Ярылгасимов Султан

Отчет по домашнему заданию №4

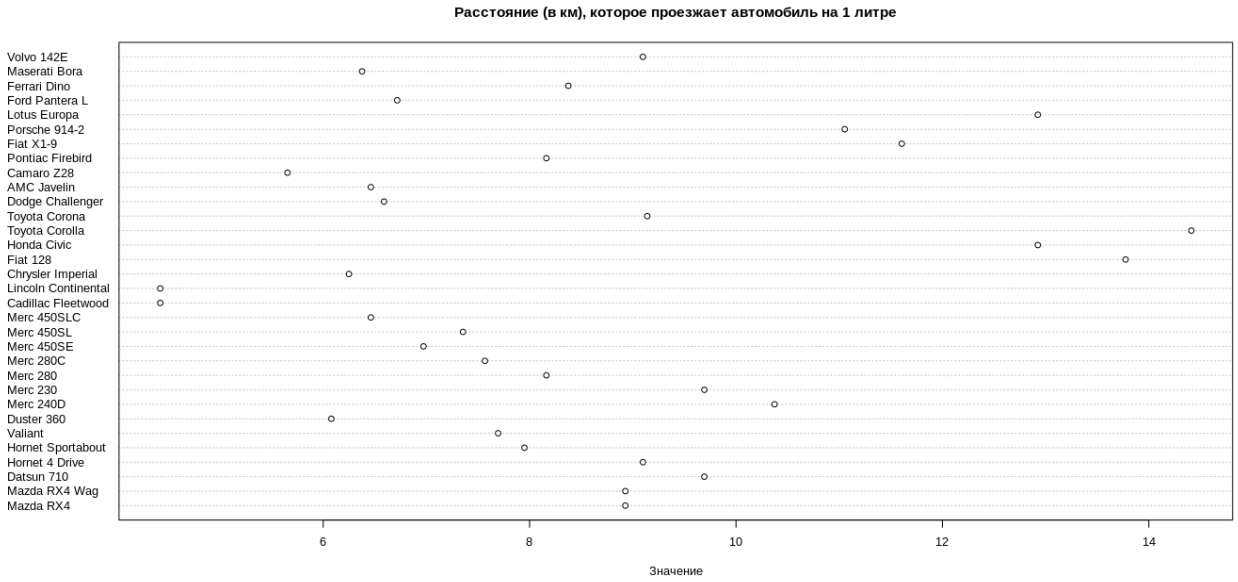
Задание 1. Выполнить задания из комментариев к занятию (см. приложенный R-файл).

1.1. Перерисовать график для расхода топлива в литрах + км

Код:

```
dotchart(mtcars$mpg * 0.425144, labels = row.names(mtcars),
         main="Расстояние (в км), которое проезжает автомобиль на 1 литре",
         xlab="Значение", cex = 0.8)
```

Вывод:

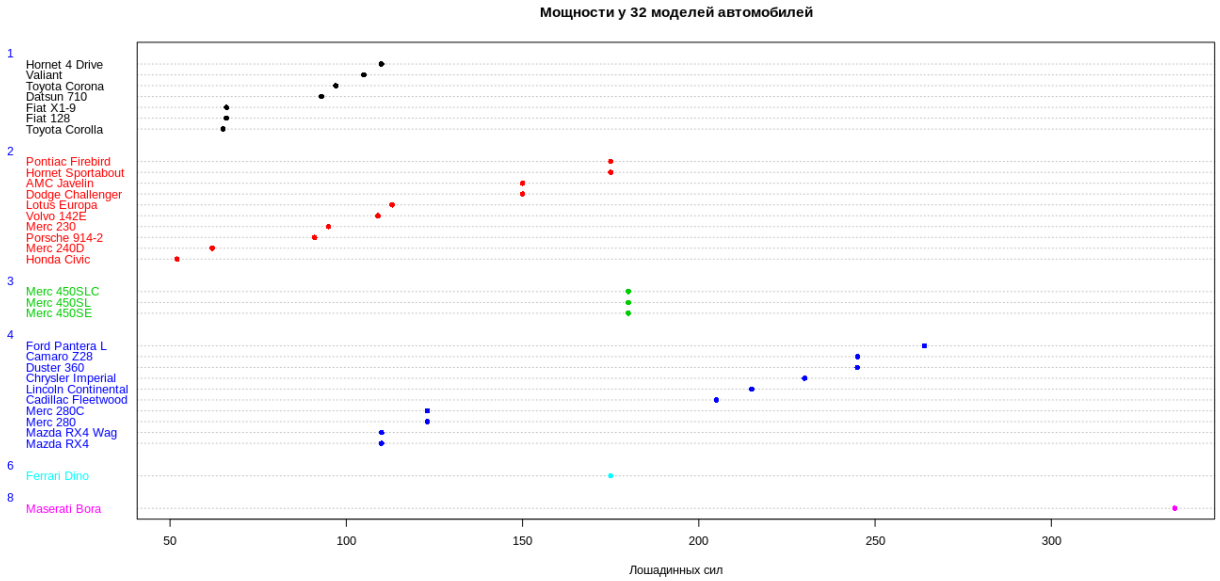


1.2. С помощью команды ?mtcars ознакомиться со структурой и описанием данных. Провести аналогичный анализ для какого-либо другого столбца.

Код:

```
x <- mtcars[order(mtcars$hp), ]
x$carb <- factor(x$carb)
cnt = 1
for (i in levels(factor(x$carb))) {
  x$color[x$carb==i] <- cnt
  cnt = cnt + 1
}

dotchart(x$hp, labels = row.names(x),
         groups = x$carb, gcolor = "blue", pch = 16,
         main="Мощности у 32 моделей автомобилей",
         xlab="Лошадинных сил", cex = 0.8, color = x$color)
```

Вывод:



Комментарий: Вывод мощностей автомобилей, сгруппированных по количеству карбюраторов.

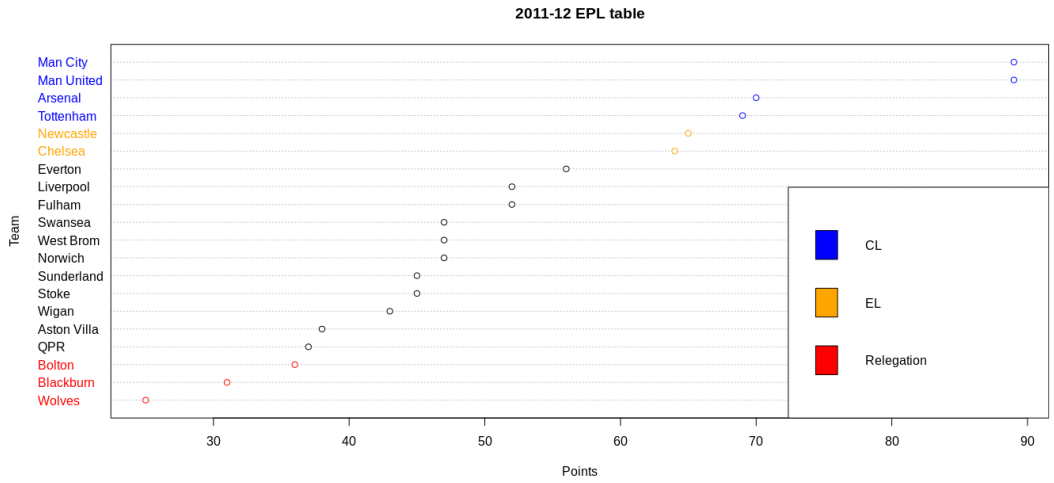
Задание 2. Реализовать анализ с помощью dotchart() и boxplot()+stripchart() для собственных данных.

2.1. dotchart
Код:

```
match_results = read.csv(file.choose())
attach(match_results)
match_results

get_goals_scored <- function(team_name, season) {
  get_goals_scored <- sum(subset(match_results, select=c(FTAG),
                           subset=(AwayTeam==team_name & Season==season)))
+
                    sum(subset(match_results, select=c(FTHG),
                           subset=(HomeTeam==team_name & Season==season)))
}

get_goals_conceded <- function(team_name, season) {
  get_goals_conceded <- sum(subset(match_results, select=c(FTHG),
                                subset=(AwayTeam==team_name &
Season==season))) +
                        sum(subset(match_results, select=c(FTAG),
                                subset=(HomeTeam==team_name &
Season==season)))
}

get_points <- function(team_name, season) {
  get_points <- 3 * length(unlist(subset(match_results, select=c(FTR),
                                subset=(HomeTeam==team_name & FTR=='H' &
Season==season)))) +
                3 * length(unlist(subset(match_results, select=c(FTR),
                                subset=(AwayTeam==team_name & FTR=='A' &
Season==season)))) +
```

```
                1 * length(unlist(subset(match_results, select=c(FTR),
                                         subset=((HomeTeam==team_name |
AwayTeam==team_name) & FTR=='D' & Season==season))))
}

get_league_table <- function(season) {
  teams = unique(subset(match_results, select=c(AwayTeam),
subset=(Season==season)))$AwayTeam
  league_table <- data.frame(
    team_name = teams,
    goals_scored = unlist(lapply(teams, season=season, get_goals_scored)),
    goals_conceded = unlist(lapply(teams, season=season,
get_goals_conceded)),
    points = unlist(lapply(teams, season=season, get_points))
  )
  league_table <- league_table[order(league_table$points,
                                     league_table$goals_scored -
league_table$goals_conceded,
                                     decreasing = T), ]
  # sort by points, then by goal difference (that's the rule)
  rownames(league_table) <- 1:nrow(league_table)
  get_league_table <- league_table
}

league_table <- get_league_table("2011-12")
league_table$color[1:4] <- 'blue'
league_table$color[5:6] <- 'orange'
league_table$color[7:17] <- 'black'
league_table$color[18:20] <- 'red'

dotchart(rev(league_table$points),
         labels=rev(league_table$team_name),
         main="2011-12 EPL table",
         xlab="Points",
         ylab="Team",
         color=rev(league_table$color)
         )  # revs for correct graph (top to bottom)
legend("bottomright",legend=c("CL", "EL", "Relegation"),
fill=c("blue","orange","red"))
```

Вывод:



Комментарий: сам не заметил как мой код разросся функциями, но все для того, чтобы вывести таблицу по итогам сезона в виде dotchart. Действительно, гораздо удобнее чем гистограммы и помещаются все команды.

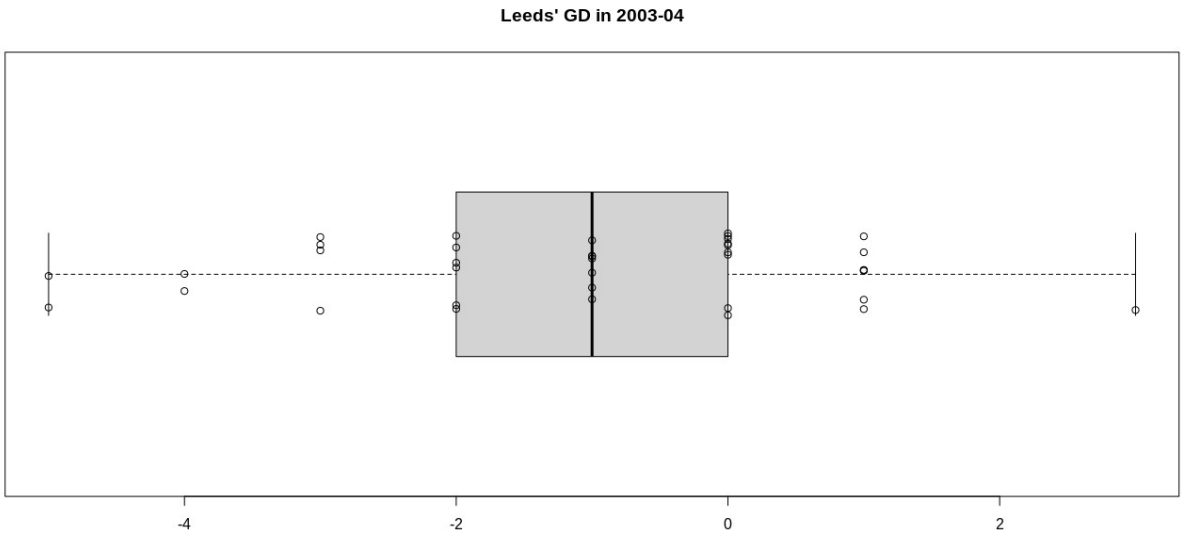
2.2. boxplot() + stripchar().

Код:

```
match_results = read.csv(file.choose())
attach(match_results)
match_results

get_goals_scored <- function(team_name, season) {
  get_goals_scored <- sum(subset(match_results, select=c(FTAG),
                           subset=(AwayTeam==team_name & Season==season)))
+
                    sum(subset(match_results, select=c(FTHG),
                           subset=(HomeTeam==team_name & Season==season)))
}

get_goals_conceded <- function(team_name, season) {
  get_goals_conceded <- sum(subset(match_results, select=c(FTHG),
                                subset=(AwayTeam==team_name &
Season==season))) +
                        sum(subset(match_results, select=c(FTAG),
                                subset=(HomeTeam==team_name &
Season==season)))
}

get_points <- function(team_name, season) {
  get_points <- 3 * length(unlist(subset(match_results, select=c(FTR),
                                subset=(HomeTeam==team_name & FTR=='H' &
Season==season)))) +
                3 * length(unlist(subset(match_results, select=c(FTR),
                                subset=(AwayTeam==team_name & FTR=='A' &
Season==season)))) +
                1 * length(unlist(subset(match_results, select=c(FTR),
                                subset=((HomeTeam==team_name |
AwayTeam==team_name) & FTR=='D' & Season==season))))
}

get_league_table <- function(season) {
  teams = unique(subset(match_results, select=c(AwayTeam),
subset=(Season==season)))$AwayTeam
  league_table <- data.frame(
    team_name = teams,
    goals_scored = unlist(lapply(teams, season=season, get_goals_scored)),
    goals_conceded = unlist(lapply(teams, season=season,
get_goals_conceded)),
    points = unlist(lapply(teams, season=season, get_points))
  )
  league_table <- league_table[order(league_table$points,
                                     league_table$goals_scored -
league_table$goals_conceded,
                                     decreasing = T), ]
  # sort by points, then by goal difference (that's the rule)
  rownames(league_table) <- 1:nrow(league_table)
  get_league_table <- league_table
}

league_table <- get_league_table("2011-12")
league_table$color[1:4] <- 'blue'
league_table$color[5:6] <- 'orange'
league_table$color[7:17] <- 'black'
league_table$color[18:20] <- 'red'
```

```
dotchart(rev(league_table$points),
         labels=rev(league_table$team_name),
         main="2011-12 EPL table",
         xlab="Points",
         ylab="Team",
         color=rev(league_table$color)
         )  # revs for correct graph (top to bottom)
legend("bottomright",legend=c("CL", "EL", "Relegation"),
fill=c("blue","orange","red"))

# let's see how particular team scores goals per season
seasons = unique(Season)
team_name='Leeds'
total_goals = sapply(seasons, team_name=team_name, get_goals_scored)
total_goals[total_goals==0] = NA  # team has skipped many seasons
total_goals  # here we see importance of NA
boxplot(total_goals, horizontal = T)
stripchart(total_goals, add=T)

# let's look at GD
home_gd=FTHG[HomeTeam==team_name & Season=='2003-04'] -
        FTAG[HomeTeam==team_name & Season=='2003-04']  # home gds
away_gd=FTAG[AwayTeam==team_name & Season=='2003-04'] -
        FTHG[AwayTeam==team_name & Season=='2003-04']  # away gds
total_gd <- c(home_gd, away_gd)  # all gds

boxplot(total_gd,
        horizontal=T,
        outline=F,
        main="Leeds GD in 2003-04")
stripchart(total_gd,
           vertical=F,
           method='jitter',
           pch=21,
           jitter=0.1,
           add=T)
```

Вывод:

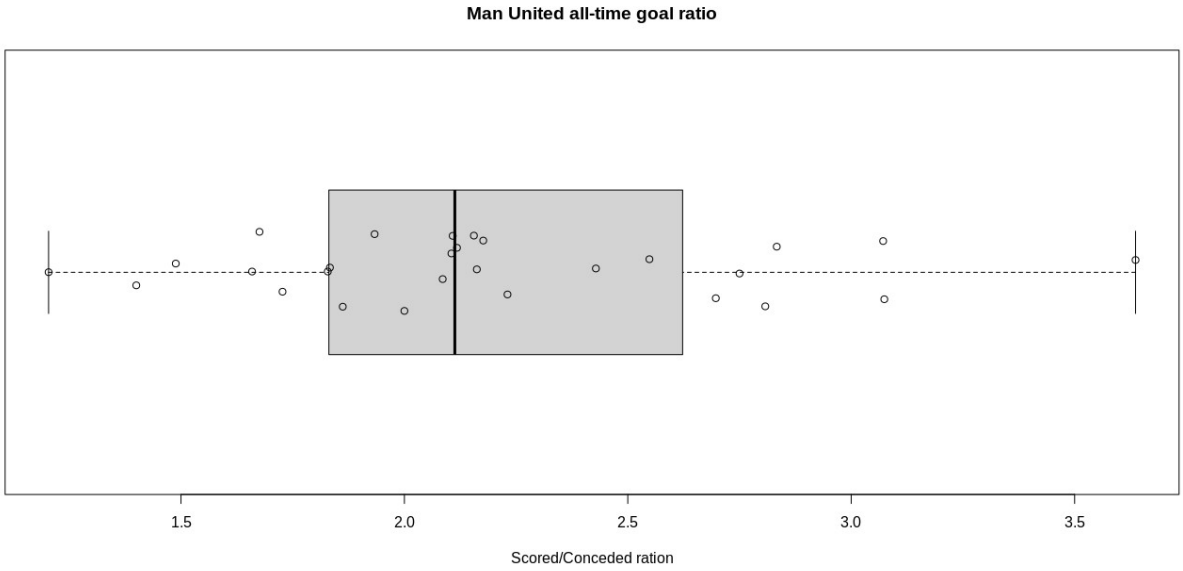


Комментарий: Продолжая тему прошлого домашнего занятия, вывожу разницу мячей определенной команды в сезоне, выглядит так конечно привлекательнее чем бездушные «ящики», но с дискретными данными все равно выглядит не очень хорошо.

Код:

```
team_name = 'Man United'
seasons = unique(Season)
scored_goals = sapply(seasons, team_name=team_name, get_goals_scored)
conceded_goals = sapply(seasons, team_name=team_name, get_goals_conceded)
ratio = scored_goals / conceded_goals
boxplot(ratio,
        horizontal=T,
        outline=F,
        main = "Man United all-time goal ratio",
        xlab = "Scored/Conceded ration")
stripchart(ratio,
           vertical=F,
           method='jitter',
           pch=21,
           jitter=0.1,
           add=T)
```

Вывод:



Комментарий: «Придумав» непрерывную величину, вывели график, который выглядит более красиво. График показывает отношение забитых и пропущенных голов команды в сезоне за все время существования турнира.

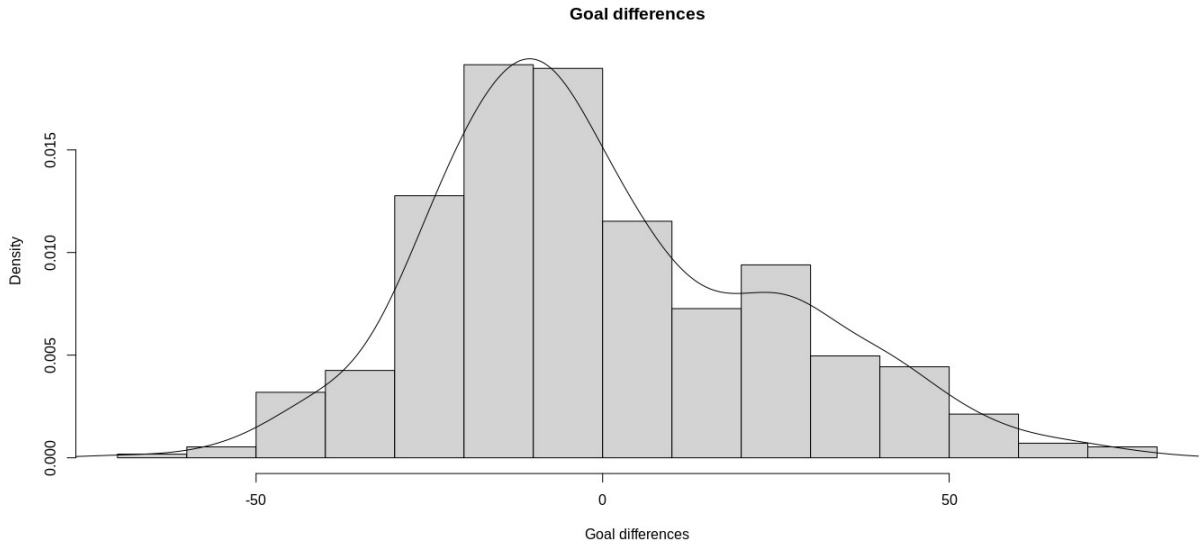
Задание 3. Проверить, являются ли наблюдения выбросами с точки зрения формальных статистических критериев Граббса и Q-теста Диксона. Снабдить решение необходимой визуализацией. Для анализа данные подготовить самостоятельно, обеспечив условия применимости критериев.

3.1. Критерий Граббса.
Комментарий: для применения критерия Граббса, нужно убедиться, что данные обладают нормальным распределением. Самым простым способом является нарисовать гистограмму и проверить имеет ли она колоколообразный симметричный вид. В дополнение к этому, выведена ядерная оценка поверх гистограммы.

Код:

```
seasons = unique(Season)
gds = c()
for (i in seasons) {
  teams = unique(match_results$AwayTeam[Season==i])
  gds <- c(gds,unlist(lapply(teams, season=i, get_goals_scored)) -
                  unlist(lapply(teams, season=i, get_goals_conceded)))
}
gds
density(gds)
hist(gds, freq=F,
     main="Goal differences",
     xlab="Goal differences")
lines(density(gds))
```

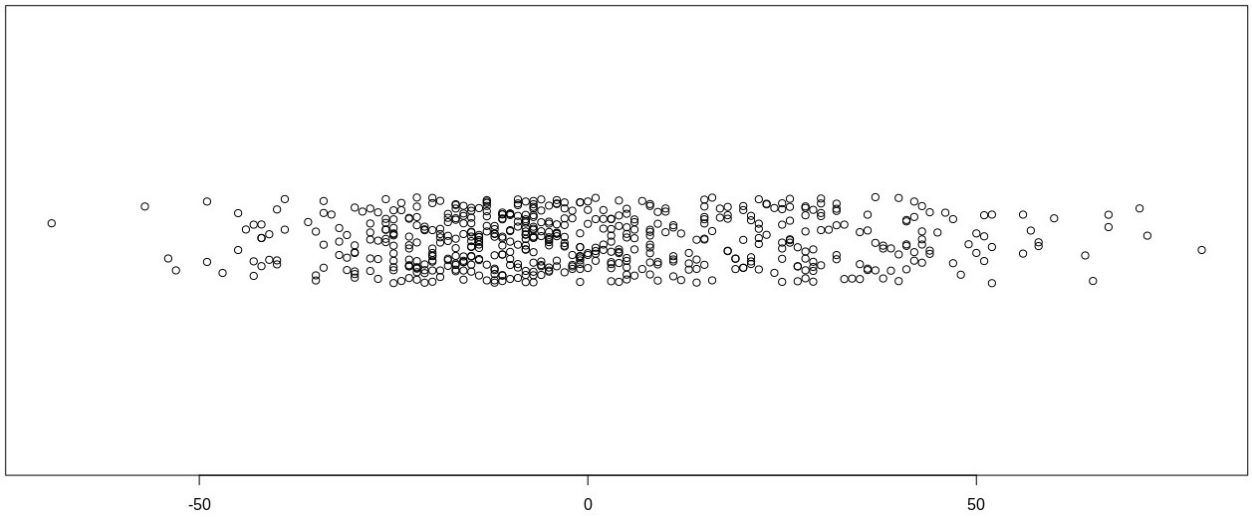
Вывод:



Комментарий: Далее, мы можем применить тест Граббса. Сначала, рассмотрим на не измененных данных, и проверим являются ли максимальные и минимальные значения выбросами.
Код:

```
library(outliers)
stripchart(gds,
           method='jitter',
           pch=21,
           jitter=0.1)
grubbs.test(gds, opposite=T)  # check if outlier on the opposite tail (min)
grubbs.test(gds)  # check if max is outlier
grubbs.test(gds, type=11)  # check if outlier is on the both sides
```

Вывод:




```
        Grubbs test for one outlier

data:  gds
G = 2.79821, U = 0.98607, p-value = 1
alternative hypothesis: lowest value -69 is an outlier

> grubbs.test(gds)  # check if max is outlier

        Grubbs test for one outlier

data:  gds
G = 3.20375, U = 0.98174, p-value = 0.3652
alternative hypothesis: highest value 79 is an outlier

> grubbs.test(gds, type=11)  # check if outlier is on the both sites

        Grubbs test for two opposite outliers

data:  gds
G = 6.00196, U = 0.96786, p-value = 1
alternative hypothesis: -69 and 79 are outliers
```

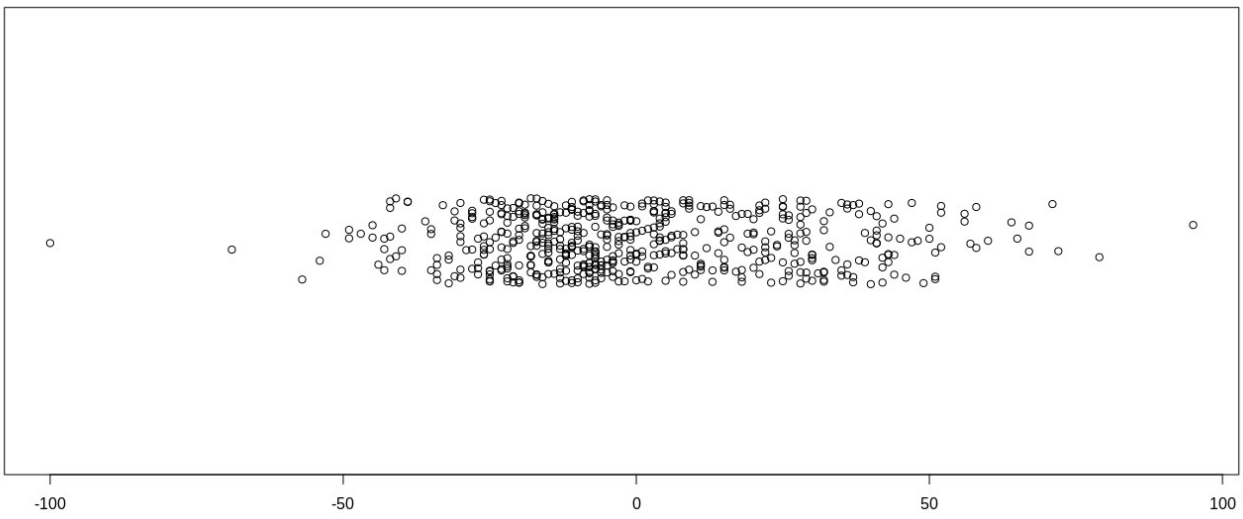
Комментарий: p-value для всех тестов больше чем 0.05 и поэтому мы не имеем достаточно доказательств, чтобы принять альтернативную гипотезу, то есть максимальные и минимальные значения не являются выбросами. Теперь «испортим» данные и добавим выбросы.

Код:

```
gds <- c(gds, 95)  # add 95 as an outlier
grubbs.test(gds)
gds <- gds[gds!=95]

gds <-c(gds, -100)  # add -100 as an outlier
grubbs.test(gds)

gds <- c(gds, 95)
grubbs.test(gds, type=11)  # outliers on both sides
stripchart(gds,
           method='jitter',
           pch=21,
           jitter=0.1)
gds = gds[gds!=95]  # delete outliers
gds = gds[gds!=-100]
```

Вывод:

```
        Grubbs test for one outlier

data:  gds
G = 3.79954, U = 0.97436, p-value = 0.03731
alternative hypothesis: highest value 95 is an outlier

> gds <- gds[gds!=95]
>
> gds <-c(gds, -100)  # add 100 as an outlier
> grubbs.test(gds)

        Grubbs test for one outlier

data:  gds
G = 3.99398, U = 0.97167, p-value = 0.01638
alternative hypothesis: lowest value -100 is an outlier

>
> gds <- c(gds, 95)
> grubbs.test(gds, type=11)  # outliers on both sides

        Grubbs test for two opposite outliers

data:  gds
G = 7.71066, U = 0.94735, p-value = 0.00546
alternative hypothesis: -100 and 95 are outliers
```

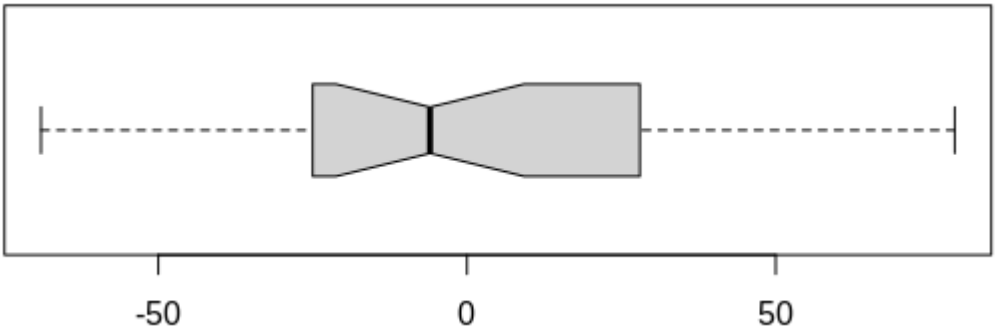


Комментарий: добавили ложные данные с каждой из сторон и с обоих одновременно. Данные подобраны так, чтобы они являлись p-value для них оказался ниже 0.05. То, как это выглядит так же доступно визуально.

3.2. Q-test Диксона.
Код:

```
short_gds <- c(min(gds), max(gds), sample(gds, 28))
boxplot(short_gds, notch=T, horizontal = T)
dixon.test(short_gds)  # check max
dixon.test(short_gds, type=11)  # check 1 elements from both sides
dixon.test(short_gds, opposite = T)  # check min
```

Вывод:




```
> dixon.test(short_gds)  # check max

        Dixon test for outliers

data:  short_gds
Q = 0.30508, p-value = 0.2888
alternative hypothesis: highest value 79 is an outlier

> dixon.test(short_gds, type=11)  # check 1 elements from both sides

        Dixon test for outliers

data:  short_gds
Q = 0.25203, p-value = 0.1594
alternative hypothesis: highest value 79 is an outlier

> dixon.test(short_gds, opposite = T)  # check min

        Dixon test for outliers

data:  short_gds
Q = 0.26786, p-value = 0.447
alternative hypothesis: lowest value -69 is an outlier
```

Комментарий: Функция реализованная в R требует от данных быть ограниченными 30 элементами, поэтому были выбраны максимальные, минимальные и 28 случайных элементов. Для их визуализации больше подошел boxplot. Проведена проверка того являются ли максимальные и минимальные данные выбросами, аналогично с предыдущим тестом. Проверим на аналогичных выбросах с первым тестом.

Код:

```
# increase max from 79 to 95
dixon.test(c(short_gds[-1], 95))
# decrease min from -69 to -100
dixon.test(c(short_gds[-2], -100))
```

Вывод:

```
        Dixon test for outliers

data:  c(short_gds[-1], 95)
Q = 0.38525, p-value = 0.08524
alternative hypothesis: highest value 95 is an outlier

> # decrease min from -69 to -100
> dixon.test(c(short_gds[-2], -100))

        Dixon test for outliers

data:  c(short_gds[-2], -100)
Q = 0.39437, p-value = 0.07244
alternative hypothesis: lowest value -100 is an outlier
```

Комментарий: Как можно увидеть те значения, которые прошлый тест считал выбросами тут таковыми не являются, но скорее всего это вызвано тем, что мы берем ограниченное количество данных, что может снижать точность.

Задание 4. Воспользоваться пакетами и функциями mice, mitools, pan, mix, aregImpute и transcan для заполнения пропусков в данных. Пропуски внести вручную и сравнить результаты заполнения с истинными значениями.

4.1. mice
Код:

```
league_table = get_league_table("2011-12")  # take table back
league_table
library(mice)
# delete 1 values from each int column at random
league_table$goals_conceded[12]
league_table$goals_scored[5]
league_table$points[10]
league_table$goals_conceded[12] = NA
league_table$goals_scored[5] = NA
league_table$points[10] = NA
league_table
summary(league_table)
md.pattern(league_table)
imputed_Data <- mice(league_table, m=3, method = 'pmm')
summary(imputed_Data)
completeData <- complete(imputed_Data)
completeData$goals_conceded[12]
completeData$goals_scored[5]
completeData$points[10]
```

Вывод:

```
> league_table$goals_conceded[12]
[1] 66
> league_table$goals_scored[5]
[1] 56
> league_table$points[10]
[1] 47
>
```

```
> completeData$goals_conceded[12]
[1] 62
> completeData$goals_scored[5]
[1] 65
> completeData$points[10]
[1] 47
>
```

Комментарий: По первым двум параметрам предсказание пропущенных данных идет не так хорошо и с каждым разом получаются числа с большим разбросом, а вот третий параметр почти всегда угадывается правильно с погрешностью в 1-2.

Код:

```
library(mice)
md.pattern(match_results)
imputed_Data <- mice(match_results, m=5, method = 'pmm')
```

```
summary(imputed_Data)
completeData <- complete(imputed_Data)
completeData
```

Вывод:

```
> completeData
    Season             DateTime         HomeTeam         AwayTeam FTHG FTAG FTR HTHG HTAG  HTR Referee HS AS HST AST HC AC HF AF HY AY HR AR
1  1993-94 1993-08-14T00:00:00Z          Arsenal         Coventry    0    3   A    0    1 <NA>    <NA> 14 17   7   9  5  6 17 10  4  1  0  0
2  1993-94 1993-08-14T00:00:00Z      Aston Villa              QPR    4    1   H    2    1 <NA>    <NA> 17  8  10   2  7  2 15 23  0  1  0  0
3  1993-94 1993-08-14T00:00:00Z          Chelsea        Blackburn    1    2   A    0    1 <NA>    <NA>  9 14   2   3  5  4 17 12  0  2  0  0
4  1993-94 1993-08-14T00:00:00Z        Liverpool   Sheffield Weds    2    0   H    0    0 <NA>    <NA>  7 17   5  11  1  7 11 12  5  3  0  1
5  1993-94 1993-08-14T00:00:00Z         Man City            Leeds    1    1   D    0    1 <NA>    <NA> 23 11   9   1  8  5 14 19  6  3  0  1
6  1993-94 1993-08-14T00:00:00Z        Newcastle        Tottenham    0    1   A    0    1 <NA>    <NA> 16 10   5   4  9  5 14 10  1  2  0  0
7  1993-94 1993-08-14T00:00:00Z           Oldham          Ipswich    0    3   A    0    2 <NA>    <NA> 13 14   5   7  3  3  6 12  0  3  0  0
8  1993-94 1993-08-14T00:00:00Z Sheffield United          Swindon    3    1   H    2    1 <NA>    <NA>  9 17   3   5  3  5  5  8  1  0  0  0
9  1993-94 1993-08-14T00:00:00Z      Southampton          Everton    0    2   A    0    1 <NA>    <NA>  7 10   2   2  3  6  9 13  1  3  0  0
10 1993-94 1993-08-14T00:00:00Z         West Ham        Wimbledon    0    2   A    0    0 <NA>    <NA> 14  7   7   5  8  4  5 17  0  2  0  0
11 1993-94 1993-08-15T00:00:00Z          Norwich      Man United    0    2   A    0    2 <NA>    <NA> 17  5   6   2 13  3 11 12  4  4  0  0
12 1993-94 1993-08-16T00:00:00Z        Tottenham          Arsenal    0    1   A    0    1 <NA>    <NA> 17 10   8   3 11  3 12 19  2  2  0  0
```

```
> match_results
    Season             DateTime         HomeTeam         AwayTeam FTHG FTAG FTR HTHG HTAG  HTR Referee HS AS HST AST HC AC HF AF HY AY HR AR
1  1993-94 1993-08-14T00:00:00Z          Arsenal         Coventry    0    3   A   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
2  1993-94 1993-08-14T00:00:00Z      Aston Villa              QPR    4    1   H   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
3  1993-94 1993-08-14T00:00:00Z          Chelsea        Blackburn    1    2   A   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
4  1993-94 1993-08-14T00:00:00Z        Liverpool   Sheffield Weds    2    0   H   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
5  1993-94 1993-08-14T00:00:00Z         Man City            Leeds    1    1   D   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
6  1993-94 1993-08-14T00:00:00Z        Newcastle        Tottenham    0    1   A   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
7  1993-94 1993-08-14T00:00:00Z           Oldham          Ipswich    0    3   A   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
8  1993-94 1993-08-14T00:00:00Z Sheffield United          Swindon    3    1   H   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
9  1993-94 1993-08-14T00:00:00Z      Southampton          Everton    0    2   A   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
10 1993-94 1993-08-14T00:00:00Z         West Ham        Wimbledon    0    2   A   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
11 1993-94 1993-08-15T00:00:00Z          Norwich      Man United    0    2   A   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
12 1993-94 1993-08-16T00:00:00Z        Tottenham          Arsenal    0    1   A   NA   NA <NA>    <NA> NA NA  NA  NA NA NA NA NA NA NA NA NA
```

Комментарий: Вплоть до 2000 года в Лиге не велась статистика таких показателей, как счет в середине игры, кол-во желтых, красных карточек, фоллов и так далее, но MICE достаточно успешно заполнил пробелы, и даже так что в середине игры счет не больше чем в конце.

4.2. aregImpute()

Код:

```
attach(mu_liverpool)
mu_liverpool = rbind(match_results[(match_results$HomeTeam=='Man United' &
match_results$AwayTeam=='Liverpool'),],
              match_results[(match_results$AwayTeam=='Liverpool' &
match_results$HomeTeam=='Man United'),])
mu_liverpool
actual=c()
space_positions=sample(1:nrow(mu_liverpool), 5)
for (i in space_positions) {
  actual<-append(actual, mu_liverpool$FTHG[i])
  mu_liverpool$FTHG[i] = NA
}
impute_arg <- aregImpute(formula=~FTHG + FTAG, data=mu_liverpool, nk=0)
mu_liverpool
impute_arg
impute_arg$imputed$FTHG[,1]
actual
mu_liverpool
```

Вывод:

```
 4306  4403 10133 30021 65461
    3     3     2     2     3
> actual
[1] 1 2 0 2 0
```

Комментарий: Наверное, будет наивно ожидать заполнение результатов матчей 1 в 1, но примерно результаты все таки попадают в погрешность.

4-задание мне далось сложнее остальных, на данный момент. Очень много функций, и я не успел со всеми разобраться. Боюсь, что не успею это сделать до конца срока сдачи. На данный момент, поддались мне MICE и aregImpute. К тому же, VIM судя по всему не имеет совместимости с новыми версиями r, и никак не установился.

Задание 5. *Изучить альтернативные методы проверки аномальности данных.*

Комментарий: третий тест, который обычно идет с теми, что мы изучили — тест Рознера. Его главное преимущество — то, что его можно использовать для обнаружения нескольких выбросов. В R функция реализована в библиотеке EnvStats под названием rosnerTest и в качестве параметров требует как минимум данные и предполагаемое количество выбросов.

Код:
```
library(EnvStats)
rosnerTest(gds, k=2)
rosnerTest(c(gds, 100), k=2)
```

Вывод:

	i	Mean.i	SD.i	Value	Obs.Num	R.i+1	lambda.i+1	Outlier
1	0	0.0000000	24.65859	79	486	3.203751	3.894791	FALSE
2	1	-0.1403197	24.45411	72	512	2.950029	3.894328	FALSE
3	2	-0.2686833	24.28531	71	335	2.934642	3.893864	FALSE

	i	Mean.i	SD.i	Value	Obs.Num	R.i+1	lambda.i+1	Outlier
1	0	0.1769912	24.99334	100	565	3.993984	3.895254	TRUE
2	1	0.0000000	24.65859	79	486	3.203751	3.894791	FALSE