Lab 6: Functions, Control Flow, dplyr and ggplot2

Stat 133, Fall 2016, Prof. Sanchez

Preparation for Midterm Project

The goal of this lab is to provide some practice with most of the concepts we have covered so far in the course:

- writing functions
- working with control flow structures
- handling strings
- data manipulation with dplyr
- producing charts with ggplot2

Women's High Jump World Record Progression

In this lab you will be working with data of world records in women's high jump (source Wikipedia). The data set is in the file `womens-high-jump-records.csv` available in the `data/` folder from the github repository:

https://raw.githubusercontent.com/ucb-stat133/stat133-fall-2016/master/data/womens-high-jump-records.csv

Assuming that the data file is already in your working directory, you can use any of the reading table functions to import it in R:

```
# read data
dat <- read.csv("womens-high-jump-records.csv", stringsAsFactors = FALSE)

# take a peek
head(dat)
```

```
##   height     first       last country day month year
## 1  1.460     Nancy   Voorhees     USA  20   May 1922
## 2  1.485 Elizabeth      Stine     USA  26   May 1923
## 3  1.485    Sophie     Eliott     GBR   6   Aug 1923
## 4  1.524   Phyllis      Green     GBR  11   Jul 1925
## 5  1.552   Phyllis      Green     GBR   2   Aug 1926
## 6  1.580     Ethel Catherwood     CAN   6   Sep 1926
```

The data set has 7 columns:

- height is the height record in meters
- first is the athlete's first name
- last is the athlete's first name
- country is the athlete's country
- day is the day of month (record's date)
- month is the name of month (record's date)
- year is the year (record's date)

Athlete's name

The function `paste()`—and its sister `paste0()`—allows you form character strings by *pasting* any number of vectors:

```r
paste("Go", "Bears", "!")
```

```
## [1] "Go Bears !"
```

```r
paste0("Go", "Bears", "!")
```

```
## [1] "GoBears!"
```

Use `paste()` to create a vector `athlete_name` that shows the athlete's full name (first and last name)

```r
# your vector athlete_name
```

Take your vector `athlete_name`, and use `paste()` again, to create a vector `athlete` in which each element shows the full name followed by the country within parentheses, for example: `Nancy Voorhees (USA)`:

```r
# your vector athlete
```

Name of the month

As you can tell, the column `month` has the names of the months in abbreviated format. But what if we want to get the full name? Interestingly, R comes with a built-in vector `month.name` that has the full English names of the months. So let's use `month.name` to write code that converts an abbreviated month into its full name.

For testing purposes, let's consider one month, say `"Jan"`. One option is to use `switch()`. Complete the following call to `switch()` with the corresponding months, and include an option `NA` at the end for when the input does not match with any of the available switch values. (Make sure to change the chunk option `eval = FALSE` to `eval = TRUE`).

```r
a <- "Jan"

switch(a,
       "Jan" = month.name[1],
       "Feb" = month.name[2],
       "Mar" = month.name[3],
       # complete the code)
```

Now encapsulate the `switch()` in a function `expand_month()`. This function takes the abbreviated name, and returns the full name. In the code chunk, add a description of what the function does, what's the expected input, and what's the returned value:

```r
# your expand_month() function


# test it with expand_month("Apr") and expand_month("Xyz")
```

Once you have your `expand_month()` function, the next task is to figure out how to use it so that you can take the vector `month` and convert all its values to abbreviated names.

For loop

One option to replace month names is to use a `for` loop. Compute a vector `new_month` by writing a `for` loop that iterates through all the elements in `month`, and switches the value to full name, using `expand_month()`:

```
# your for loop to get new_month
```

Another alternative is to use one of the functions from the `apply` family: `lapply()` and `sapply()`. Check the documentation for `lapply()` and `sapply()`, and see some examples. Even better, take the column `month`, your `expand_month()` function, and pass them to both `lapply()` and `sapply()` and see what happens:

```
# use lapply and sapply
```

Build a vector `full_month` with the full name:

```
# your vector full_month
```

Record dates

Take the vector `day`, `full_month`, and `year`, and `paste()` them in order to build a vector `dates`: e.g. `"20 May 1922", "26 May 1923", ...`

```
# your vector dates
```

Your vector `dates` is just a character vector. But you can use it get a vector of class `"Date"`.

Check the documentation of the function `as.Date()`. Use `as.Date()` to reformat your vector `dates` with `"%d %B %Y"` format:

```
# date formatting
# complete as necessary: dates <- as.Date(dates, ...)
```

Derived Data Frame

The next task consists of building a new data frame `womens` using the vectors `height`, `athlete_name`, `country`, and `date`. When building the data frame use the following column names:

- `Height = height`
- `Athlete = athlete_name`
- `Country = country`
- `Date = dates`

```
# your data frame "womens"
```

Manipulating data with `"dplyr"`

Let's do some data manipulation and aggregation with the R package `"dplyr"`:

```r
library(dplyr)
```

Extract distinct (unique) countries:

```r
# distinct countries
```

The column `height` in the original data frame `dat` is in meters. You can use `transmute()` to compute a vector `height_inches` (1 meter = 39.3701 inches):

```r
# compute vector of height in inches
```

Likewise, you can use the function `mutate()` to add a new column `height_inches` to `dat`:

```r
# add column height_inches to dat
```

Take the original data frame `dat` and use `"dplyr"` to compute the:

- number of records per country
- number of records per country in decreasing order
- number of records per year
- number of records per year in decresing order

```r
# records per country

# records per country in descending order

# records per year

# records per year in descending order
```

Now take the data frame `womens` and use `"dplyr"` to compute the:

- number of records per athlete
- number of records per athlete in decreasing order

```r
# number of records per country

# number of records per country in decreasing order
```

Visualizing Records with `"ggplot2"`

Now that you have the `dat` and `womens` data frame, let's use `"ggplot2"` to get some plots:

```r
library(ggplot2)
```

Take `dat` and get a bar-chart with the number of records per country, with bars in descending order:

```r
# your scatterplot
```

Now take `womens` and use `geom_point()` to get a scatterplot of `x = Date` and `y = Height`. Add a title `"Women's High Jump Record Progression"`.

```r
# your scatterplot
```

Now add a line, you can try using `geom_line()`

```r
# scatterplot with line
```

Instead of adding a simple line, use the `geom_step()` line:

```r
# scatterplot with step line
```

Now use `Country` to color the points

```r
# scatterplot with step line, color points by country
```

Solutions

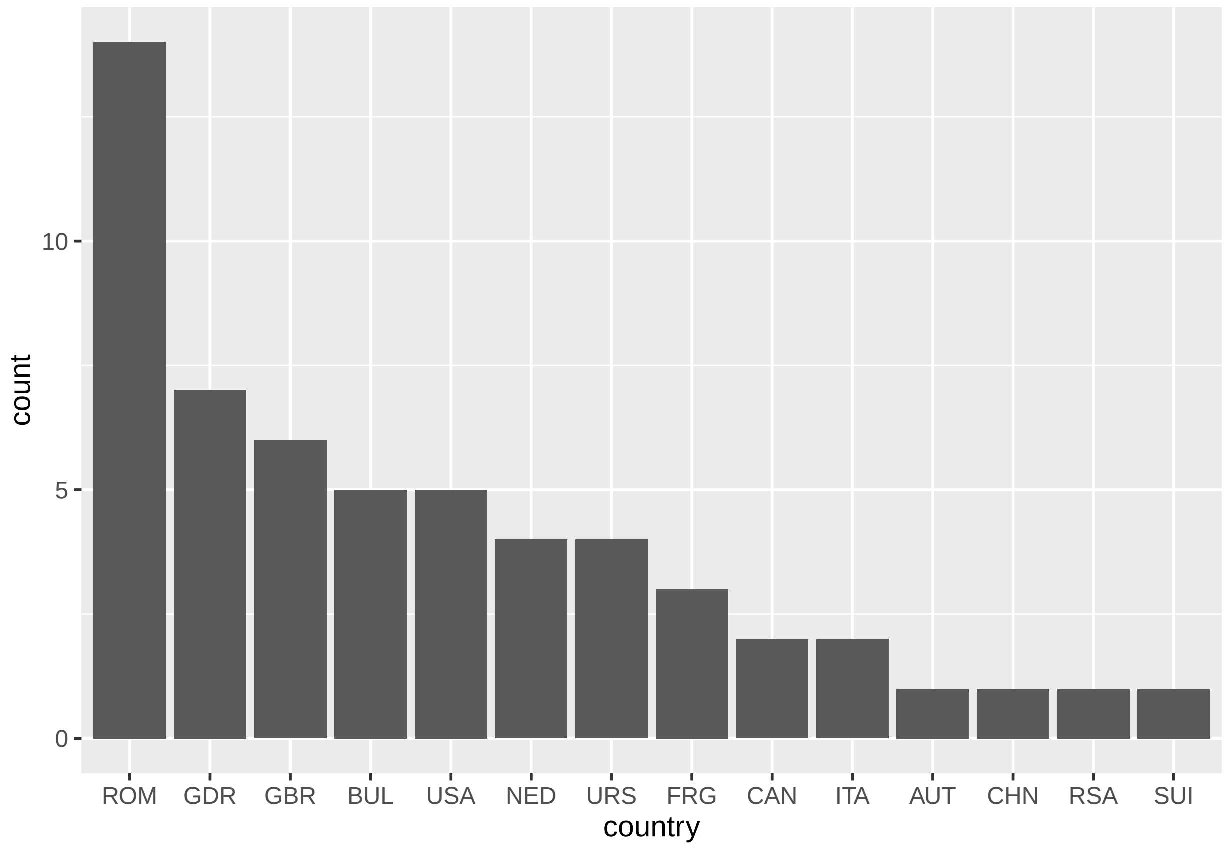
```r
# vector athlete_name
athlete_name <- paste(dat$first, dat$last)

# vector athlete
athlete <- paste0(athlete_name, " (", dat$country, ")")

# month in full name
a <- "Jan"

switch(a,
       "Jan" = month.name[1],
       "Feb" = month.name[2],
       "Mar" = month.name[3],
       "Apr" = month.name[4],
       "May" = month.name[5],
       "Jun" = month.name[6],
       "Jul" = month.name[7],
       "Aug" = month.name[8],
       "Sep" = month.name[9],
       "Oct" = month.name[10],
       "Nov" = month.name[11],
       "Dec" = month.name[12],
       NA)
```

```
## [1] "January"
```

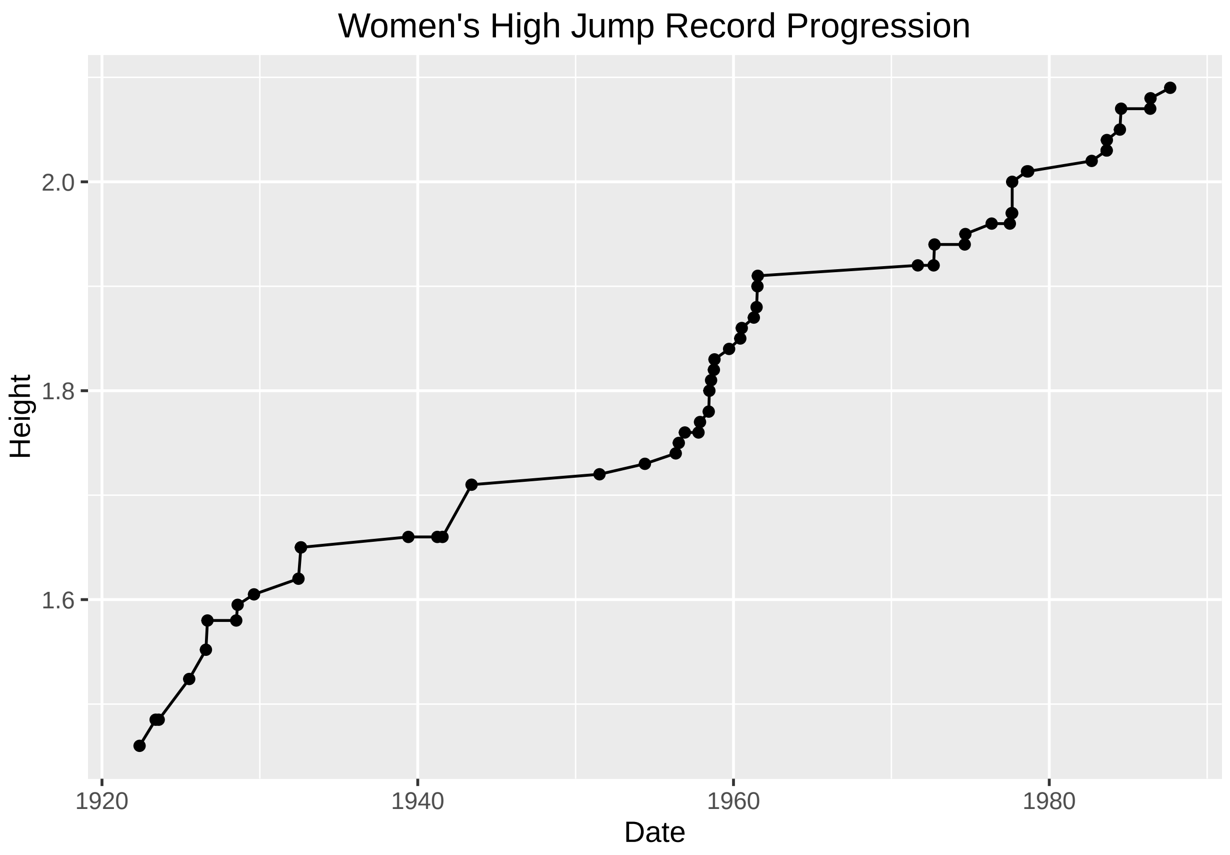
```r
# your expand_month() function
expand_month <- function(mon = 'Jan') {
  switch(mon,
       "Jan" = month.name[1],
       "Feb" = month.name[2],
       "Mar" = month.name[3],
       "Apr" = month.name[4],
       "May" = month.name[5],
       "Jun" = month.name[6],
       "Jul" = month.name[7],
       "Aug" = month.name[8],
       "Sep" = month.name[9],
       "Oct" = month.name[10],
       "Nov" = month.name[11],
       "Dec" = month.name[12],
       NA)
}


# Your for loop to get new_month
# 1st) initialize an empty character vector
new_month <- character(length(dat$month))

# now the loop
```

```r
for (m in 1:length(dat$month)) {
  new_month[m] <- expand_month(dat$month[m])
}


# use lapply and sapply to get full name of month
lapply_month <- lapply(dat$month, expand_month)
sapply_month <- sapply(dat$month, expand_month)

# vector full_month
full_month <- sapply(dat$month, expand_month)

# vector dates
dates <- paste(dat$day, full_month, dat$year)

# format date
dates <- as.Date(dates, "%d %B %Y")

# data frame "womens"
womens <- data.frame(
  Height = dat$height,
  Athlete = athlete_name,
  Country = dat$country,
  Date = dates
)


# ===================================================================
# Data Manipulation with dplyr
# ===================================================================

# distinct countries
unique_countries <- dat %>% distinct(country)

# compute vector of height in inches
height_inches <- transmute(dat, height_inches = height * 39.3701)

# add column height_inches to 'dat'
dat <- mutate(dat, height_inches = height * 39.3701)

# records per country
per_country <- dat %>%
  group_by(country) %>%
  summarise(records = n())

# records per country in descending order
per_country_desc <- per_country %>%
  arrange(desc(records))

# records per year
per_year <- dat %>%
  group_by(year) %>%
  summarise(records = n())
```

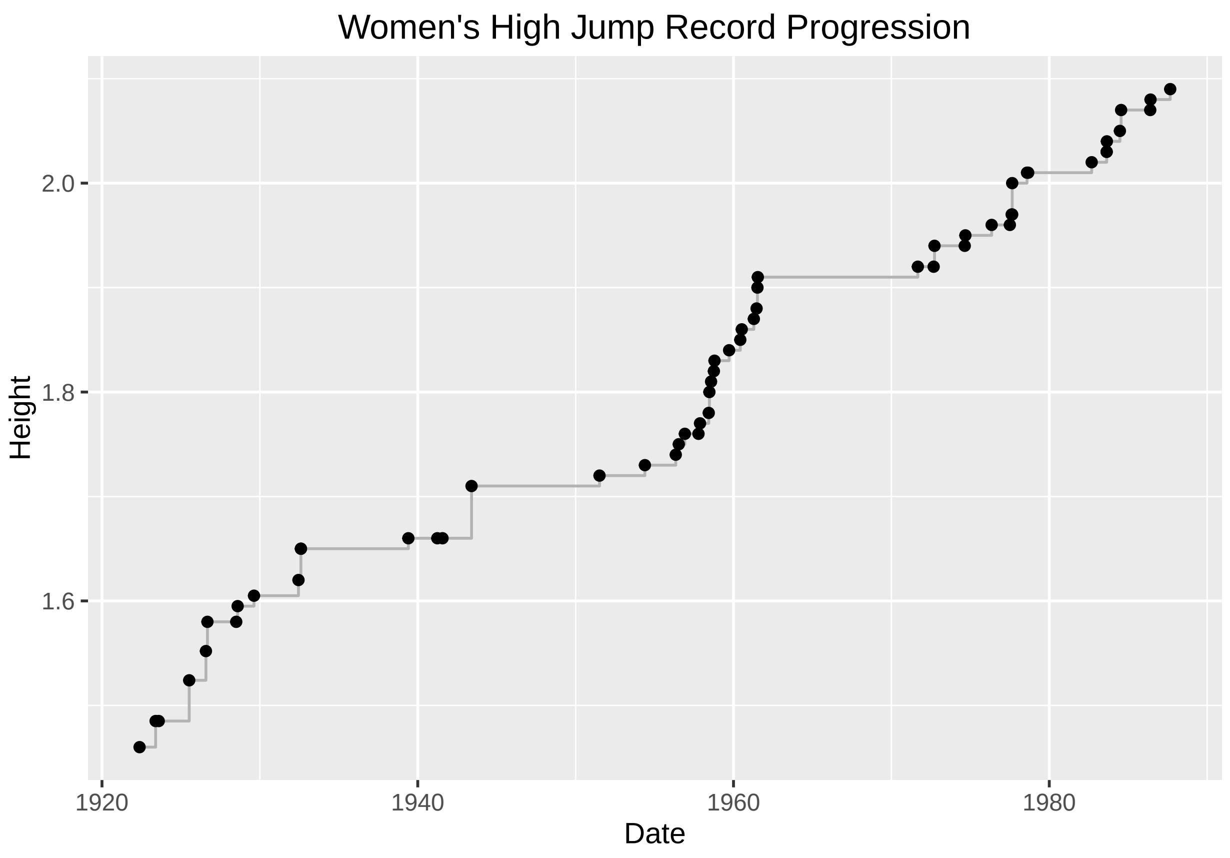
```
# records per year in descending order
per_year_desc <- per_year %>%
  arrange(desc(records))


# records per athlete
per_athlete <- womens %>%
  group_by(Athlete) %>%
  summarise(records = n())

# number of records per country in decreasing order
per_athlete_desc <- per_country %>%
  arrange(desc(records))


# ================================================================
# Data Visualiztion with ggplot2
# ================================================================

# bar-chart with the number of records per country:
# to order the bars, you can use scale_x_discrete() and specify
# the order of the bars
ggplot(dat, aes(x = country)) +
  geom_bar() +
  scale_x_discrete(limits = per_country_desc$country)
```



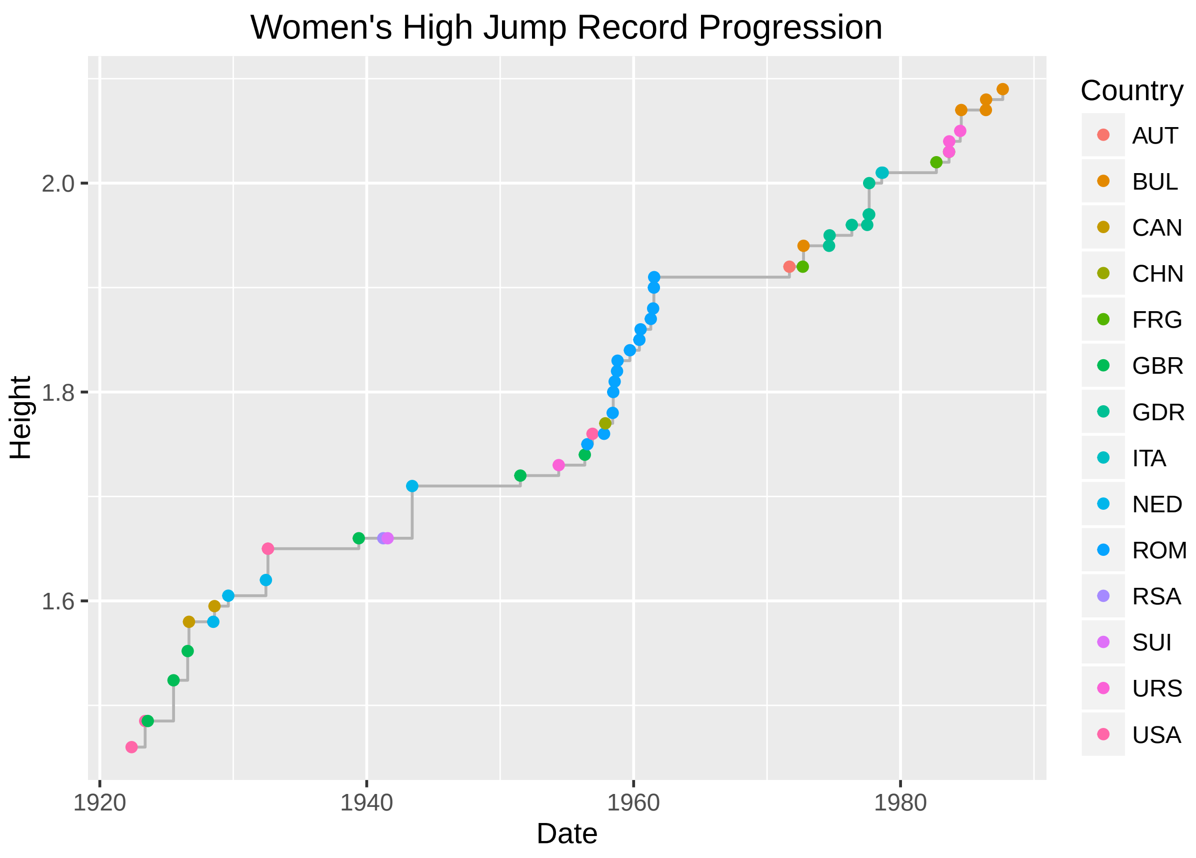
```
# ggplot object (for convenient purposes and save retyping)
progression <- ggplot(data = womens, aes(x = Date, y = Height)) +
  ggtitle("Women's High Jump Record Progression")

# scatterplot with line
progression + geom_line() + geom_point()
```



```
# scatterplot with step line
progression + geom_step(color = "gray70") + geom_point()
```



```r
# scatterplot with step line, color points by country
progression +
  geom_step(color = "gray70") +
  geom_point(aes(color = Country))
```


Women's High Jump Record Progression
Height
2.0
1.8
1.6
1920
1940
1960
1980
Date
Country
AUT
BUL
CAN
CHN
FRG
GBR
GDR
ITA
NED
ROM
RSA
SUI
URS
USA