# Bios 6301: Assignment 6

Max Rohde

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------- tidy
```

```
## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ------------------------------------------------------------------------- tidyverse_
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

**Question 1**

**16 points**

Obtain a copy of the football-values lecture. Save the five 2020 CSV files in your working directory.

Modify the code to create a function. This function will create dollar values given information (as arguments) about a league setup.

It will return a data.frame and write this data.frame to a CSV file.

The final data.frame should contain the columns 'PlayerName', 'pos', 'points', 'value' and be ordered by value (descending). Do not round dollar values.

Note that the returned data.frame should have `sum(posReq)*nTeams` rows.

Define the function as such (10 points):

```
# path: directory path to input files
# file: name of the output file; it should be written to path
# nTeams: number of teams in league
# cap: money available to each team
# posReq: number of starters for each position
# points: point allocation for each category

ffvalues <- function(path, file='outfile.csv', nTeams=12, cap=200, posReq=c(qb=1, rb=2, wr=3, te=1, k=1
                     points=c(fg=4, xpt=1, pass_yds=1/25, pass_tds=4, pass_ints=-2,
                              rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6)) {
```

```
## read in CSV files
k <- read_csv("proj_k20.csv", col_types = cols())
qb <- read_csv("proj_qb20.csv", col_types = cols())
rb <- read_csv("proj_rb20.csv", col_types = cols())
te <- read_csv("proj_te20.csv", col_types = cols())
wr <- read_csv("proj_wr20.csv", col_types = cols())

# add the position to each row
k[,'pos'] <- 'k'
qb[,'pos'] <- 'qb'
rb[,'pos'] <- 'rb'
te[,'pos'] <- 'te'
wr[,'pos'] <- 'wr'

# merge the data frames
dplyr::bind_rows(k,qb,rb,te,wr) %>%
  mutate(across(where(is.numeric), ~coalesce(.x, 0))) -> x

# calculate new columns
# convert NFL stat to fantasy points
x[,'p_fg'] <- x[,'fg']*points['fg']
x[,'p_xpt'] <- x[,'xpt']*points['xpt']
x[,'p_pass_yds'] <- x[,'pass_yds']*points['pass_yds']
x[,'p_pass_tds'] <- x[,'pass_tds']*points['pass_tds']
x[,'p_pass_ints'] <- x[,'pass_ints']*points['pass_ints']
x[,'p_rush_yds'] <- x[,'rush_yds']*points['rush_yds']
x[,'p_rush_tds'] <- x[,'rush_tds']*points['rush_tds']
x[,'p_fumbles'] <- x[,'fumbles']*points['fumbles']
x[,'p_rec_yds'] <- x[,'rec_yds']*points['rec_yds']
x[,'p_rec_tds'] <- x[,'rec_tds']*points['rec_tds']

# sum selected column values for every row
# this is total fantasy points for each player
x[,'points'] <- rowSums(x[,grep("^p_", names(x))])

x2 <- x[order(x[,'points'], decreasing=TRUE),]

k.ix <- which(x2[,'pos']=='k')
qb.ix <- which(x2[,'pos']=='qb')
rb.ix <- which(x2[,'pos']=='rb')
te.ix <- which(x2[,'pos']=='te')
wr.ix <- which(x2[,'pos']=='wr')

# calculate marginal points by subtracting "baseline" player's points
# I know this is bad... need to reduce copy paste
if(posReq['k'] == 0){
  x2[k.ix, 'marg'] <- 0
}
else{
  x2[k.ix, 'marg'] <- x2[k.ix,'points'] - x2[[k.ix[nTeams * posReq['k']]],'points']]
}

if(posReq['qb'] == 0){
```

```
    x2[qb.ix, 'marg'] <- 0
  }
  else{
      x2[qb.ix, 'marg'] <- x2[qb.ix,'points'] - x2[[qb.ix[nTeams * posReq['qb']]],'points']]
  }

  if(posReq['rb'] == 0){
    x2[rb.ix, 'marg'] <- 0
  }
  else{
    x2[rb.ix, 'marg'] <- x2[rb.ix,'points'] - x2[[rb.ix[nTeams * posReq['rb']]],'points']]
  }

  if(posReq['te'] == 0){
    x2[te.ix, 'marg'] <- 0
  }
  else{
    x2[te.ix, 'marg'] <- x2[te.ix,'points'] - x2[[te.ix[nTeams * posReq['te']]],'points']]
  }

  if(posReq['wr'] == 0){
    x2[wr.ix, 'marg'] <- 0
  }
  else{
    x2[wr.ix, 'marg'] <- x2[wr.ix,'points'] - x2[[wr.ix[nTeams * posReq['wr']]],'points']]
  }

  # create a new data.frame subset by non-negative marginal points
  x3 <- x2 %>% filter(marg >= 0)

  # re-order by marginal points
  x3 <- x3 %>% arrange(desc(marg))

  # reset the row names
  rownames(x3) <- NULL

  # calculation for player value
  x3[,'value'] <- x3[,'marg']*(nTeams*cap-nrow(x3))/sum(x3[,'marg']) + 1

  # create a data.frame with more interesting columns
  x4 <- x3[,c('PlayerName','pos','points','marg','value')]
  head(x4)
  tail(x4)

  ## save dollar values as CSV file
  write_csv(x4, path=file)

  ## return data.frame with dollar values
  return(x4)
}
```

1. Call `x1 <- ffvalues('.')`

```
x1 <- ffvalues('.')
```

1. How many players are worth more than $20? (1 point)

```
# 46 players are worth more than $20
x1 %>% filter(value > 20) %>% nrow()
```

```
## [1] 46
```

1. Who is 15th most valuable running back (rb)? (1 point)

```
# Todd Gurley is the 15th most valuable 'rb'
x1 %>%
  filter(pos=='rb') %>%
  arrange(desc(value)) %>%
  slice(15)
```

```
## # A tibble: 1 x 5
##   PlayerName  pos   points  marg value
##   <chr>       <chr>  <dbl> <dbl> <dbl>
## 1 Todd Gurley rb       159.  24.5  26.7
```

1. Call `x2 <- ffvalues(getwd(), '16team.csv', nTeams=16, cap=150)`

```
x2 <- ffvalues(getwd(), '16team.csv', nTeams=16, cap=150)
```

1. How many players are worth more than $20? (1 point)

```
# 43 players are worth more than $20
x2 %>% filter(value > 20) %>% nrow()
```

```
## [1] 43
```

1. How many wide receivers (wr) are in the top 40? (1 point)

```
# 8 wide receivers (wr) are in the top 40
x2 %>%
      arrange(desc(value)) %>%
      head(40) %>%
      filter(pos=='wr') %>%
      nrow()
```

```
## [1] 8
```

1. Call:

   ```
   x3 <- ffvalues('.', 'qbheavy.csv', posReq=c(qb=2, rb=2, wr=3, te=1, k=0),
           points=c(fg=0, xpt=0, pass_yds=1/25, pass_tds=6, pass_ints=-2,
                   rush_yds=1/10, rush_tds=6, fumbles=-2, rec_yds=1/20, rec_tds=6))
   ```

   1. How many players are worth more than $20? (1 point)

   ```
   # 41 players are worth more than $20
   x3 %>% filter(value > 20) %>% nrow()
   ```

   ```
   ## [1] 41
   ```

   1. How many quarterbacks (qb) are in the top 30? (1 point)

```
# 12 qb are in the top 30
x3 %>%
      arrange(desc(value)) %>%
      head(30) %>%
```

```
      filter(pos=='qb') %>%
      nrow()
```

```
## [1] 12
```

### Question 2

**24 points**

Import the HAART dataset (`haart.csv`) from the GitHub repository into R, and perform the following manipulations: (4 points each)

```
# Read in the dataset
df <- read_csv("https://raw.githubusercontent.com/couthcommander/Bios6301/master/datasets/haart.csv",
               col_types=cols())
```

1. Convert date columns into a usable (for analysis) format. Use the `table` command to display the counts of the year from `init.date`.

```
# Parse dates using lubridate
df$init.date <- mdy(df$init.date)
df$last.visit <- mdy(df$last.visit)
df$date.death <- mdy(df$date.death)

# Create of table showing the frequency of each year for `init.date`
year(df$init.date) %>% table()
```

```
## .
## 1998 2000 2001 2002 2003 2004 2005 2006 2007
##    1    5   17   60  270  292  207  104   44
```

2. Create an indicator variable (one which takes the values 0 or 1 only) to represent death within 1 year of the initial visit. How many observations died in year 1?

```
df$years_to_death <- lubridate::time_length(difftime(df$date.death, df$init.date), "years")
df$within_one_year <- (df$years_to_death <= 1) %>% replace_na(0)

# 92 subjects died in year 1
sum(df$within_one_year)
```

```
## [1] 92
```

3. Use the `init.date`, `last.visit` and `death.date` columns to calculate a followup time (in days), which is the difference between the first and either the last visit or a death event (whichever comes first). If these times are longer than 1 year, censor them (this means if the value is above 365, set followup to 365). Print the quantile for this new variable.

```
# last.record is first of date.death or last.visit
df$last.record <- pmin(df$date.death, df$last.visit, na.rm=TRUE)

# get followup time in days
df$followup <- lubridate::time_length(difftime(df$last.record, df$init.date, ), "days")

# Censor at 1 year
df$followup <- pmin(df$followup, 365)

quantile(df$followup)
```

```
##     0%    25%    50%    75%   100%
```

```
##   0.00 320.75 365.00 365.00 365.00
```

4. Create another indicator variable representing loss to followup; this means the observation is not known to be dead but does not have any followup visits after the first year. How many records are lost-to-followup?

```
# lost to followup if 1) not dead and 2) no followup after first year
df$lost.to.followup <- (!df$death) & (df$followup != 365)

# 173 subjects were lost to followup
sum(df$lost.to.followup)
```

```
## [1] 173
```

5. Recall our work in class, which separated the `init.reg` field into a set of indicator variables, one for each unique drug. Create these fields and append them to the database as new columns. Which drug regimen are found over 100 times?

```
df %>%
  separate_rows(init.reg) %>%
  count(init.reg) %>%
  arrange(desc(n))
```

```
## # A tibble: 18 x 2
##    init.reg     n
##    <chr>    <int>
##  1 3TC        973
##  2 AZT        794
##  3 EFV        516
##  4 NVP        358
##  5 D4T        146
##  6 RTV         79
##  7 ABC         56
##  8 DDI         38
##  9 LPV         31
## 10 SQV         29
## 11 IDV         27
## 12 TDF         10
## 13 FTC          8
## 14 NFV          8
## 15 ATV          2
## 16 FPV          2
## 17 DDC          1
## 18 T20          1
```

```
# We see that 3TC, AZT, EFV, NVP, and D4T are found over 100 times

# Create indicator columns for init.reg
df %>%
  separate_rows(init.reg) %>%
  pivot_wider(names_from=init.reg,
              values_from=1,
              values_fill=0) -> df
```

6. The dataset `haart2.csv` contains a few additional observations for the same study. Import these and append them to your master dataset (if you were smart about how you coded the previous steps, cleaning the additional observations should be easy!). Show the first five records and the last five

records of the complete (and clean) data set.

```
# Read in haart
df1 <- read_csv("https://raw.githubusercontent.com/couthcommander/Bios6301/master/datasets/haart.csv",
                col_types=cols())


# Read in haart2
df2 <- read_csv("https://raw.githubusercontent.com/couthcommander/Bios6301/master/datasets/haart2.csv",
                col_types=cols())

df <- bind_rows(df1,df2)

clean <- function(df){
  # Parse dates using lubridate
  df$init.date <- mdy(df$init.date)
  df$last.visit <- mdy(df$last.visit)
  df$date.death <- mdy(df$date.death)

  # Create indicator for dying within 1 year
  df$years_to_death <- lubridate::time_length(difftime(df$date.death, df$init.date), "years")
  df$within_one_year <- (df$years_to_death <= 1) %>% replace_na(0)

  # Create indicator for followup
  # ----------------------------------
  # last.record is first of date.death or last.visit
  df$last.record <- pmin(df$date.death, df$last.visit, na.rm=TRUE)
  # get followup time in days
  df$followup <- lubridate::time_length(difftime(df$last.record, df$init.date, ), "days")
  # Censor at 1 year
  df$followup <- pmin(df$followup, 365)

  # lost to followup if 1) not dead and 2) no followup after first year
  df$lost.to.followup <- (!df$death) & (df$followup != 365)

    # Create indicator columns for init.reg
  df %>%
    separate_rows(init.reg) %>%
    pivot_wider(names_from=init.reg,
                values_from=1,
                values_fill=0) -> df

  return(df)
}

df <- clean(df)

head(df)
```

```
## # A tibble: 6 x 33
##     age  aids cd4baseline logvl weight hemoglobin init.date  last.visit death
##   <dbl> <dbl>       <dbl> <dbl>  <dbl>      <dbl> <date>     <date>     <dbl>
## 1    25     0          NA    NA   NA           NA 2003-07-01 2007-02-26     0
## 2    49     0         143    NA   58.1         11 2004-11-23 2008-02-22     0
## 3    42     1         102    NA   48.1          1 2003-04-30 2005-11-21     1
```

```
## 4    33     0         107    NA   46           NA 2006-03-25 2006-05-05     1
## 5    27     0          52     4   NA           NA 2004-09-01 2007-11-13     0
## 6    34     0         157    NA   54.9         NA 2003-12-02 2008-02-28     0
## # ... with 24 more variables: date.death <date>, years_to_death <dbl>,
## #   within_one_year <dbl>, last.record <date>, followup <dbl>,
## #   lost.to.followup <lgl>, `3TC` <dbl>, AZT <dbl>, EFV <dbl>, NVP <dbl>,
## #   D4T <dbl>, ABC <dbl>, DDI <dbl>, IDV <dbl>, LPV <dbl>, RTV <dbl>,
## #   SQV <dbl>, FTC <dbl>, TDF <dbl>, DDC <dbl>, NFV <dbl>, T20 <dbl>,
## #   ATV <dbl>, FPV <dbl>
```

```
tail(df)
```

```
## # A tibble: 6 x 33
##     age  aids cd4baseline logvl weight hemoglobin init.date  last.visit death
##   <dbl> <dbl>       <dbl> <dbl>  <dbl>      <dbl> <date>     <date>     <dbl>
## 1  31       0         102 NA      61.7         11 2003-05-22 2008-03-07     0
## 2  40       1         131 NA      46.3          8 2003-07-03 2008-02-29     0
## 3  27       0         232 NA      NA           NA 2003-12-01 2004-01-05     0
## 4  38.7     0         170 NA      84           NA 2002-09-26 2004-03-29     0
## 5  23      NA         154  4.00   65.5         14 2007-01-31 2007-04-16     0
## 6  31       0         236 NA      45.8         NA 2003-12-03 2007-10-11     0
## # ... with 24 more variables: date.death <date>, years_to_death <dbl>,
## #   within_one_year <dbl>, last.record <date>, followup <dbl>,
## #   lost.to.followup <lgl>, `3TC` <dbl>, AZT <dbl>, EFV <dbl>, NVP <dbl>,
## #   D4T <dbl>, ABC <dbl>, DDI <dbl>, IDV <dbl>, LPV <dbl>, RTV <dbl>,
## #   SQV <dbl>, FTC <dbl>, TDF <dbl>, DDC <dbl>, NFV <dbl>, T20 <dbl>,
## #   ATV <dbl>, FPV <dbl>
```