74      31.9
```

6) Shooting Distance (20 pts)

Consider the following question: the shorter the shooting distance, the higher the chance to successfully make a shot? Intuition and experience will suggest that YES. To confirm this, you will have to calculate, for each distance value, the proportion of made shots.

6.1) dplyr table (10 pts)

More precisely, use dplyr operations to obtain a tibble with two columns: `shot_distance` and `made_shot_prop`. The first row of the tibble should contain the value of distance = 0 ft, and the associated proportion of made shots (of all the five analyzed players). The second row should contain the value of distance = 1 ft, and the corresponding proportion of made shots; and so on.

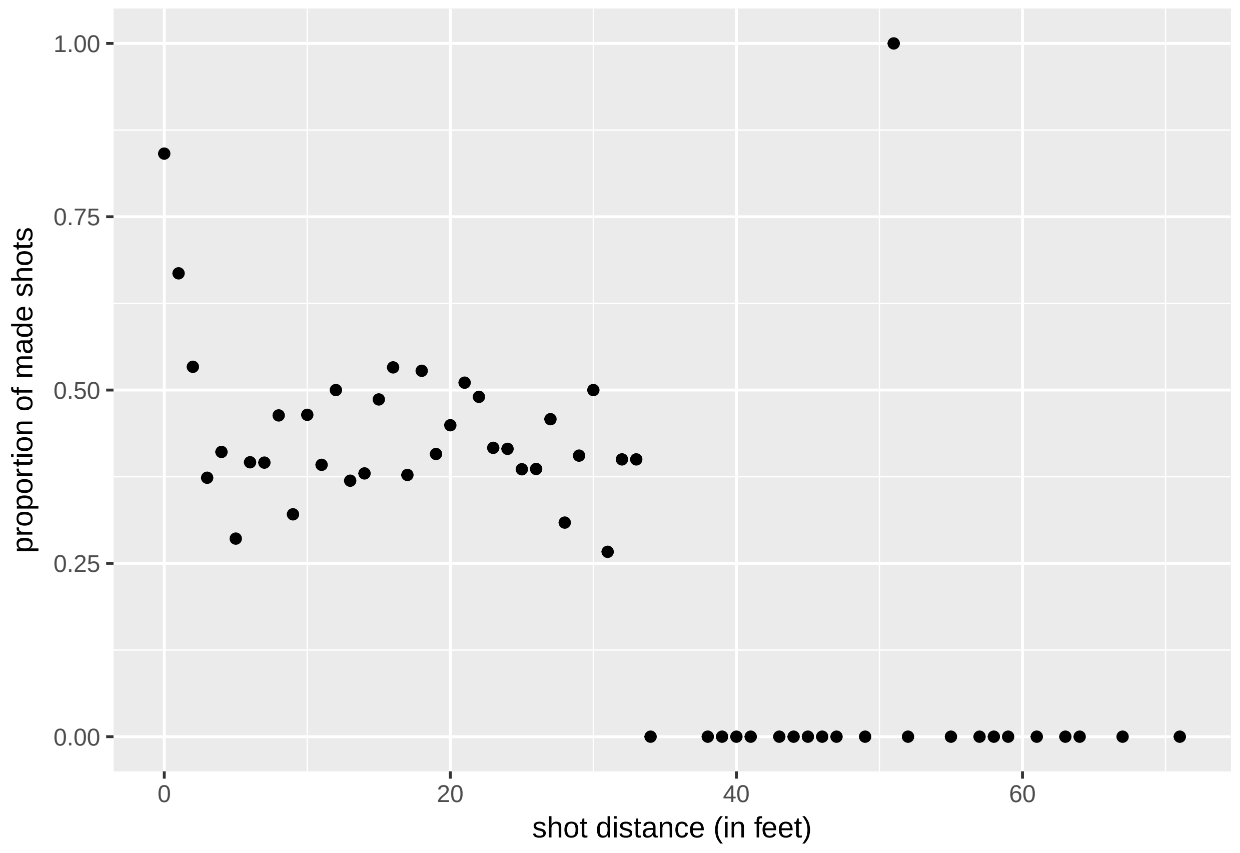
```
# distance effectivity
shots_dist_made <- dat %>%
  group_by(shot_distance) %>%
  select(shot_distance, shot_made_flag) %>%
  summarise(
    total_shots = n(),
    made_shots = sum(shot_made_flag == 'made shot')) %>%
  mutate(prop_made = made_shots / total_shots)

shots_dist_made
```

```
## # A tibble: 56 x 4
##    shot_distance total_shots made_shots prop_made
##            <int>       <int>      <int>     <dbl>
##  1             0         409        344     0.841
##  2             1         392        262      0.668
##  3             2         283        151      0.534
##  4             3          83         31      0.373
##  5             4          56         23      0.411
##  6             5          42         12      0.286
##  7             6          48         19      0.396
##  8             7          43         17      0.395
##  9             8          41         19      0.463
## 10             9          53         17      0.321
## # ... with 46 more rows
```

6.2) ggplot (10 pts)

With the tibble created in 6.1), use `ggplot()` to make a scatterplot with the variables `shot_distance` and `made_shot_prop`. Use the x-axis for the shot distance, and the y-axis for the proportion of made shots.

```
ggplot(data = shots_dist_made,
       aes(x = shot_distance, y = prop_made)) +
  geom_point() +
  xlab('shot distance (in feet)') +
  ylab('proportion of made shots')
```



What do you observe? Based on the obtained scatterplot, we can observe a decreasing trend between shot distance and proportion of made shots.

```
outlier_shot <- shots_dist_made %>%
  filter(shot_distance > 40) %>%
  filter(prop_made == 1)

outlier_shot$shot_distance
```

```
## [1] 51
```

Also, there's an atypical (outlier) shot successfully made from 51 feet.

The shorter the distance, the more effective the shots? We can confirm that as the distance of shots gets smaller, the proportion of made shots increases. Notice the large proportions from 0 feet and 1 foot, compared to the rest of the distances:

```
shots_dist_made %>%
  filter(shot_distance %in% c(0,1)) %>%
  select(prop_made)
```

```
## # A tibble: 2 x 1
##   prop_made
##       <dbl>
## 1     0.841
## 2     0.668
```

Can you guesstimate a distance threshold beyond which the chance of making a successful shot is basically null? One possible answer to this question is by looking at the distance in which the first null proportion occurs:

```
shots_dist_made %>%
  filter(prop_made == 0) %>%
  select(shot_distance) %>%
  head(1)
```

```
## # A tibble: 1 x 1
##   shot_distance
##           <int>
## 1            34
```

Distances that tend to have a percentage (of making a shot) of 50% or more?

```
shots_dist_made %>%
  filter(prop_made >= 0.5)
```

```
## # A tibble: 9 x 4
##   shot_distance total_shots made_shots prop_made
##           <int>       <int>      <int>     <dbl>
## 1             0         409        344     0.841
## 2             1         392        262     0.668
## 3             2         283        151     0.534
## 4            12          64         32     0.500
## 5            16         107         57     0.533
## 6            18         108         57     0.528
## 7            21          47         24     0.511
## 8            30          12          6     0.500
## 9            51           1          1     1.00
```

7) Total number of shots by minute of occurrence (10 pts)

The last part of the assignment involves looking at the *total number of shots (made and missed) by minute of occurrence.*

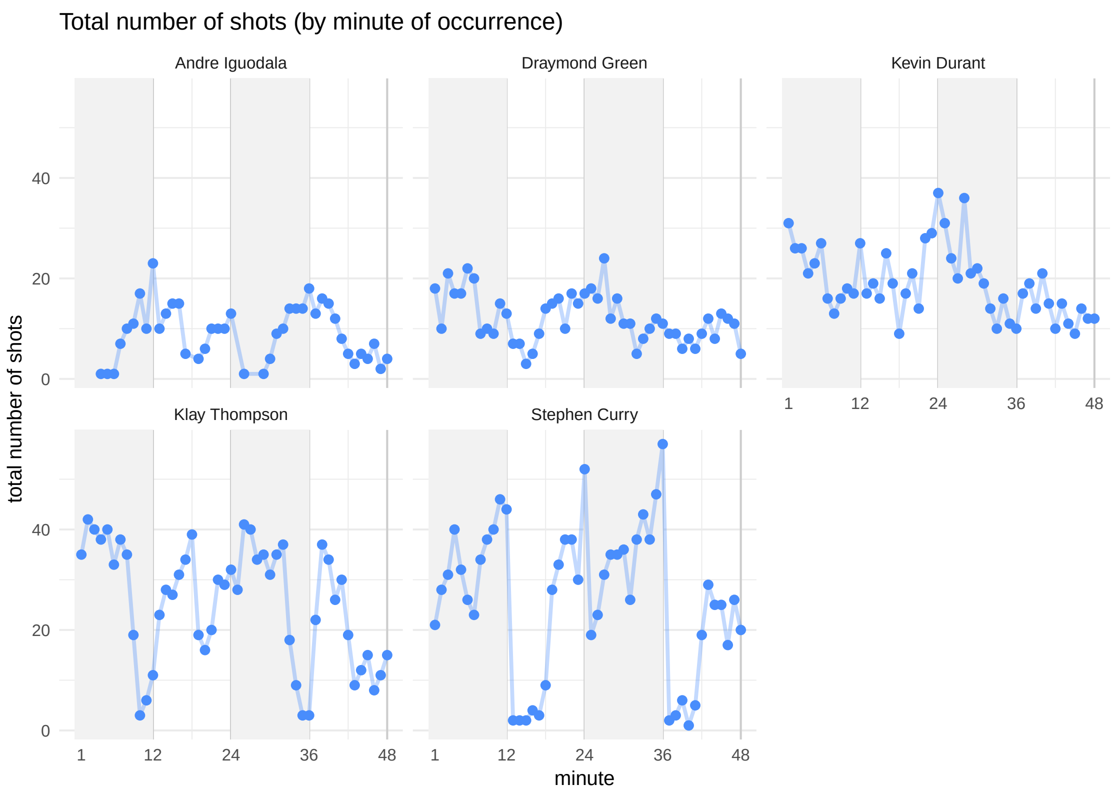
The appearance of your graph will very, very likely be somewhat different from the one in the instructions. The main purpose behind this part was to do a bit of *reverse engineering*, trying to approximate the format of the plot as close as possible: using various secondary ggplot functions to play with rectangles, background colors, line color, and dot colors, x-axis elemnts, as well as theme elements.

For reference purposes, I've included the data behind the required ggplot:

```
shots_in_minute <- dat %>%
  group_by(name, minute) %>%
  summarise(number_shots = n())

shots_in_minute
```

```
## # A tibble: 233 x 3
## # Groups:   name [?]
##    name           minute number_shots
##    <fct>           <int>        <int>
##  1 Andre Iguodala      4            1
##  2 Andre Iguodala      5            1
##  3 Andre Iguodala      6            1
##  4 Andre Iguodala      7            7
##  5 Andre Iguodala      8           10
##  6 Andre Iguodala      9           11
##  7 Andre Iguodala     10           17
##  8 Andre Iguodala     11           10
##  9 Andre Iguodala     12           23
## 10 Andre Iguodala     13           10
## # ... with 223 more rows
```

```
shots_in_mn_plot <- ggplot(shots_in_minute,
                           aes(x = minute, y = number_shots)) +
  geom_vline(xintercept = c(12, 24, 36, 48), color = 'gray80') +
  scale_x_continuous(breaks = c(1, 12, 24, 36, 48)) +
  ylab('total number of shots') +
  geom_rect(aes(xmin = 0, xmax = 12, ymin = -Inf, ymax = Inf),
            fill = "gray95") +
  geom_rect(aes(xmin = 24, xmax = 36, ymin = -Inf, ymax = Inf),
            fill = "gray95") +
  theme_minimal() +
  geom_path(color = '#498dfc55', size = 1) +
  geom_point(color = "#498dfc", size = 2) +
  facet_wrap(~ name) +
  ggtitle('Total number of shots (by minute of occurrence)')
```


Total number of shots (by minute of occurrence)
Andre Iguodala
Draymond Green
Kevin Durant
Klay Thompson
Stephen Curry
total number of shots
minute
0
20
40
1
12
24
36
48