RYAN STRAIGHT

THE OLYMPICS

Introduction

The Olympics has become a financial juggernaut from the viewpoint of both the hosting country and the value of being a medalist. In 2016, the Rio Olymipcs cost $13.1 billion US dollars (7.23 billion reals) to host.[1]. This included "a subway line, a doping laboratory, a renovated port and cleanup of pouted Guanabara Bay."

[1] Rio Olympics cost $13.1 billion

The value of *gold* is more than the value of the medal, of course. Countries reward their medalists depending on which medal they bring home and these rewards vary drastically from country to country. Singapore, for example, rewards gold medalists with $1 million USD while Canada pays a comparatively paltry $15,000 USD[2]. Advertising sales during the Rio Olympics in 2016 totaled $1.2 billion USD.[3]

[2] Here's how much Olympic athletes earn in 12 different countries

[3] NBC says it has topped $1 billion in national ad sales for 2020 Summer Olympics

Needless to say, it behooves interested parties to be able to predict just who, when, and where medalists will crop up, whether this is in an effort to determine if the 12-year-old male gymnast in Sweden is likely to be a 16-year-old gold medalist in four years or if this or that country is worth scouting in for talent given their past medal winnings.

The Chosen Data

The purpose of this project is to apply a variety of classification and predictive methodologies to a chosen data set for the purposes of demonstrating knowledge and skills developed throughout the semester. The dataset chosen for this project is 120 years of Olympic history: athletes and results[4] This particular dataset was chosen for a variety of reasons:

[4] Described by the creator as "basic bio data on athletes and medal results from Athens 1896 to Rio 2016."

- It is relatively large, coming in 271,116 rows when loaded raw.
- There is a variety of variable types to work with, providing a range of options when it comes to different classification and preditive tests.
- It affords a certain level of approachability and familiarity by virtue of its content; after all, we all know Olympic medalists.

The purpose of this study, then, is to examine the particulars of the Olympic historical record and attempt to identify trends and make predictions thereby. Three possibilities for this data in this context come to mind:

1. What trends are apparent in nations' medal totals?
2. What demographics contribute to medaling?
3. Can we predict a medal based on a collection of an athlete's demographics?

A Description of the Data

The data originates in a Kaggle.com dataset provided by Randi Griffin. According to Griffin,

> This is a historical dataset on the modern Olympic Games, including all the Games from Athens 1896 to Rio 2016 [scraped from] www.sports-reference.com in May 2018.... Note that the Winter and Summer Games were held in the same year up until 1992. After that, they staggered them such that Winter Games occur on a four year cycle starting with 1994, then Summer in 1996, then Winter in 1998, and so on. A common mistake people make when analyzing this data

> is to assume that the Summer and Winter Games have always been staggered.
>
> — Randi Griffin

The dataset is delivered in two files: `athlete_events.csv` and `noc_regions.csv`. The descriptions are provided in the data source.

File	Variable	Data type	Data format	Description
athlete_events.csv				
	ID	ind	int	Unique number for each athlete
	Name	ind	chr	Athlete's name
	Sex	dep	chr	M or F
	Age	ind	int	Integer
	Height	ind	int	In centimeters
	Weight	ind	num	In kilograms
	Team	ind	chr	Team name
	NOC	ind	chr	National Olympic Committee 3 letter code
	Games	ind	chr	Year and season
	Year	ind	int	Integer
	Season	ind	chr	Summer or Winter
	City	ind	chr	City
	Sport	ind	chr	Sport
	Event	ind	chr	Event
	Medal	dep	chr	Gold, Silver, Bronze, or `NA`
noc_regions.csv				
	NOC	ind	chr	National Olympic Committee 3 letter code
	region	ind	chr	Country name (matches with regions in `map_data("world")`
	notes	ind	chr	Notes

Data Pre-Processing

Fortunately, Griffin's scraping techniques prove tidy and in need of very little cleaning, all things considered. The entirety of the loading and tidying is as follows:

```
# tidy up the titles
athlete_events <- athlete_events %>% clean_names()
noc_regions <- noc_regions %>% clean_names()

# Join up athlete_events and noc_regions to get a nice country name
olympics <- as_tibble(athlete_events %>% left_join(noc_regions, by = "noc"))

# Switch to factors
olympics <- olympics %>%
  mutate(across(c("sex", "team", "noc", "games", "year", "sport", "city",
    "region", "season"), factor))

# Replace NAs in "medal" with "None"
olympics$medal <- olympics$medal %>%
  replace_na("none") %>%
  as_factor()

# There are way too many sports and a few only happened a couple times.
# Pare those down to the top 50, naming the rest "Other."
olympics$sport <- olympics$sport %>%
  fct_lump_n(n = 51)
```

With this we can now explore the data a bit easier. Note that each row in this dataset is for an *athlete*.

Summary table of relevant data

age

height

weight

year

medal

Min. :10.00

Min. :127.0

Min. : 25.0

1992 : 16413

none :231333

1st Qu.:21.00

1st Qu.:168.0

1st Qu.: 60.0

1988 : 14676

Gold : 13372

Median :24.00

Median :175.0

Median : 70.0

2000 : 13821

Bronze: 13295

Mean :25.56

Mean :175.3

Mean : 70.7

1996 : 13780

Silver: 13116

3rd Qu.:28.00

3rd Qu.:183.0

3rd Qu.: 79.0

2016 : 13688

NA

Max. :97.00

Max. :226.0

Max. :214.0

2008 : 13602

NA

NA's :9474

NA's :60171

NA's :62875

(Other):185136

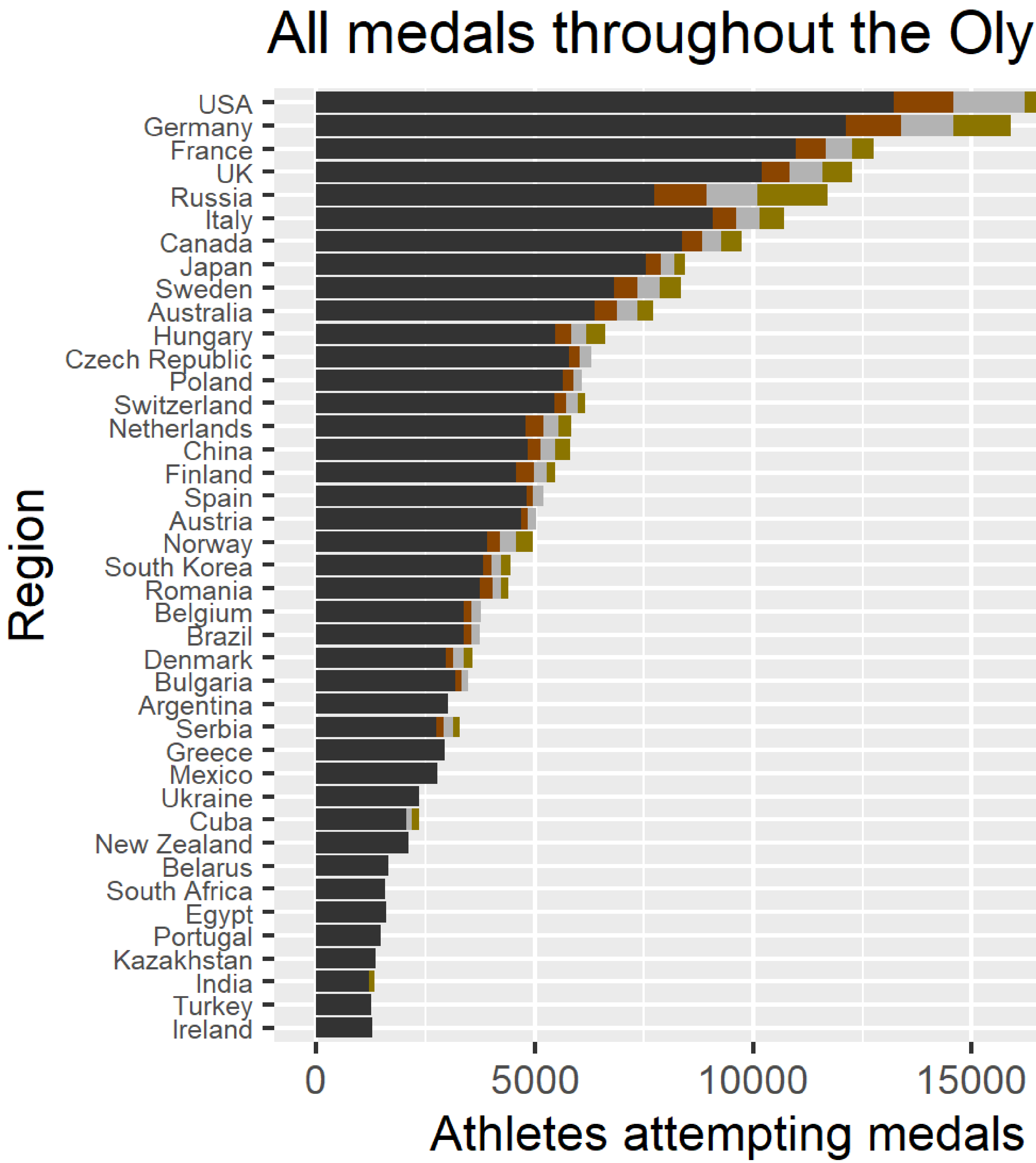
NA

Follows is a look at the structure of the numerical data to verify data formats are as expected.

variable	class	first__values
id	double	1, 2, 3, 4, 5, 5
name	character	A Dijiang, A Lamusi, Gunnar Nielsen Aaby, Edgar Lindenau Aabye, Christine Jacoba Aaftink, C
sex	integer	M, M, M, M, F, F
age	double	24, 23, 24, 34, 21, 21
height	double	180, 170, NA, NA, 185, 185
weight	double	80, 60, NA, NA, 82, 82
team	integer	China, China, Denmark, Denmark/Sweden, Netherlands, Netherlands
noc	integer	CHN, CHN, DEN, DEN, NED, NED
games	integer	1992 Summer, 2012 Summer, 1920 Summer, 1900 Summer, 1988 Winter, 1988 Winter
year	integer	1992, 2012, 1920, 1900, 1988, 1988
season	integer	Summer, Summer, Summer, Summer, Winter, Winter
city	integer	Barcelona, London, Antwerpen, Paris, Calgary, Calgary
sport	integer	Basketball, Judo, Football, Other, Speed Skating, Speed Skating
event	character	Basketball Men’s Basketball, Judo Men’s Extra-Lightweight, Football Men’s Football, Tug-Of-Wa
medal	integer	none, none, none, Gold, none, none
region	integer	China, China, Denmark, Denmark, Netherlands, Netherlands
notes	character	NA, NA, NA, NA, NA, NA
decade	integer	1990s, 2010s, 1920s, 1900s, 1980s, 1980s

Descriptive Analysis

Due to the sheer number of regions that have existed and send athletes to the modern Olympic games (each of which may attempt multiple medal attempts in different events), the following depicts medal attempts by regions in the top 50% of all medal attempt totals.


All medals throughout the Oly
Region
USA
Germany
France
UK
Russia
Italy
Canada
Japan
Sweden
Australia
Hungary
Czech Republic
Poland
Switzerland
Netherlands
China
Finland
Spain
Austria
Norway
South Korea
Romania
Belgium
Brazil
Denmark
Bulgaria
Argentina
Serbia
Greece
Mexico
Ukraine
Cuba
New Zealand
Belarus
South Africa
Egypt
Portugal
Kazakhstan
India
Turkey
Ireland
0
5000
10000
15000
Athletes attempting medals

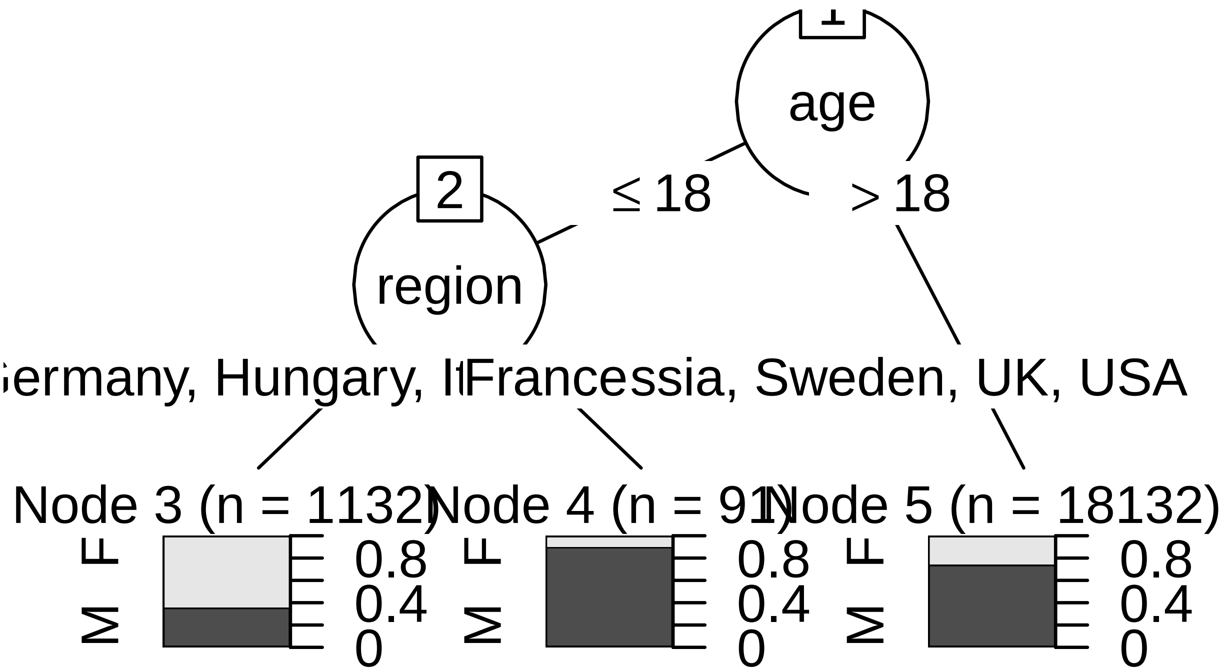
Decision Tree

Given the top ten medal-winning countries, a decision tree can be produced that demonstrates where gold medal winners are likely to come from, what gender they're likely to be, and what age.

```
##
## Call:
## C5.0.default(x = train[, medal_predictors], y = train$sex)
##
##
## C5.0 [Release 2.07 GPL Edition]      Tue Aug 25 12:51:37 2020
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 19355 cases (4 attributes) from undefined.data
##
## Decision tree:
##
## age > 18: M (18244.6/4707.5)
## age <= 18:
## :...region in {Australia,Canada,Germany,Hungary,Italy,Russia,Sweden,UK,
##     :          USA}: F (1057.1/332.7)
##     region = France: M (53.4/8.1)
##
##
## Evaluation on training data (19355 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       3 5038(26.0%)   <<
##
##
```

```
##      (a)   (b)    <-classified as
##     ----  ----
##      724  4716    (a): class F
##      322 13593    (b): class M
##
##
##  Attribute usage:
##
##   98.61% age
##    7.05% region
##
##
## Time: 0.0 secs
```

And the plotted decision tree is as follows:

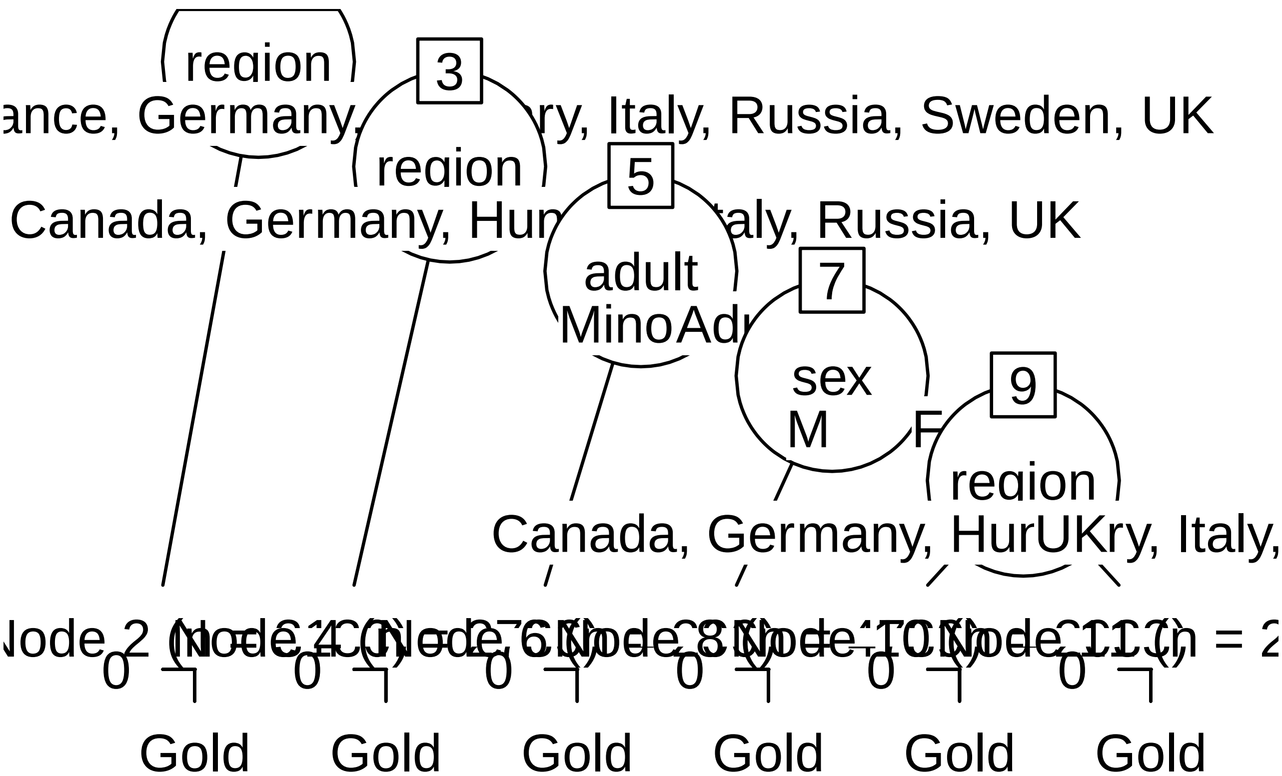


As we can see, if over 18, that athlete is considerably more likely to be male, which makes up the vast majority of athletes throughout the modern Olympic's history. Once athletes' age drops under 18 years old, the region they come from and then further age breakdown are the determining factors for predicting the athlete's sex.

The gender makeup has changed considerably over the past half century. Let's look at that now. Examining the factors that go into medal winning while paying special attention to whether the athlete is an adult or a minor.

```
##
## Call:
## C5.0.default(x = trainB[, medal_predictorsB], y = trainB$medal)
##
##
## C5.0 [Release 2.07 GPL Edition]      Tue Aug 25 12:51:37 2020
## -------------------------------
##
## Class specified by attribute `outcome'
##
## Read 13190 cases (4 attributes) from undefined.data
##
## Decision tree:
##
## region = USA: Gold (3130/1693)
## region in {Australia,Canada,France,Germany,Hungary,Italy,Russia,Sweden,UK}:
## :...region in {Australia,France,Sweden}: Bronze (2786/1786)
##     region in {Canada,Germany,Hungary,Italy,Russia,UK}:
##     :...adult = Minor: Silver (118.6/70.5)
##         adult = Adult:
##         :...sex = M: Gold (4865.5/3006)
##             sex = F:
##             :...region in {Canada,Germany,Hungary,Italy,
##                 :          Russia}: Gold (2033/1287)
##                 region = UK: Bronze (256.9/145)
##
##
## Evaluation on training data (13190 cases):
##
##      Decision Tree
##    ----------------
##    Size      Errors
##
##       6 7988(60.6%)   <<
##
##
##     (a)   (b)   (c)    <-classified as
##    ----  ----  ----
##    4043   867    34    (a): class Gold
```

```
##     2952  1112    35    (b): class Bronze
##     3036  1064    47    (c): class Silver
##
##
##  Attribute usage:
##
##  100.00% region
##   54.27% sex
##   53.93% adult
##
##
## Time: 0.0 secs
```

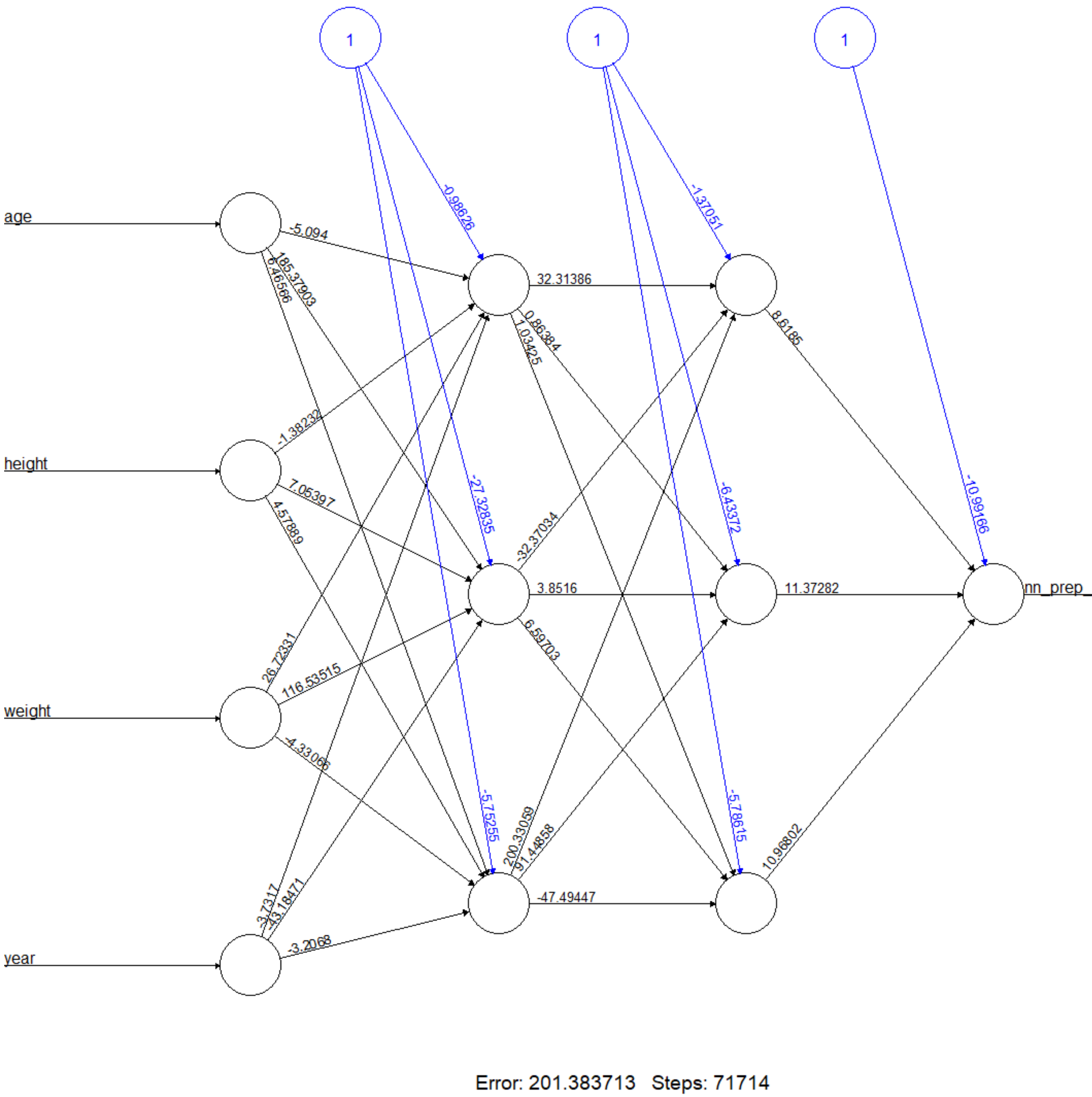


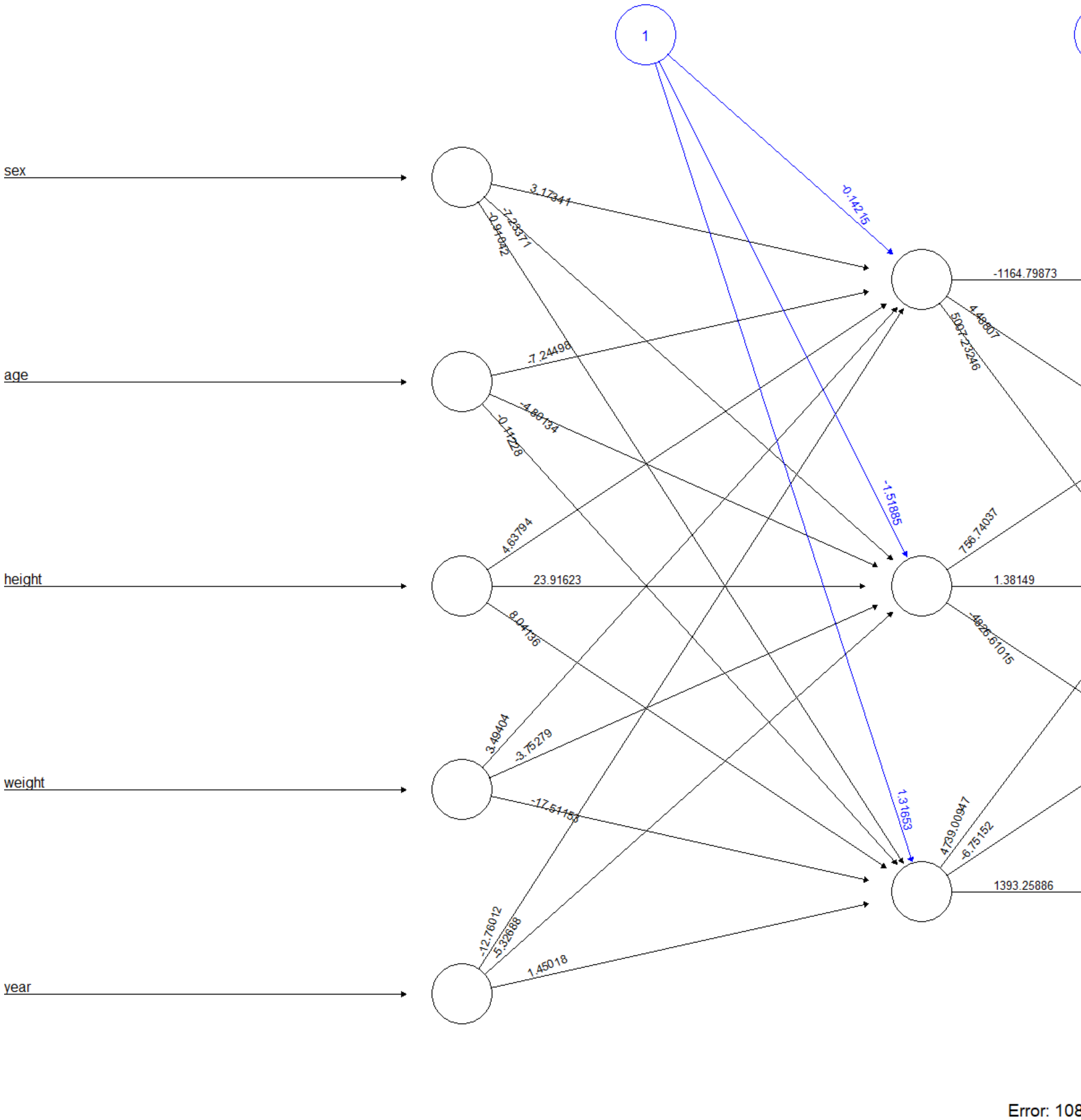
Neural Networks

There are, of course, other ways of predicting attributes. One such way that has garnered renewed interest over recent years is the *neural network*, made possible here by the `neuralnet` package.

By feeding the network with age, height, weight, and year, we can determine the sex of the athlete.

To determine a gold medal winner…


age
height
weight
year
nn_prep_
1
1
1
-5.094
185.37903
6.46566
-1.38232
7.05397
4.57889
26.72331
116.53515
-4.33066
-3.7317
-43.18471
-3.2068
-0.98626
-27.32835
-5.75255
32.31386
0.86384
1.03425
-32.37034
3.8516
6.59703
200.33059
91.44858
-47.49447
-1.37051
-6.43372
-5.78615
8.6185
11.37282
10.96802
-10.99166
Error: 201.383713 Steps: 71714



1
sex
age
height
weight
year
3.17341
-7.23371
-0.91042
-0.14215
-7.24498
-4.80134
-0.11228
4.63794
23.91623
8.04136
3.49404
-3.75279
-17.51153
-12.76012
-5.32688
1.45018
-1.51885
1.31653
-1164.79873
4.48807
5007.23246
756.74037
1.38149
-4826.61015
4739.00947
-6.75152
1393.25886

References

```
devtools::session_info()
```

```
## - Session info ---------------------------------------------------------------
##  setting  value
##  version  R version 4.0.2 (2020-06-22)
##  os       Windows 10 x64
##  system   x86_64, mingw32
##  ui       RTerm
##  language (EN)
##  collate  English_United States.1252
##  ctype    English_United States.1252
##  tz       America/Phoenix
##  date     2020-08-25
##
## - Packages -------------------------------------------------------------------
##  package     * version    date       lib source
##  assertthat    0.2.1      2019-03-21 [1] CRAN (R 4.0.0)
##  backports     1.1.8      2020-06-17 [1] CRAN (R 4.0.0)
##  blob          1.2.1      2020-01-20 [1] CRAN (R 4.0.0)
##  broom         0.7.0.9000 2020-05-18 [1] Github (tidymodels/broom@aae4322)
##  C50         * 0.1.3.1    2020-05-26 [1] CRAN (R 4.0.2)
##  callr         3.4.3      2020-03-28 [1] CRAN (R 4.0.0)
##  cellranger    1.1.0      2016-07-27 [1] CRAN (R 4.0.0)
##  citr        * 0.3.2      2019-08-19 [1] CRAN (R 4.0.0)
##  cli           2.0.2      2020-02-28 [1] CRAN (R 4.0.0)
##  colorspace    1.4-1      2019-03-18 [1] CRAN (R 4.0.0)
##  crayon        1.3.4      2017-09-16 [1] CRAN (R 4.0.0)
##  Cubist        0.2.3      2020-01-10 [1] CRAN (R 4.0.2)
##  DBI           1.1.0      2019-12-15 [1] CRAN (R 4.0.0)
##  dbplyr        1.4.4      2020-05-27 [1] CRAN (R 4.0.0)
##  desc          1.2.0      2018-05-01 [1] CRAN (R 4.0.0)
##  devtools      2.3.1      2020-07-21 [1] CRAN (R 4.0.2)
##  digest        0.6.25     2020-02-23 [1] CRAN (R 4.0.0)
##  dplyr       * 1.0.1      2020-07-31 [1] CRAN (R 4.0.2)
##  ellipsis      0.3.1      2020-05-15 [1] CRAN (R 4.0.0)
```

```
## evaluate      0.14      2019-05-28 [1] CRAN (R 4.0.0)
## fansi         0.4.1     2020-01-08 [1] CRAN (R 4.0.0)
## farver        2.0.3     2020-01-16 [1] CRAN (R 4.0.0)
## fastmap       1.0.1     2019-10-08 [1] CRAN (R 4.0.0)
## forcats     * 0.5.0     2020-03-01 [1] CRAN (R 4.0.0)
## formatR       1.7       2019-06-11 [1] CRAN (R 4.0.2)
## Formula       1.2-3     2018-05-03 [1] CRAN (R 4.0.0)
## fs            1.5.0     2020-07-31 [1] CRAN (R 4.0.2)
## generics      0.0.2     2018-11-29 [1] CRAN (R 4.0.0)
## gganimate   * 1.0.6     2020-07-08 [1] CRAN (R 4.0.2)
## ggplot2     * 3.3.2     2020-06-19 [1] CRAN (R 4.0.2)
## glue          1.4.1     2020-05-13 [1] CRAN (R 4.0.0)
## gtable        0.3.0     2019-03-25 [1] CRAN (R 4.0.0)
## haven         2.3.1     2020-06-01 [1] CRAN (R 4.0.0)
## here        * 0.1       2017-05-28 [1] CRAN (R 4.0.0)
## highr         0.8       2019-03-20 [1] CRAN (R 4.0.0)
## hms           0.5.3     2020-01-08 [1] CRAN (R 4.0.0)
## htmltools     0.5.0     2020-06-16 [1] CRAN (R 4.0.2)
## httpuv        1.5.4     2020-06-06 [1] CRAN (R 4.0.2)
## httr          1.4.2     2020-07-20 [1] CRAN (R 4.0.2)
## inum          1.0-1     2019-04-25 [1] CRAN (R 4.0.2)
## janitor     * 2.0.1     2020-04-12 [1] CRAN (R 4.0.2)
## jsonlite      1.7.0     2020-06-25 [1] CRAN (R 4.0.2)
## kableExtra  * 1.1.0     2019-03-16 [1] CRAN (R 4.0.0)
## knitr       * 1.29      2020-06-23 [1] CRAN (R 4.0.2)
## later         1.1.0.1   2020-06-05 [1] CRAN (R 4.0.0)
## lattice       0.20-41   2020-04-02 [1] CRAN (R 4.0.2)
## libcoin       1.0-5     2019-08-27 [1] CRAN (R 4.0.2)
## lifecycle     0.2.0     2020-03-06 [1] CRAN (R 4.0.0)
## lubridate     1.7.9     2020-06-08 [1] CRAN (R 4.0.2)
## magick        2.4.0     2020-06-23 [1] CRAN (R 4.0.0)
## magrittr    * 1.5       2014-11-22 [1] CRAN (R 4.0.0)
## Matrix        1.2-18    2019-11-27 [1] CRAN (R 4.0.2)
## memoise       1.1.0     2017-04-21 [1] CRAN (R 4.0.0)
## mime          0.9       2020-02-04 [1] CRAN (R 4.0.0)
## miniUI        0.1.1.1   2018-05-18 [1] CRAN (R 4.0.0)
## modelr        0.1.8     2020-05-19 [1] CRAN (R 4.0.0)
## munsell       0.5.0     2018-06-12 [1] CRAN (R 4.0.0)
## mvtnorm       1.1-1     2020-06-09 [1] CRAN (R 4.0.0)
## neuralnet   * 1.44.2    2019-02-07 [1] CRAN (R 4.0.2)
## pander      * 0.6.3     2018-11-06 [1] CRAN (R 4.0.0)
## partykit      1.2-9     2020-07-10 [1] CRAN (R 4.0.2)
## pillar        1.4.6     2020-07-10 [1] CRAN (R 4.0.2)
## pkgbuild      1.1.0     2020-07-13 [1] CRAN (R 4.0.2)
```

```
## pkgconfig     2.0.3    2019-09-22 [1] CRAN (R 4.0.0)
## pkgload       1.1.0    2020-05-29 [1] CRAN (R 4.0.0)
## plyr          1.8.6    2020-03-03 [1] CRAN (R 4.0.2)
## png           0.1-7    2013-12-03 [1] CRAN (R 4.0.0)
## prettyunits   1.1.1    2020-01-24 [1] CRAN (R 4.0.0)
## processx      3.4.3    2020-07-05 [1] CRAN (R 4.0.2)
## progress      1.2.2    2019-05-16 [1] CRAN (R 4.0.0)
## promises      1.1.1    2020-06-09 [1] CRAN (R 4.0.2)
## ps            1.3.4    2020-08-11 [1] CRAN (R 4.0.2)
## purrr       * 0.3.4    2020-04-17 [1] CRAN (R 4.0.0)
## R6            2.4.1    2019-11-12 [1] CRAN (R 4.0.0)
## Rcpp          1.0.5    2020-07-06 [1] CRAN (R 4.0.0)
## readr       * 1.3.1    2018-12-21 [1] CRAN (R 4.0.0)
## readxl        1.3.1    2019-03-13 [1] CRAN (R 4.0.0)
## remotes       2.2.0    2020-07-21 [1] CRAN (R 4.0.2)
## reprex        0.3.0    2019-05-16 [1] CRAN (R 4.0.0)
## reshape2      1.4.4    2020-04-09 [1] CRAN (R 4.0.0)
## rlang         0.4.7    2020-07-09 [1] CRAN (R 4.0.0)
## rmarkdown     2.3      2020-06-18 [1] CRAN (R 4.0.2)
## rpart         4.1-15   2019-04-12 [1] CRAN (R 4.0.2)
## rprojroot     1.3-2    2018-01-03 [1] CRAN (R 4.0.0)
## rstudioapi    0.11     2020-02-07 [1] CRAN (R 4.0.0)
## rvest         0.3.6    2020-07-25 [1] CRAN (R 4.0.2)
## scales        1.1.1    2020-05-11 [1] CRAN (R 4.0.0)
## sessioninfo   1.1.1    2018-11-05 [1] CRAN (R 4.0.0)
## shiny         1.5.0    2020-06-23 [1] CRAN (R 4.0.2)
## snakecase     0.11.0   2019-05-25 [1] CRAN (R 4.0.2)
## stringi       1.4.6    2020-02-17 [1] CRAN (R 4.0.0)
## stringr     * 1.4.0    2019-02-10 [1] CRAN (R 4.0.0)
## survival      3.2-3    2020-06-13 [1] CRAN (R 4.0.2)
## testthat      2.3.2    2020-03-02 [1] CRAN (R 4.0.0)
## tibble      * 3.0.3    2020-07-10 [1] CRAN (R 4.0.2)
## tidyr       * 1.1.1    2020-07-31 [1] CRAN (R 4.0.2)
## tidyselect    1.1.0    2020-05-11 [1] CRAN (R 4.0.0)
## tidyverse   * 1.3.0    2019-11-21 [1] CRAN (R 4.0.0)
## tufte       * 0.6      2020-05-08 [1] CRAN (R 4.0.2)
## tweenr        1.0.1    2018-12-14 [1] CRAN (R 4.0.2)
## usethis       1.6.1    2020-04-29 [1] CRAN (R 4.0.2)
## vctrs         0.3.2    2020-07-15 [1] CRAN (R 4.0.0)
## viridisLite   0.3.0    2018-02-01 [1] CRAN (R 4.0.0)
## webshot       0.5.2    2019-11-22 [1] CRAN (R 4.0.0)
## withr         2.2.0    2020-04-20 [1] CRAN (R 4.0.0)
## xfun          0.16     2020-07-24 [1] CRAN (R 4.0.2)
## xml2          1.3.2    2020-04-23 [1] CRAN (R 4.0.0)
```

```
##  xtable        1.8-4      2019-04-21 [1] CRAN (R 4.0.2)
##  yaml          2.2.1      2020-02-01 [1] CRAN (R 4.0.0)
##
## [1] C:/Program Files/R/R-4.0.2/library
```