Entrega

> Github link aonde tem o link do Colab
> https://github.com/BrunaKrasotaMatos/ParadigmasProgramacao/blob/1661f56509f447f788dd8b4144cd436668e5c1a7/Trilha7_ExercicioAprofundamentoBrunaKrasotaMatos.ipynb

Exercício de Aprofundamento - Trilha 7

Faça as manipulações e explorações visuais de acordo com as perguntas que precisam ser respondidas

Análise de dados da NFL

Pacote do R: https://cran.r-project.org/web/packages/nflfastR/index.html

Este pacote permite que dados da NFL sejam analisados, jogada a jogada, habilitando diversos tipos de tomada de decisão a partir de manipulação dos dados e geração de gráficos.

Nesta atividade de aprofundamento, vamos explorar itens estudados tanto na trilha 6 com o pacote **Tidyverse** quanto na trilha 7 com o pacote **ggplot2**.

Algumas partes desta atividade já estão prontas, como por exemplo, o carregamento do conjunto de dados geral, a impressão dos escudos dos times e a segmentação de sub-conjuntos de dados para permitir uma manipulação mais simples na atividade.

Começamos então, com a instalação do pacote *nflfastR* e os carregamentos dos pacotes necessários.

```
install.packages("nflfastR")
install.packages("ggimage")
install.packages("imager")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘RApiSerialize’, ‘stringfish’, ‘globals’, ‘liste

Installing package into ‘/usr/local/lib/R/site-library’
```

```
(as ‘lib’ is unspecified)

also installing the dependencies ‘gridGraphics’, ‘yulab.utils’, ‘ggfun’, ‘ggplot

Warning message in install.packages("ggimage"):
“installation of package ‘magick’ had non-zero exit status”
Warning message in install.packages("ggimage"):
“installation of package ‘ggimage’ had non-zero exit status”
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependencies ‘bmp’, ‘tiff’, ‘png’, ‘jpeg’, ‘readbitmap’, ‘dow
```

```
install.packages("ggimage")
```

```
Installing package into ‘/usr/local/lib/R/site-library’
(as ‘lib’ is unspecified)

also installing the dependency ‘magick’

Warning message in install.packages("ggimage"):
“installation of package ‘magick’ had non-zero exit status”
Warning message in install.packages("ggimage"):
“installation of package ‘ggimage’ had non-zero exit status”
```

```
library(nflfastR)

library(tidyverse)
library(ggplot2)

library(imager)
#library(ggimage)
```

Como este pacote permite baixar dados de todas as temporadas, jogada a jogada, desde 1999, faremos um recorte apenas de 2014. A escolha deste ano foi aleatória, mesmo que possa parecer que foi escolhido de forma proposital por ser ultimo ano no qual *Seattle Seahawks* ganhou o *Super Bowl* (que é o jogo final da temporada e define o vencedor do campeonato). Fique a vontade para escolher qualquer outro ano, caso deseje estudar.

> Contudo, para este exercício de aprofundamento, **mantenha o ano de 2014**.

```
temporada <- load_pbp(2014) #Carregamento dos dados, jogada a jogada, de 2014
```

Repare que para a seleção do subconjunto de dados, foi informado o ano da temporada desejado.

Poderiam ser um intervalo de outros anos, para isso, seria necessário definir o valor como **anoInicio:anoFim**, por exemplo: 2014:2018 e neste caso os dados seriam de 2014 até 2018.

> temporada <- load_pbp(2014:2018)

Repare que este conjunto de dados de pbp (*play-by-play* -- jogada a jogada) possui muitas variáveis. Ao chamar a função *names* colocando o nome do conjunto de dados, são retornadas todas as variáveis. Execute o bloco abaixo e conheça quais são estas variáveis.

```
names(temporada)
```

'play_id' · 'game_id' · 'old_game_id' · 'home_team' · 'away_team' · 'season_type' · 'week' · 'posteam' · 'posteam
'side_of_field' · 'yardline_100' · 'game_date' · 'quarter_seconds_remaining' · 'half_seconds_remaining' · 'game
'game_half' · 'quarter_end' · 'drive' · 'sp' · 'qtr' · 'down' · 'goal_to_go' · 'time' · 'yrdln' · 'ydstogo' · 'ydsnet' · 'desc'
'yards_gained' · 'shotgun' · 'no_huddle' · 'qb_dropback' · 'qb_kneel' · 'qb_spike' · 'qb_scramble' · 'pass_length'
'air_yards' · 'yards_after_catch' · 'run_location' · 'run_gap' · 'field_goal_result' · 'kick_distance' · 'extra_point_re
'two_point_conv_result' · 'home_timeouts_remaining' · 'away_timeouts_remaining' · 'timeout' · 'timeout_team'
'td_player_name' · 'td_player_id' · 'posteam_timeouts_remaining' · 'defteam_timeouts_remaining' · 'total_home
'total_away_score' · 'posteam_score' · 'defteam_score' · 'score_differential' · 'posteam_score_post' · 'defteam_
'score_differential_post' · 'no_score_prob' · 'opp_fg_prob' · 'opp_safety_prob' · 'opp_td_prob' · 'fg_prob' · 'safet
'extra_point_prob' · 'two_point_conversion_prob' · 'ep' · 'epa' · 'total_home_epa' · 'total_away_epa' · 'total_hom
'total_away_rush_epa' · 'total_home_pass_epa' · 'total_away_pass_epa' · 'air_epa' · 'yac_epa' · 'comp_air_epa
'total_home_comp_air_epa' · 'total_away_comp_air_epa' · 'total_home_comp_yac_epa' · 'total_away_comp_y
'total_home_raw_air_epa' · 'total_away_raw_air_epa' · 'total_home_raw_yac_epa' · 'total_away_raw_yac_epa
'home_wp' · 'away_wp' · 'wpa' · 'vegas_wpa' · 'vegas_home_wpa' · 'home_wp_post' · 'away_wp_post' · 'vegas_
'total_home_rush_wpa' · 'total_away_rush_wpa' · 'total_home_pass_wpa' · 'total_away_pass_wpa' · 'air_wpa'
'comp_air_wpa' · 'comp_yac_wpa' · 'total_home_comp_air_wpa' · 'total_away_comp_air_wpa' · 'total_home_c
'total_away_comp_yac_wpa' · 'total_home_raw_air_wpa' · 'total_away_raw_air_wpa' · 'total_home_raw_yac_
'total_away_raw_yac_wpa' · 'punt_blocked' · 'first_down_rush' · 'first_down_pass' · 'first_down_penalty' · 'third_
'third_down_failed' · 'fourth_down_converted' · 'fourth_down_failed' · 'incomplete_pass' · 'touchback' · 'intercep
'punt_inside_twenty' · 'punt_in_endzone' · 'punt_out_of_bounds' · 'punt_downed' · 'punt_fair_catch' · 'kickoff_in
'kickoff_in_endzone' · 'kickoff_out_of_bounds' · 'kickoff_downed' · 'kickoff_fair_catch' · 'fumble_forced' · 'fumble
'fumble_out_of_bounds' · 'solo_tackle' · 'safety' · 'penalty' · 'tackled_for_loss' · 'fumble_lost' · 'own_kickoff_reco
'own_kickoff_recovery_td' · 'qb_hit' · 'rush_attempt' · 'pass_attempt' · 'sack' · 'touchdown' · 'pass_touchdown' ·
'return_touchdown' · 'extra_point_attempt' · 'two_point_attempt' · 'field_goal_attempt' · 'kickoff_attempt' · 'punt_
'complete_pass' · 'assist_tackle' · 'lateral_reception' · 'lateral_rush' · 'lateral_return' · 'lateral_recovery' · 'passer
'passer_player_name' · 'passing_yards' · 'receiver_player_id' · 'receiver_player_name' · 'receiving_yards' · 'rus
'rusher_player_name' · 'rushing_yards' · 'lateral_receiver_player_id' · 'lateral_receiver_player_name' · 'lateral_
'lateral_rusher_player_id' · 'lateral_rusher_player_name' · 'lateral_rushing_yards' · 'lateral_sack_player_id' ·
'lateral_sack_player_name' · 'interception_player_id' · 'interception_player_name' · 'lateral_interception_playe

Para conhecer os times que jogam na NFL, é possível ter um retorno de dados básicos dos cada um deles. Este retorno básico pode ser transformado em um data frame, para posteriormente, ser utilizado como filtro da estrutura.

```
times <- teams_colors_logos %>% unique()
```

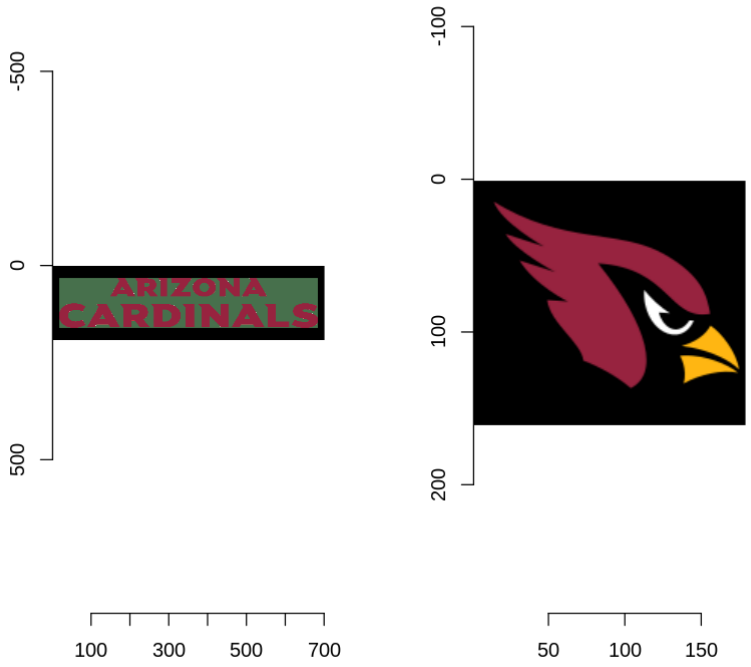
```
names(times)
```

'team_abbr' · 'team_name' · 'team_id' · 'team_nick' · 'team_color' · 'team_color2' · 'team_color3' · 'team_color4' · 'team_logo_wikipedia' · 'team_logo_espn' · 'team_wordmark'

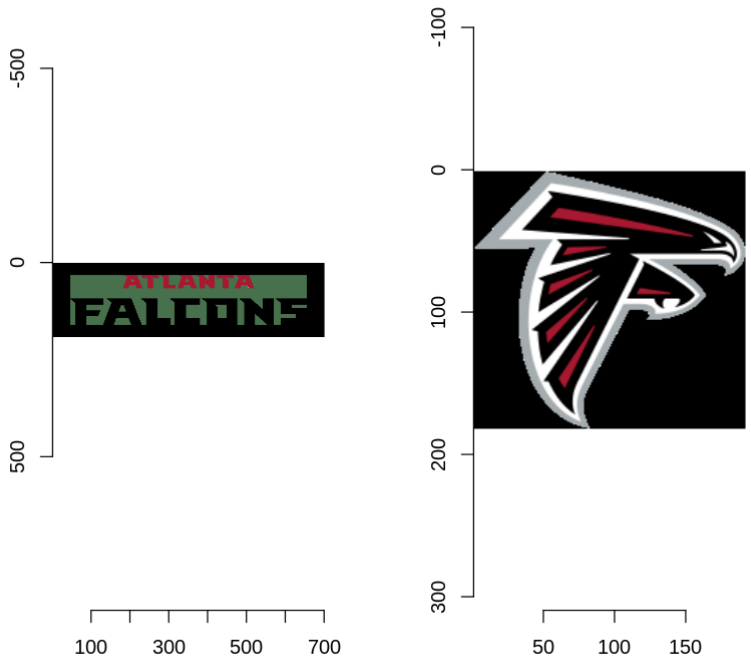
Aproveitando que estamos observando os times, é possível mostrar todos os seus escudos e nomes. Para isso, execute o bloco de codigo abaixo, e veja como é a saída:

```
for (i in 1:dim(times)[1]){
    par(mfrow=c(1,2))
    load.image(as.character(times[i,'team_wordmark'])) %>% plot ;
    load.image(as.character(times[i,'team_logo_wikipedia'])) %>% plot ;
    print(paste(times[i,'team_name'],times[i,'team_abbr'],sep=' >> '));
    par(mfrow=c(1,1))
}
```

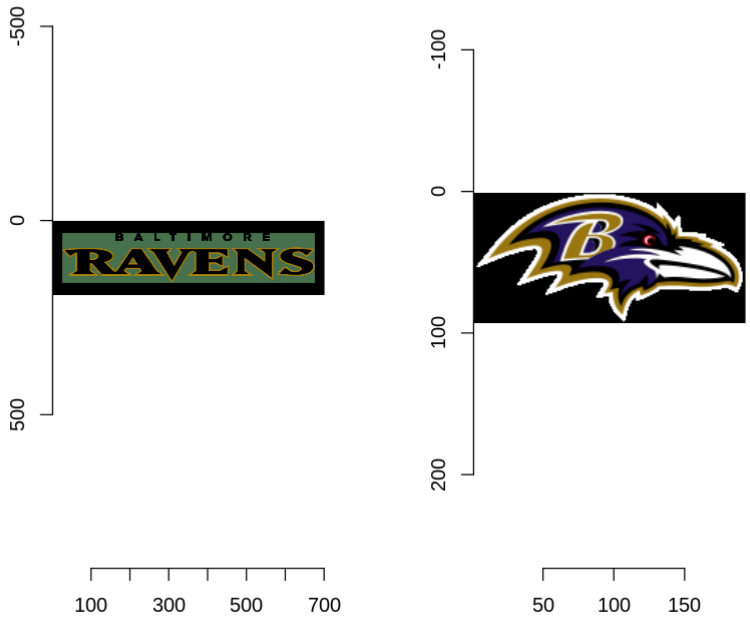
```
[1] "Arizona Cardinals >> ARI"
[1] "Atlanta Falcons >> ATL"
```



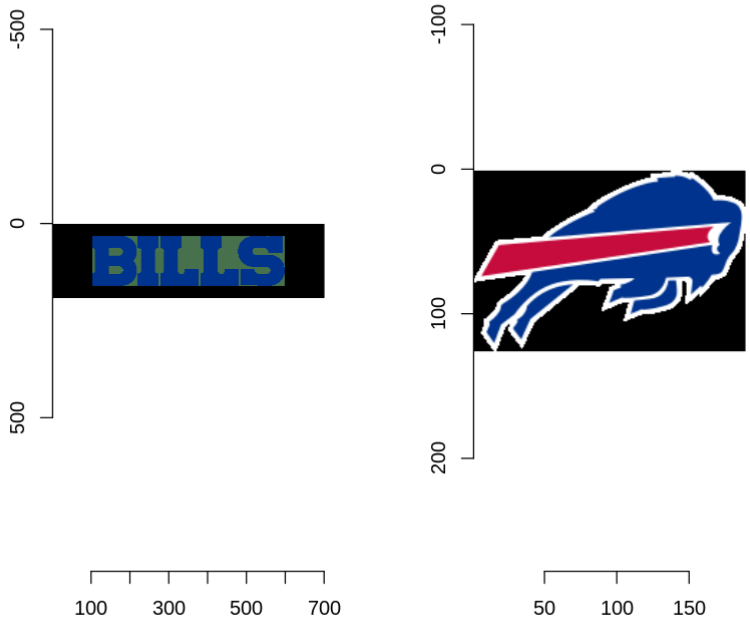

```
[1] "Baltimore Ravens >> BAL"
```



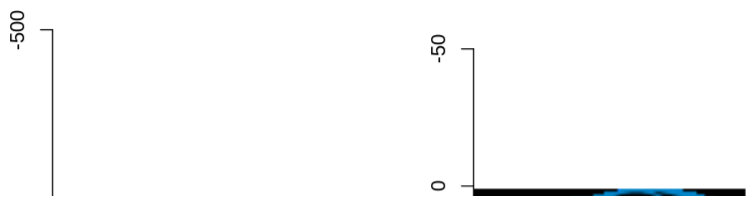

```
[1] "Buffalo Bills >> BUF"
```

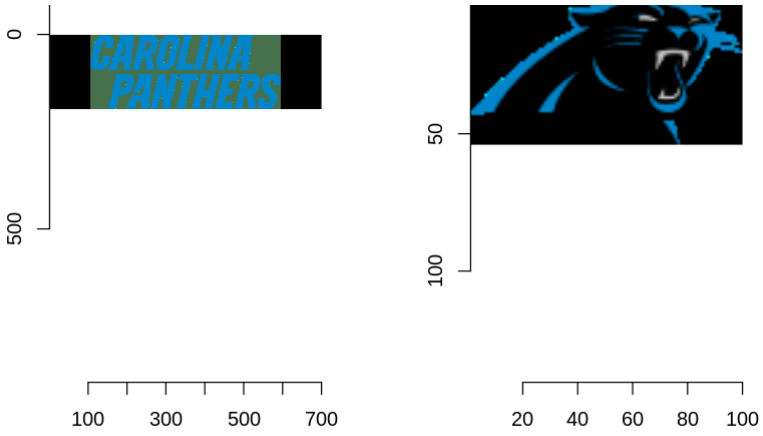



```
[1] "Carolina Panthers >> CAR"
```

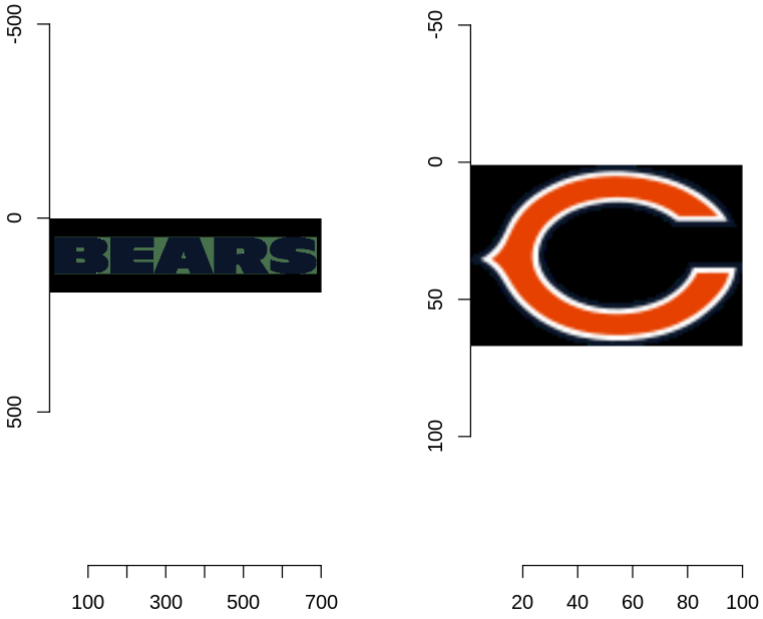



```
[1] "Chicago Bears >> CHI"
```

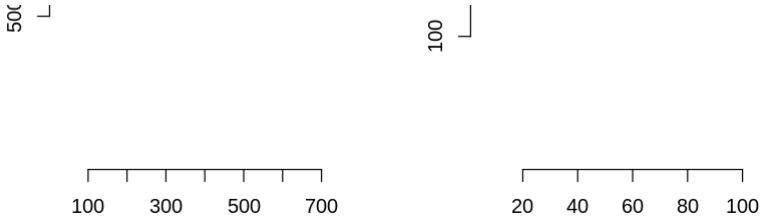




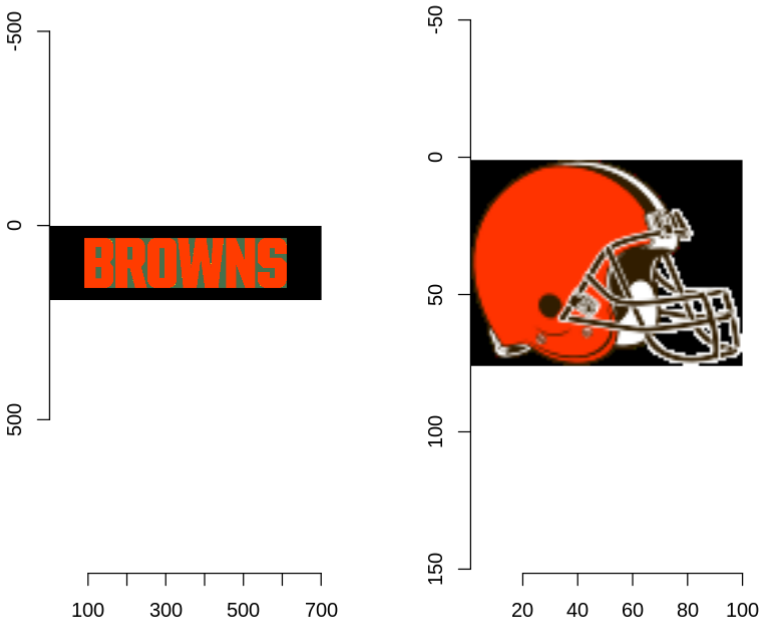

```
[1] "Cincinnati Bengals >> CIN"
```



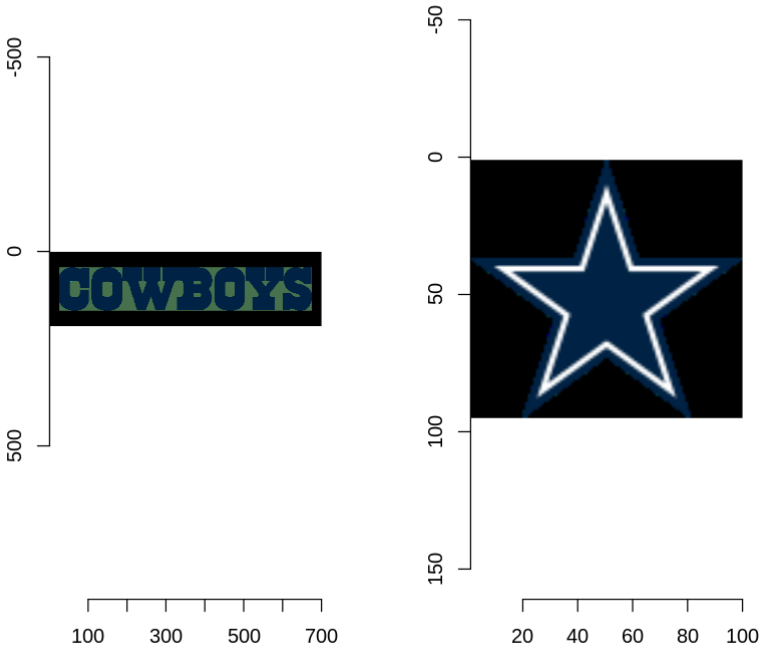

```
[1] "Cleveland Browns >> CLE"
```



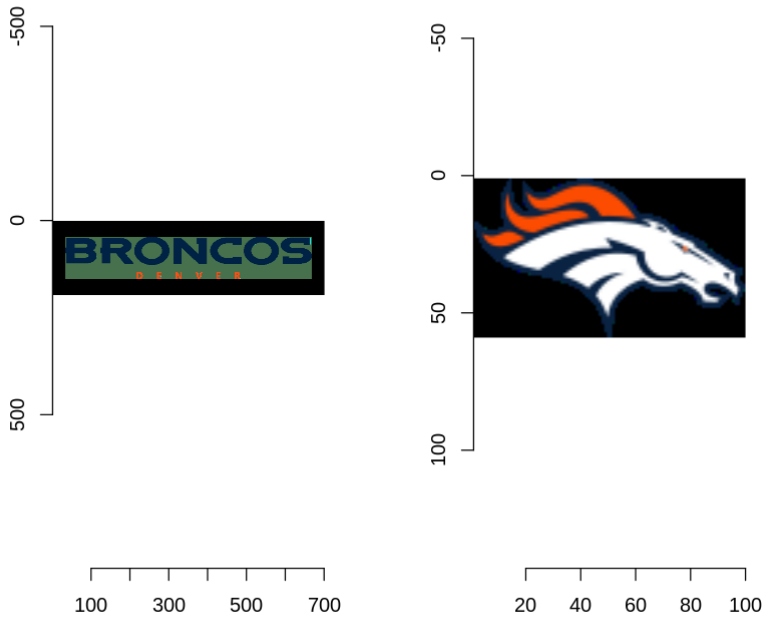

```
[1] "Dallas Cowboys >> DAL"
```



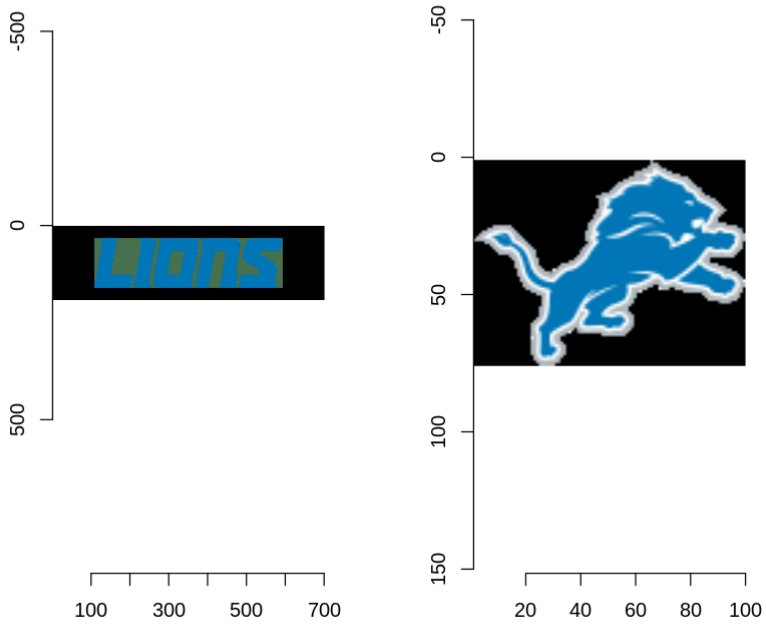

```
[1] "Denver Broncos >> DEN"
```



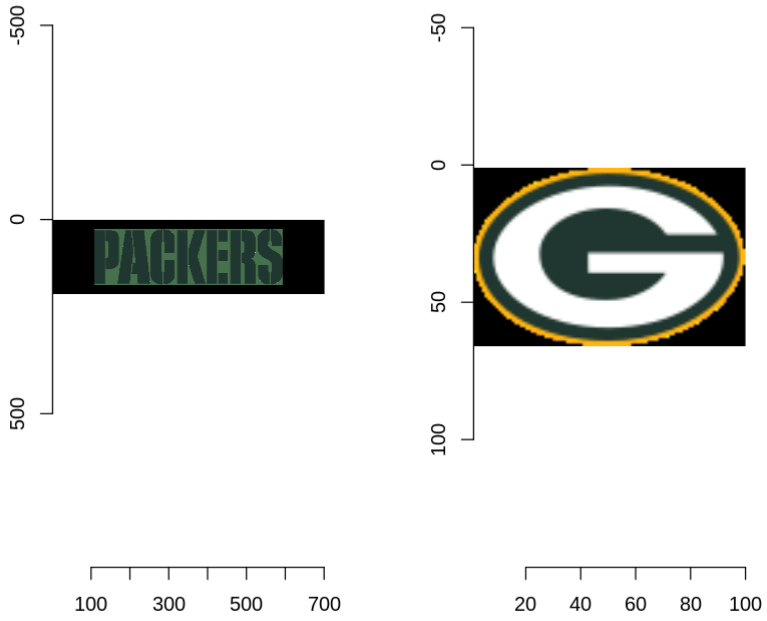
```
[1] "Detroit Lions >> DET"
```



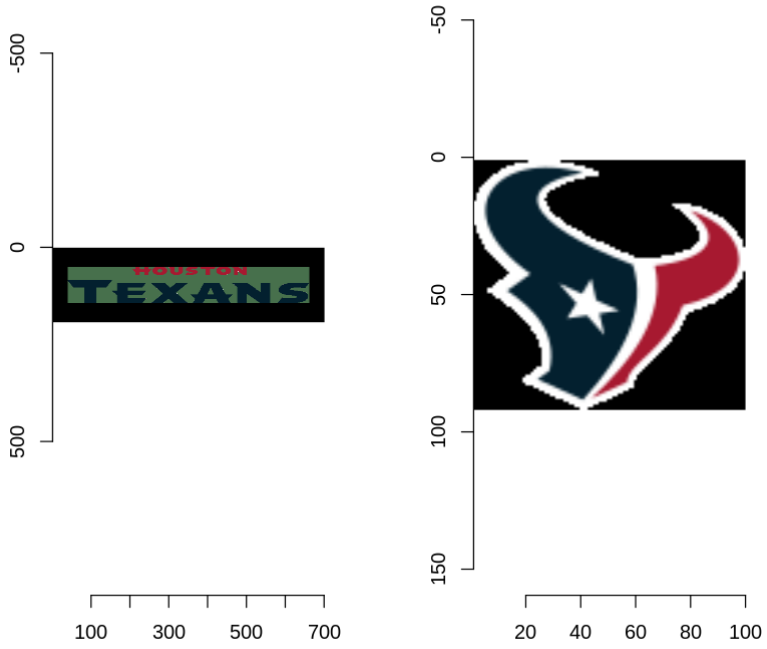

```
[1] "Green Bay Packers >> GB"
```



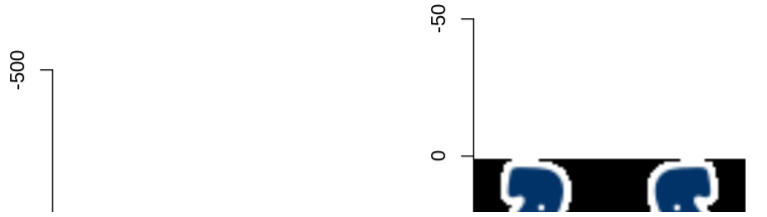

```
[1] "Houston Texans >> HOU"
```

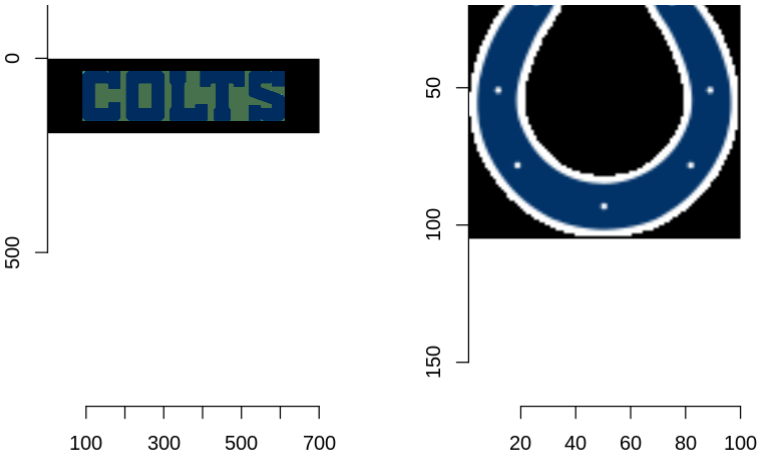



```
[1] "Indianapolis Colts >> IND"
```

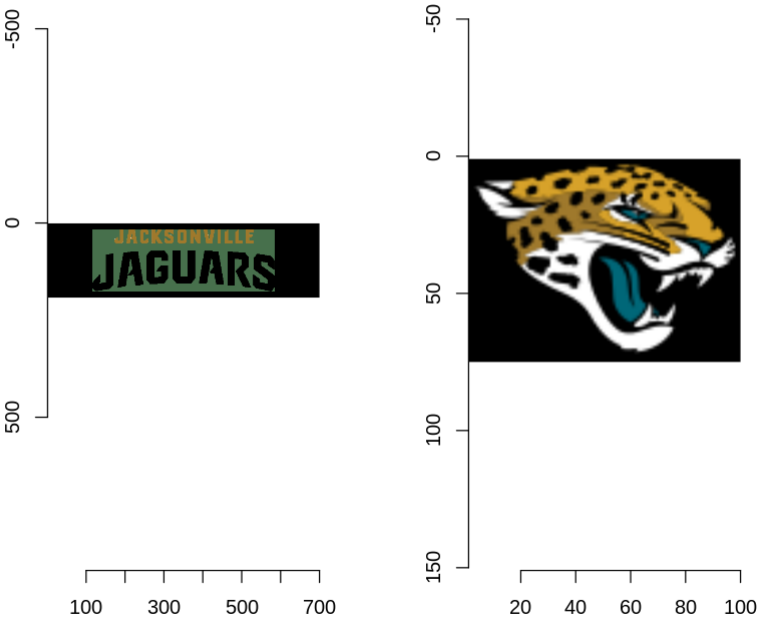



```
[1] "Jacksonville Jaguars >> JAX"
```

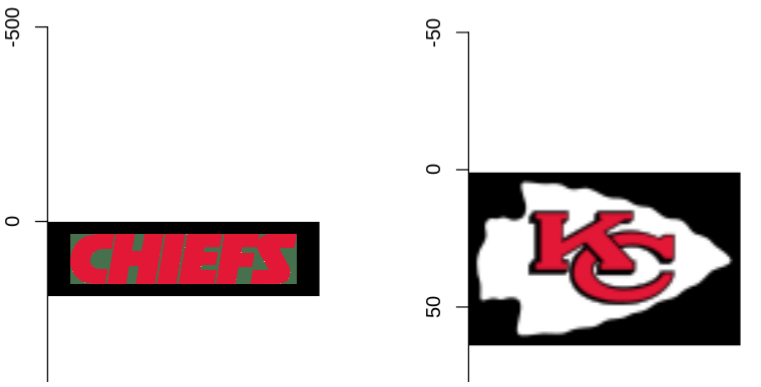


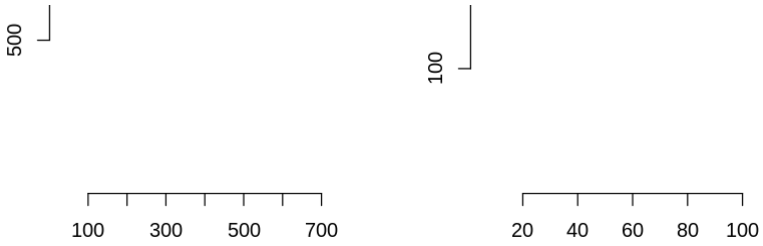



```
[1] "Kansas City Chiefs >> KC"
```

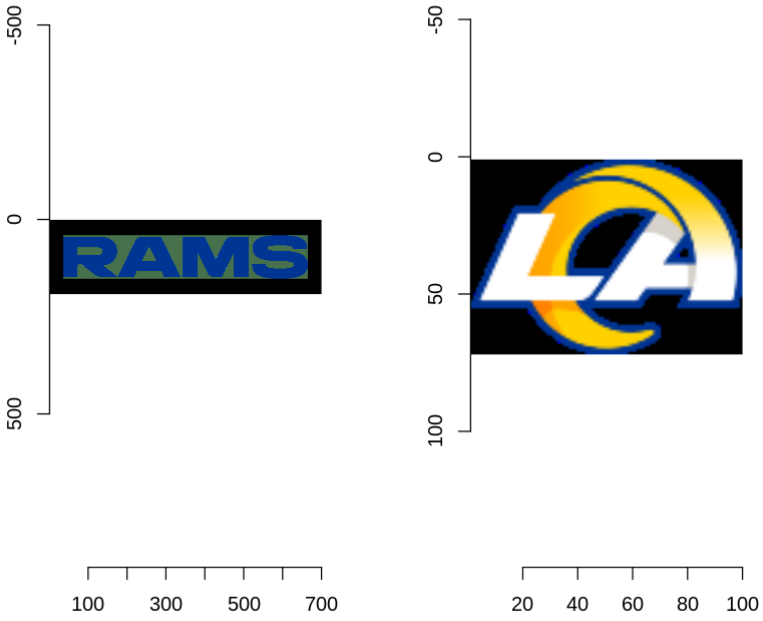



```
[1] "Los Angeles Rams >> LA"
```

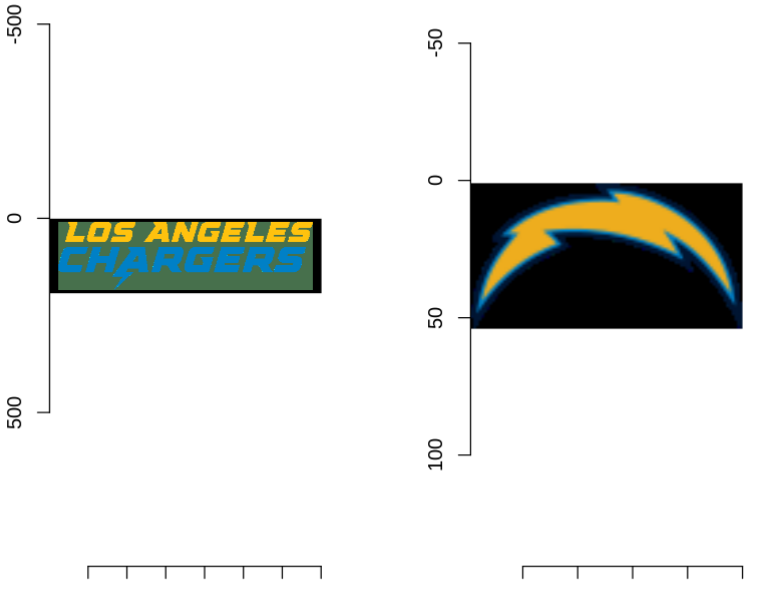




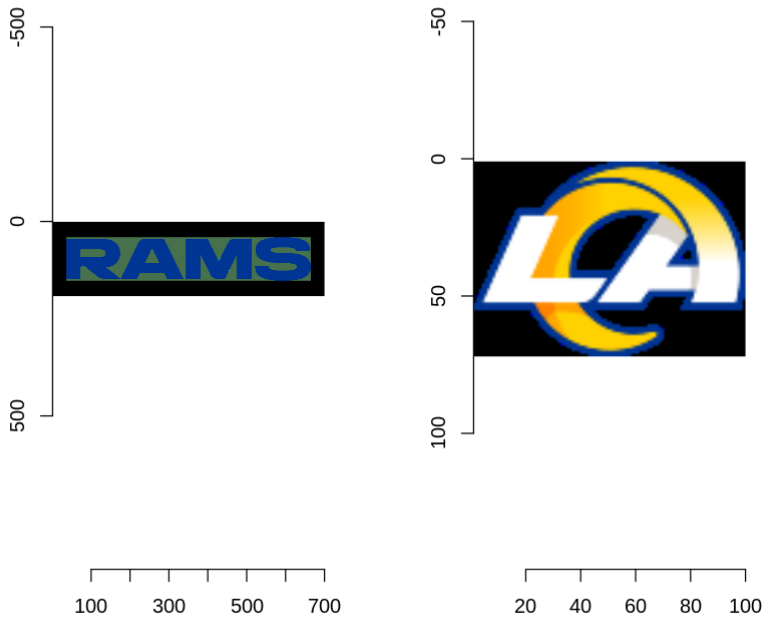

```
[1] "Los Angeles Chargers >> LAC"
```



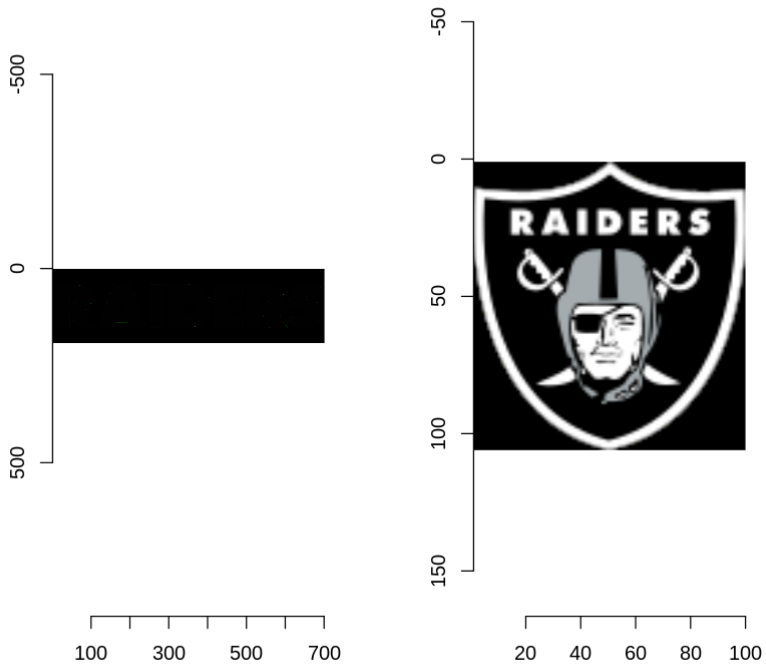

```
[1] "Los Angeles Rams >> LAR"
```



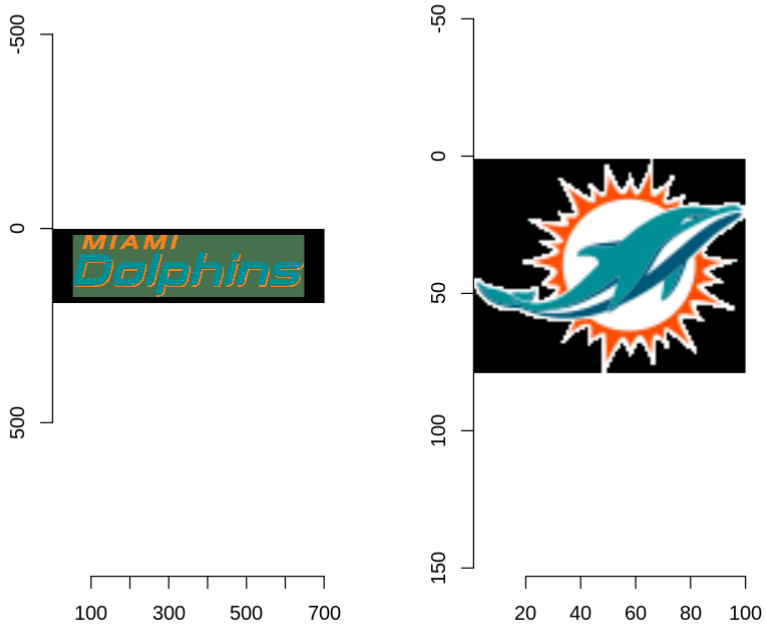

```
[1] "Las Vegas Raiders >> LV"
```



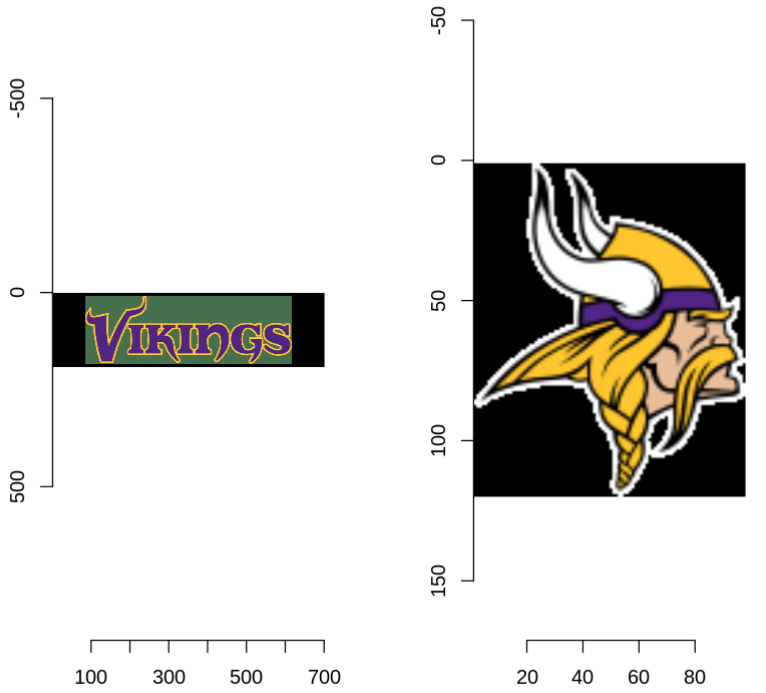

```
[1] "Miami Dolphins >> MIA"
```



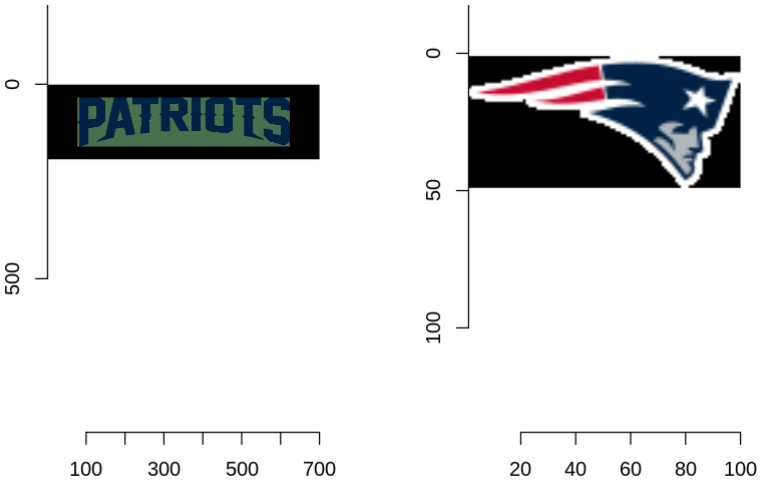

```
[1] "Minnesota Vikings >> MIN"
```



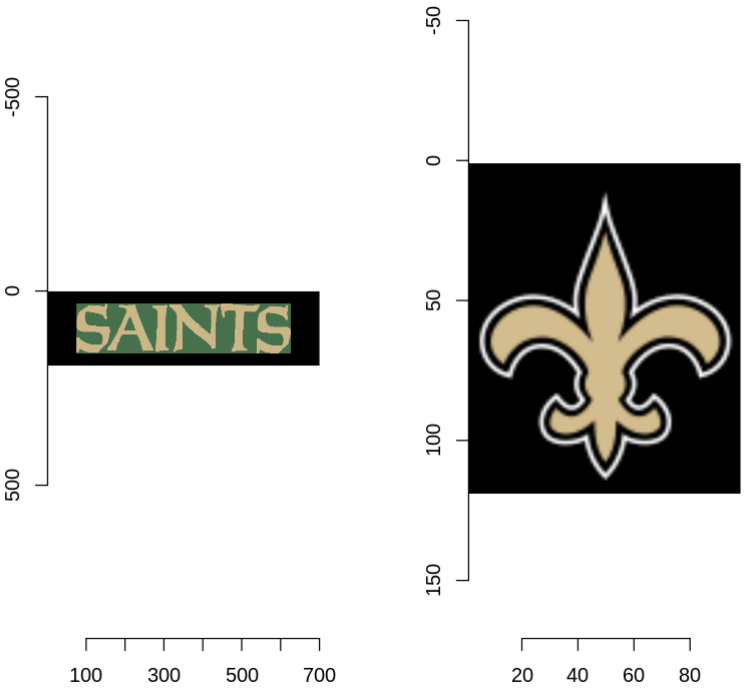

```
[1] "New England Patriots >> NE"
```



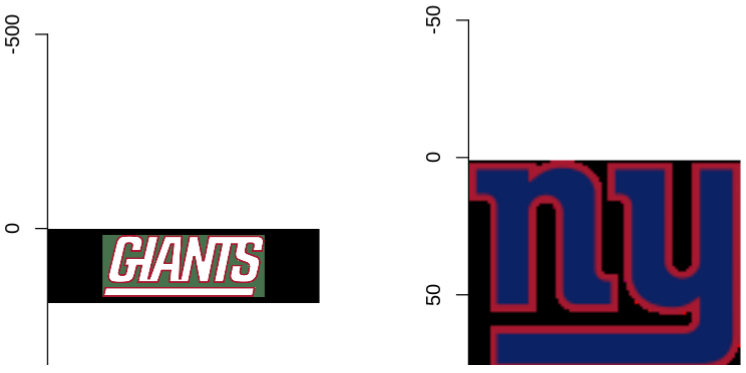

```
[1] "New Orleans Saints >> NO"
```

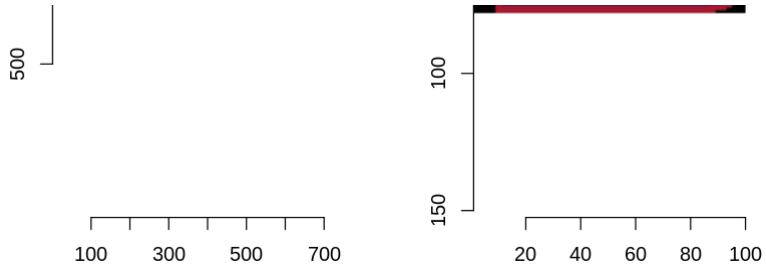



```
[1] "New York Giants >> NYG"
```

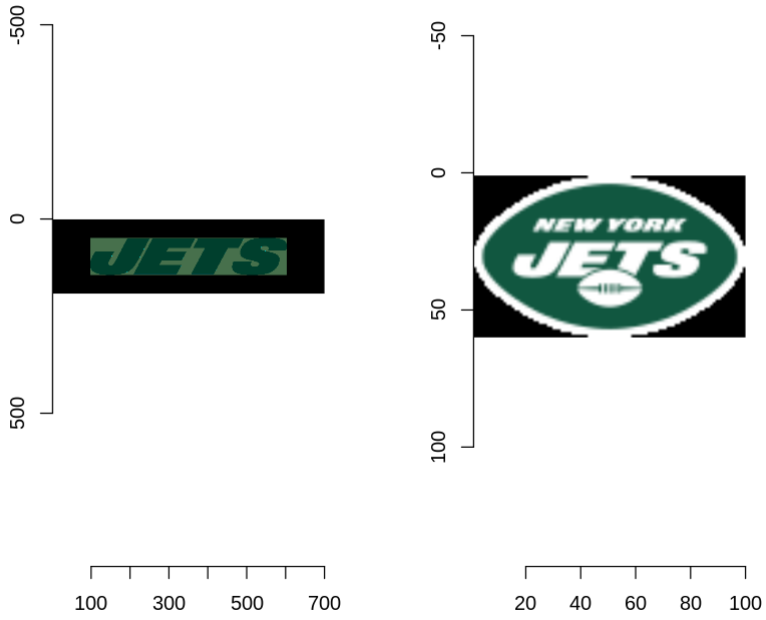



```
[1] "New York Jets >> NYJ"
```

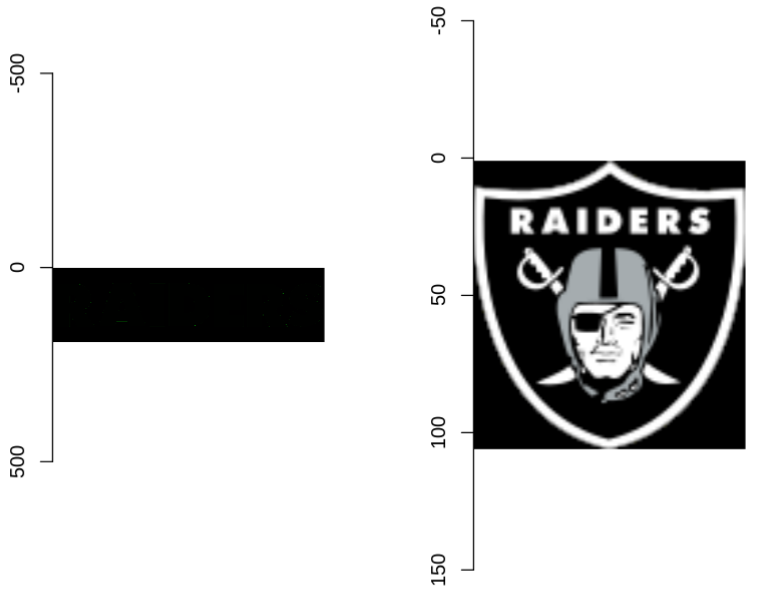




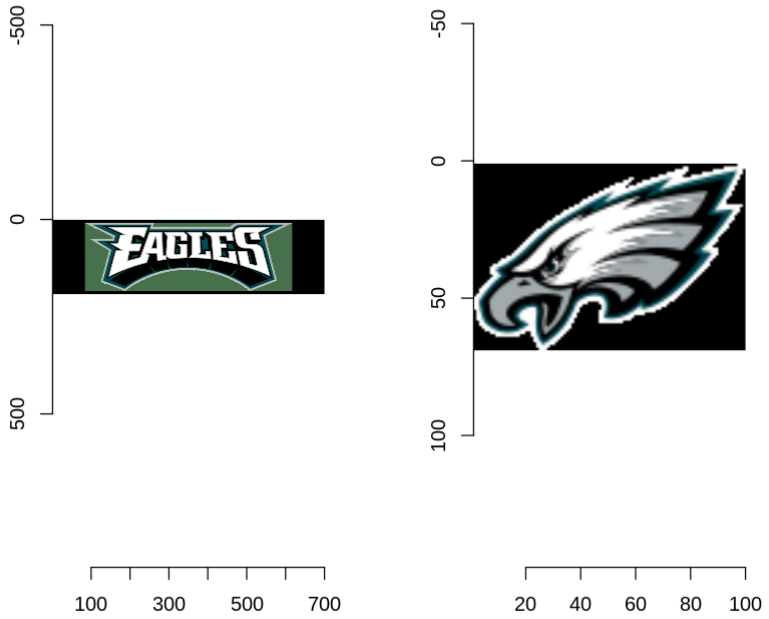

```
[1] "Oakland Raiders >> OAK"
```




```
[1] "Philadelphia Eagles >> PHI"
```

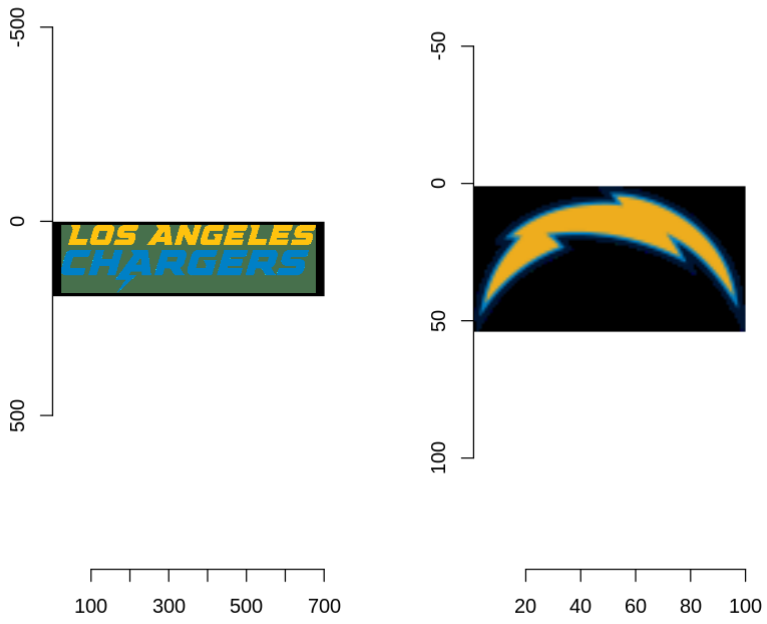



```
[1] "Pittsburgh Steelers >> PIT"
```

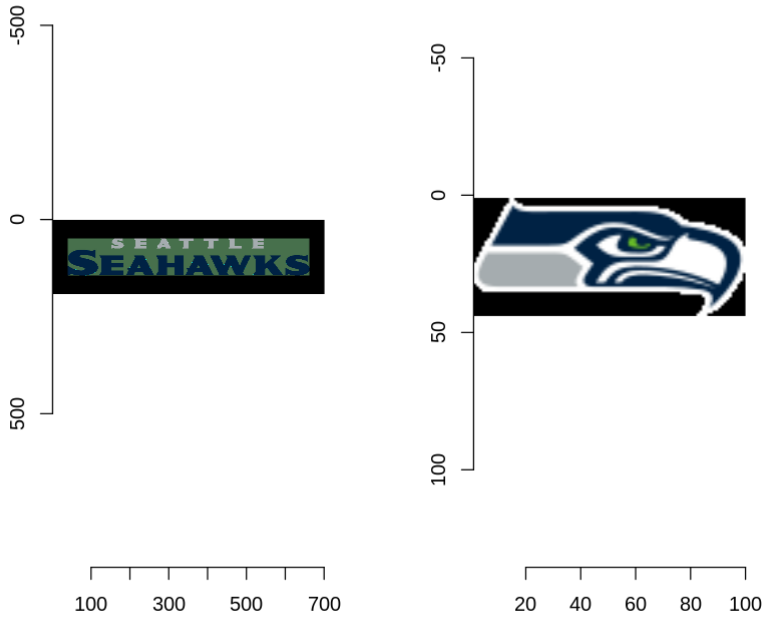



```
[1] "San Diego Chargers >> SD"
```

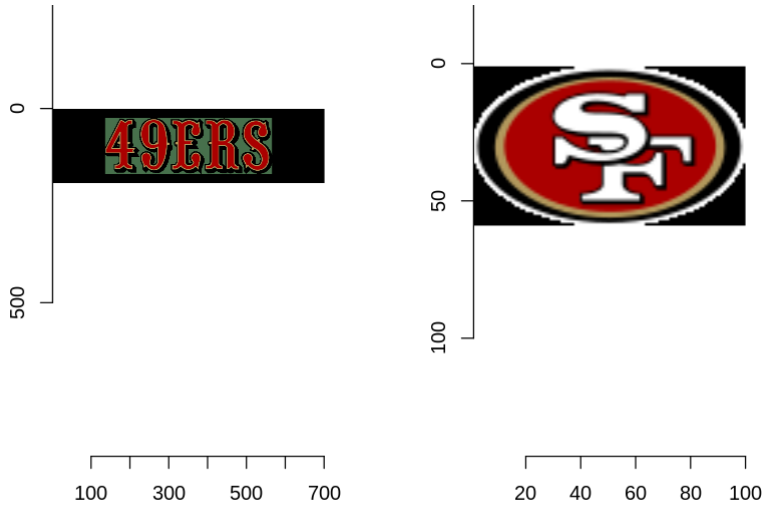

```
[1] "Seattle Seahawks >> SEA"
```



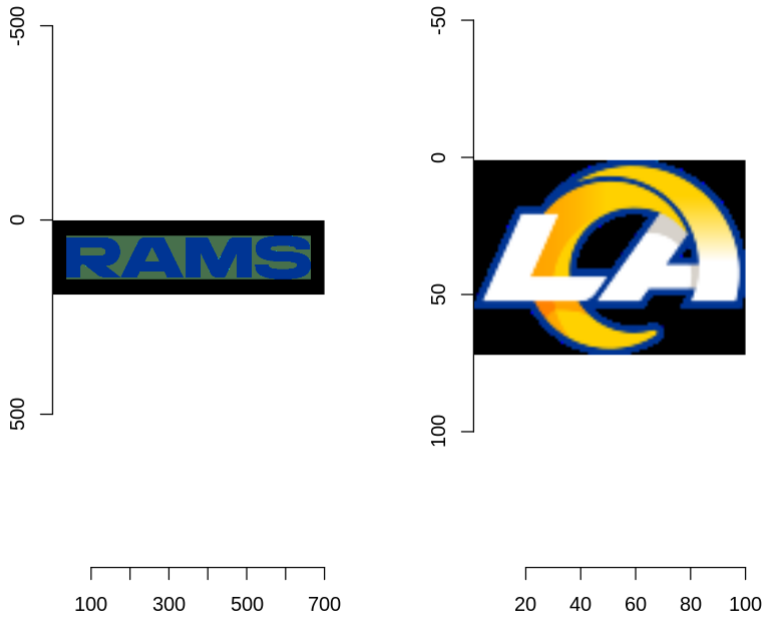

```
[1] "San Francisco 49ers >> SF"
```



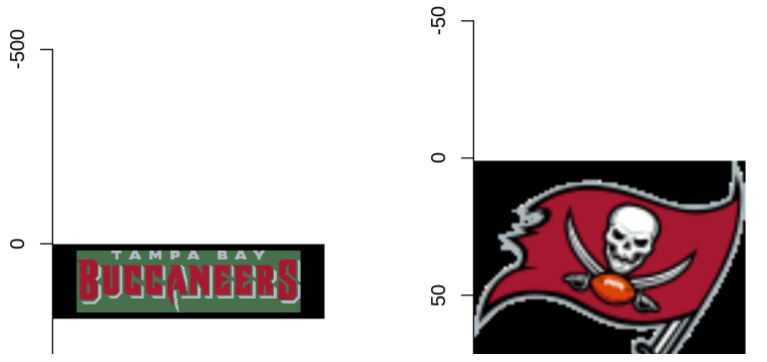

```
[1] "St. Louis Rams >> STL"
```

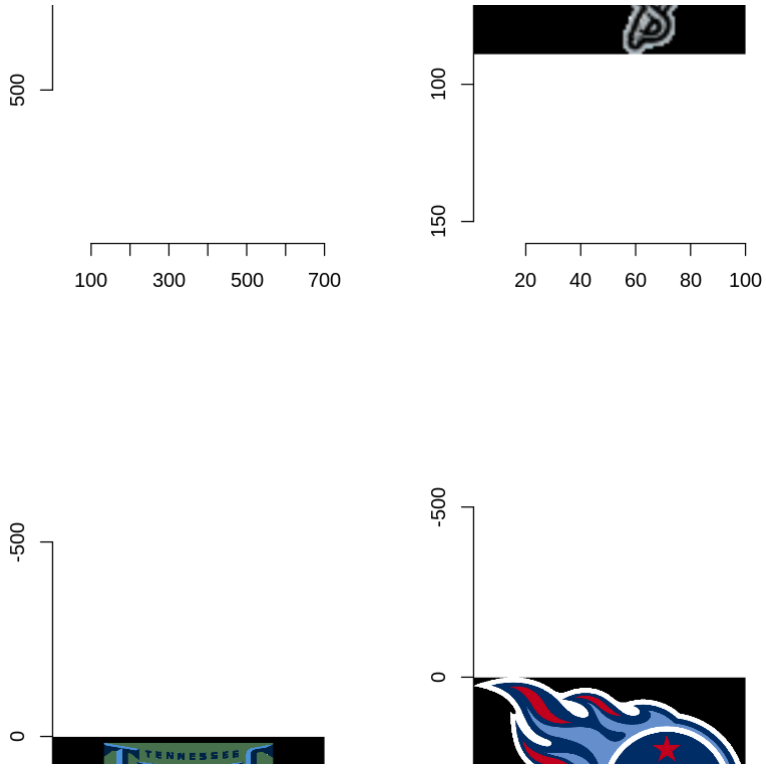



```
[1] "Tampa Bay Buccaneers >> TB"
```




```
[1] "Tennessee Titans >> TEN"
```





Vejam que é possível ter análises bem complexas e elaboradas, como por exemplo este bloco de código abaixo que foi adaptado do *Exemplo 5* de [Get Start with nflfastR](#).

> Não é objetivo desta disciplina de introdução exigir estes elementos. Coloquei aqui apenas para caráter informativo e mostrar que é possível realizar análises tão complexas quanto desejarmos.


```
#offense <- temporada %>%
#  dplyr::group_by(posteam) %>%
#  dplyr::summarise(off_epa = mean(epa, na.rm = TRUE))

#defense <- temporada %>%
#  dplyr::group_by(defteam) %>%
#  dplyr::summarise(def_epa = mean(epa, na.rm = TRUE))

#logos <- teams_colors_logos %>% dplyr::select(team_abbr, team_logo_espn)

#offense %>%
#  dplyr::inner_join(defense, by = c("posteam" = "defteam")) %>%
#  dplyr::inner_join(logos, by = c("posteam" = "team_abbr")) %>%
#  ggplot2::ggplot(aes(x = off_epa, y = def_epa)) +
#  ggplot2::geom_abline(slope = -1.5, intercept = c(.4, .3, .2, .1, 0, -.1, -.2, -.3),
#  ggplot2::geom_hline(aes(yintercept = mean(off_epa)), color = "red", linetype = "das
#  ggplot2::geom_vline(aes(xintercept = mean(def_epa)), color = "red", linetype = "das
#  ggimage::geom_image(aes(image = team_logo_espn), size = 0.10, asp = 16 / 9) +
#  ggplot2::labs(
#    x = "Ataque EPA/jogada",
#    y = "Defesa EPA/jogada",
#    caption = "Dados: @nflfastR",
```

```
#    caption = "Dados: nflfastR",
#    title = "2014 NFL Ataque e Defesa EPA por jogada"
#  ) +
#  ggplot2::theme_bw() +
#  ggplot2::theme(
#    aspect.ratio = 9 / 16,
#    plot.title = ggplot2::element_text(size = 12, hjust = 0.5, face = "bold")
#  ) +
#  ggplot2::scale_y_reverse()
```

Manipulação de dados

Criação dos *datasets* segmentados por variáveis

Pense no seguinte problema. Sabendo que o time joga tanto em casa (*home_team*) quanto fora de casa (*away_team*), em qual semana o time escolhido ficou de folga. Ou seja, não há entrada de dados na variável *week*.

Para esta atividade de aprofundamento mantenha o time 'SEA' escolhido, mesmo que você explore outras oportunidades posteriormente.

```
timeEscolhido <- 'SEA'

jogosTimeEscolhido <- temporada %>% filter(home_team == timeEscolhido | away_team == t

table(jogosTimeEscolhido$away_team, jogosTimeEscolhido$week)
```

	1	2	3	5	6	7	8	9	10	11	12	13	14	15	16	17	19	20
ARI	0	0	0	0	0	0	0	0	0	0	154	0	0	0	0	0	0	0
CAR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	167	0
DAL	0	0	0	0	173	0	0	0	0	0	0	0	0	0	0	0	0	0
DEN	0	0	203	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Criação dos datasets específicos, segmentando o dataset original, para facilitar a manipulação dos dados e responder às perguntas de negócio.

Utilizando o pacote **Tidyverse**, crie novos conjuntos de dados a partir da função *select*. Garanta que todos datasets estejam fazendo um filtro apenas da semana 1.

> Dica: para o filtro da semana 1, utilize a condição **week==1** na função *filter*

jogo com as variáveis *play_id*, *home_team*, *away_team*, *away_score*, *home_score*, *total*

acoesJogadas com as variáveis *play_id*, *rush_attempt*, pass_attempt*, *field_goal_attempt*, *down*, *time*, *qtr*, *ydstogo*, *yards_gained*

pontuacaoJogadas com as variáveis *play_id*, *posteam*, *defteam*, *posteam_score*, *defteam_score*, *rush*, *pass*, *name*, *passer*, *rusher*, *receiver*, *interception*, *play_type*, *pass_length*, *air_yards*, *kick_distance*, *drive*, *touchdown*, *td_team*

descricaoJogadas com as variáveis *play_id*,*desc*, *passer_player_name*, *passing_yards*, *receiver_player_name*, *punt_returner_player_name*,*name*

> Repare que **TODOS** conjuntos de dados criados possuem a variável *play_id*, porque ela fará o relacionamento entre os conjuntos de dados, caso você queria/precise combinar conjuntos de dados para chegar à uma solução

```
jogo <- jogosTimeEscolhido %>%
    filter(week==1) %>%
    select(play_id,
           home_team, away_team, away_score, home_score, total
          )
```

```
acoesJogadas <- jogosTimeEscolhido %>%
    filter(week==1) %>%
    select(play_id,
           rush_attempt, pass_attempt, field_goal_attempt, down, time, qtr, ydstogo, y
          )
```

```
pontuacaoJogadas <- jogosTimeEscolhido %>%
    filter(week==1) %>%
    select(play_id,
```

```
           posteam, defteam, posteam_score, defteam_score, rush, pass, passer, rusher,
          )
descricaoJogadas <- jogosTimeEscolhido %>%
    filter(week==1) %>%
    select(play_id,
           desc, passer_player_name, passing_yards, receiver_player_name, punt_returne
          )
```

```
head(jogo)
head(acoesJogadas)
head(pontuacaoJogadas)
head(descricaoJogadas)
```

A nflverse_data: 6 × 6

play_id	home_team	away_team	away_score	home_score	total
<dbl>	<chr>	<chr>	<int>	<int>	<int>
1	SEA	GB	16	36	52
36	SEA	GB	16	36	52
58	SEA	GB	16	36	52
79	SEA	GB	16	36	52
111	SEA	GB	16	36	52
132	SEA	GB	16	36	52

A nflverse_data: 6 × 9

play_id	rush_attempt	pass_attempt	field_goal_attempt	down	time	qtr	yds
<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<chr>	<dbl>	<(
1	NA	NA	NA	NA	15:00	1	
36	0	0	0	NA	15:00	1	

O uso da função *inner_join* no pacote **Tidyverse** é muito útil para combinar conjunto de dados. Veja, nos exemplos abaixo, como fica a combinação destes *datasets* que foram criados anteriormente.

Pense nos seguintes desafios:

1) Combinar o resultado de **pontuacaoJogadas** que tem a informação de quando um time fez *touchdown* (significa que marcou 6 pontos no jogo) e **descricaoJogadas** onde há uma descrição da jogada. Estes conjuntos de dados estão segmentados, cada um deles possui uma parte da informação. Ao combinar estes dois conjuntos de dados é possível ter todas as variáveis juntas como se fossem um único *dataset*. Eles se combinam a partir da variável *play_id*, que é comum entre eles. A partir desta combinação, a manipulação é similar ao que já foi estudado anteriormente.

```
pontuacaoJogadas %>% #primeiro dataset
inner_join(descricaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(play_id, posteam, touchdown, td_team, desc) %>% #seleção de variáveis
filter(touchdown == 1) #filtro de dados
```

A nflverse_data: 6 × 5

play_id	posteam	touchdown	td_team	desc
<dbl>	<chr>	<dbl>	<chr>	<chr>
802	GB	1	GB	(1:30) 30-J.Kuhn left guard for 2 yards, TOUCHDOWN.

(13:08) (Shotgun) 3-B.Wilson pass short left to 83-

2) Mostrar qual foi o jogador do time da casa e quando ele recebeu o primeiro passe que permitiu correr 5 ou mais jardas.

Para isso, é necessário cobinar 3 conjuntos de dados. No dataset **jogo** é possível retornar qual é o time da casa. Já em **acoesJogadas** é possível saber quantas jardar foram conquistadas (com a variável *yards_gained*). E por fim, em **pontuacaoJogadas** há o nome de quem correu com a bola (variável *rusher*). Vamos ver como fica essa combinação?

```
jogo %>% #primeiro dataset
inner_join(acoesJogadas, by='play_id') %>% #segundo dataset combinando com o primeiro
inner_join(pontuacaoJogadas, by='play_id') %>% #terceiro dataset combinando com o prir
select(play_id, home_team,posteam, rusher, yards_gained, time, qtr ) %>% #seleção de v
filter( posteam == home_team | yards_gained >=5 ) %>% #filtro de dados
head(1) #retorno apenas de 1 linha
```

A nflverse_data: 1 × 7

play_id	home_team	posteam	rusher	yards_gained	time	qtr
<dbl>	<chr>	<chr>	<chr>	<dbl>	<chr>	<dbl>
58	SEA	GB	E.Lacy	6	14:56	1

Desafios de manipulação de dados

Com base no dataset específico **pontuacaoJogadas**, apresente os dados somente quando ouve *rush* ou *pass* na jogada. Garanta que exista também o nome ou abreviatura do time que está atacando (variável *posteam*), além dos nomes dos jogadores que estão fazendo passe, correndo ou recebendo a bola (variáveis *passer*, *rusher* e *receiver*)

```
resposta1 <- pontuacaoJogadas %>% #dataset escolhido
   filter(rush==1) %>%
   select(play_id,
           rush, posteam, passer, rusher, receiver
          )
head(resposta1)
```

A nflverse_data: 6 × 6

play_id	rush	posteam	passer	rusher	receiver
<dbl>	<dbl>	<chr>	<chr>	<chr>	<chr>
58	1	GB	NA	E.Lacy	NA
79	1	GB	NA	E.Lacy	NA
111	1	GB	NA	E.Lacy	NA
132	1	GB	NA	J.Starks	NA
245	1	SEA	NA	M.Lynch	NA
314	1	SEA	NA	R.Turbin	NA

Utilizando o subconjunto de dados **acoesJogadas** e **pontuacaoJogadas**, crie uma análise que retorne qual foi o jogador que conquistou mais jardas no terceiro quarto.

```
acoesJogadas %>% #primeiro dataset
inner_join(pontuacaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(play_id, posteam, rusher, yards_gained, time, qtr ) %>% #seleção de variáveis
filter( qtr == 3 & rusher != 'NA') %>% #filtro de dados
group_by(rusher) %>%
mutate(yards_sum = sum(yards_gained, na.rm = TRUE)) %>%
select(rusher, yards_sum) %>%
unique() %>%
arrange(desc(yards_sum)) %>%
head(1) #retorno apenas de 1 linha
```

A grouped_df: 1 × 2

rusher	yards_sum
<chr>	<dbl>
M.Lynch	34

Desafio de geração de gráfico

Crie um gráfico de linhas, mostrando a pontuação de cada time em cada *quarter*. O resultado deve ter duas linhas, uma para cada time, e cada linha será composta pela pontuação de cada um dos *quarters* sendo uma cor para cada time. O eixo X terá os *quarters* e o eixo y terá a pontuação.

```
time1 = 'SEA'
time2 = 'GB'
```

```
#-------------------------------------------------------------------------------
#                                  TIME1
#-------------------------------------------------------------------------------

# Selecionando as informacoes para o primeiro time quando ele está como Post
time1Post <- acoesJogadas %>% #primeiro dataset
inner_join(pontuacaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(posteam, defteam, posteam_score, defteam_score, qtr ) %>% #seleção de variáveis
filter(posteam == time1 & posteam_score != 0) %>%
group_by(qtr) %>%
unique()

# Selecionando as informacoes para o primeiro time quando ele está como Def
time1Def <- acoesJogadas %>% #primeiro dataset
inner_join(pontuacaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(posteam, defteam, posteam_score, defteam_score, qtr ) %>% #seleção de variáveis
filter(defteam == time1 & defteam_score != 0) %>%
group_by(qtr) %>%
unique()

# Selecionando apenas os maiores valores em cada quarter do time 1 como Post
time1PostQuarters <- time1Post %>%
group_by(qtr) %>%
slice(which.max(posteam_score))

time1PostQuarters

# Selecionando apenas os maiores valores em cada quarter do time 1 como Def
time1DefQuarters <- time1Def %>%
group_by(qtr) %>%
slice(which.max(defteam_score))

time1DefQuarters

# craindo dataframes para uma manipulacao mais rápida
time1_dataP <- data.frame(  time1 = time1PostQuarters$posteam,
                            score = time1PostQuarters$posteam_score,
                          quarter = time1PostQuarters$qtr)
time1_dataD <- data.frame(  time1 = time1DefQuarters$defteam,
                            score = time1DefQuarters$defteam_score,
                          quarter = time1DefQuarters$qtr)

# colocando os dataframes em apenas uma tabela para uma comparacao posterior
time1_join <- time1_dataP %>%
inner_join(time1_dataD, by = 'quarter')

# criando o votero aonde teremos os valores de cada quarter para o time 1
quarter_score_time1 <- c()
time1_name <- c()
```

```
# loop para fazer a comparacao entre os scores como Post e como Def
for (i in 1:length(time1_join$quarter)){
  if (time1_join$score.x[i] > time1_join$score.y[i])
    quarter_score_time1 <- append(quarter_score_time1,time1_join$score.x[i])
  else (
    quarter_score_time1 <- append(quarter_score_time1,time1_join$score.y[i])
  )
  time1_name <- append(time1_name,time1)
  #print(quarter_score_time1[i])
}

#-------------------------------------------------------------------------------
#                                 TIME2
#-------------------------------------------------------------------------------

# Selecionando as informacoes para o primeiro time quando ele está como Post
time2Post <- acoesJogadas %>% #primeiro dataset
inner_join(pontuacaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(posteam, defteam, posteam_score, defteam_score, qtr ) %>% #seleção de variáveis
filter(posteam == time2 & posteam_score != 0) %>%
group_by(qtr) %>%
unique()

# Selecionando as informacoes para o primeiro time quando ele está como Def
time2Def <- acoesJogadas %>% #primeiro dataset
inner_join(pontuacaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(posteam, defteam, posteam_score, defteam_score, qtr ) %>% #seleção de variáveis
filter(defteam == time2 & defteam_score != 0) %>%
group_by(qtr) %>%
unique()

# Selecionando apenas os maiores valores em cada quarter do time 2 como Post
time2PostQuarters <- time2Post %>%
group_by(qtr) %>%
slice(which.max(posteam_score))

# Selecionando apenas os maiores valores em cada quarter do time 2 como Def
time2DefQuarters <- time2Def %>%
group_by(qtr) %>%
slice(which.max(defteam_score))

# craindo dataframes para uma manipulacao mais rápida
time2_dataP <- data.frame(  time2 = time2PostQuarters$posteam,
                            score = time2PostQuarters$posteam_score,
                          quarter = time2PostQuarters$qtr)
time2_dataD <- data.frame(  time2 = time2DefQuarters$defteam,
                            score = time2DefQuarters$defteam_score,
                          quarter = time2DefQuarters$qtr)

# colocando os dataframes em apenas uma tabela para uma comparacao posterior
time2_join <- time2_dataP %>%
```

```
inner_join(time2_dataD, by = 'quarter')

# criando o votero aonde teremos os valores de cada quarter para o time 2
quarter_score_time2 <- c()
quarter <- c()
time2_name <- c()

# loop para fazer a comparacao entre os scores como Post e como Def
for (i in 1:length(time2_join$quarter)){
  if (time2_join$score.x[i] > time2_join$score.y[i])
    quarter_score_time2 <- append(quarter_score_time2,time2_join$score.x[i])
  else (
    quarter_score_time2 <- append(quarter_score_time2,time2_join$score.y[i])
  )
  quarter <- append(quarter,i)
  time2_name <- append(time2_name,time2)
  #print(quarter_score_time2[i])
}
```

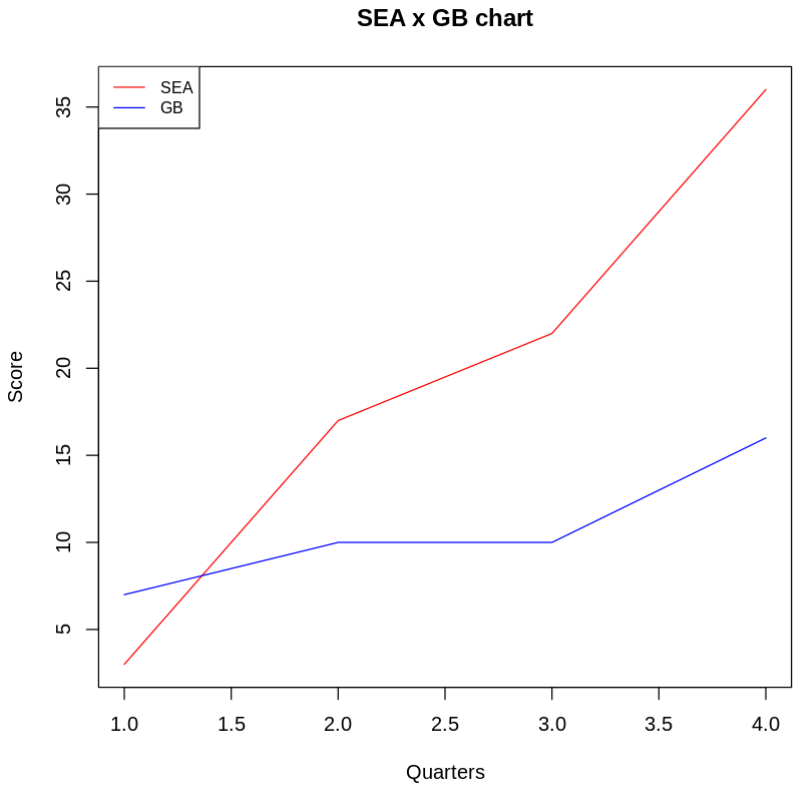
A grouped_df: 4 × 5

posteam	defteam	posteam_score	defteam_score	qtr
<chr>	<chr>	<dbl>	<dbl>	<dbl>
SEA	GB	3	7	1
SEA	GB	17	10	2
SEA	GB	22	10	3
SEA	GB	36	16	4

A grouped_df: 4 × 5

posteam	defteam	posteam_score	defteam_score	qtr
<chr>	<chr>	<dbl>	<dbl>	<dbl>
GB	SEA	0	3	1
GB	SEA	10	17	2
GB	SEA	10	20	3
GB	SEA	16	36	4

```
plot(quarter, quarter_score_time1, type="l", col="red", xlab="Quarters", ylab="Score",
# Add a line
lines(quarter, quarter_score_time2, col="blue", type="l")
# Add a legend
legend("topleft", legend=c(time1, time2),
       col=c("red", "blue"), lty=1:1, cex=0.8)
```

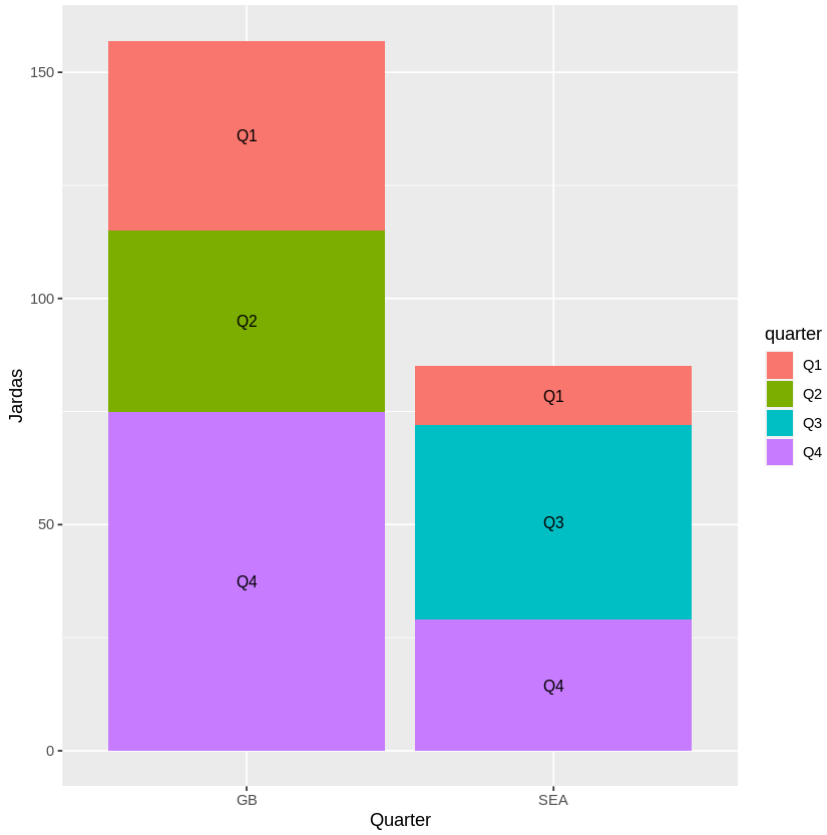


Crie um gráfico de barras empilhada (colunas verticais), utilizando somente as jogadas que tiveram entre 10 e 20 jardas conquistas. O empilhamento das barras será feito pela quantidade de jardas conquistadas (entre 10 e 20). Mantenha as barras verticais segmentadas por quarter do jogo, e por fim, crie a faceta baseada nos times.

```
yards_df_q1 <- data.frame(acoesJogadas %>% #primeiro dataset
inner_join(pontuacaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(posteam, yards_gained, qtr ) %>% #seleção de variáveis
filter( yards_gained > 10 & yards_gained <20 & qtr == 1) %>% #filtro de dados
group_by(posteam) %>%
mutate(yards_sum = sum(yards_gained, na.rm = TRUE)) %>%
select(posteam, qtr, yards_sum) %>%
unique() %>%
mutate(quarter = 'Q1')
)

yards_df_q2 <- data.frame(acoesJogadas %>% #primeiro dataset
inner_join(pontuacaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(posteam, yards_gained, qtr ) %>% #seleção de variáveis
filter( yards_gained > 10 & yards_gained <20 & qtr == 2) %>% #filtro de dados
group_by(posteam) %>%
mutate(yards_sum = sum(yards_gained, na.rm = TRUE)) %>%
select(posteam, qtr, yards_sum) %>%
unique() %>%
mutate(quarter = 'Q2')
)
```

```
yards_df_q3 <- data.frame(acoesJogadas %>% #primeiro dataset
inner_join(pontuacaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(posteam, yards_gained, qtr ) %>% #seleção de variáveis
filter( yards_gained > 10 & yards_gained <20 & qtr == 3) %>% #filtro de dados
group_by(posteam) %>%
mutate(yards_sum = sum(yards_gained, na.rm = TRUE)) %>%
select(posteam, qtr, yards_sum) %>%
unique() %>%
mutate(quarter = 'Q3')
)

yards_df_q4 <- data.frame(acoesJogadas %>% #primeiro dataset
inner_join(pontuacaoJogadas, by='play_id') %>% #segundo dataset combinando com o prime
select(posteam, yards_gained, qtr ) %>% #seleção de variáveis
filter( yards_gained > 10 & yards_gained <20 & qtr == 4) %>% #filtro de dados
group_by(posteam) %>%
mutate(yards_sum = sum(yards_gained, na.rm = TRUE)) %>%
select(posteam, qtr, yards_sum) %>%
unique() %>%
mutate(quarter = 'Q4')
)

#yards_df_q1
#yards_df_q2
#yards_df_q3
#yards_df_q4
```

```
yards_df <- bind_rows(yards_df_q1 , yards_df_q2, yards_df_q3, yards_df_q4)

#yards_df

ggplot(data=yards_df,
       aes(x=posteam, weights=yards_sum)) +
  geom_bar(aes(fill=quarter) ) +
  geom_text(aes(x=posteam, y=yards_sum, group=quarter, label= quarter),
            position = position_stack(vjust = 0.5), size=3.3) +
  xlab("Quarter") + ylab("Jardas")
```



✓ 0s completed at 2:37 PM