Exploratory analysis

Haoran Hu

2019-3-26

Overview

Questions

Analysis

Load the data

```
fifa = read_csv("..\\Data\\CompleteDataset.csv")
```

Set the seed

```
set.seed(3)
```

data cleaning

Values for some players were recorded in "M" unit, and value for other players were recorded in "K" unit. So I first change them into the same scale.

```
calc_expression = function(fmula) {
    eval(parse(text = fmula))
}

data = fifa %>%
    janitor::clean_names() %>%
    select(-c(x1, name, photo, flag, club_logo, wage, overall, id)) %>%
    #mutate(value = readr::parse_number(value))
    mutate(value = str_replace(value, "K", "/ 1000"),
           value = str_replace(value, "M", ""),
           value = str_replace(value, "€", "")) %>%
    mutate(value = map(value, calc_expression),
           value = as.numeric(value))
```

Next, I will transform variables into suitable data type.

```
trans2int_cols = c(7:40)
trans2fct_cols = c(2, 4)
data[trans2int_cols] = map(data[trans2int_cols], as.integer)
data[trans2fct_cols] = map(data[trans2fct_cols], as.factor)
```

Variables that are important to all soccer players:

- `age`: (int)age of players
- `nationality`: (fct)nationality of players
- `potential`: (int)score of players' potential
- `club`: (fct)club of players
- `special`: (int)aggregated score of players' several special abilities(important abilities for all soccer players)
- `acceleration`: (int)score of players' acceleration ability(an important ability for all soccer players)
- `aggression`: (int)score evaluating players' level of aggression(important for all soccer players)
- `agility`: (int)score of players' agility(an important ability for all soccer players)
- `balance`: (int)score of players' balancing ability(an important ability for all soccer players)
- `ball control`: (int)score of players' ball control ability(an important ability for all soccer players)
- `composure`: (int)score of players' self-possession ability(an important ability for all soccer players)
- `curve`: (int)score of players' curve skill(an important skill for all soccer players)
- `dribble`: (int)score of players' dribbling skill(an important skill for all soccer players)
- `free_kick_accuracy`: (int)score of players' free-kicking skill(an important skill for all soccer players)
- `interceptions`: (int)score of players' interceptions(an important ability for all soccer players)
- `jumping`: (int)score of players' jumping ability(an important ability for all soccer players)
- `long_passing`: (int)score of players' long passing ability(an important ability for all soccer players)
- `penalty`: (int)score of players' penalty skill(an important skill for all soccer players)
- `positioning`: (int)score of players' positioning skill(an important skill for soccer players except goal keepers)
- `reactions`: (int)score of players' reaction speed(important for soccer players except goal keepers)
- `short_passing`: (int)score of players' short passing skill(an attacking skill, important for soccer players except goal keepers)
- `shot_power`: (int)score of players' shot power(important for soccer players except goal keepers)
- `sprint_speed`: (int)score of players' shot power(important for all soccer players)
- `stamina`: (int)score of players' stamina(important for all soccer players)
- `strength`: (int)score of players' strength(important for all soccer players)
- `vision`: (int)score of players' vision(important for all soccer players)
- `sliding_tackle`:(int)score of players' sliding tackle skill(an defending skill, but important for all soccer players)
- `standing_tackle`:(int)score of players' standing tackle skill(an defending skill, but important for all soccer players)

Variables on goal-keeping skills:

- `gk_diving`: (int)score of goal keepers' diving skill(an important skill for goal keepers)
- `gk_handling`: (int)score of goal keepers' handling skill(an important skill for goal keepers)
- `gk_kicking`: (int)score of goal keepers' kicking skill(an important skill for goal keepers)
- `gk_positioning`: (int)score of goal keepers' positioning skill(an important skill for goal keepers)
- `gk_reflexes`: (int)score of goal keepers' reflex speed(important for goal keepers)

Variables on attacking skills:

- `crossing`: (int)score of players' crossing skill(an attacking skill, important for soccer players except goal keepers, especially important for forward players)
- `finishing`: (int)score of players' finishing skill(an attacking skill, important for soccer players except goal keepers, especially important for forward players)
- `long_shots`: (int)score of players' long shot skill(an attacking skill, important for soccer players except goal keepers, especially important for forward players)
- `heading`: (int)score of players' heading skill(an attacking skill, important for soccer players except goal keepers)
- `volleys`: (int)score of players' volleying skill(an attacking skill, important for soccer players except goal keepers)

Variables on defending skills:

- `marking`: (int)score of players' marking skill(an defending skill, important for soccer players except goal keepers)

Outcome:

- `value`: (num)market value of players

Other:

- `preferred_possitions`: (chr)indicates players' preferred positions

Each of the other variables in the dataset evaluates players' ability when playing at a specific position.

In the next chunk, I will:

- seperate the column `preferred_positions` into multiple columns(`pos1`, `pos2`, `pos3`, `pos4`), so that each column indicates only one position.
- create a new variable `single_pos` indicating if a player's have only one preferred position or not. If YES, `multi_pos` = TRUE. Otherwise, `multi_pos` = FALSE.
- create a new variable `max_pos_score`, which shows the maximum position score of a player. Goal keepers' maximum position score will be set to 0, since in the original dataset, their position scores are all NA's, and their goal keeping skills are indicated by other variables.
- create a new variable `best_pos`, which shows the position where the player's position score gets the maximum. Goal keepers' best positions will be directly specified as "gk".

```
pos_data = data %>% #a temporary data frame used to get the maximum position score and the position whe
    select(cam:st) %>%
    select(-preferred_positions, everything()) %>%
    #mutate(., max_pos_score = purrr::pmap(.[, 1:(ncol(.) - 1)], max)) %>%
    mutate(., max_pos_score = apply(.[, 1:(ncol(.) - 1)], 1, max)) %>% #get the maximum position score
    mutate(., best_pos = colnames(.)[max.col(.[, 1:(ncol(.) - 2)],ties.method = "first")]) %>%  #get th
    select(preferred_positions, max_pos_score, best_pos)

gk_pos_score = function(Pos, scor){
    if (Pos == "GK") {
        scor = 0
    }
    else
        scor = scor
    scor
}#a function to set GKs' position scores to be 0

gk_pos = function(Pos, pos_out){
    if (Pos == "GK") {
        pos_out = "gk"
    }
    else
        pos_out = pos_out
    pos_out
}#a function to set GKs' best position to be "gk"

pos_data = pos_data %>%
    mutate(max_pos_score = map2(preferred_positions, max_pos_score, gk_pos_score),
           best_pos = map2(preferred_positions, best_pos, gk_pos))


data = data %>%
    mutate(max_pos_score = as.integer(pos_data$max_pos_score),
           best_pos = as.character(pos_data$best_pos)) %>%
    mutate(best_pos = as.factor(best_pos)) %>%
    separate(preferred_positions, c("pos1", "pos2", "pos3", "pos4"), sep = " ", remove = TRUE) %>%
    mutate(single_pos = map(pos2, is.na)) %>%
    mutate(single_pos = unlist(single_pos))
```

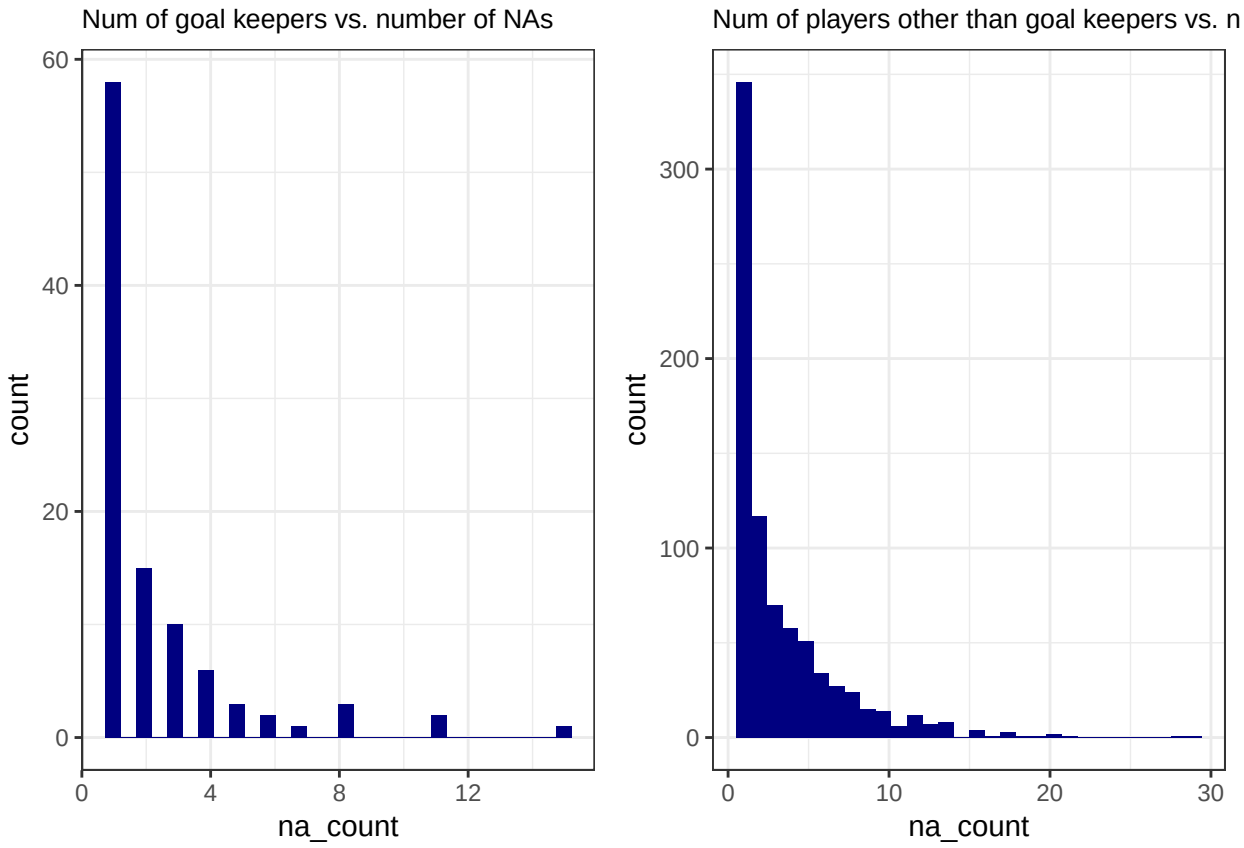
Since soccer players have relatively stable positions, their values should be mainly depend on their performance at their preferred position. Then, it's reasonable to put more emphasis on their scores at the best position for each of them.

In the following chunk, I will:

- check missing data in each observation in the dataset. Since a lot of NA's are caused by missing position score data of goal keepers, and These NA's do not matter, I will amalyze goal-keeper data and non goal-keeper data seperately.

```
data_gk = data %>%
    filter(best_pos == "gk") %>%
    select(-(cam:st)) %>%
    mutate(., na_count = apply(., 1, function(x) sum(is.na(x))))#the na_count shows number of NA's that
```

```
data_other = data %>%
    filter(best_pos != "gk") %>%
    select(-(pos1:pos4)) %>%
    mutate(., na_count = apply(., 1, function(x) sum(is.na(x))))#the na_count shows number of NA's in e

P1 = data_gk %>%
    filter(na_count > 0) %>%
    ggplot(aes(x = na_count)) + geom_histogram(fill = "navy") + theme_bw() + labs(title = "Num of goal

P2 = data_other %>%
    filter(na_count > 0) %>%
    ggplot(aes(x = na_count)) + geom_histogram(fill = "navy") + theme_bw() + labs(title = "Num of playe

P1 + P2
```

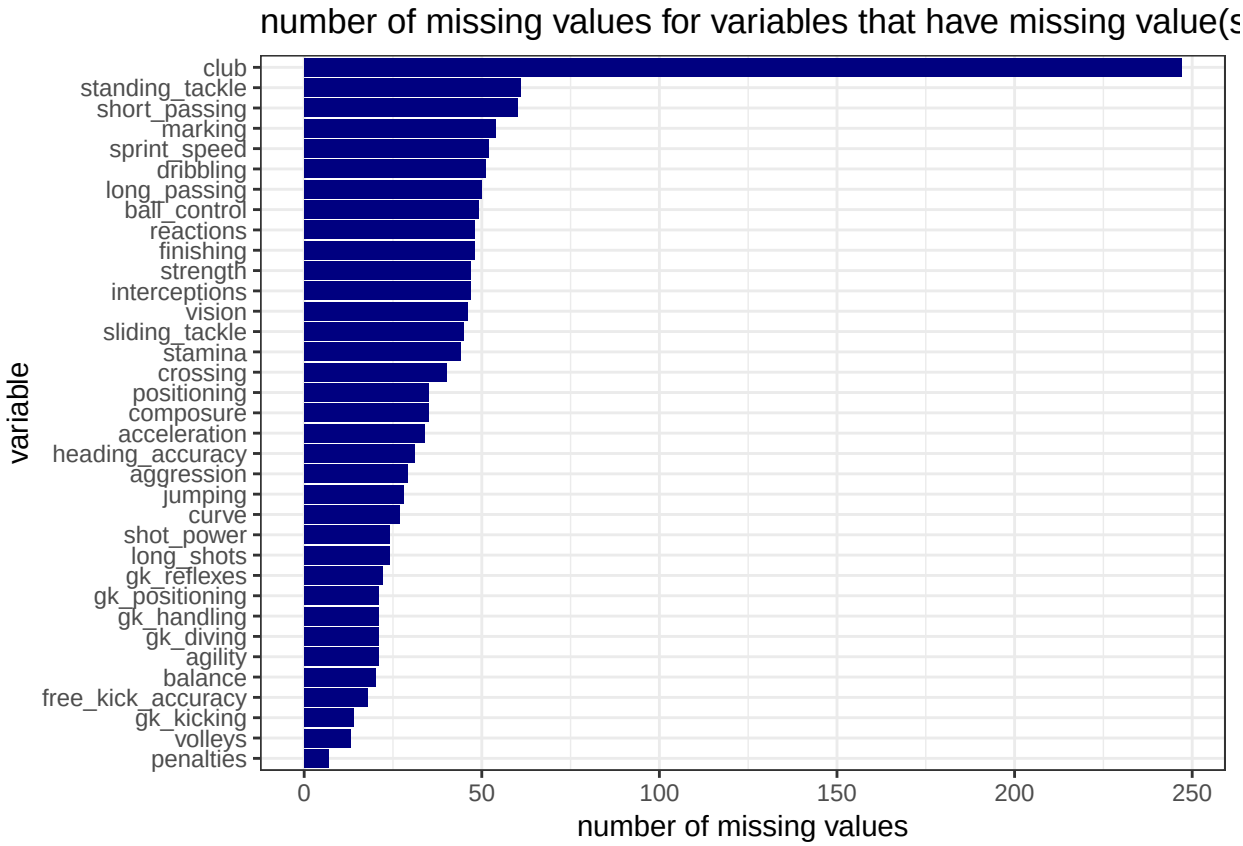


From the plot, it seems to be reasonable to drop observations that have more than 5 missing values. In the next chunk, I will:

- drop observations that have more than 5 missing values
- check number of NA's in each variable

```
data_gk = data_gk %>%
    filter(na_count <= 5)
```

```r
data_other = data_other %>%
    filter(na_count <= 5)

na_gk = colSums(is.na(data_gk)) %>%
    as.list() %>%
    as.data.frame() %>%
    select(-na_count) %>%
    gather(age:single_pos, key = "variable", value = "num_of_na")

na_other = colSums(is.na(data_other)) %>%
    as.list() %>%
    as.data.frame() %>%
    select(-na_count) %>%
    gather(age:single_pos, key = "variable", value = "num_of_na")

na_all = merge(na_gk, na_other, by = "variable", all = TRUE) %>%
    mutate(., num_of_na.x = replace_na(.$num_of_na.x, 0)) %>%
    mutate(., num_of_na = .$num_of_na.x + .$num_of_na.y)


na_all %>%
    filter(num_of_na > 0) %>%
    mutate(variable = fct_reorder(variable, num_of_na)) %>%
    ggplot(aes(x = variable, y = num_of_na)) +
  geom_col(fill = "navy") +
  theme(legend.position = "bottom") +
  labs(title = "number of missing values for variables that have missing value(s)", y = "number of missi
  coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8)) + theme_bu
```



In total, 35 variables have missing values. The "club" variable has the largest number of missing values(247 missing values). Note that all the players that have missing "club" values have market values = 0. This actually means they do not belong to any club for the time being. It seems reasonable to create a new category for the missing clubs, i.e., substitute NA's in "club" column with the word "none". Each of the other variables that have missing values has at most 61 missing values. Compared to the sample size, that's not a big loss of data, and thus all the variables can be kept in the model. The missing values of a varible will be replaced by the average observed value of the variable.

In the next chunk, I will:

- replace NA's in "club" column a new category "other club"
- replaced NA's in each column except "club" column by the mean of that column. This will be done seperately for goal keepers and for non goal keepers.

```
#get the names of variables that have missing values, except for "club"
vbls_replace_NA = na_all %>%
    filter(num_of_na > 0 & variable != "club")

#for variables(except for "club") that have missing values, replace NA's with mean
for (i in 1:34) {
    vbl_name = vbls_replace_NA$variable[i]
    #col_num = which(colnames(data_gk) == vbl_name) %>%
        #as.numeric()
       vec = data_gk[vbl_name] %>%
        unlist() %>%
        unname()
```

```
        #data_gk[is.na(data_gk[,col_num]), col_num] = round(mean(vec, na.rm = TRUE))
        data_gk[is.na(data_gk[,vbl_name]), vbl_name] = round(mean(vec, na.rm = TRUE))

        vec = data_other[vbl_name] %>%
         unlist() %>%
         unname()
        data_other[is.na(data_other[,vbl_name]), vbl_name] = round(mean(vec, na.rm = TRUE))
}


data_gk = data_gk %>%
    mutate(club = forcats::fct_explicit_na(club, na_level = "none"))
data_other = data_other %>%
    mutate(club = forcats::fct_explicit_na(club, na_level = "none"))

sum(is.na(data_gk)) #=0
```

```
## [1] 0
```

```
sum(is.na(data_other)) #=0
```

```
## [1] 0
```

In the next chunk, I will:

- bind the rows of goal keepers' data and non goal keepers' data.

```
data = bind_rows(data_gk, data_other) %>%
    select(-na_count)

rm(data_gk)
rm(data_other)
```

Tables for descriptive statistics

```
descrip_list = data %>%
  dplyr::select(-max_pos_score, -(cam:st)) %>% #max_pos_score need to be analyzed seperately, since the
  skimr::skim_to_list()

descrip_list_maxpos = data %>%
    filter(max_pos_score > 0) %>%
    select(max_pos_score, cam:st) %>%
    skimr::skim_to_list() #analyzed max_pos_score seperately

descrip_list[[1]] %>%
    select(variable, n_unique) %>%
    dplyr::rename("unique levels" = n_unique) %>%
    knitr::kable(caption = "Factor variables")
```

Table 1: Factor variables

variable	unique levels
best_pos	11
club	648
nationality	165

```
bind_rows(descrip_list[[2]] , descrip_list_maxpos[[1]]) %>% #combine with max_pos_score
    mutate(variable = recode(variable, x = 'max_pos_score')) %>%
    dplyr::select(variable,
                  Min = p0,
                  `1st Q` = p25,
                  Mean = mean,
                  Median = p50,
                  `3rd Q` = p75,
                  Max = p100,
                  `Std Dev` = sd) %>%
  knitr::kable(digits = 3)
```

variable	Min	1st Q	Mean	Median	3rd Q	Max	Std Dev
age	16	21	25.16	25	28	47	4.62
potential	46	67	71.17	71	75	94	6.1
special	728	1448	1593.51	1633	1785	2291	272.53
max_pos_score	45	62	65.95	66	71	92	6.78

```
tibble(levels = c("TRUE", "FALSE"),
       proportion = c(0.46, 0.54)) %>%
    knitr::kable(caption = "Levels of single_pos variable(logi)")
```

Table 3: Levels of single_pos variable(logi)

levels	proportion
TRUE	0.46
FALSE	0.54

```
bind_rows(descrip_list[[4]], descrip_list_maxpos[[2]])  %>%
      dplyr::select(variable,
                  Min = p0,
                  `1st Q` = p25,
                  Mean = mean,
                  Median = p50,
                  `3rd Q` = p75,
                  Max = p100,
                  `Std Dev` = sd) %>%
    knitr::kable(caption = "Integer variables")
```

Table 4: Integer variables

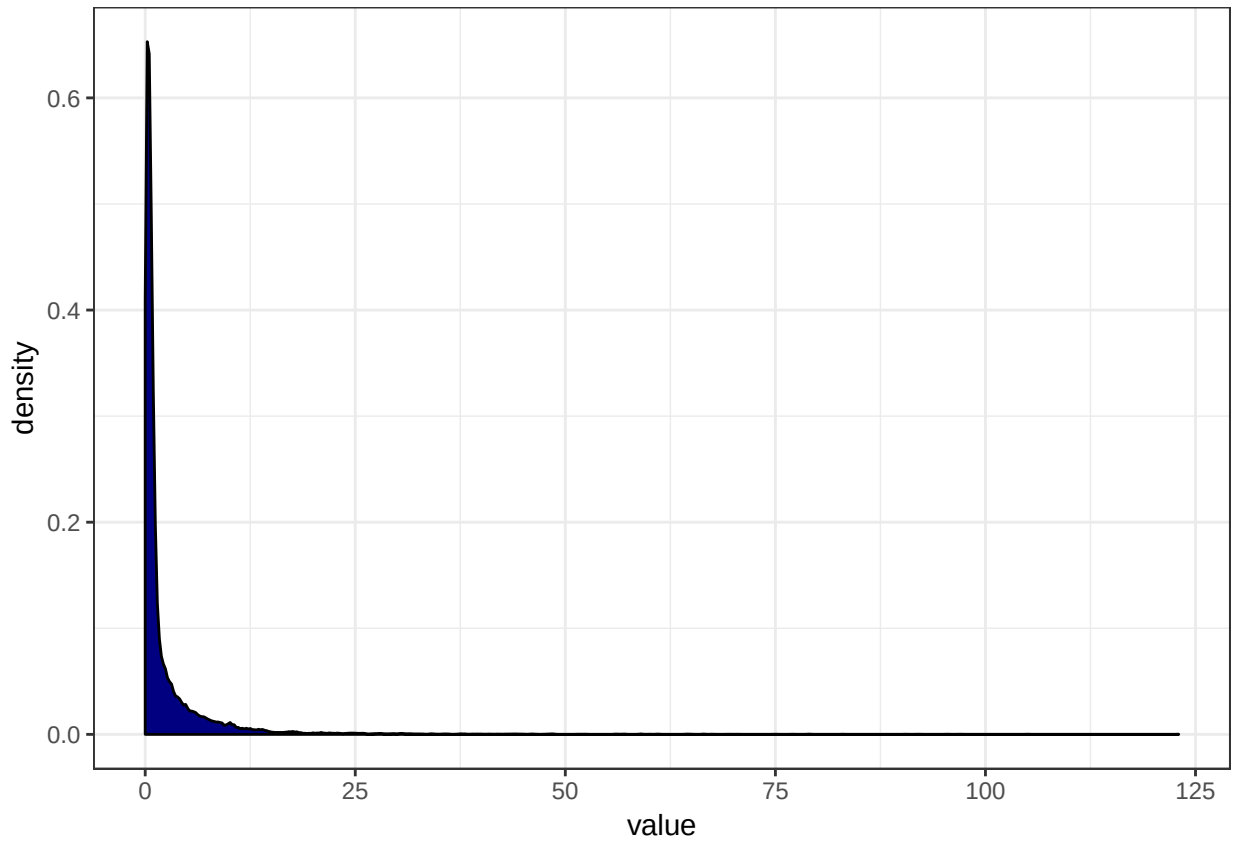
variable	Min	1st Q	Mean	Median	3rd Q	Max	Std Dev
acceleration	11	56	64.55	67	75	96	14.91
aggression	11	43	55.8	59	69	96	17.48
agility	14	55	63.3	65	74	96	14.8
balance	11	56	63.79	66	74	96	14.11
ball_control	8	53	58.07	63	69	95	16.82
composure	5	51	57.87	60	67	96	12.93
crossing	5	38	49.73	54	64	91	18.46
curve	6	34	47.25	48	62	92	18.46
dribbling	2	48	55.01	60	68	97	19
finishing	2	29	45.22	48	61	95	19.46
free_kick_accuracy	4	31	43.13	42	57	93	17.57
gk_diving	1	8	16.77	11	14	91	17.82
gk_handling	1	8	16.54	11	14	91	17.05
gk_kicking	1	8	16.42	11	14	95	16.64
gk_positioning	1	8	16.53	11	14	91	17.16
gk_reflexes	1	8	16.9	11	14	90	18.11
heading_accuracy	4	45	52.31	56	64	94	17.42
interceptions	4	26	46.53	51	64	92	20.68
jumping	15	58	64.88	66	73	95	11.88
long_passing	7	42	52.41	56	64	93	15.53
long_shots	3	32	47.17	51	62	92	19.28
marking	4	22	44.07	48	63	92	21.56
penalties	5	39	48.96	50	61	92	15.83
positioning	2	38	49.59	54	64	95	19.44
reactions	28	55	61.9	62	68	96	9.16
short_passing	10	53	58.26	62	68	92	14.93
shot_power	3	46	55.62	59	69	94	17.4
sliding_tackle	4	24	45.53	51	64	91	21.47
sprint_speed	11	57	64.77	67	75	96	14.63
stamina	12	56	63.21	66	74	95	15.94
standing_tackle	4	26	47.4	54	66	92	21.82
strength	20	58	65.29	66	74	98	12.61
value	0	0.3	2.38	0.68	2.1	123	5.36
vision	10	43	52.97	54	64	94	14.38
volleys	4	30	43.18	44	57	91	17.74
cam	27	53	59.24	60	66	92	9.89
cb	25	45	55.56	57	65	87	12.2
cdm	26	49	56.87	58	65	85	10.32
cf	27	53	59.01	60	66	92	9.93
cm	30	53	58.5	59	65	87	8.9
lam	27	53	59.24	60	66	92	9.89
lb	30	50	56.98	58	64	84	9.8
lcb	25	45	55.56	57	65	87	12.2
lcm	30	53	58.5	59	65	87	8.9
ldm	26	49	56.87	58	65	85	10.32
lf	27	53	59.01	60	66	92	9.93
lm	28	54	60.04	61	67	90	9.35
ls	31	52	58.19	59	65	92	9.18
lw	26	53	59.34	60	66	91	9.98
lwb	31	51	57.7	58	64	84	9.15

variable	Min	1st Q	Mean	Median	3rd Q	Max	Std Dev
ram	27	53	59.24	60	66	92	9.89
rb	30	50	56.98	58	64	84	9.8
rcb	25	45	55.56	57	65	87	12.2
rcm	30	53	58.5	59	65	87	8.9
rdm	26	49	56.87	58	65	85	10.32
rf	27	53	59.01	60	66	92	9.93
rm	28	54	60.04	61	67	90	9.35
rs	31	52	58.19	59	65	92	9.18
rw	26	53	59.34	60	66	91	9.98
rwb	31	51	57.7	58	64	84	9.15
st	31	52	58.19	59	65	92	9.18

Figures for descriptive statistics

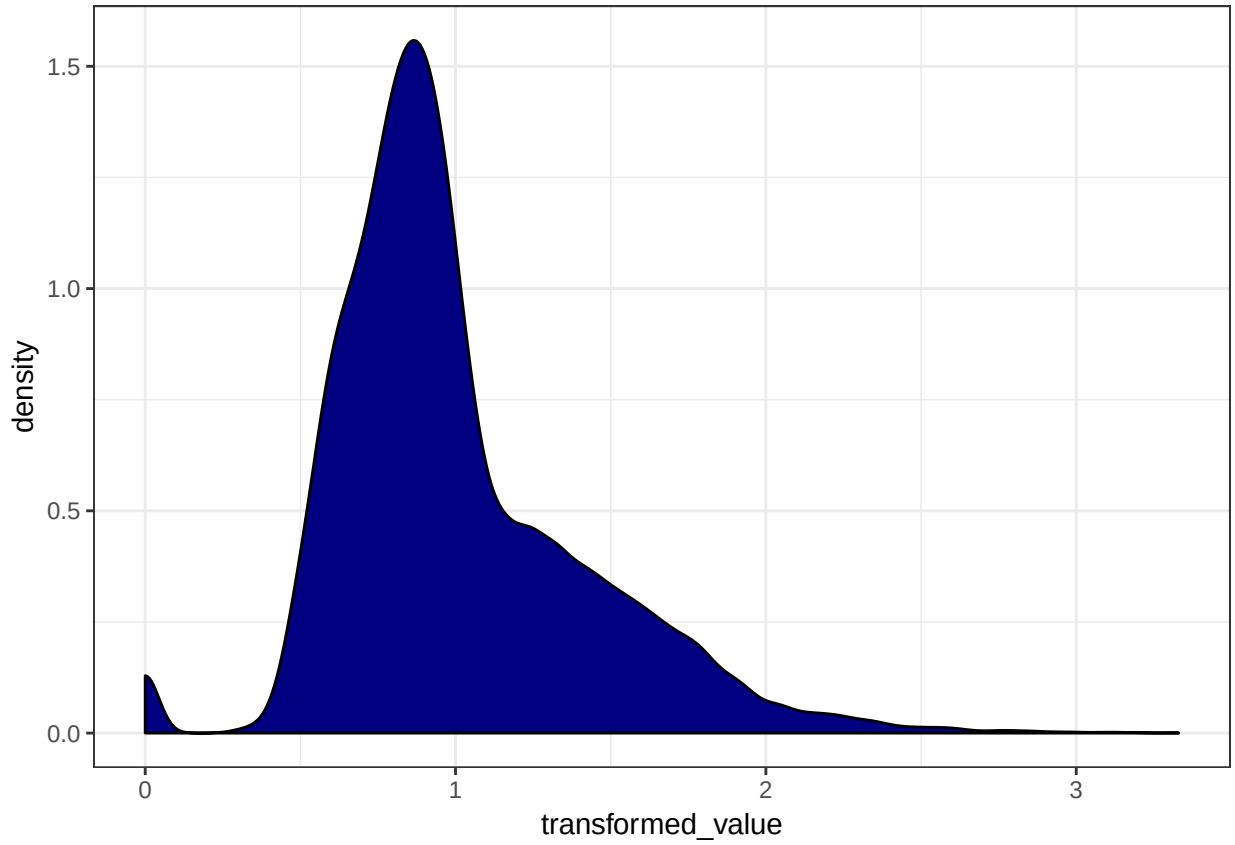
For the outcome(value)

```
data %>%
    ggplot(aes(x = value)) + geom_density(fill = "navy") + theme_bw()
```



The outcome data is skewed, so I will transform it. New_value = Old_valueˆ(1/4)

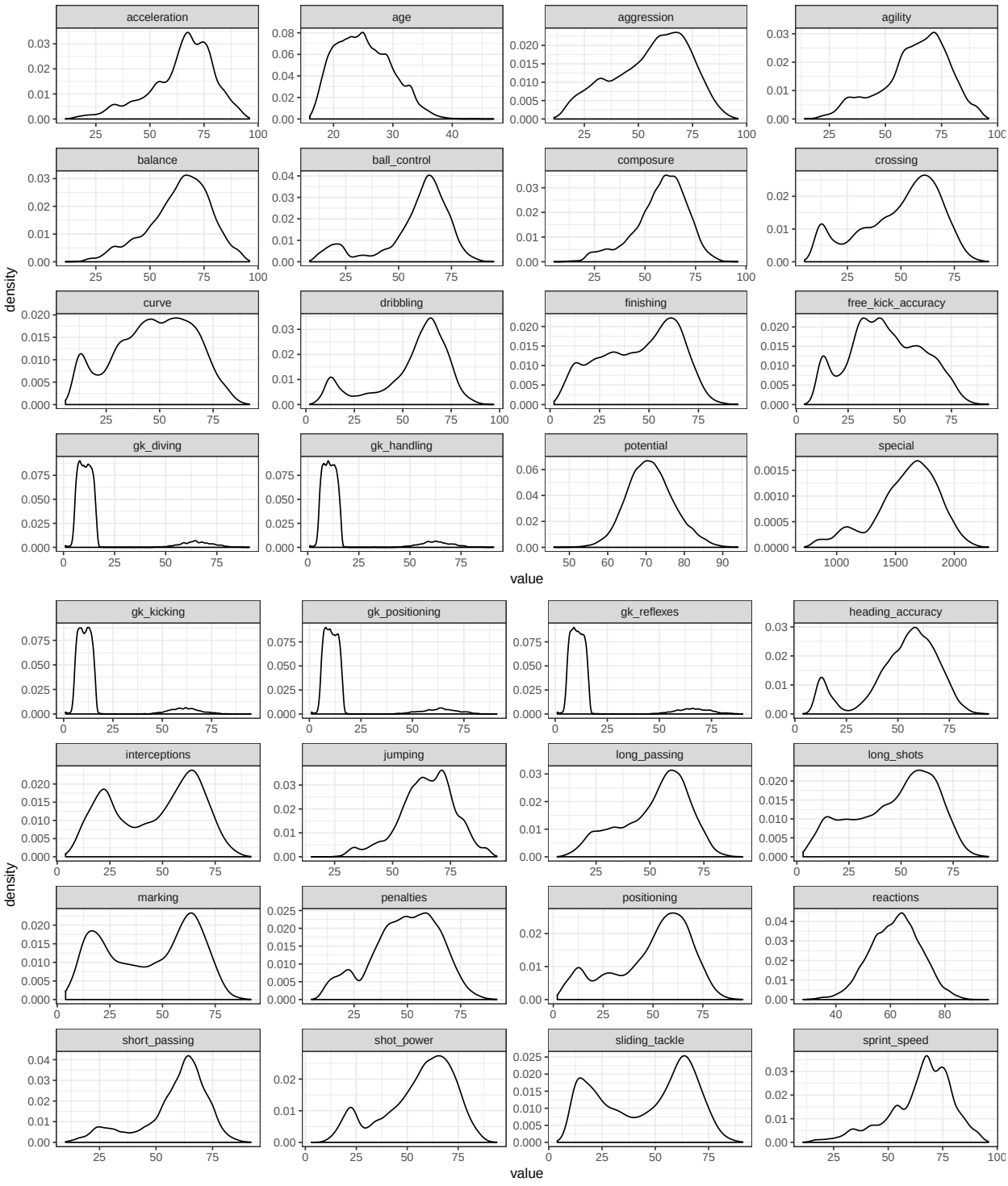
```r
#The effect of log transformation is not ideal
#data = data %>%
#   mutate(value = log(value + 2)

data = data %>%
    mutate(value = value^(1/4)) %>%
    rename("transformed_value" = value)

data %>%
    ggplot(aes(x = transformed_value)) + geom_density(fill = "navy") + theme_bw()
```



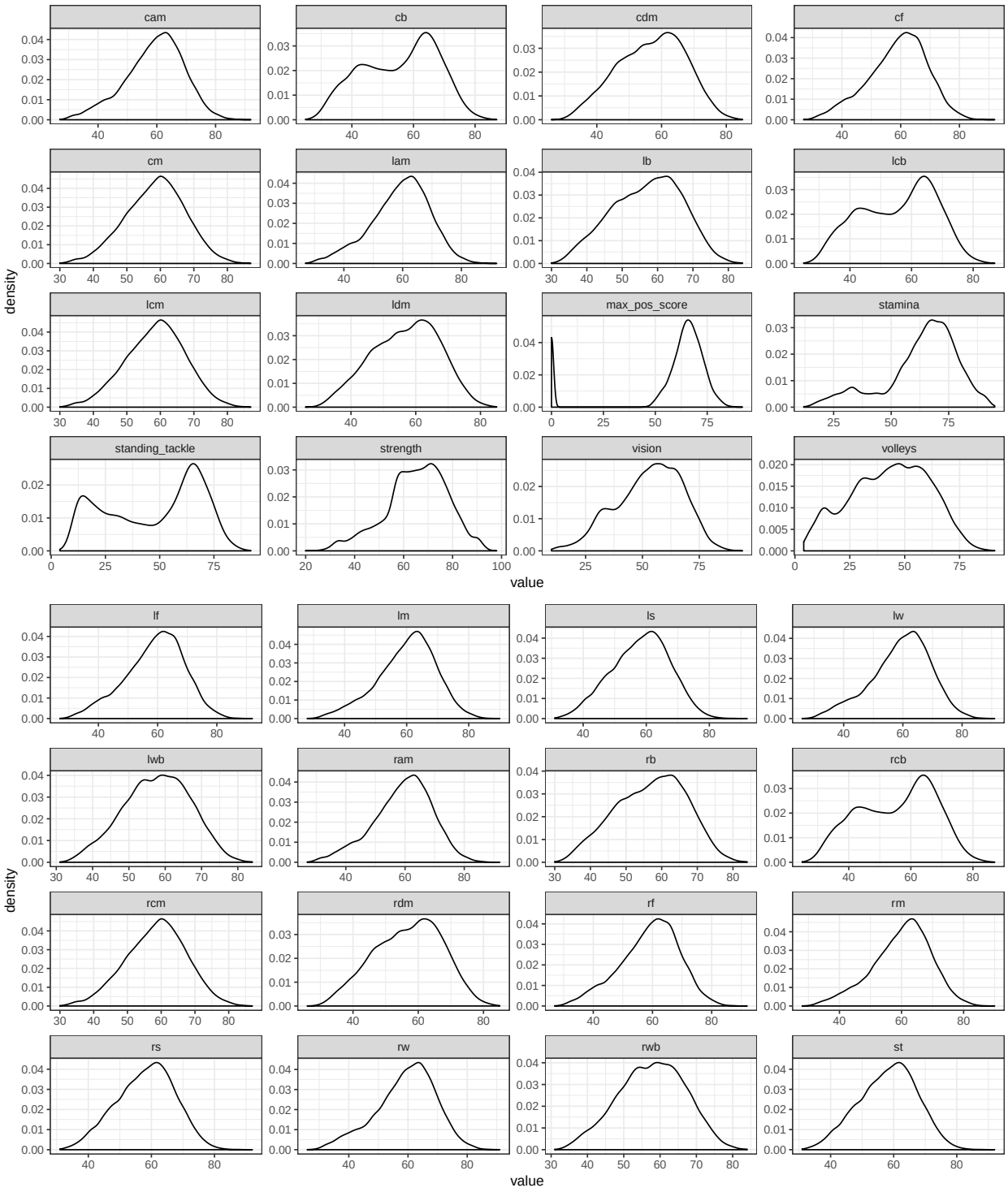
Check the distribution for each numeric/integer predictor

```r
#library(gridExtra)

p1 = data[,1:19] %>%
    select(-transformed_value) %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density() +
    theme_bw()
```

```r
p2 = data[,20:35] %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density() +
    theme_bw()

p3 = data[,36:53] %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density()  +
    theme_bw()

p4 = data[,54:69] %>%
  keep(is.numeric) %>%                     # Keep only numeric columns
  gather() %>%                             # Convert to key-value pairs
  ggplot(aes(value)) +                     # Plot the values
    facet_wrap(~ key, scales = "free") +   # In separate panels
    geom_density()  +
    theme_bw()
p1/p2
```


acceleration
age
aggression
agility
balance
ball_control
composure
crossing
curve
dribbling
finishing
free_kick_accuracy
gk_diving
gk_handling
potential
special
density
value
gk_kicking
gk_positioning
gk_reflexes
heading_accuracy
interceptions
jumping
long_passing
long_shots
marking
penalties
positioning
reactions
short_passing
shot_power
sliding_tackle
sprint_speed
density
value

```
p3/p4
```



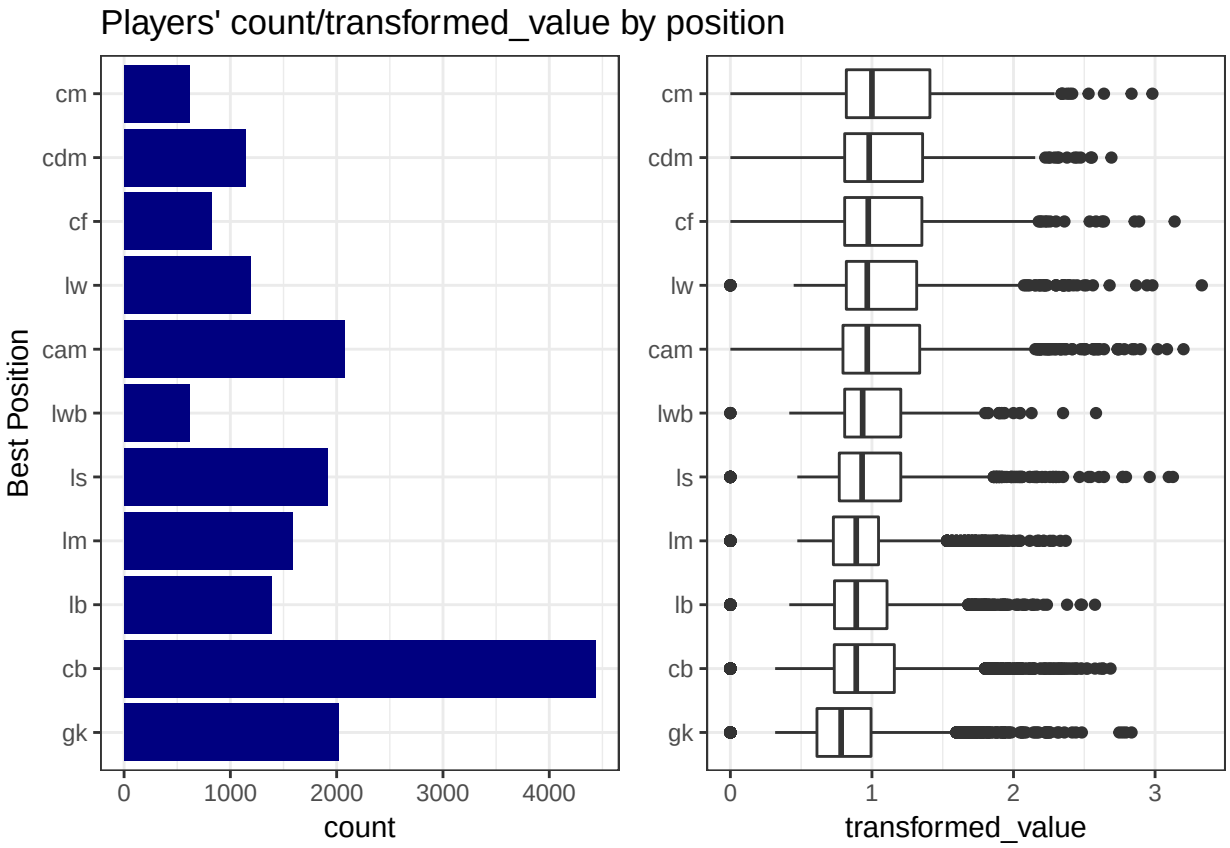
There seems not to be any data error. The variable `max_pos_score` has unusually large amount of 0 zero values because I set it to be 0 for goal keepers. For the variables whose density plots have two peaks, the smaller peaks are generally caused by goal keepers, and the larger peaks are generally caused by non-goal keepers.

plot for variables

for factor variables

- best_pos

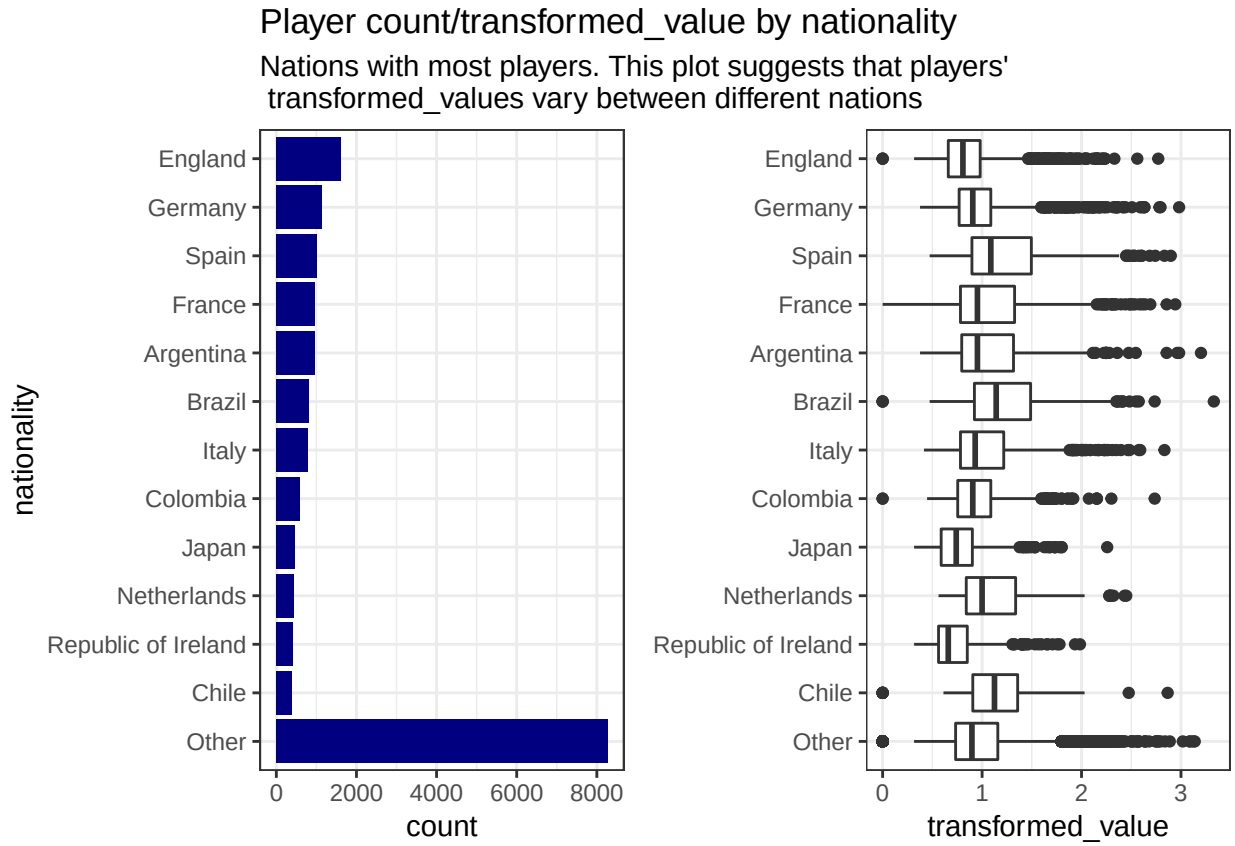
```
pos_box = data %>%
mutate(best_pos = fct_reorder(best_pos, transformed_value)) %>%
ggplot(aes(x = best_pos, y = transformed_value)) +
geom_boxplot() +
theme(legend.position = "bottom") +
labs(x = NULL) +
coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8)) + theme_bw()

pos_hist = data %>%
mutate(best_pos = fct_reorder(best_pos, transformed_value)) %>%
ggplot(aes(x = best_pos)) +
geom_bar(fill = "navy") + labs(title = "Players' count/transformed_value by position", x = "Best Positi
    coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8)) + theme_

pos_hist + pos_box
```



- nationality

plot the situation for nations with the most number of players:

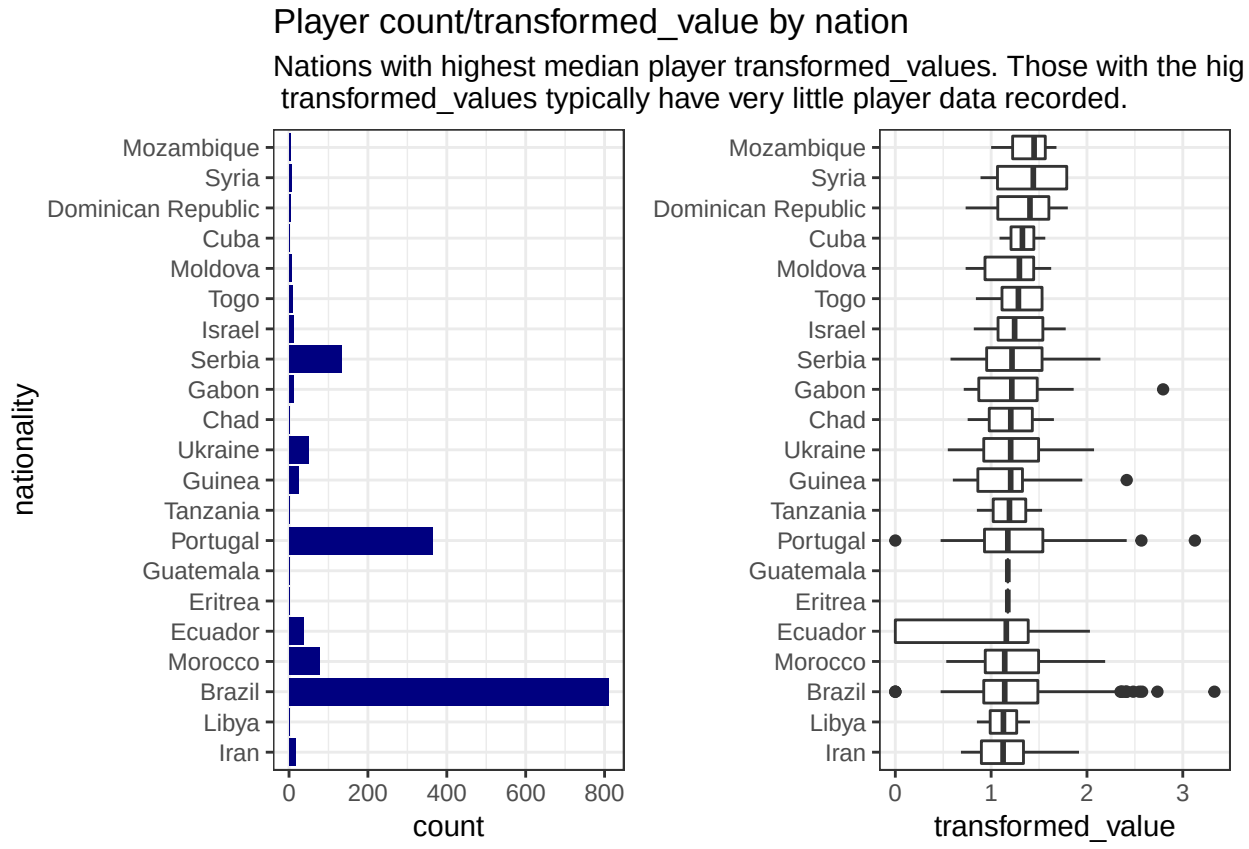
```r
nation_box = data %>%
    mutate(nationality = fct_lump(nationality, 12)) %>%
    mutate(nationality = fct_infreq(nationality)) %>%
    mutate(nationality = fct_rev(nationality)) %>%
    #move "Other" level to the last:
    mutate(nationality = fct_relevel(nationality, "Other", after = 0)) %>%
    ggplot(aes(x = nationality, y = transformed_value)) +
geom_boxplot() +
theme(legend.position = "bottom") +
labs( x = NULL) +
coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8)) + theme_bw()

nation_hist = data %>%
    #If focus on most common nations:
    mutate(nationality = fct_lump(nationality, 12)) %>% #nations that have fewer players will be denote
    mutate(nationality = fct_infreq(nationality)) %>%
    mutate(nationality = fct_rev(nationality)) %>%
    #move "Other" level to the last:
    mutate(nationality = fct_relevel(nationality, "Other", after = 0)) %>%
    ggplot(aes(x = nationality)) +
    geom_bar(fill = "navy") +
    theme(legend.position = "bottom") +
    labs(title = "Player count/transformed_value by nationality", subtitle = "Nations with most players
    coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8))  +
    theme_bw()

nation_hist + nation_box
```



plot the nations with highest players' transformed_values:

```
nation_box = data %>%
    group_by(nationality) %>%
    mutate(med_by_nation = median(transformed_value)) %>%
    ungroup() %>%
    mutate(nationality = fct_reorder(nationality, med_by_nation)) %>%
    filter(as.integer(nationality) >= 145) %>%  #get the nations that have the largest median player tr
    ggplot(aes(x = nationality, y = transformed_value)) +
    geom_boxplot() +
    theme(legend.position = "bottom") +
    labs( x = NULL) +
    coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8)) + theme

nation_hist = data %>%
    group_by(nationality) %>%
    mutate(med_by_nation = median(transformed_value)) %>%
    ungroup() %>%
    mutate(nationality = fct_reorder(nationality, med_by_nation)) %>%
    filter(as.integer(nationality) >= 145) %>%  #get the nations that have the largest median player tr
    ggplot(aes(x = nationality)) +
    geom_bar(fill = "navy") +
    theme(legend.position = "bottom") +
    labs(title = "Player count/transformed_value by nation", subtitle = "Nations with highest median pla
    coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8))  + theme
```

```
nation_hist + nation_box
```

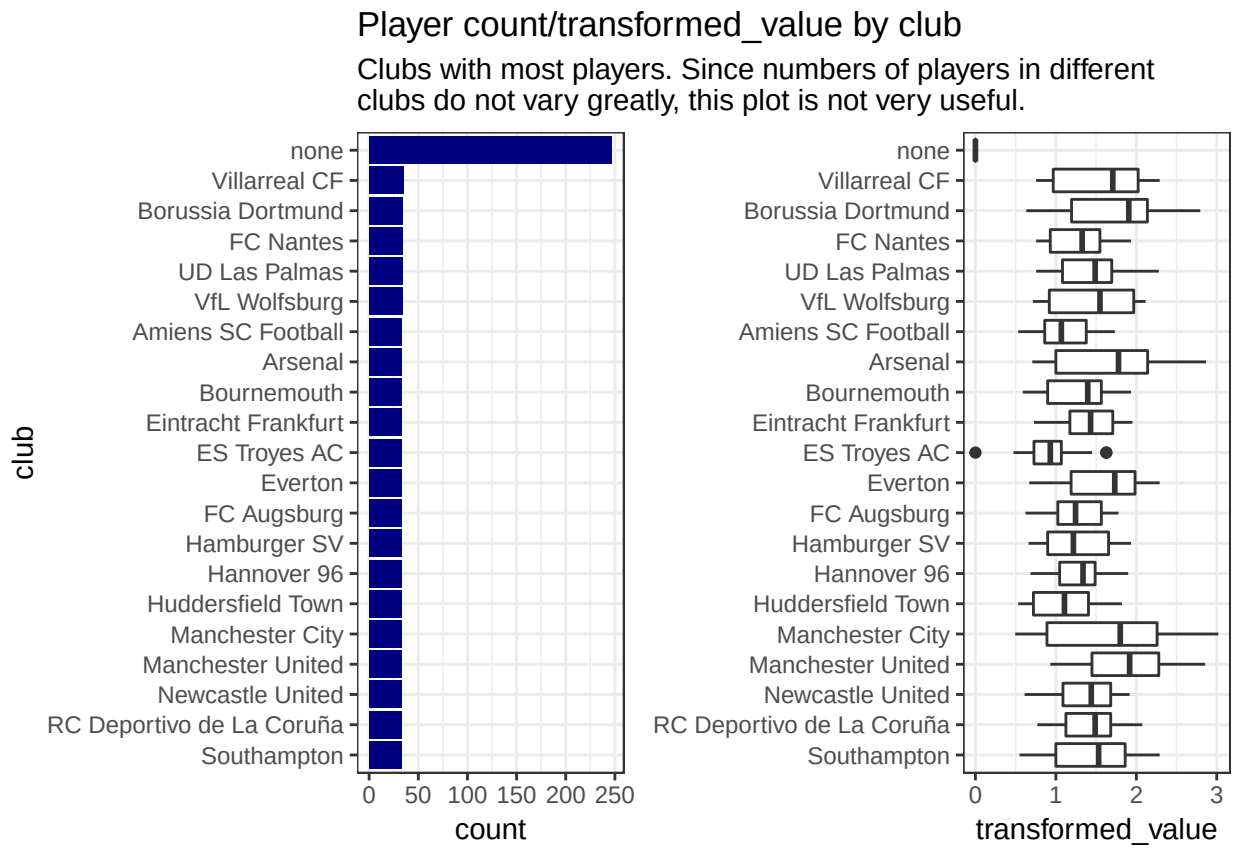


- club

clubs that have the largest number of players

```
club_box = data %>%
    mutate(club = fct_lump(club, 12)) %>%
    mutate(club = fct_infreq(club)) %>%
    mutate(club = fct_rev(club)) %>%
    #move "Other" level to the last:
    filter(club != "Other") %>%
    ggplot(aes(x = club, y = transformed_value)) +
geom_boxplot() +
theme(legend.position = "bottom") +
labs( x = NULL) +
coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8)) + theme_bw()

club_hist = data %>%
    mutate(club = fct_lump(club, 12)) %>%
    mutate(club = fct_infreq(club)) %>%
    mutate(club = fct_rev(club)) %>%
    #move "Other" level to the last:
    filter(club != "Other") %>%
```

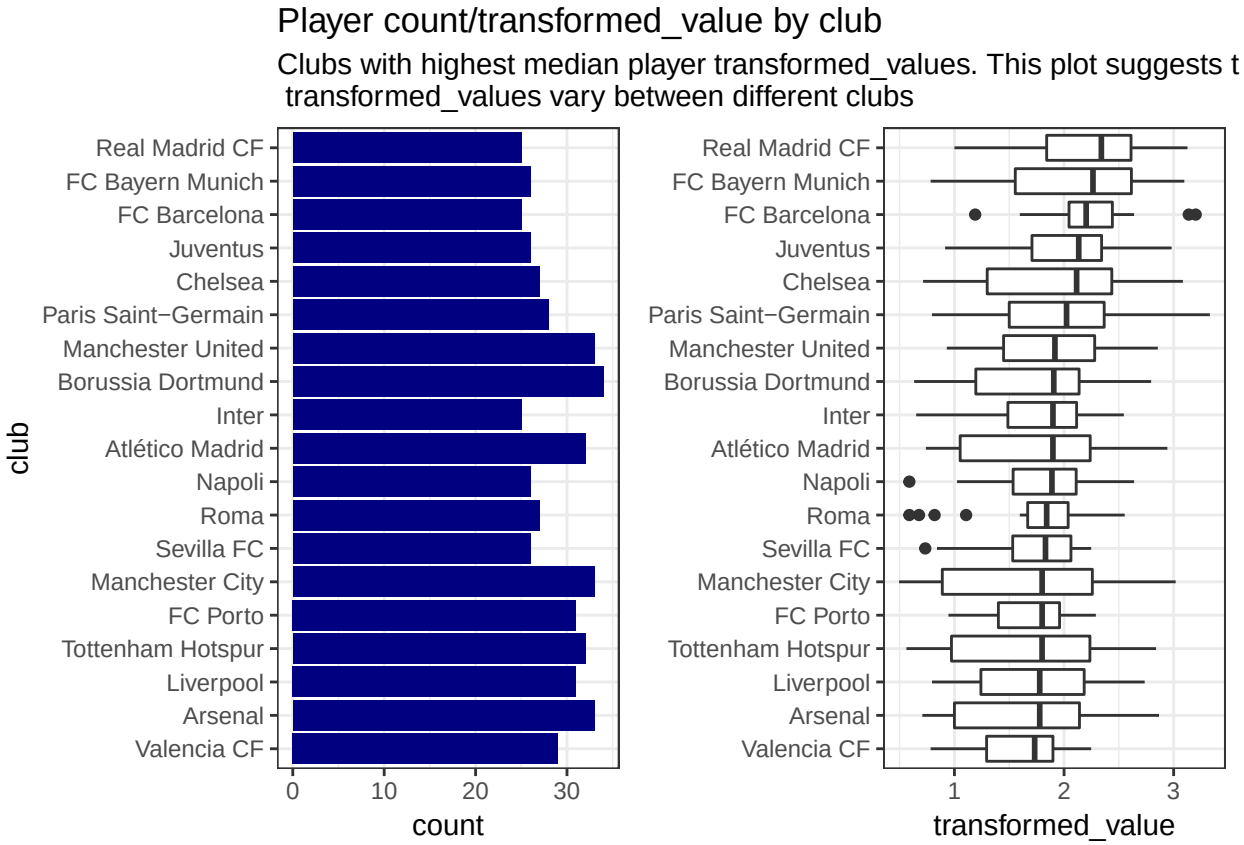
```
    ggplot(aes(x = club)) +
    geom_bar(fill = "navy") +
    theme(legend.position = "bottom") +
    labs(title = "Player count/transformed_value by club", subtitle = "Clubs with most players. Since nu
    coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8))  +
    theme_bw()

club_hist + club_box
```



clubs that have the highest player median transformed_values

```
club_box = data %>%
    group_by(club) %>%
    mutate(med_by_club = median(transformed_value)) %>%
    ungroup() %>%
    mutate(club = fct_reorder(club, med_by_club)) %>%
    filter(as.integer(club) >= 630) %>%  #get the clubs that have the highest median transformed_values
    ggplot(aes(x = club, y = transformed_value)) +
    geom_boxplot() +
    theme(legend.position = "bottom") +
    labs( x = NULL) +
    coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8))  + theme

club_hist = data %>%
    group_by(club) %>%
    mutate(med_by_club = median(transformed_value)) %>%
```

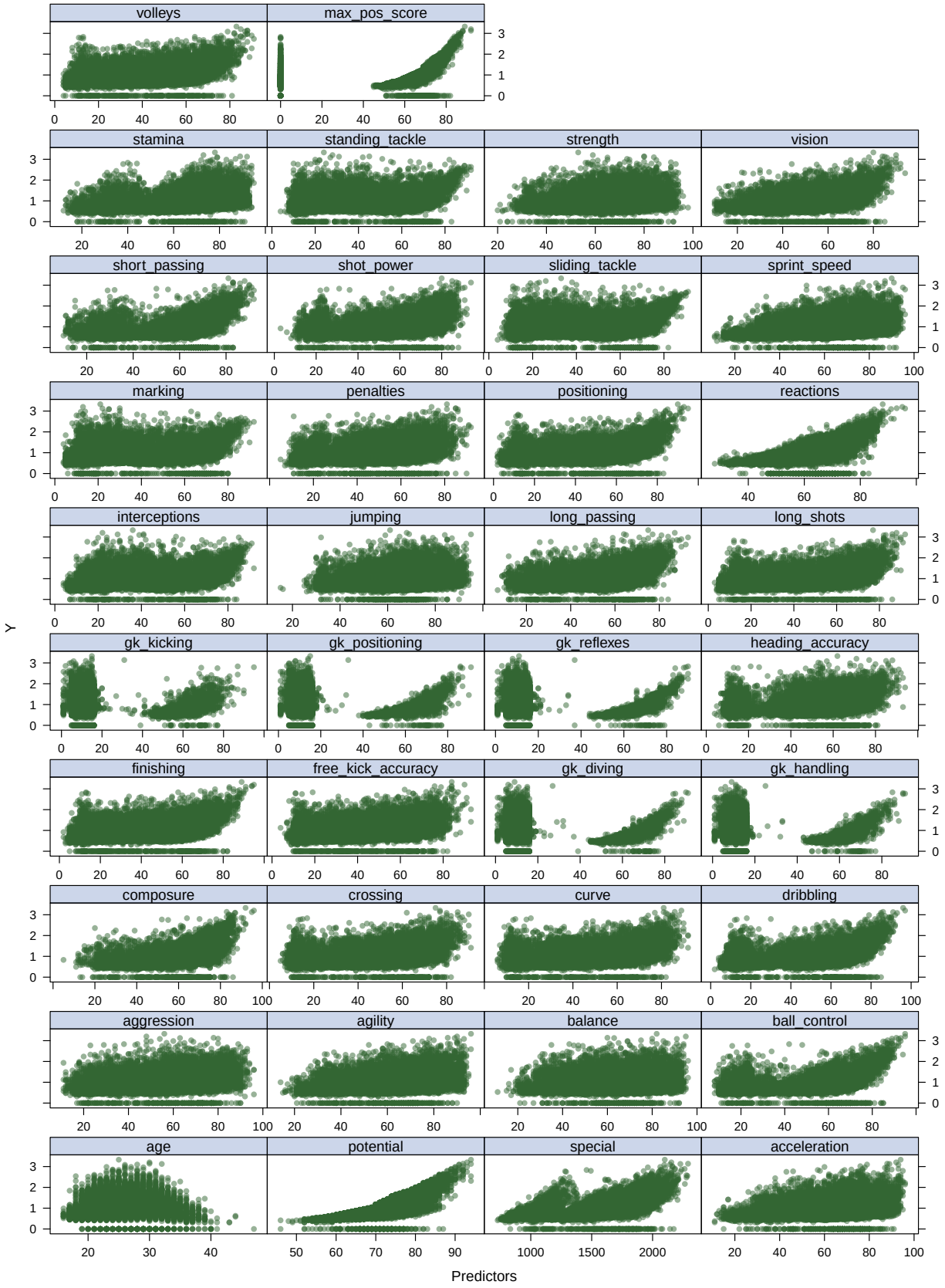
```
    ungroup() %>%
    mutate(club = fct_reorder(club, med_by_club)) %>%
    filter(as.integer(club) >= 630) %>%  #get the clubs that have the highest median transformed_values
    ggplot(aes(x = club)) +
    geom_bar(fill = "navy") +
    theme(legend.position = "bottom") +
    labs(title = "Player count/transformed_value by club", subtitle = "Clubs with highest median player
    coord_flip() + theme(axis.text.x = element_text(face = "plain", color = "black", size = 8))  + theme

club_hist + club_box
```



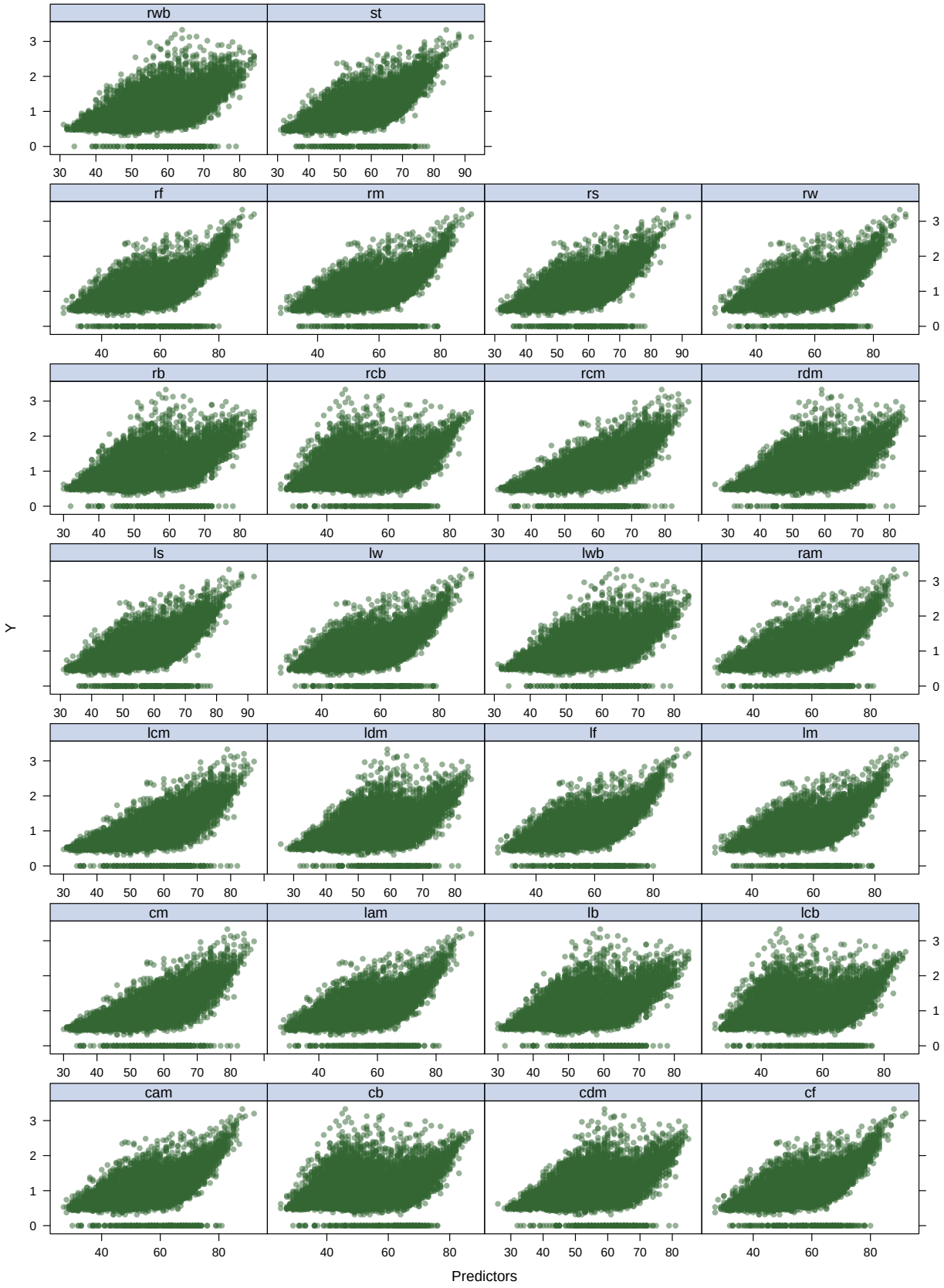
for int/num variables

```
# matrix of predictors

#for numeric/integer variables that do not contain NA
data_num1  = data %>%
    keep(is.numeric) %>%
    select_if(~ !any(is.na(.))) %>%
    select(-transformed_value)

#for numeric/integer variables that are NA's for gks
data_num2  = data %>%
```

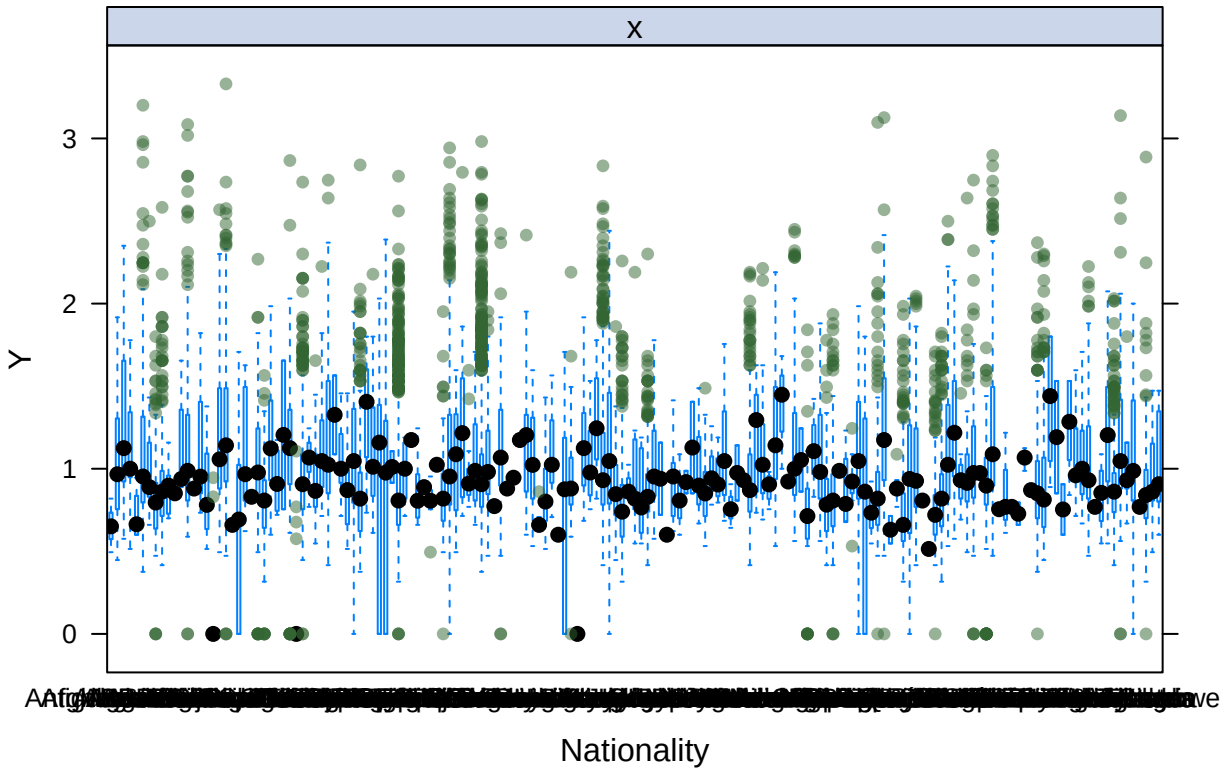
```
    keep(is.numeric) %>%
    select_if(~ any(is.na(.)))

#for factor variables
data_fct  = data %>%
    mutate(single_pos = as.factor(single_pos)) %>%
    select_if(~ !any(is.na(.))) %>%
    select(-transformed_value) %>%
    select_if(~ is.factor(.))


# vector of response
y <- data$transformed_value


featurePlot(data_num1,
y,
plot = "scatter",
span = .5,
labels = c("Predictors","Y"),
type = "p",
layout = c(4, 10))
```


volleys
max_pos_score
stamina
standing_tackle
strength
vision
short_passing
shot_power
sliding_tackle
sprint_speed
marking
penalties
positioning
reactions
interceptions
jumping
long_passing
long_shots
gk_kicking
gk_positioning
gk_reflexes
heading_accuracy
finishing
free_kick_accuracy
gk_diving
gk_handling
composure
crossing
curve
dribbling
aggression
agility
balance
ball_control
age
potential
special
acceleration
Y
Predictors

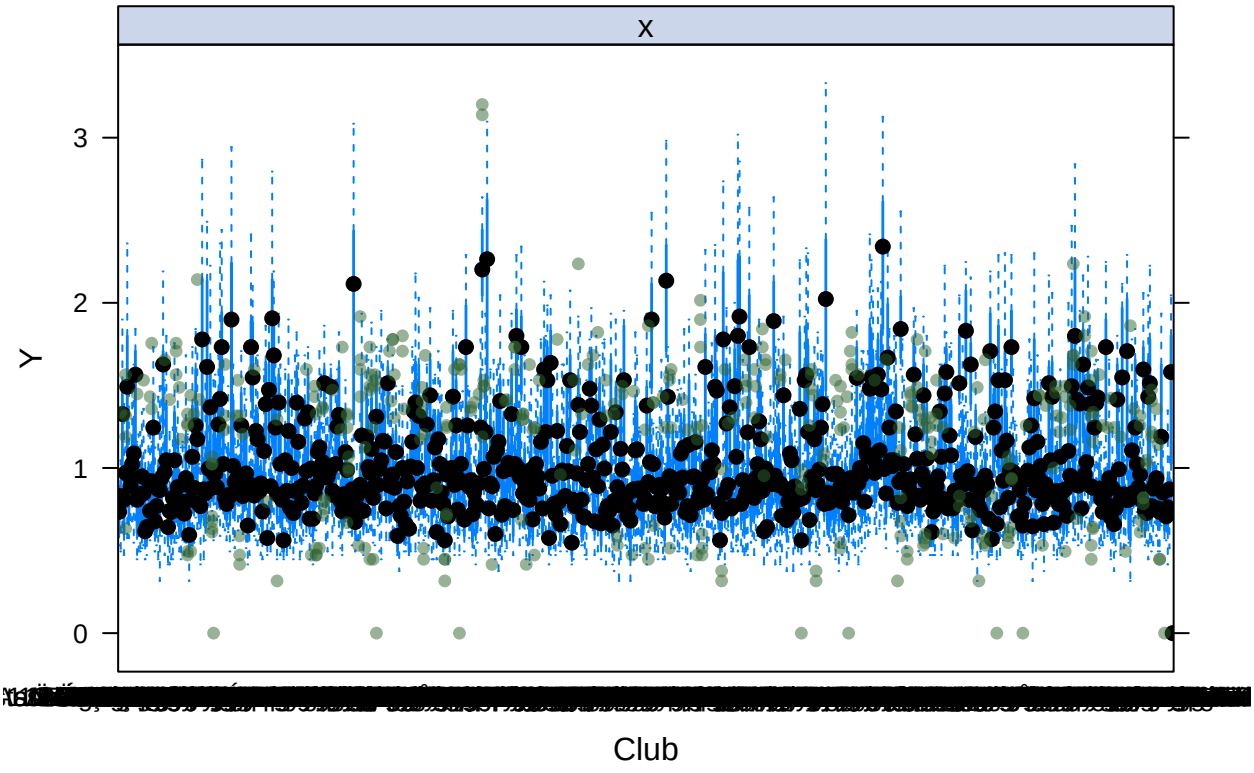
```
featurePlot(data_num2,
y,
plot = "scatter",
span = .5,
labels = c("Predictors","Y"),
type = "p",
layout = c(4, 7))
```


rwb
st
rf
rm
rs
rw
rb
rcb
rcm
rdm
ls
lw
lwb
ram
lcm
ldm
lf
lm
cm
lam
lb
lcb
cam
cb
cdm
cf
Y
Predictors

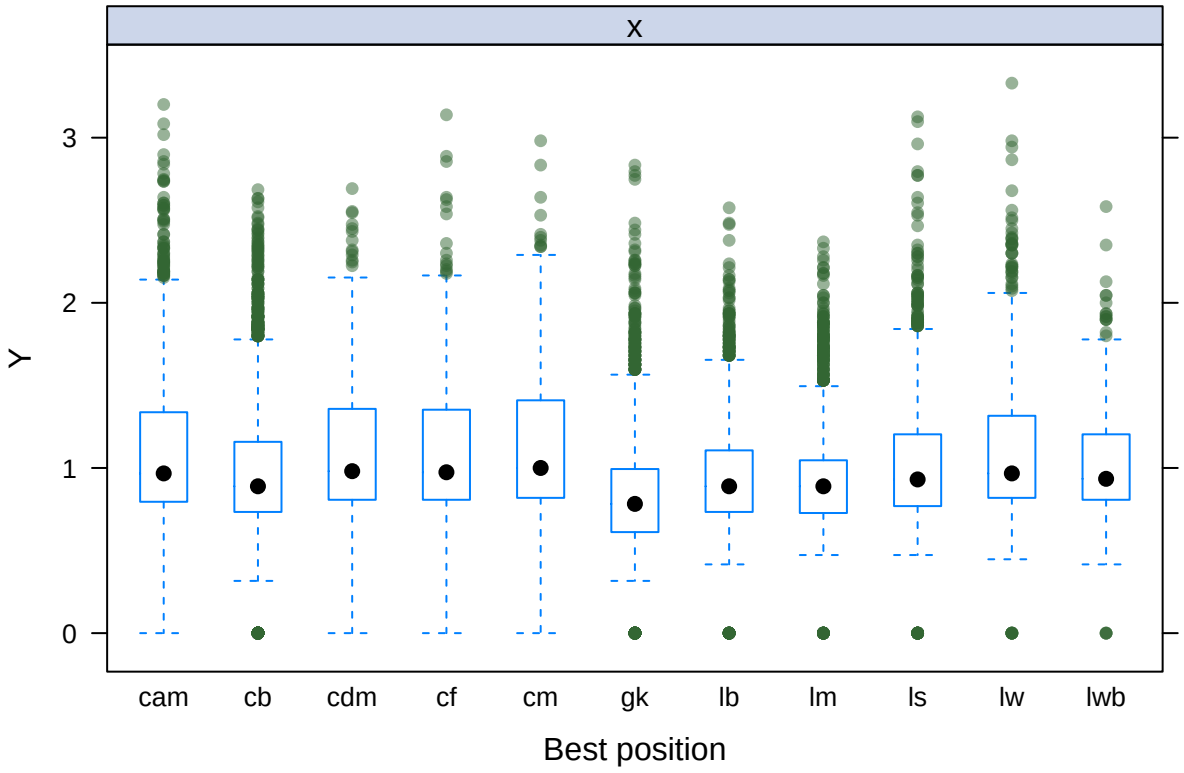
```
featurePlot(data$transformed_value, data_fct$nationality, "box", labels = c("Nationality","Y"))
```



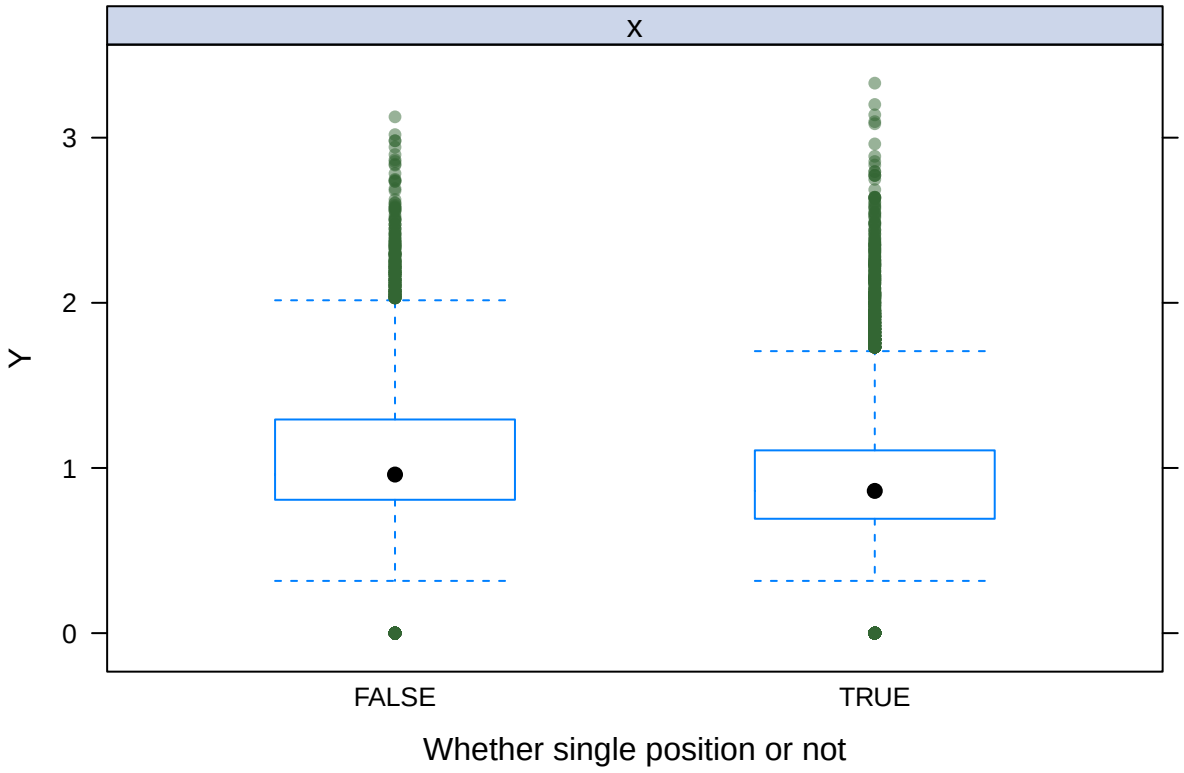

```
featurePlot(data$transformed_value, data_fct$club, "box", labels = c("Club","Y"))
```




```
featurePlot(data$transformed_value, data_fct$best_pos, "box", labels = c("Best position","Y"))
```




```
featurePlot(data$transformed_value, data_fct$single_pos, "box", labels = c("Whether single position or
```



Split the data set into training and testing data

```
trRows = createDataPartition(data$transformed_value, p = .75, list = FALSE)
train = data[trRows,] %>% write_csv("..\\exploratory analysis\\train.csv")
test = data[-trRows,] %>% write_csv("..\\exploratory analysis\\test.csv")
```