ggg

Daryn Ramsden

2019-08-16

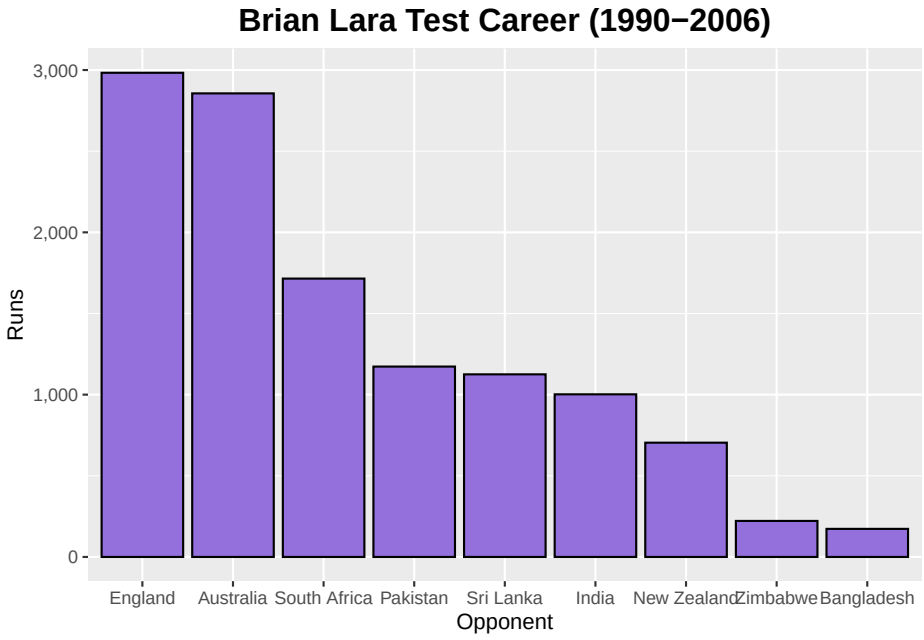
Contents

This site is intended to provide a gallery of plots of different types created using the *ggplot2* R package.

In order that the visitors can reproduce the plots here, I will maintain an R package, *gcubed* containing all the data sets used that are not available in base R or via CRAN.

Examples

Brian Lara's runs in test cricket by Opponent



Data

This plot uses the *lara_tests* data frame of the *gcubed* package.

This data was obtained from ESPN Cricinfo.

```
library(gcubed)
head(lara_tests)
```

```
## # A tibble: 6 x 8
```

```
##    Runs Inning Notout DNB   Opp          Ground     `Start Date` MatchNum
##   <int> <fct>  <lgl>  <lgl> <chr>        <chr>      <chr>        <chr>
## 1    44 1      FALSE  FALSE Pakistan     Lahore     6-Dec-90     1158
## 2     5 2      FALSE  FALSE Pakistan     Lahore     6-Dec-90     1158
## 3    17 1      FALSE  FALSE South Africa Bridgetown 18-Apr-92    1188
## 4    64 2      FALSE  FALSE South Africa Bridgetown 18-Apr-92    1188
## 5    58 1      FALSE  FALSE Australia    Brisbane   27-Nov-92    1202
## 6     0 2      FALSE  FALSE Australia    Brisbane   27-Nov-92    1202
```

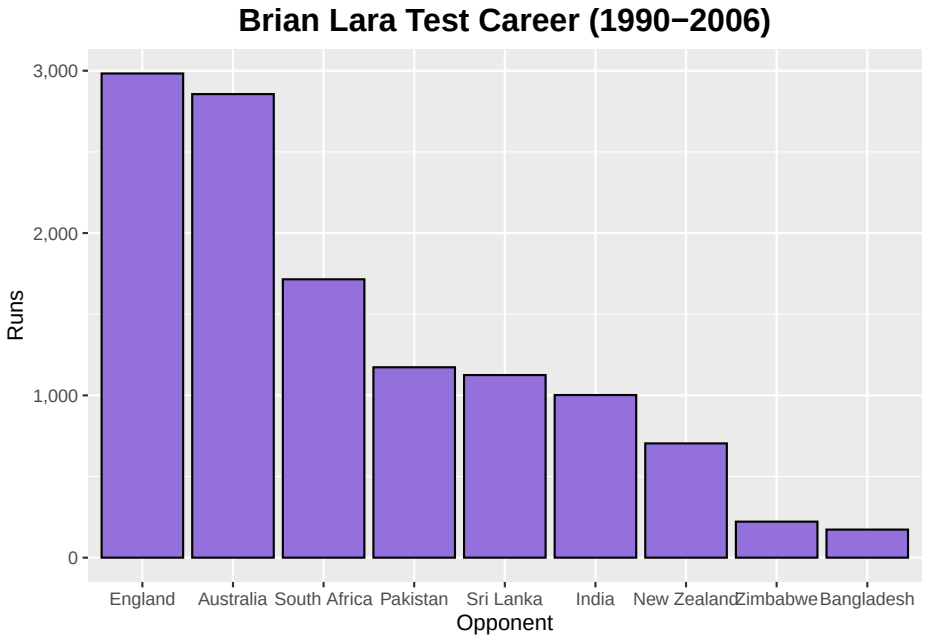
Code for plot

First, create a data frame aggregating runs by opponent. (Note that this can also be done using the **aggregate** function of base R):

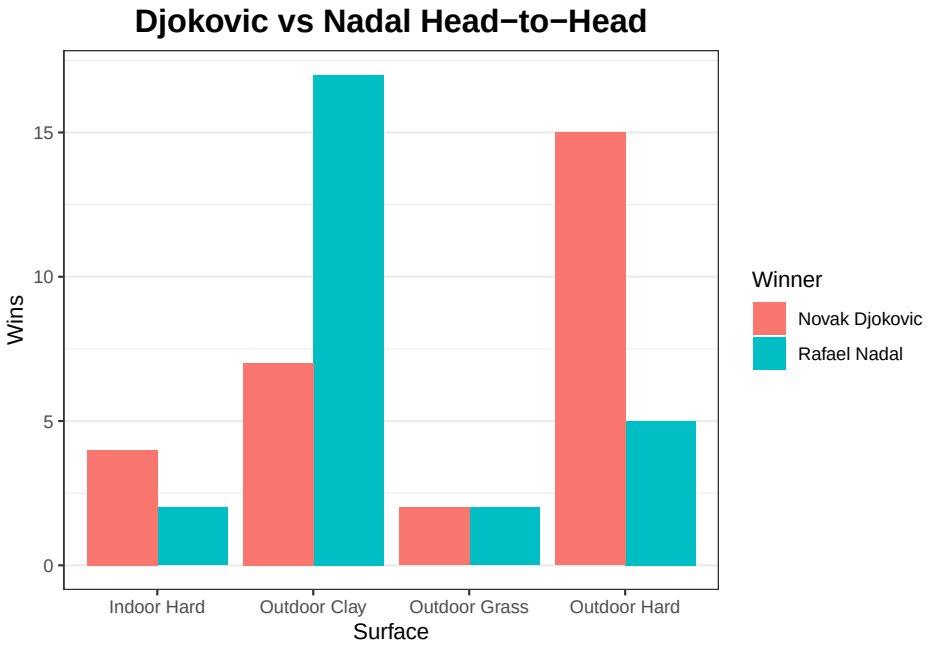
```
library(dplyr)

df <- group_by(lara_tests, Opp) %>%
  summarise(Runs = sum(Runs, na.rm = TRUE)) %>%
  arrange(desc(Runs))

head(df)
```

```
## # A tibble: 6 x 2
##   Opp           Runs
##   <chr>        <int>
## 1 England       2983
## 2 Australia     2856
## 3 South Africa  1715
## 4 Pakistan      1173
## 5 Sri Lanka     1125
## 6 India         1002
```

```
library(ggplot2)
library(scales) #to get commas in formatting numerical values on the y-axis

df$Opp <- factor(df$Opp, levels =df$Opp) # gets the order of the bars right

bcl_runs_plt <- ggplot(df, aes(x = Opp, y = Runs)) +
  geom_bar(stat = "identity", fill = "mediumpurple", colour = "black") +
  xlab("Opponent") +
  ggtitle("Brian Lara Test Career (1990-2006)")+
  scale_y_continuous(label = comma)+
  theme( plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

bcl_runs_plt
```


Brian Lara Test Career (1990−2006)
3,000
2,000
1,000
0
Runs
England
Australia
South Africa
Pakistan
Sri Lanka
India
New Zealand
Zimbabwe
Bangladesh
Opponent

Djokovic vs Nadal Head-to-head



Data

This plot uses the *rafa_novak* data frame from the *gcubed* library. This data frame has one row for every match played between Novak Djokovic and Rafael Nadal over the course of their professional careers. In particular, the column *Winner* has the name of the winner of the match.

This data was sourced from the ATP's Head2Head Comparison tool and was current as of 2019-08-15.

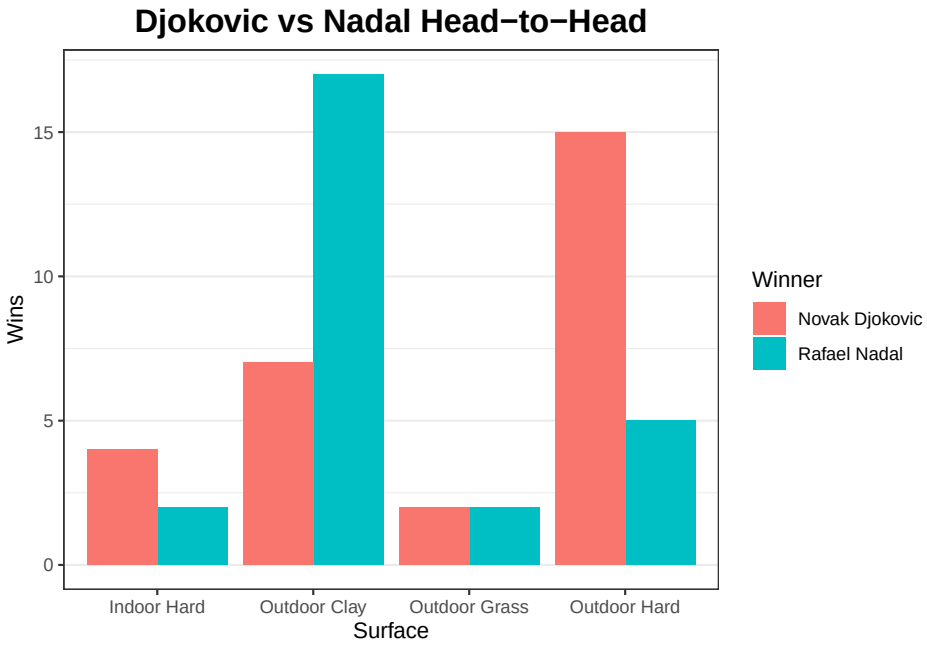
```
library(gcubed)
head(rafa_novak)
```

```
## # A tibble: 6 x 8
##    Year Event        Location   Surface   RND   Winner   Result    Loser
##   <dbl> <chr>        <chr>      <chr>     <chr> <chr>    <chr>     <chr>
## 1  2019 ATP Masters~ Italy      Outdoor ~ F     Rafael ~ 60 46 61  Novak D~
## 2  2019 Australian ~ Australia  Outdoor ~ F     Novak D~ 63 62 63  Rafael ~
## 3  2018 Wimbledon    Great Bri~ Outdoor ~ SF    Novak D~ 64 36 76~ Rafael ~
## 4  2018 ATP Masters~ Italy      Outdoor ~ SF    Rafael ~ 764 63    Novak D~
## 5  2017 ATP Masters~ Spain      Outdoor ~ SF    Rafael ~ 62 64     Novak D~
## 6  2016 ATP Masters~ Italy      Outdoor ~ QF    Novak D~ 75 764    Rafael ~
```

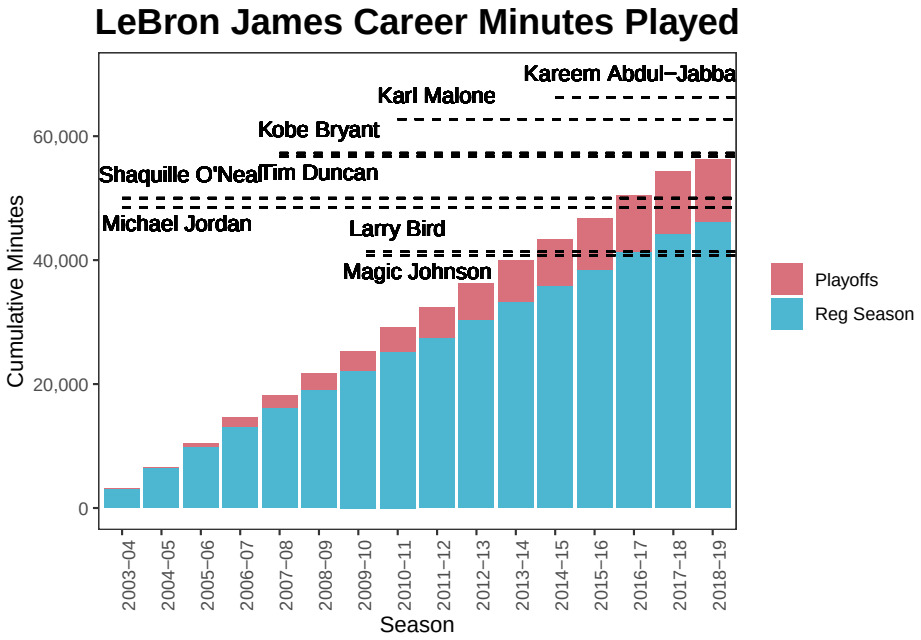
This data frame is already suitable for making the plot.

Code

```
library(ggplot2)

rafa_novak_plt <- ggplot(rafa_novak, aes(x = Surface, fill = Winner)) +
  geom_bar(position = "dodge") +
  ylab("Wins") +
  ggtitle("Djokovic vs Nadal Head-to-Head") +
  theme_bw() +
  theme(panel.grid.major.x = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

rafa_novak_plt
```


Djokovic vs Nadal Head−to−Head
Wins
15
10
5
0
Indoor Hard
Outdoor Clay
Outdoor Grass
Outdoor Hard
Surface
Winner
Novak Djokovic
Rafael Nadal

LeBron James Career Minutes



Data

This plot uses the *lebron_mp* data frame from the *gcubed* library. This data frame has columns *MPR* and *MPP* for minutes played by LeBron James during the regular season and playoffs respectively of the corresponding season.

The original data source is Basketball Reference. [1]

[1] I did do some data manipulation to arrive at only these columns. More complete statistics for LeBron James' career are in the *lebron* and *lebron_playoffs* data sets also contained in the *gcubed* package. The career totals for the other players in the plot were also sourced from their respective player pages on Basketball Reference.

```
library(gcubed)
head(lebron_mp)
```

```
## # A tibble: 6 x 3
##   Season    MPR   MPP
##   <chr>   <dbl> <dbl>
## 1 2003-04  3122     0
## 2 2004-05  3388     0
## 3 2005-06  3361   604
## 4 2006-07  3190   893
## 5 2007-08  3027   552
## 6 2008-09  3054   580
```

Code for plot

First, add columns for cumulative career minutes for both playoffs and the regular season.

```
library(dplyr)

df <- mutate(lebron_mp, Playoffs = cumsum(MPP),
                        `Reg Season` = cumsum(MPR))
head(df)
```

```
## # A tibble: 6 x 5
##   Season    MPR   MPP Playoffs `Reg Season`
##   <chr>   <dbl> <dbl>    <dbl>        <dbl>
## 1 2003-04  3122     0        0         3122
## 2 2004-05  3388     0        0         6510
## 3 2005-06  3361   604      604         9871
## 4 2006-07  3190   893     1497        13061
## 5 2007-08  3027   552     2049        16088
## 6 2008-09  3054   580     2629        19142
```

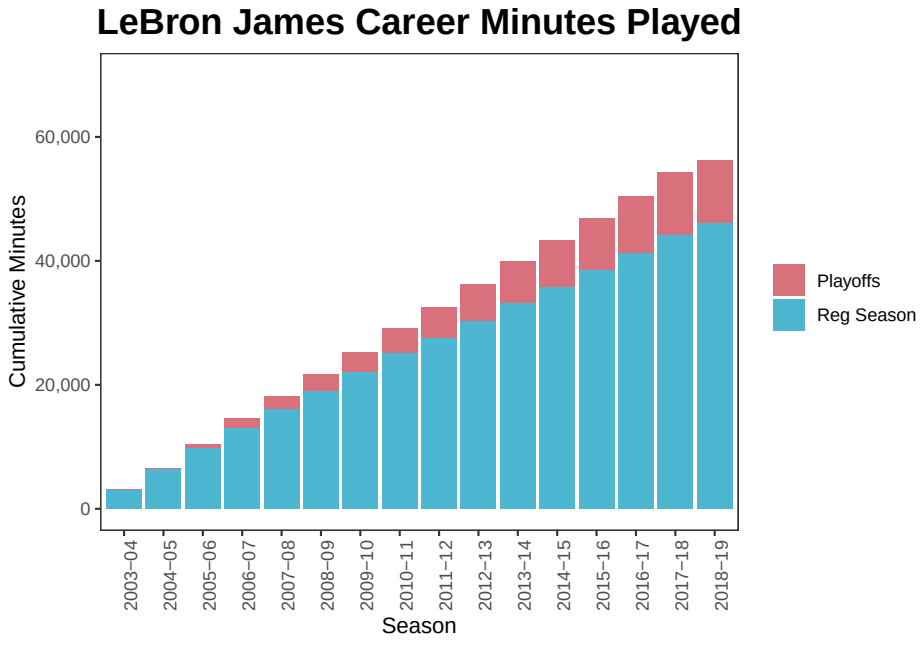
```
library(tidyr)
df <- gather(df, key = RegPlayoffs, value = MP, Playoffs:`Reg Season`)

head(df)
```

```
## # A tibble: 6 x 5
##   Season    MPR   MPP RegPlayoffs    MP
##   <chr>   <dbl> <dbl> <chr>       <dbl>
## 1 2003-04  3122     0 Playoffs        0
```

```
## 2 2004-05  3388     0 Playoffs        0
## 3 2005-06  3361   604 Playoffs      604
## 4 2006-07  3190   893 Playoffs     1497
## 5 2007-08  3027   552 Playoffs     2049
## 6 2008-09  3054   580 Playoffs     2629
```

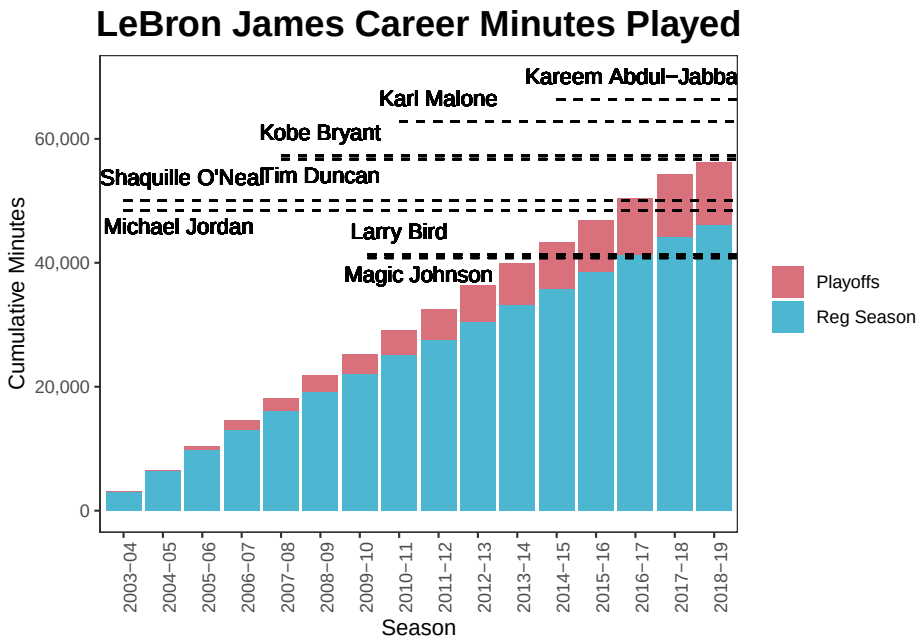
First version of the plot:

```
library(ggplot2)
library(scales)

lbj_plt <- ggplot(df, aes(x = Season, y = MP, fill = RegPlayoffs)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("#D9717D", "#4DB6D0")) +
  scale_y_continuous(label=comma, limits = c(0,70000)) +
  theme_bw() + #change the background colour to white
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        axis.text.x = element_text(angle = 90),
        plot.title = element_text(size = 18, face = "bold", hjust = 0.5),
        legend.title = element_blank()
        )+
  ylab("Cumulative Minutes") +
  ggtitle("LeBron James Career Minutes Played")

lbj_plt
```



Adding annotations for other significant NBA players:

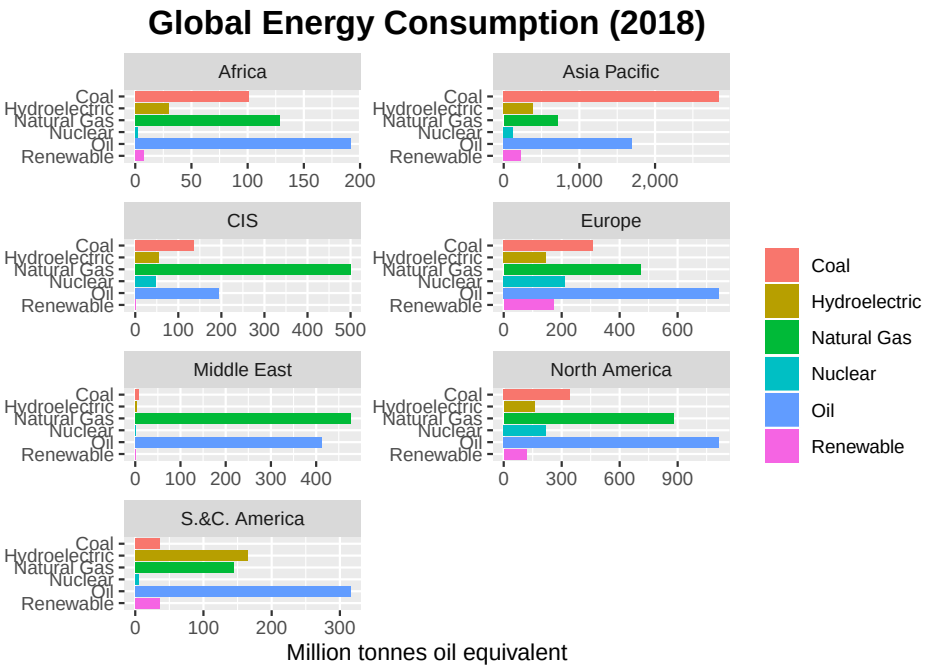
```
lbj_plt <- lbj_plt +
  geom_segment(x = 12, xend = 17, y=66297, yend = 66297, linetype="dashed") +
  geom_text(aes(14,66297,label = "Kareem Abdul-Jabbar", vjust = -1)) +
  geom_segment(x = 8, xend = 17, y = 62759, yend = 62759, linetype="dashed") +
  geom_text(aes(9,62759,label = "Karl Malone", vjust = -1)) +
  geom_segment(x = 5, xend = 17, y = 57278, yend = 57278, linetype="dashed") +
  geom_text(aes(6,57278,label = "Kobe Bryant", vjust = -1)) +
  geom_segment(x = 5, xend = 17, y = 56738, yend = 56738, linetype="dashed") +
  geom_text(aes(6,56738,label = "Tim Duncan", vjust = 1.5)) +
  geom_segment(x = 1, xend = 17, y = 50016, yend = 50016, linetype="dashed") +
  geom_text(aes(2.5,50016,label = "Shaquille O'Neal", vjust = -1)) +
  geom_segment(x = 1, xend = 17, y = 48485, yend = 48485, linetype="dashed") +
  geom_text(aes(2.4,48485,label = "Michael Jordan", vjust = 1.5)) +
  geom_segment(x = 7.2, xend = 17, y = 41329, yend = 41329, linetype="dashed") +
  geom_text(aes(8,41329,label = "Larry Bird", vjust = -1)) +
  geom_segment(x = 7.2, xend = 17, y = 40783, yend = 40783, linetype="dashed") +
  geom_text(aes(8.5,40783,label = "Magic Johnson", vjust = 1.5))

lbj_plt
```



0.0.0.0.1 Code for complete plot

```
lbj_plt <- ggplot(df, aes(x = Season, y = MP, fill = RegPlayoffs)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("#D9717D", "#4DB6D0")) +
  scale_y_continuous(label=comma, limits = c(0,70000)) +
  theme_bw() + #change the background colour to white
  theme(panel.grid.major = element_blank(), panel.grid.minor = element_blank(),
        axis.text.x = element_text(angle = 90),
        plot.title = element_text(size = 18, face = "bold", hjust = 0.5),
        legend.title = element_blank()
        )+
  ylab("Cumulative Minutes") +
  ggtitle("LeBron James Career Minutes Played") +
  geom_segment(x = 12, xend = 17, y=66297, yend = 66297, linetype="dashed") +
  geom_text(aes(14,66297,label = "Kareem Abdul-Jabbar", vjust = -1)) +
  geom_segment(x = 8, xend = 17, y = 62759, yend = 62759, linetype="dashed") +
  geom_text(aes(9,62759,label = "Karl Malone", vjust = -1)) +
  geom_segment(x = 5, xend = 17, y = 57278, yend = 57278, linetype="dashed") +
  geom_text(aes(6,57278,label = "Kobe Bryant", vjust = -1)) +
  geom_segment(x = 5, xend = 17, y = 56738, yend = 56738, linetype="dashed") +
  geom_text(aes(6,56738,label = "Tim Duncan", vjust = 1.5)) +
  geom_segment(x = 1, xend = 17, y = 50016, yend = 50016, linetype="dashed") +
  geom_text(aes(2.5,50016,label = "Shaquille O'Neal", vjust = -1)) +
  geom_segment(x = 1, xend = 17, y = 48485, yend = 48485, linetype="dashed") +
  geom_text(aes(2.4,48485,label = "Michael Jordan", vjust = 1.5)) +
  geom_segment(x = 7.2, xend = 17, y = 41329, yend = 41329, linetype="dashed") +
  geom_text(aes(8,41329,label = "Larry Bird", vjust = -1)) +
  geom_segment(x = 7.2, xend = 17, y = 40783, yend = 40783, linetype="dashed") +
  geom_text(aes(8.5,40783,label = "Magic Johnson", vjust = 1.5))
```

Global Energy Consumption 2018



Data

This plot uses the *energy18* data frame of the *gcubed* package.

The data set contains energy consumption by country(or groups of countries in some cases) and also has a column identifying that country's region.

The original source of this data is the BP Statistical Report of World Energy 2019.

The first few rows of *energy18* look like this:

```
library(gcubed)
head(energy18)
```

```
## # A tibble: 6 x 8
##   Countries   Oil `Natural Gas`  Coal Nuclear Hydroelectric Renewable
##   <chr>     <dbl>         <dbl> <dbl>   <dbl>         <dbl>     <dbl>
## 1 Canada    110            99.5  14.4    22.6          87.6      10.3
## 2 Mexico     82.8          77    11.9     3.1           7.3       4.8
## 3 US        920.          703.  317     192.           65.3     104.
## 4 Argentina  30.1          41.9   1.2     1.6           9.4       0.9
## 5 Brazil    136.           30.9  15.9     3.5          87.7      23.6
## 6 Chile      18.1           5.5   7.7     0             5.2       3.5
## # ... with 1 more variable: Region <chr>
```

Data wrangling for plot

First get totals for each energy source (natural gas, oil, coal, nuclear, hydroelectric, renewable) for each region:

```
library(dplyr)

df <- group_by(energy18, Region) %>%
  summarise(Oil = sum(Oil),
            `Natural Gas` = sum(`Natural Gas`),
            Coal = sum(Coal),
            Nuclear = sum(Nuclear),
            Hydroelectric = sum(Hydroelectric),
            Renewable = sum(Renewable))
head(df)
```

```
## # A tibble: 6 x 7
##   Region          Oil `Natural Gas`   Coal Nuclear Hydroelectric Renewable
##   <chr>         <dbl>         <dbl>  <dbl>   <dbl>         <dbl>     <dbl>
## 1 Africa         191.          129.  101.      2.5          30.1       7.2
## 2 Asia Pacific  1695.          710. 2841.    125.          389.      225.
## 3 CIS            194.          499.  135.     46.8          55.4       0.5
## 4 Europe         742.          472.  307.    212.          145.      172.
## 5 Middle East    412           475.    8.1     1.6           3.4       1.6
## 6 North America 1112.          879.  343.    218.          160.      119.
```

Now use the **gather** command of the **tidyr** package to get all the energy values into the same column while creating an accompanying column to indicate the energy source.

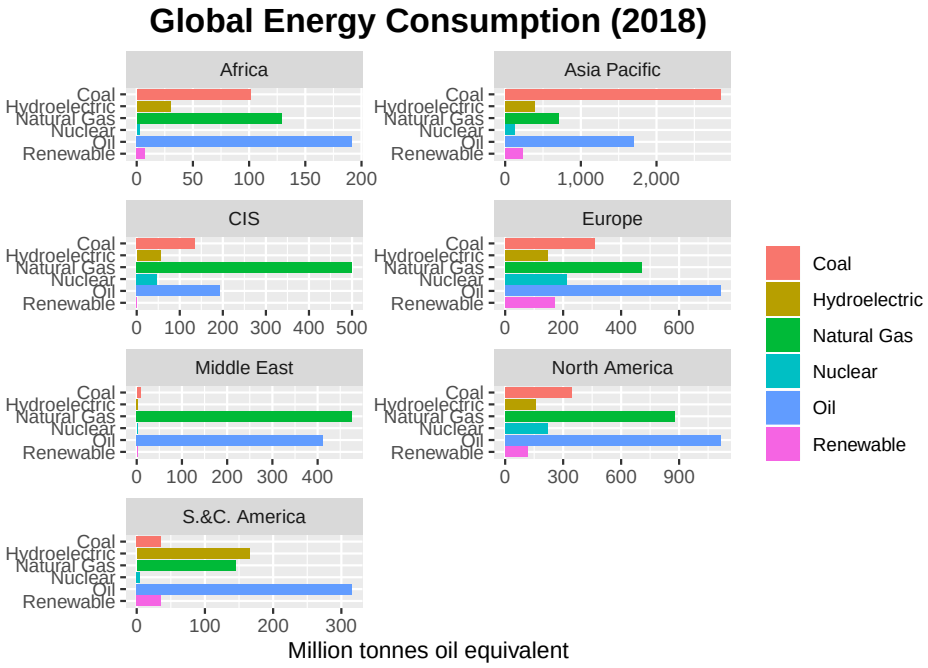
```
library(tidyr)
df <- gather(df, key = Type, value = Energy, -1)

head(df)
```

```
## # A tibble: 6 x 3
##   Region        Type  Energy
##   <chr>         <chr>  <dbl>
## 1 Africa        Oil     191.
## 2 Asia Pacific  Oil    1695.
## 3 CIS           Oil     194.
## 4 Europe        Oil     742.
## 5 Middle East   Oil     412
## 6 North America Oil    1112.
```

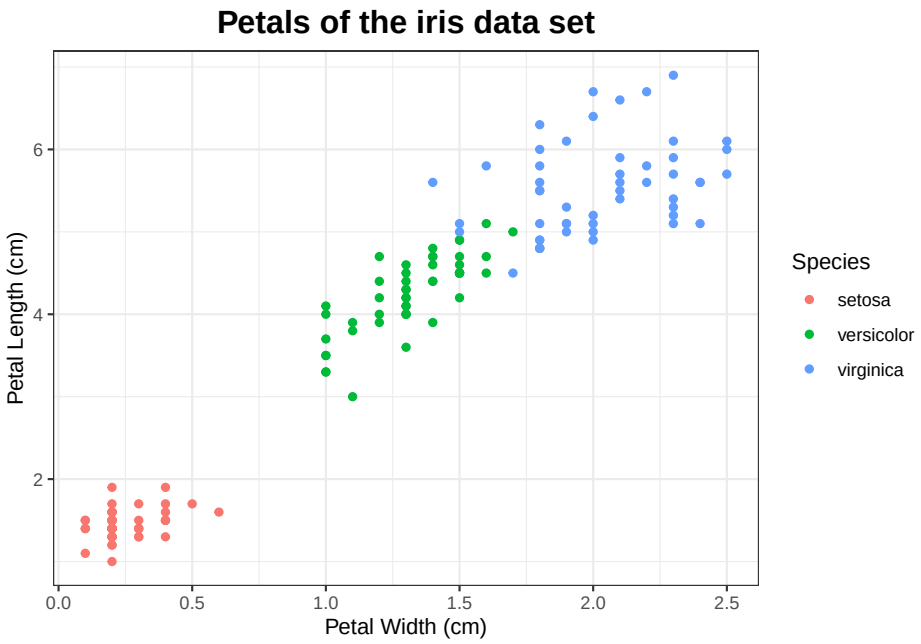
The *df* data frame is now suitable for making the plot.

Code

```
library(ggplot2)
library(scales) #for formatting the axes labels to have commas e.g. 1,000

df$Type <- factor(df$Type)
energy_plt <- ggplot(df, aes(Type, Energy, fill = Type)) + geom_bar(stat = "identity") +
  facet_wrap(~Region, ncol = 2, scale = "free") + coord_flip() +
  scale_x_discrete(limits = rev(levels(df$Type))) +
  scale_y_continuous(label = comma) +
  xlab("") +
  ylab("Million tonnes oil equivalent") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        legend.title = element_blank()) +
  ggtitle("Global Energy Consumption (2018)")

energy_plt
```


Global Energy Consumption (2018)
Africa
Asia Pacific
CIS
Europe
Middle East
North America
S.&C. America
Coal
Hydroelectric
Natural Gas
Nuclear
Oil
Renewable
Million tonnes oil equivalent

Iris Data Set



Data

This plot uses the *iris* data set that comes with R. This data frame contains the widths and lengths of the petals and sepals of 150 iris flowers. The flowers are of three different species: setosa, versicolor and virginica. There are 50 specimens of each species.

The 1st, 51st and 101st rows look like:

```
iris[c(1,51,101), ]
```

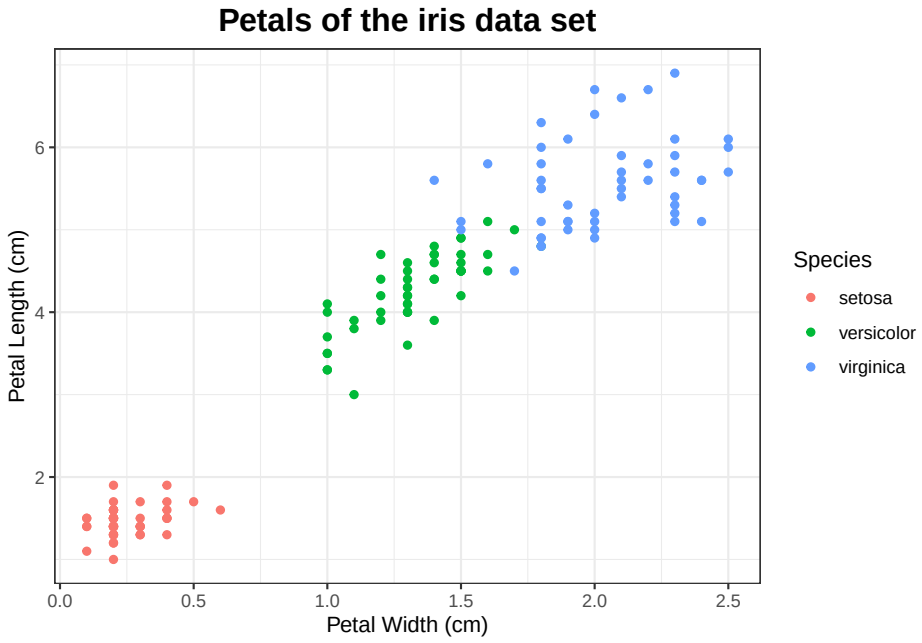
```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
```

```
## 1          5.1         3.5          1.4         0.2     setosa
## 51         7.0         3.2          4.7         1.4 versicolor
## 101        6.3         3.3          6.0         2.5  virginica
```

Code for plot

To get this scatter plot, we use the *geom_point* geometry. To get the species coloured differently, the *colour* aesthetic is used and mapped to the *Species* variable.

```
library(ggplot2)

iris_petal_plot <- ggplot(data = iris, aes(x = Petal.Width, y = Petal.Length, colour =
  geom_point() + theme_bw() +
  xlab("Petal Width (cm)") +
  ylab("Petal Length (cm)") +
  ggtitle("Petals of the iris data set") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

iris_petal_plot
```



10 Yr History of US Unemployment: Line Chart

Data

This plot uses the *us_unemp* data frame of the *gcubed* package. This data frame contains unemployment rates for the United States published by the Bureau of Labor and Statistics. Rates are published monthly.

```
library(gcubed)
head(us_unemp)
```

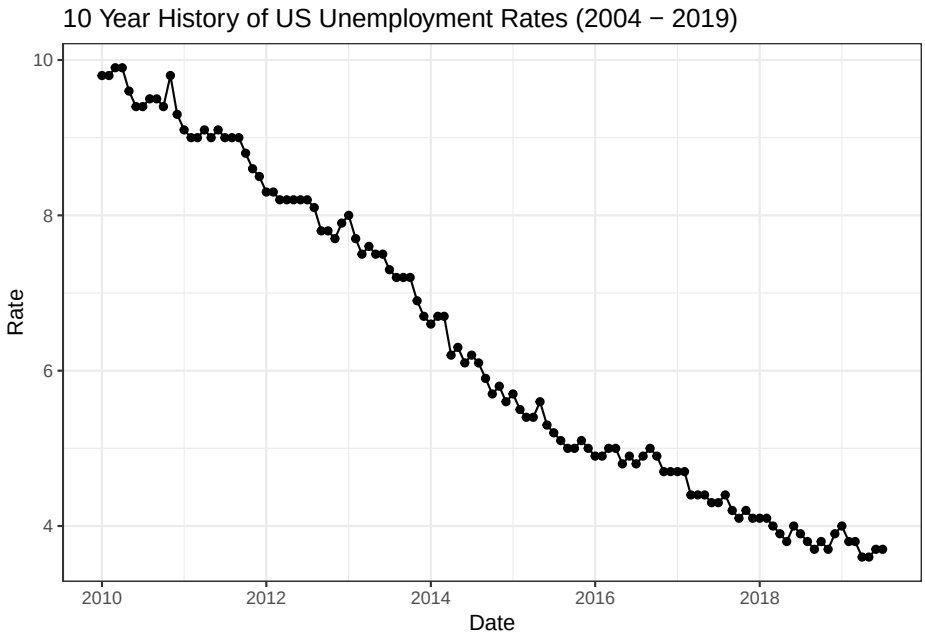
```
## # A tibble: 6 x 2
##   Date        Rate
##   <date>     <dbl>
## 1 2010-01-01   9.8
## 2 2010-02-01   9.8
## 3 2010-03-01   9.9
## 4 2010-04-01   9.9
## 5 2010-05-01   9.6
## 6 2010-06-01   9.4
```

Code for plot

This plot uses *geom_line* to create a line chart.

```
library(ggplot2)

unemp_plt <- ggplot(us_unemp, aes(x = Date, y = Rate)) +
  geom_line() +
  ggtitle("10 Year History of US Unemployment Rates (2004 - 2019)")

unemp_plt
```

10 Yr History of US Unemployment: Line and Point plot



Data

This plot uses the *us_unemp* data frame of the *gcubed* package. This data frame contains unemployment rates for the United States published by the Bureau of Labor and Statistics. Rates are published monthly.

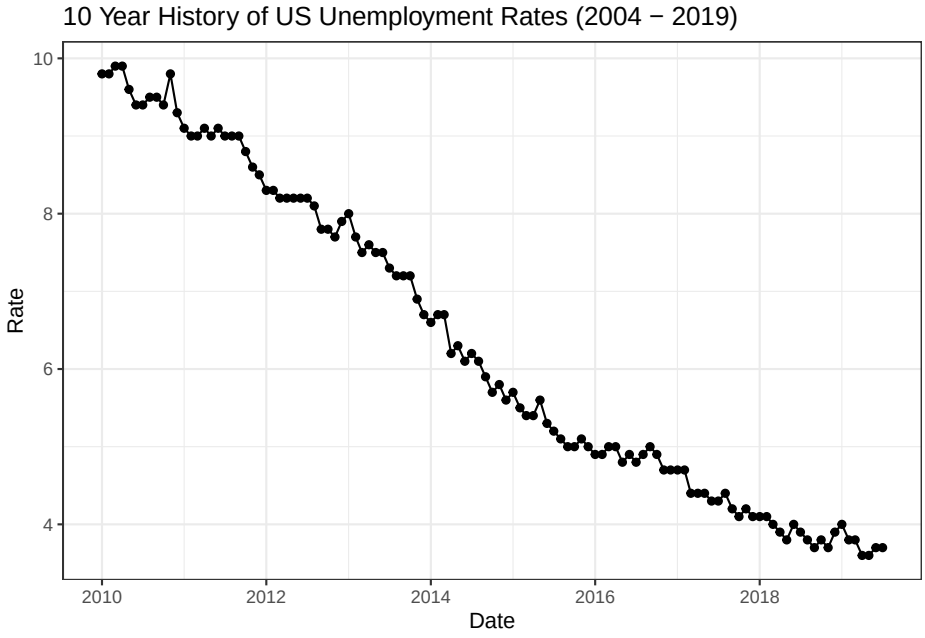
```
library(gcubed)
head(us_unemp)
```

```
## # A tibble: 6 x 2
##   Date        Rate
##   <date>     <dbl>
## 1 2010-01-01   9.8
## 2 2010-02-01   9.8
## 3 2010-03-01   9.9
## 4 2010-04-01   9.9
## 5 2010-05-01   9.6
## 6 2010-06-01   9.4
```

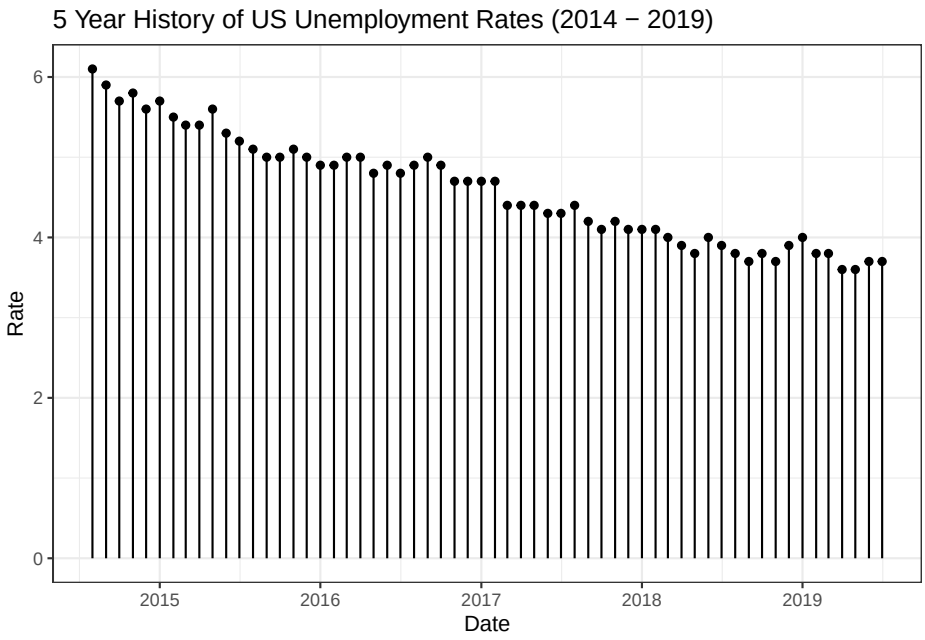
Code for plot

This plot uses *geom_line* (to create the line chart) and *geom_point* to highlight the data points simultaneously.

```
library(ggplot2)

unemp_ptline_plt <- ggplot(us_unemp, aes(x = Date, y = Rate)) +
  geom_line() + geom_point() +
  ggtitle("10 Year History of US Unemployment Rates (2004 - 2019)")  +
  theme_bw()

unemp_ptline_plt
```


10 Year History of US Unemployment Rates (2004 − 2019)
10
8
6
4
Rate
2010
2012
2014
2016
2018
Date

5 Yr History of US Unemployment: Lollipop chart



Data

This plot uses the *us_unemp* data frame of the *gcubed* package. This data frame contains unemployment rates for the United States published by the Bureau of Labor and Statistics. Rates are published monthly.

```
library(gcubed)
head(us_unemp)
```

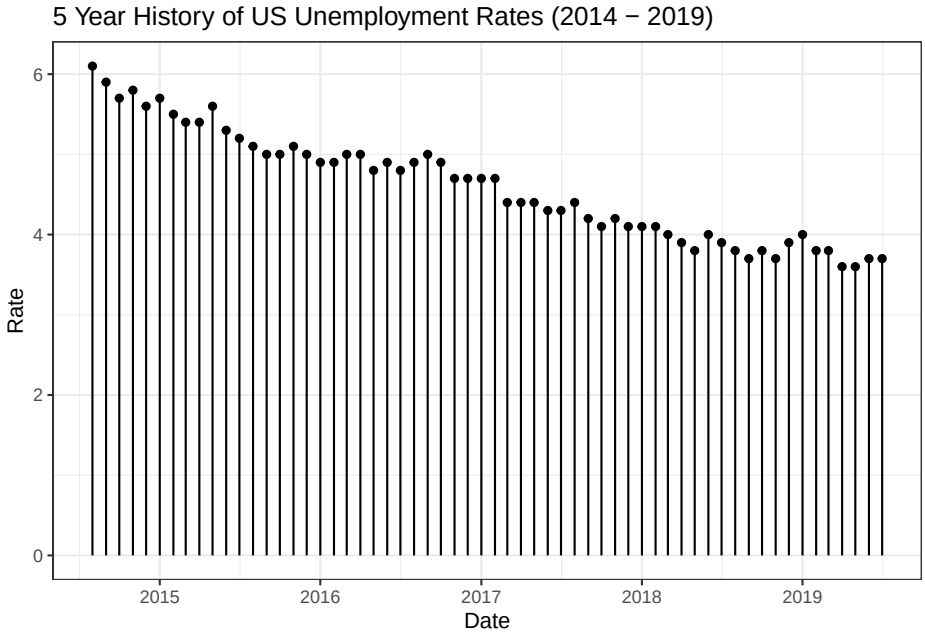
```
## # A tibble: 6 x 2
##   Date        Rate
##   <date>     <dbl>
## 1 2010-01-01   9.8
## 2 2010-02-01   9.8
## 3 2010-03-01   9.9
## 4 2010-04-01   9.9
## 5 2010-05-01   9.6
## 6 2010-06-01   9.4
```

To get only 5 years of rates, we filter:

```
df <- us_unemp[us_unemp$Date > as.Date("2014-07-01"), ]
```

Code for plot

```
library(ggplot2)

unemp_plt <- ggplot(df, aes(x = Date, y = Rate)) + geom_point() +
  geom_segment(aes(x = Date, xend = Date, y = 0, yend = Rate)) +
  ggtitle("5 Year History of US Unemployment Rates (2014 - 2019)")  +
  theme_bw()


unemp_plt
```


5 Year History of US Unemployment Rates (2014 − 2019)
Rate
6
4
2
0
2015
2016
2017
2018
2019
Date

10 Yr History of US Unemployment: Step Plot

Data

This plot uses the *us_unemp* data frame of the *gcubed* package. This data frame contains unemployment rates for the United States published by the Bureau of Labor and Statistics. Rates are published monthly.

```
library(gcubed)
head(us_unemp)
```

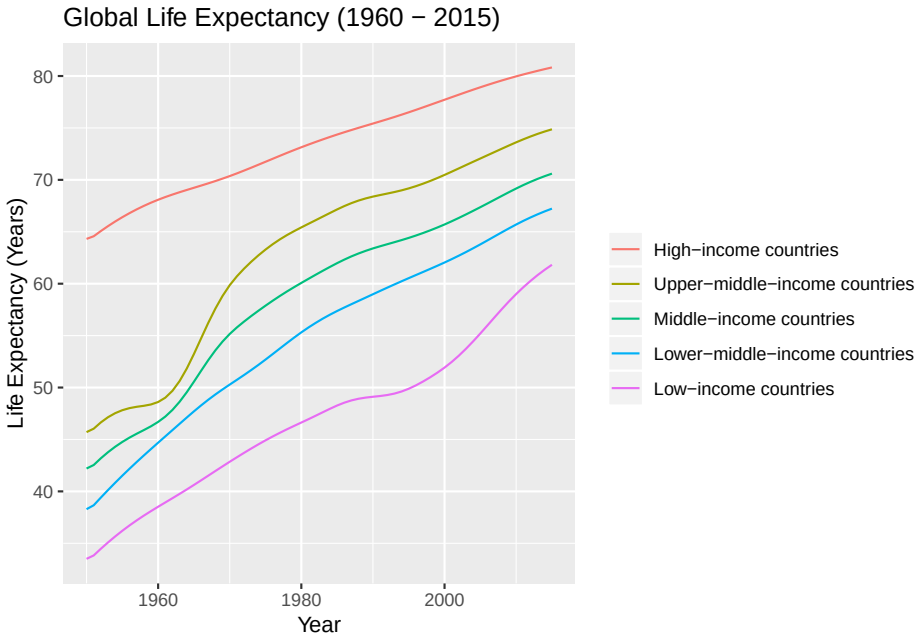
```
## # A tibble: 6 x 2
##   Date        Rate
##   <date>     <dbl>
## 1 2010-01-01   9.8
## 2 2010-02-01   9.8
## 3 2010-03-01   9.9
## 4 2010-04-01   9.9
## 5 2010-05-01   9.6
## 6 2010-06-01   9.4
```

Code for plot

This plot uses the *geom_step* geometry to get a step function appearance as opposed to the look of using *geom_line*.

```
library(ggplot2)

unemp_step_plt <- ggplot(us_unemp, aes(x = Date, y = Rate)) + geom_step() +
  ggtitle("10 Year History of US Unemployment Rates (2004 - 2019)")


unemp_step_plt
```

Global life expectancy: Line chart



Data

This plot uses the *life_ex* data frame of the *gcubed* package. This data frame contains life expectancy data for numerous countries and groups of countries in the *Entity* column.

```
library(gcubed)
head(life_ex)
```

```
## # A tibble: 6 x 4
```

```
##   Entity      Code   Year    LE
##   <chr>       <chr> <dbl> <dbl>
## 1 Afghanistan AFG    1950  27.5
## 2 Afghanistan AFG    1951  27.8
## 3 Afghanistan AFG    1952  28.4
## 4 Afghanistan AFG    1953  28.9
## 5 Afghanistan AFG    1954  29.4
## 6 Afghanistan AFG    1955  29.9
```

Code for plot

First, we will restrict the data set to only those rows that contain the life expectancy values for the country groups we are interested in. (Note that this filtering of rows could also have been done using base R.)

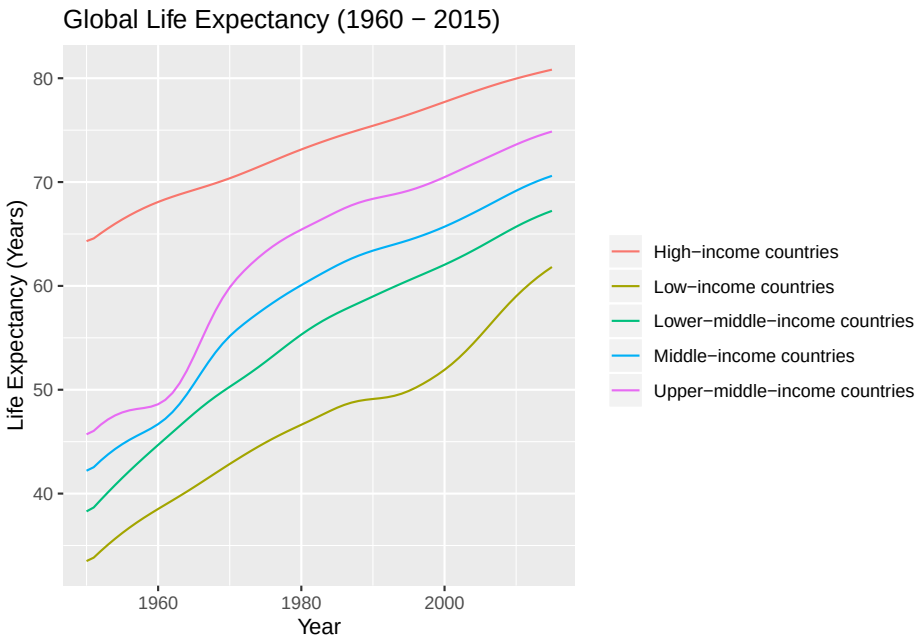
```
groups <- c("Upper-middle-income countries", "Middle-income countries", "Low-income cou
  "Lower-middle-income countries","High-income countries")

library(dplyr)
df <- filter(life_ex, Entity %in% groups)
head(df)
```

```
## # A tibble: 6 x 4
##   Entity                Code   Year    LE
##   <chr>                 <chr> <dbl> <dbl>
## 1 High-income countries <NA>   1950  64.3
## 2 High-income countries <NA>   1951  64.6
## 3 High-income countries <NA>   1952  65.1
## 4 High-income countries <NA>   1953  65.5
## 5 High-income countries <NA>   1954  66.0
## 6 High-income countries <NA>   1955  66.4
```

The data is already in the correct shape to be used by *geom_line*: all the life expectancy values are in the single column, *LE*. To get different lines for each income group, the *group* aesthetic is used in the creation of the *ggplot* object. To give each line a different colour, the *colour* aesthetic is used.

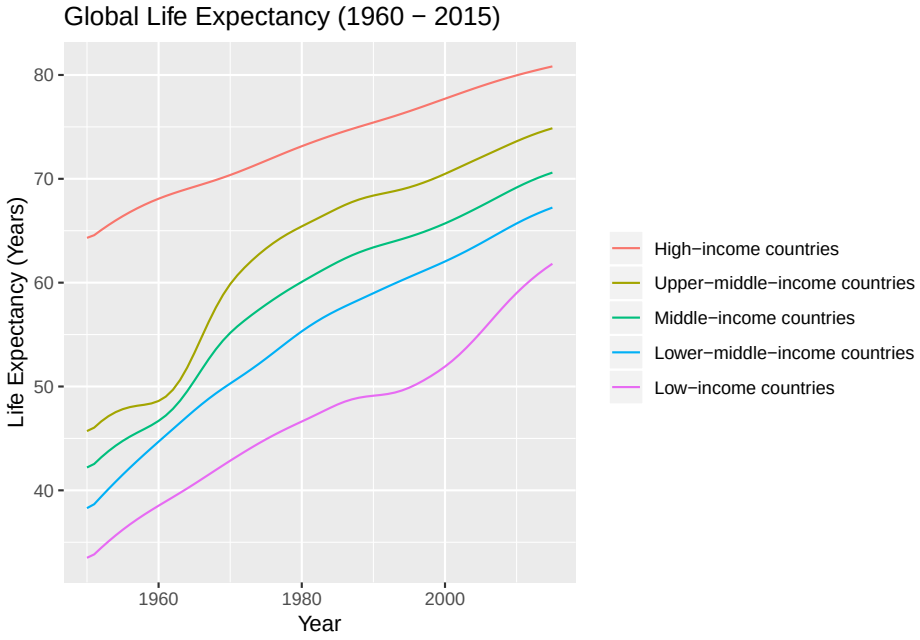
```
library(ggplot2)

le_plt <- ggplot(df, aes(x = Year, y = LE, group = Entity, colour = Entity)) +
  geom_line() +
  ylab("Life Expectancy (Years)") +
  ggtitle("Global Life Expectancy (1960 - 2015)") +
  theme(legend.title = element_blank())
```

```
le_plt
```

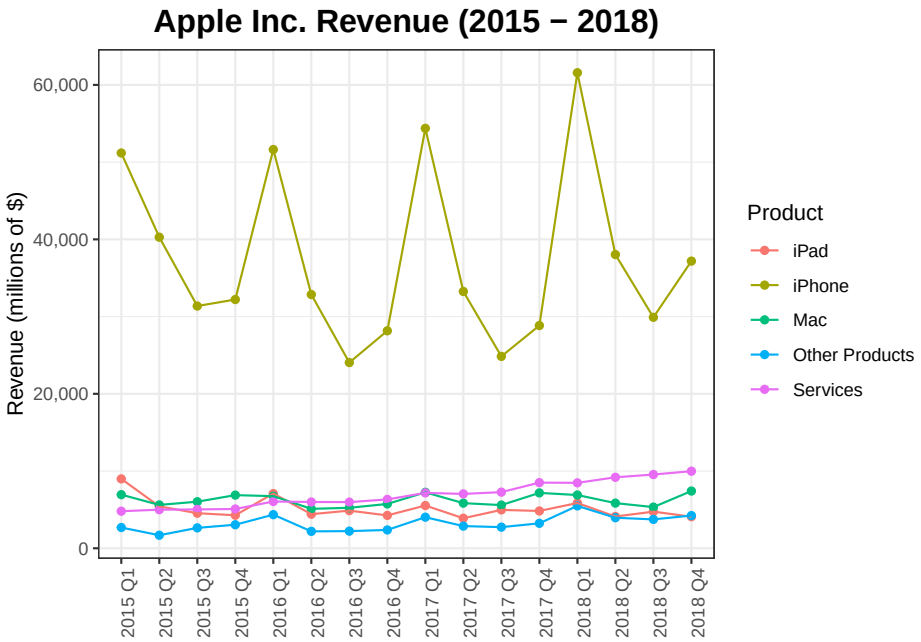


We can reorder the country groups so that the legend shows in the same order that the lines do in the plot.

```
library(ggplot2)
group_order <- c("High-income countries",  "Upper-middle-income countries", "Middle-income countr

df$Entity <- factor(df$Entity, levels = group_order)


le_plt <- ggplot(df, aes(x = Year, y = LE, group = Entity, colour = Entity)) +
  geom_line() +
  ylab("Life Expectancy (Years)") +
  ggtitle("Global Life Expectancy (1960 - 2015)") +
  theme(legend.title = element_blank())

le_plt
```


Global Life Expectancy (1960 − 2015)
Life Expectancy (Years)
80
70
60
50
40
1960
1980
2000
Year
High−income countries
Upper−middle−income countries
Middle−income countries
Lower−middle−income countries
Low−income countries

Apple Inc Revenue



Data

This plot uses the *apple* data frame of the *gcubed* package. This data frame contains the revenue (in millions of dollars) for each of Apple's product lines for the period 2015 to 2018. This data was collated from Apple's quarterly reports over that period.[2]

```
library(gcubed)
head(apple)
```

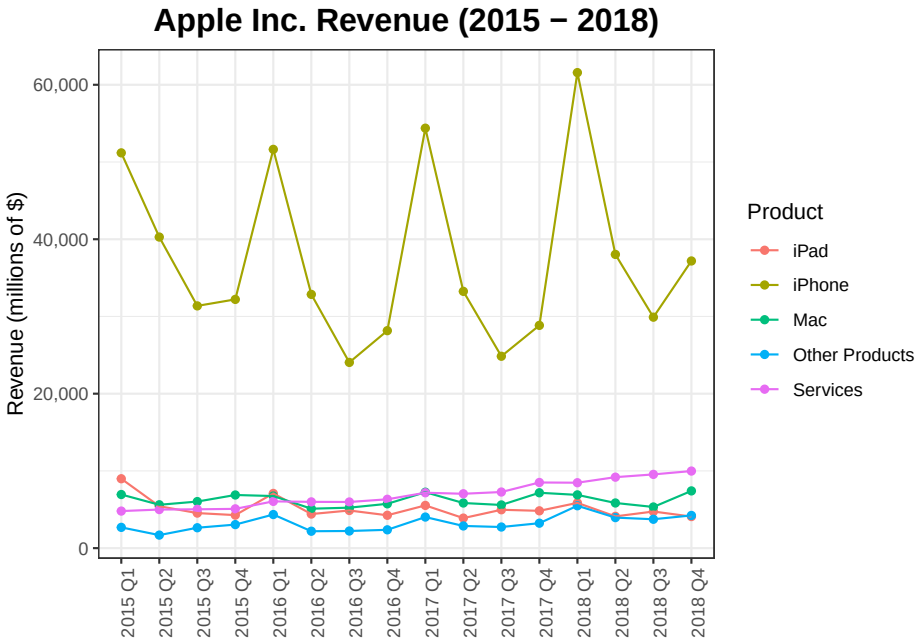
[2] One example can be found here

```
## # A tibble: 6 x 5
##    Year Quarter Product        Units Revenue
##   <int>   <int> <chr>          <dbl>   <dbl>
## 1  2015       1 iPad           21419    8985
## 2  2015       1 iPhone         74468   51182
## 3  2015       1 Mac             5519    6944
## 4  2015       1 Other Products    NA    2689
## 5  2015       1 Services          NA    4799
## 6  2015       2 iPad           12623    5428
```

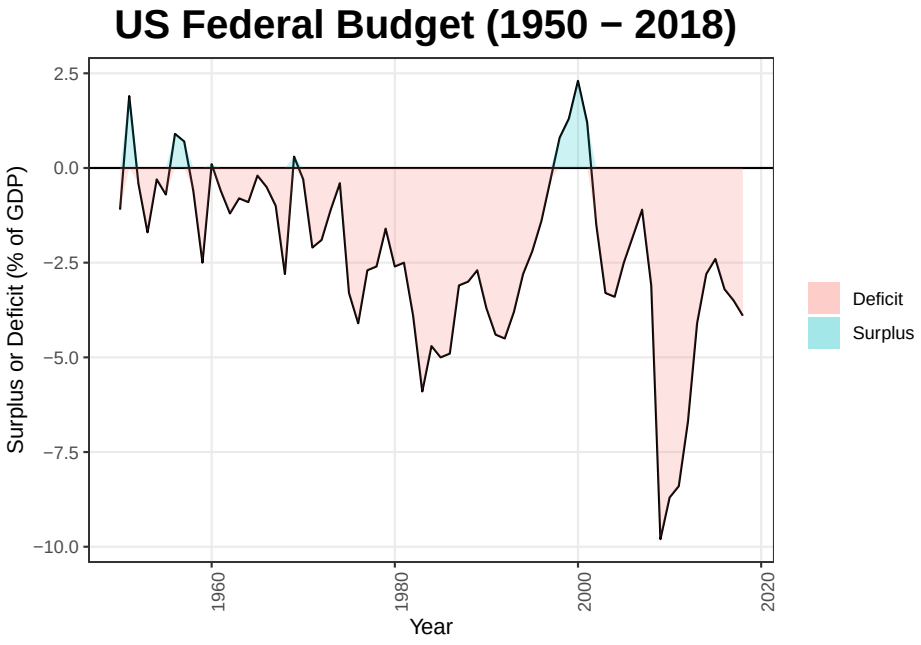
Code for plot

The code makes use of both *geom_point* and *geom_line* as well as *group* and *colour* aesthetics.

```
library(ggplot2)
library(scales) #for formatting the numerical y-axis values

apple_rev_plot <- ggplot(data = apple, aes(x = paste(Year, Quarter, sep = " Q"), y = Re
  geom_point() +
  geom_line() +
  ggtitle("Apple Inc. Revenue (2015 - 2018)") +
  ylab("Revenue (millions of $)") +
  theme_bw() +
  scale_y_continuous(label=comma) +
  theme(axis.text.x = element_text(angle = 90),
        axis.title.x = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

apple_rev_plot
```


Apple Inc. Revenue (2015 − 2018)
Revenue (millions of $)
60,000
40,000
20,000
0
2015 Q1
2015 Q2
2015 Q3
2015 Q4
2016 Q1
2016 Q2
2016 Q3
2016 Q4
2017 Q1
2017 Q2
2017 Q3
2017 Q4
2018 Q1
2018 Q2
2018 Q3
2018 Q4
Product
iPad
iPhone
Mac
Other Products
Services

Budget Surplus or Deficit



Data

This plot uses the *budget* data frame of the *gcubed* package. In particular, the columns *Year* and *SurpDef_pg* are used. *SurpDef_pg* represents the surplus/deficit as a percentage of the US GDP for the given year. Some rows of the data frame are shown below.

The source of this data was the Federal Reserve Economic Data (FRED) of the St. Louis Fed.

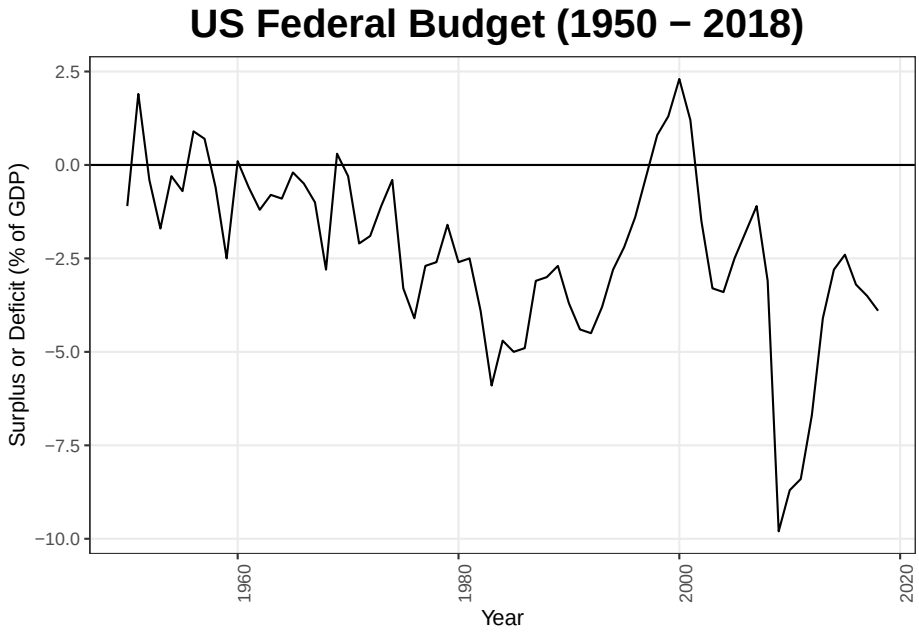
```
library(gcubed)
budget[budget$Year %in% c(1970, 1980, 1990, 2000, 2010),
       c("Year", "SurpDef_pg")]
```

```
## # A tibble: 5 x 2
##    Year SurpDef_pg
##   <dbl>      <dbl>
## 1  1970       -0.3
## 2  1980       -2.6
## 3  1990       -3.7
## 4  2000        2.3
## 5  2010       -8.7
```

Code

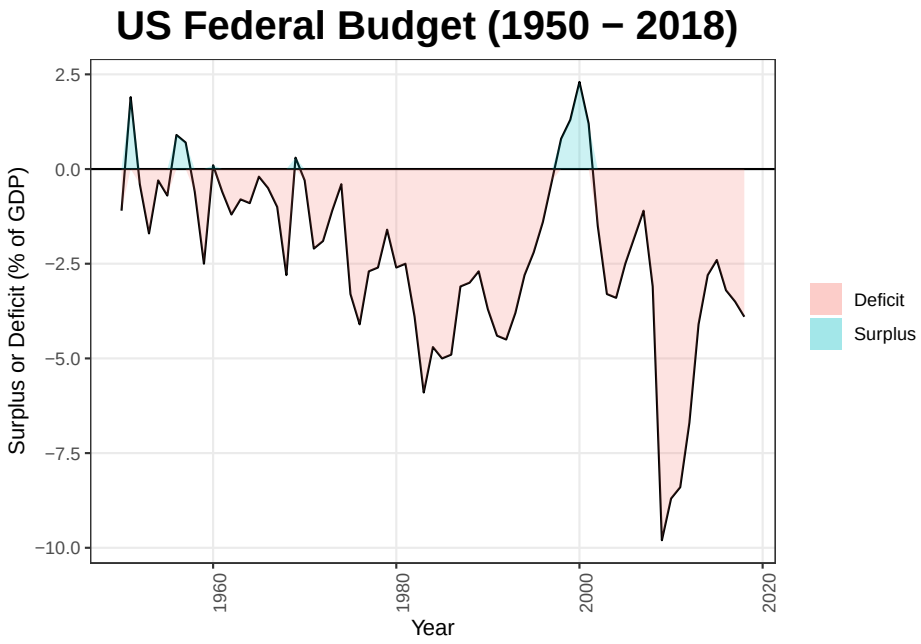
First, we make a plot using *geom_line*. *geom_hline* is also used to create the x-axis.

```
library(ggplot2)
df <- budget[budget$Year >= 1950, ]

budget_plt <- ggplot(data = df, aes(x = Year, y = SurpDef_pg)) +
  geom_line() +
  geom_hline(yintercept = 0) + #deficit ribbon below
  theme_bw() +
  ylab("Surplus or Deficit (% of GDP)") +
  ggtitle("US Federal Budget (1950 - 2018) ") +
    theme(panel.grid.minor = element_blank(),
        axis.text.x = element_text(angle = 90),
        plot.title = element_text(size = 20, face = "bold", hjust = 0.5),
        legend.title = element_blank())

budget_plt
```



The filled-in regions are added using the *geom_ribbon* geometry.

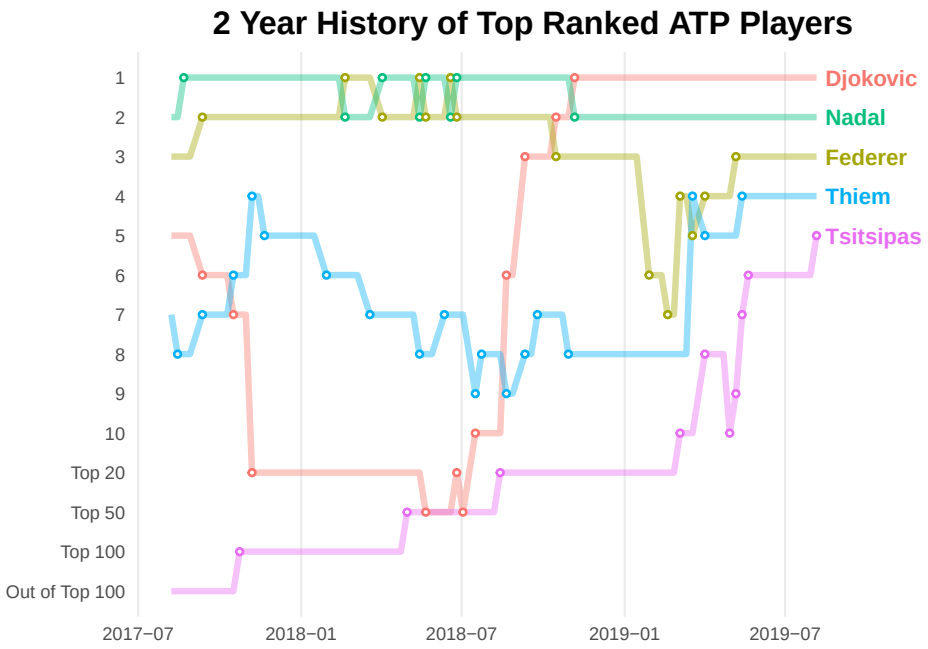
```
budget_plt <- budget_plt +
  geom_ribbon(aes(ymin = ifelse(SurpDef_pg < 0, SurpDef_pg, 0),
                  ymax = 0,
                  fill = "Deficit"), alpha = 0.2) +
  geom_ribbon(aes(ymin = 0,
                  ymax = ifelse(SurpDef_pg > 0, SurpDef_pg, 0),
                  fill = "Surplus"), alpha = 0.2)


budget_plt
```



Alternatively, all the code for the entire plot is shown below.

```
library(ggplot2)
library(gcubed)

df <- budget[budget$Year >= 1950, ]

budget_plt <- ggplot(data = df, aes(x = Year, y = SurpDef_pg)) +
  geom_line() +
  geom_hline(yintercept = 0) + #deficit ribbon below
  theme_bw() +
  ylab("Surplus or Deficit (% of GDP)") +
  ggtitle("US Federal Budget (1950 - 2018) ") +
    theme(panel.grid.minor = element_blank(),
        axis.text.x = element_text(angle = 90),
        plot.title = element_text(size = 20, face = "bold", hjust = 0.5),
        legend.title = element_blank()) +
  geom_ribbon(aes(ymin = ifelse(SurpDef_pg < 0, SurpDef_pg, 0),
                  ymax = 0,
                  fill = "Deficit"), alpha = 0.2) +
  geom_ribbon(aes(ymin = 0,
                  ymax = ifelse(SurpDef_pg > 0, SurpDef_pg, 0),
                  fill = "Surplus"), alpha = 0.2)
```

2 Year History of Top-ranked ATP Players



Data

For this plot, we will use the *atp_rankings* data frame of the *gcubed* package.

This data was originally obtained from the ATP[3].

[3] I collated each of the players' individual rankings history. For example, Novak Djokovic's ranking history

```
library(gcubed)
head(atp_rankings)
```

```
## # A tibble: 6 x 6
##    Year Month   Day Singles Player    Date
##   <dbl> <dbl> <dbl>   <int> <chr>     <dttm>
## 1  2017     8     7       5 Djokovic  2017-08-07 12:00:00
## 2  2017     8     7       2 Nadal     2017-08-07 12:00:00
## 3  2017     8     7       3 Federer   2017-08-07 12:00:00
## 4  2017     8     7       7 Thiem     2017-08-07 12:00:00
## 5  2017     8     7     168 Tsitsipas 2017-08-07 12:00:00
## 6  2017     8    14       5 Djokovic  2017-08-14 12:00:00
```

First, create a new variable, *Ranking* that preserves the rankings when the player is in the top 10. When the player is not in the top 10, the new variable is set to: 11 if the player is in the top 20; 12 if the player is ranked between 21 and 50 (inclusive); 13 if the player is ranked between 51 and 100 (inclusive); 14 if the player is ranked lower than 100.

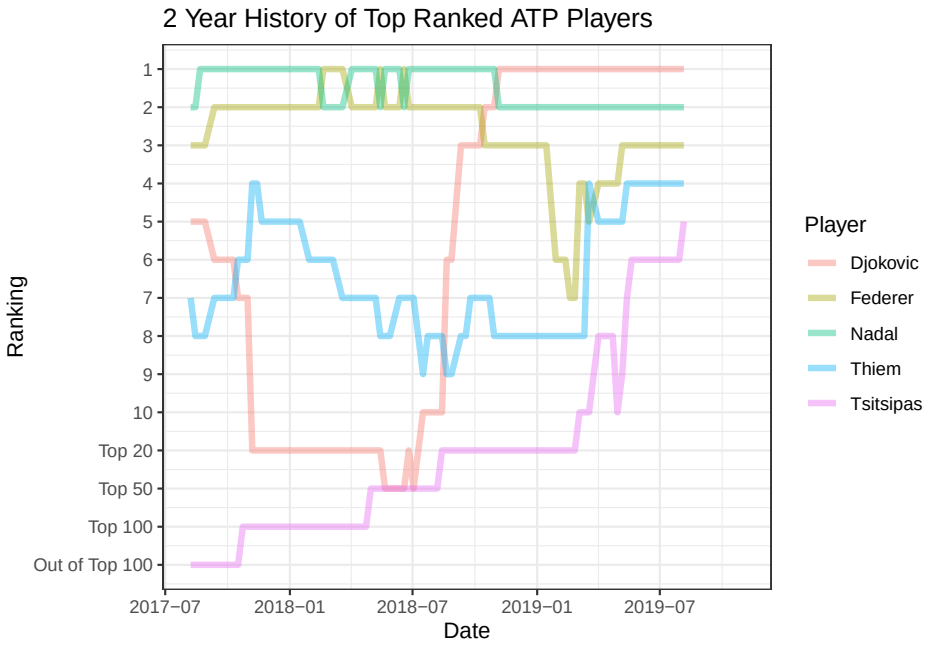
Also, we create a variable *Change* to be used later to identify the points in time when the players' rankings changed.

```
library(dplyr)

rankings <- mutate(atp_rankings, Ranking = ifelse(Singles > 100, 14,
                           ifelse(Singles > 50, 13,
                                  ifelse(Singles > 20, 12,
                                         ifelse(Singles > 10, 11, Singles))))) %>%
  group_by(Player) %>% mutate(Change = c(0,diff(Ranking))) %>% ungroup()
```

Code for plot

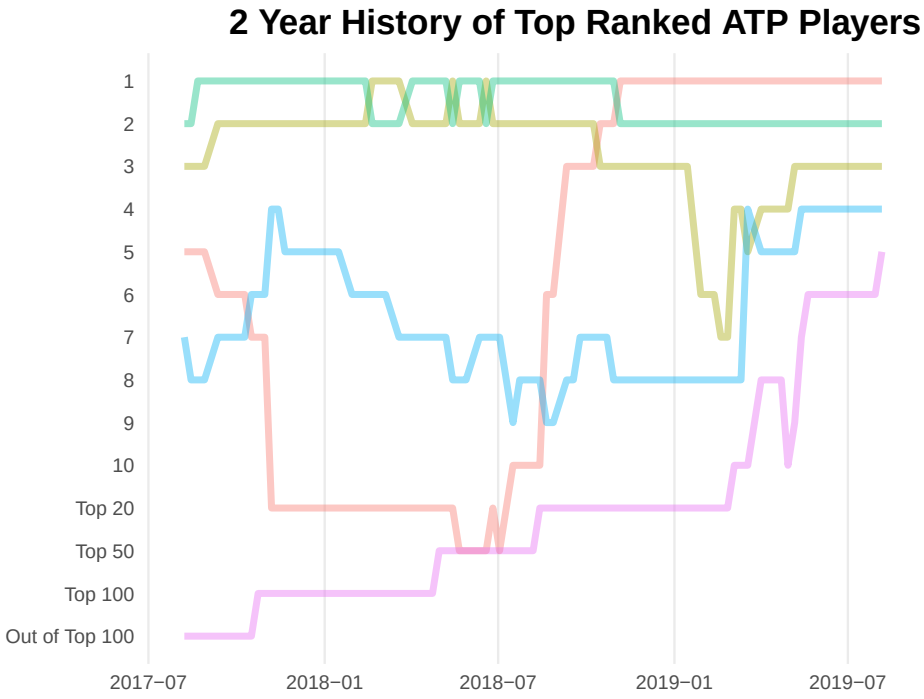
```
library(ggplot2)


ylabels <- c(1:10, "Top 20", "Top 50", "Top 100", "Out of Top 100")


show_date <- ISOdate(2019, 11,1)
begin_date <- ISOdate(2017, 8, 7)
next_date <- ISOdate(2019, 8, 15)
```

```
atp_plt <- ggplot(data = rankings, aes(x = Date, y = Ranking, group = Player)) +
  geom_line(aes(color = Player), alpha = 0.4, size = 1.5) +
  scale_y_continuous(breaks = c(1:14), labels = ylabels, trans = "reverse") +
  ggtitle("2 Year History of Top Ranked ATP Players") +
  xlim(c(begin_date, show_date)) +
  theme_bw()

atp_plt
```



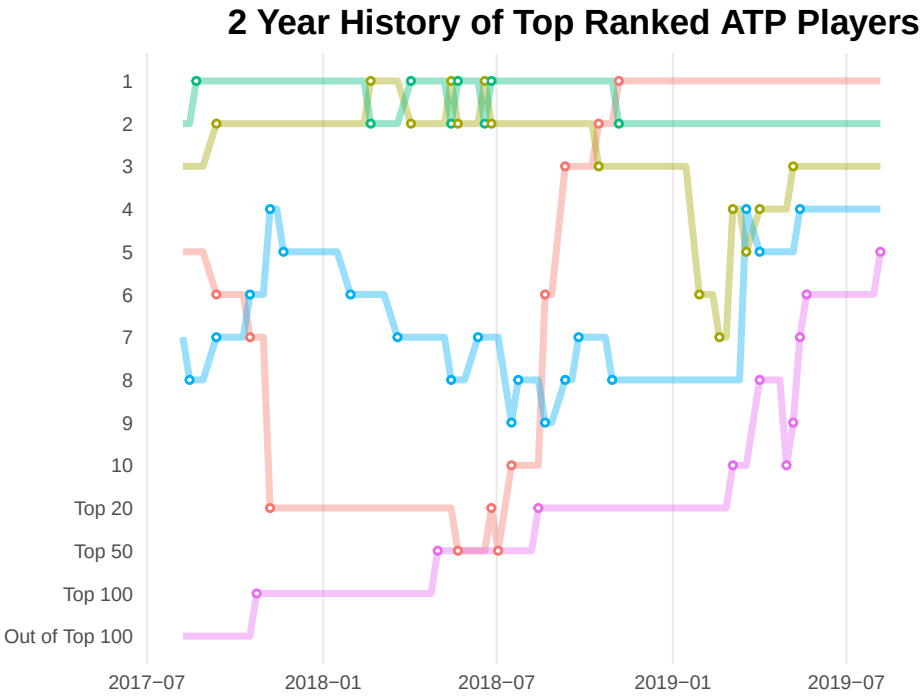
Next, we can change the overall look of the plot using the *theme* function to change several details of the graph.

```
atp_plt <- atp_plt +
  theme(panel.grid.major.y = element_blank(), panel.grid.minor.y = element_blank(),
        panel.grid.minor.x = element_blank(), axis.ticks = element_blank(),
        legend.position = "none", panel.border = element_blank(),
        axis.title.x = element_blank(), axis.title.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

atp_plt
```



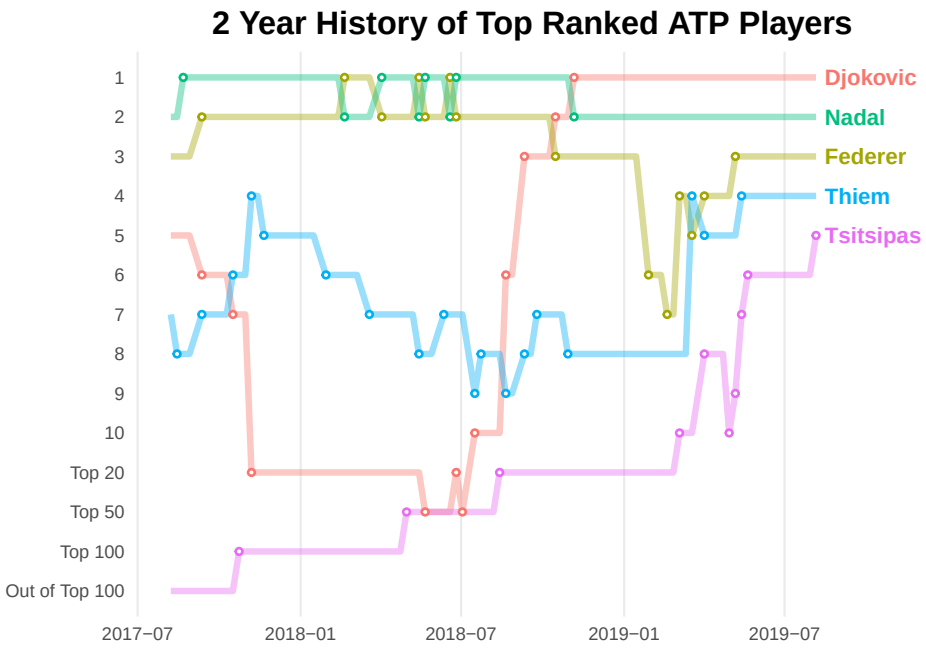
Adding some points to signify the times at which the players' rankings changed using *geom_point*. We are going to use two *geom_point* geometries to create a smaller white circle inside the coloured larger circles.

```
changes <- filter(rankings, Change != 0)


atp_plt <- atp_plt + geom_point(data = changes, aes(x = Date, y = Ranking, color = Pla
  geom_point(data = changes, color = "#FFFFFF", size = 0.25)

atp_plt
```



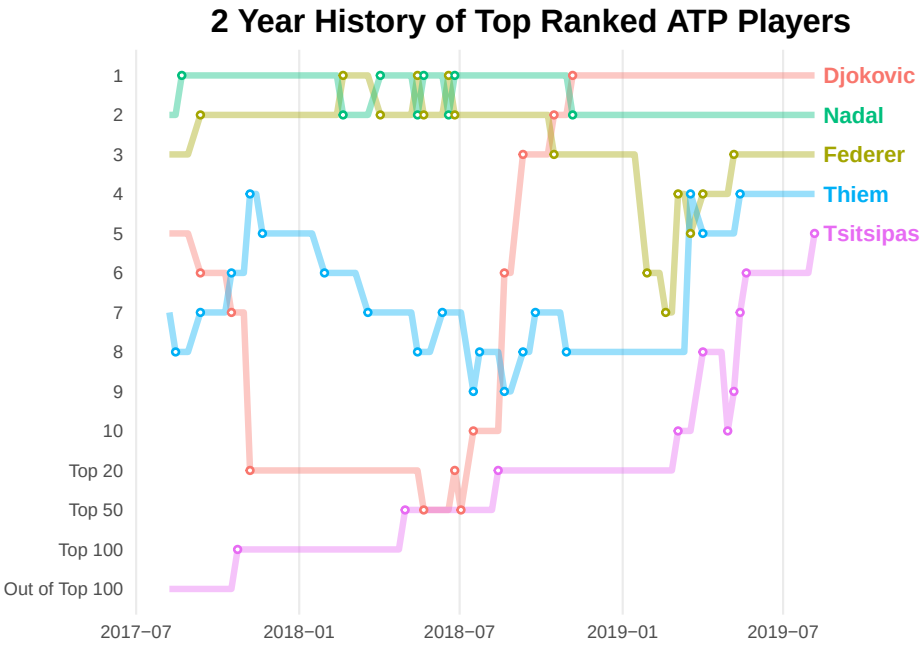
Now to add the annotation of the players' names using *geom_text*.

```
last_rankings <- rankings %>% top_n(5, Date)

last_rankings$nextd <- next_date

atp_plt <- atp_plt + geom_text(data = last_rankings,
            aes(label = Player, x = nextd,  colour = Player, y = Ranking) , hjust = 0,
            fontface = "bold", size = 4)

atp_plt
```

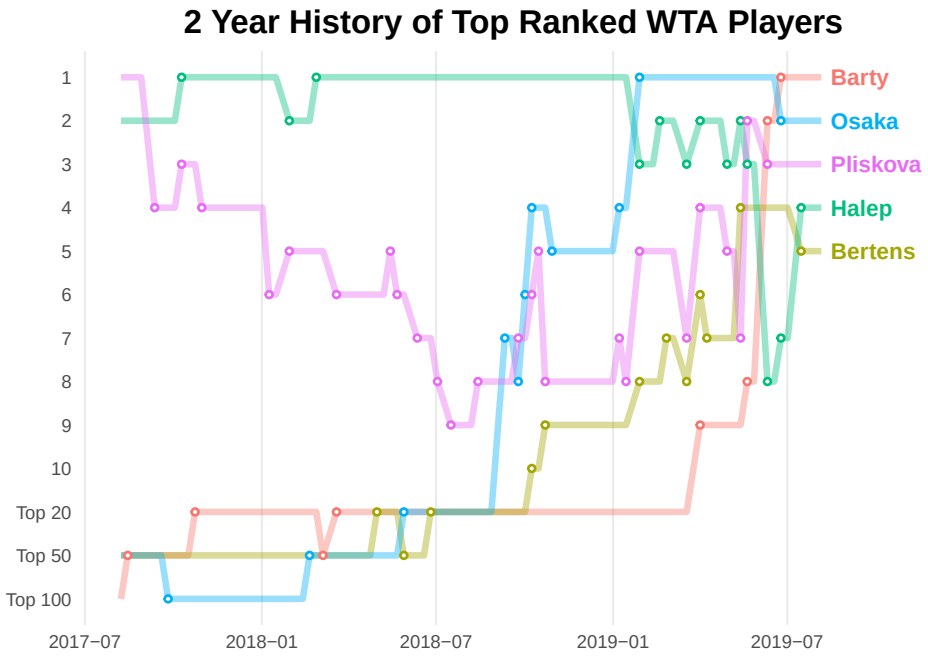


The complete code for the plot:

```
atp_plt <- ggplot(data = rankings, aes(x = Date, y = Ranking, group = Player)) +
  geom_line(aes(color = Player), alpha = 0.4, size = 1.5) +
  scale_y_continuous(breaks = c(1:14), labels = ylabels, trans = "reverse") +
  ggtitle("2 Year History of Top Ranked ATP Players") +
  xlim(c(begin_date, show_date)) +
  theme_bw() +
  theme(panel.grid.major.y = element_blank(), panel.grid.minor.y = element_blank(),
        panel.grid.minor.x = element_blank(), axis.ticks = element_blank(),
        legend.position = "none", panel.border = element_blank(),
        axis.title.x = element_blank(), axis.title.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5)) +
  geom_point(data = changes, aes(x = Date, y = Ranking, color = Player)) +
  geom_point(data = changes, color = "#FFFFFF", size = 0.25) +
  geom_text(data = last_rankings,
            aes(label = Player, x = nextd, y = Ranking, colour = Player) , hjust = 0,
            fontface = "bold", size = 4)

atp_plt
```


2 Year History of Top Ranked ATP Players
1
2
3
4
5
6
7
8
9
10
Top 20
Top 50
Top 100
Out of Top 100
2017−07
2018−01
2018−07
2019−01
2019−07
Djokovic
Nadal
Federer
Thiem
Tsitsipas

2 Year History of Top-ranked WTA Players



Data

For this plot, we will use the *wta_rankings* data frame of the *gcubed* package.

This data was originally obtained from the WTA[4].

[4]I collated each of the players' individual rankings history. For example, Naomi Osaka's ranking history

```
library(gcubed)
head(wta_rankings)
```

```
## # A tibble: 6 x 6
##   Month   Day  Year Singles Player   Date
##   <dbl> <dbl> <dbl>   <int> <chr>    <dttm>
## 1     8     7  2017      58 Barty    2017-08-07 12:00:00
## 2     8     7  2017      50 Osaka    2017-08-07 12:00:00
## 3     8     7  2017       1 Pliskova 2017-08-07 12:00:00
## 4     8     7  2017       2 Halep    2017-08-07 12:00:00
## 5     8     7  2017      27 Bertens  2017-08-07 12:00:00
## 6     8    14  2017      48 Barty    2017-08-14 12:00:00
```

First, create a new variable, *Ranking* that preserves the rankings when the player is in the top 10. When the player is not in the top 10, the new variable is set to: 11 if the player is in the top 20; 12 if the player is ranked between 21 and 50 (inclusive); 13 if the player is ranked between 51 and 100 (inclusive); 14 if the player is ranked lower than 100.

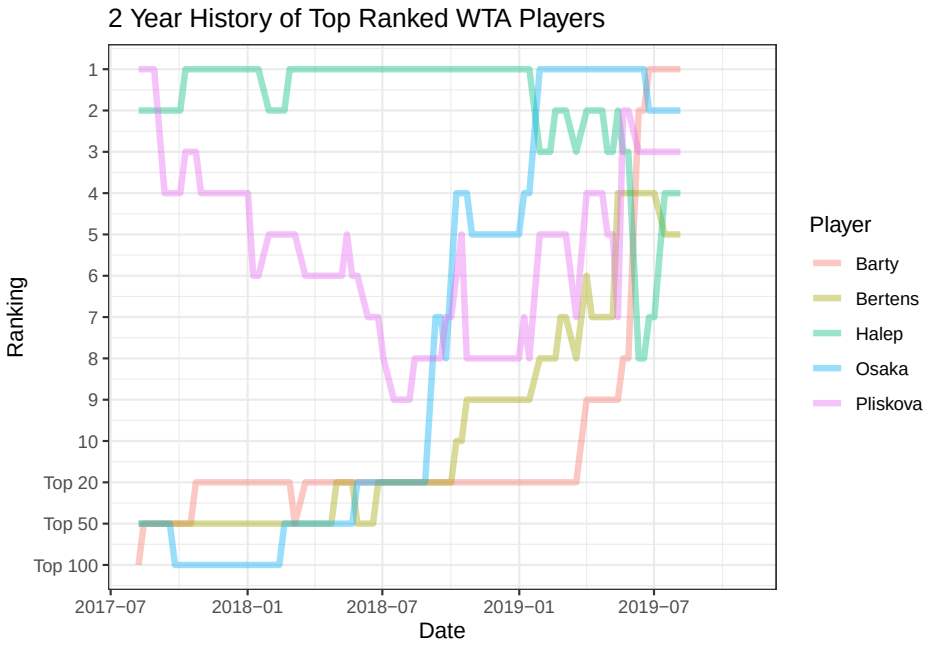
Also, we create a variable *Change* to be used later to identify the points in time when the players' rankings changed.

```
library(dplyr)
rankings <- mutate(wta_rankings, Ranking = ifelse(Singles > 100, 14,
                           ifelse(Singles > 50, 13,
                                  ifelse(Singles > 20, 12,
                                         ifelse(Singles > 10, 11, Singles))))) %>%
  group_by(Player) %>% mutate(Change = c(0,diff(Ranking))) %>% ungroup()
```

Code for plot

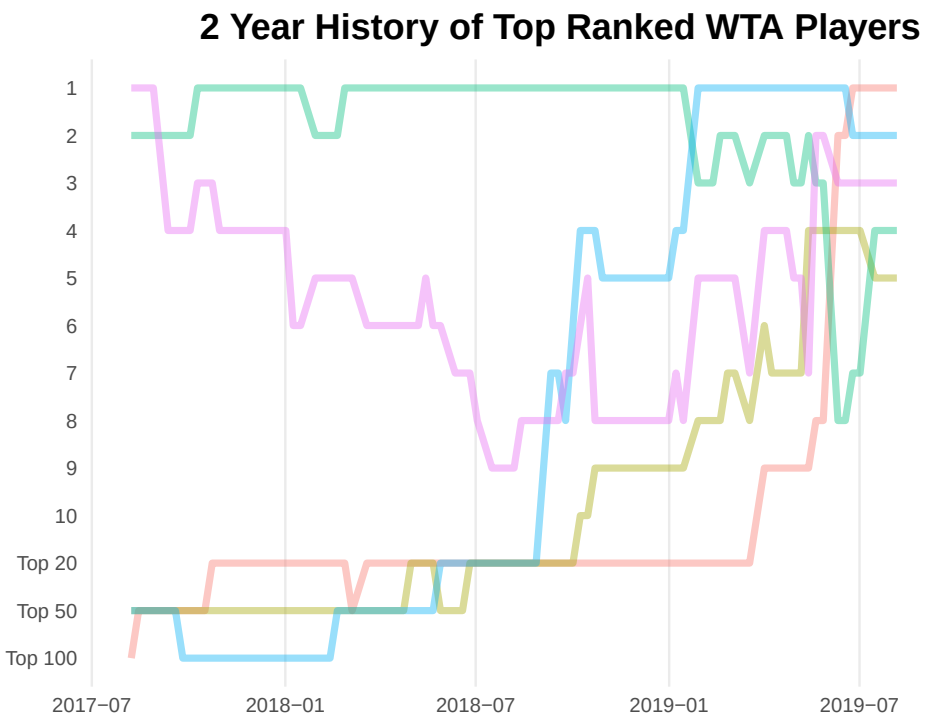
```
library(ggplot2)
ylabels <- c(1:10, "Top 20", "Top 50", "Top 100", "Out of Top 100")


show_date <- ISOdate(2019, 11,1)
begin_date <- ISOdate(2017, 8, 7)
next_date <- ISOdate(2019, 8, 15)



wta_plt <- ggplot(data = rankings, aes(x = Date, y = Ranking, group = Player)) +
  geom_line(aes(color = Player), alpha = 0.4, size = 1.5) +
```

```
  scale_y_continuous(breaks = c(1:14), labels = ylabels, trans = "reverse") +
  ggtitle("2 Year History of Top Ranked WTA Players") +
  xlim(c(begin_date, show_date)) +
  theme_bw()

wta_plt
```



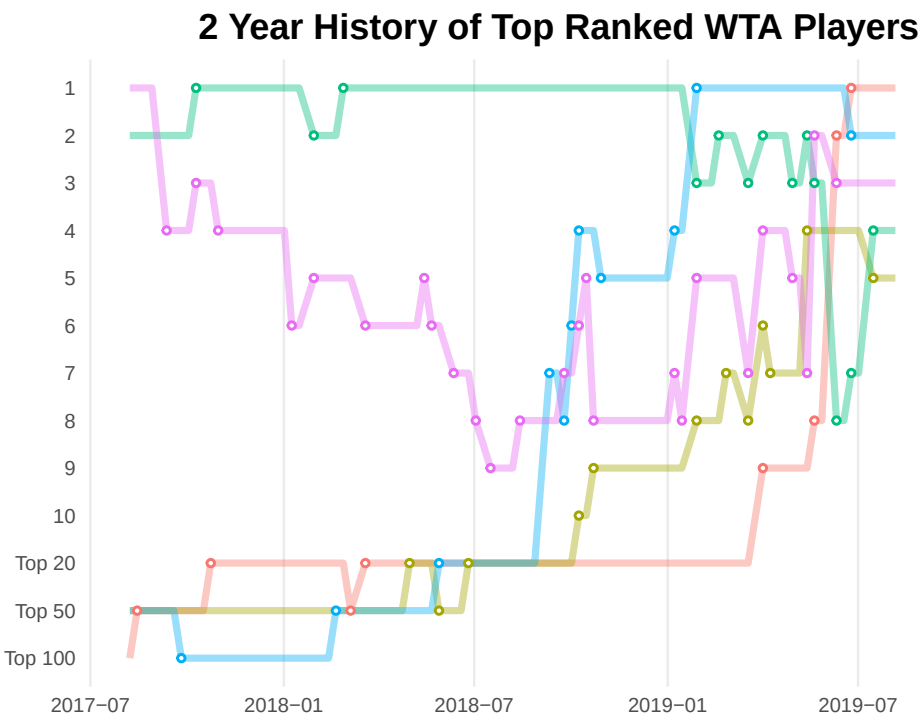
Next, we can change the overall look of the plot using the *theme* function to change several details of the graph.

```
wta_plt <- wta_plt +
  theme(panel.grid.major.y = element_blank(), panel.grid.minor.y = element_blank(),
        panel.grid.minor.x = element_blank(), axis.ticks = element_blank(),
        legend.position = "none", panel.border = element_blank(),
        axis.title.x = element_blank(), axis.title.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

wta_plt
```



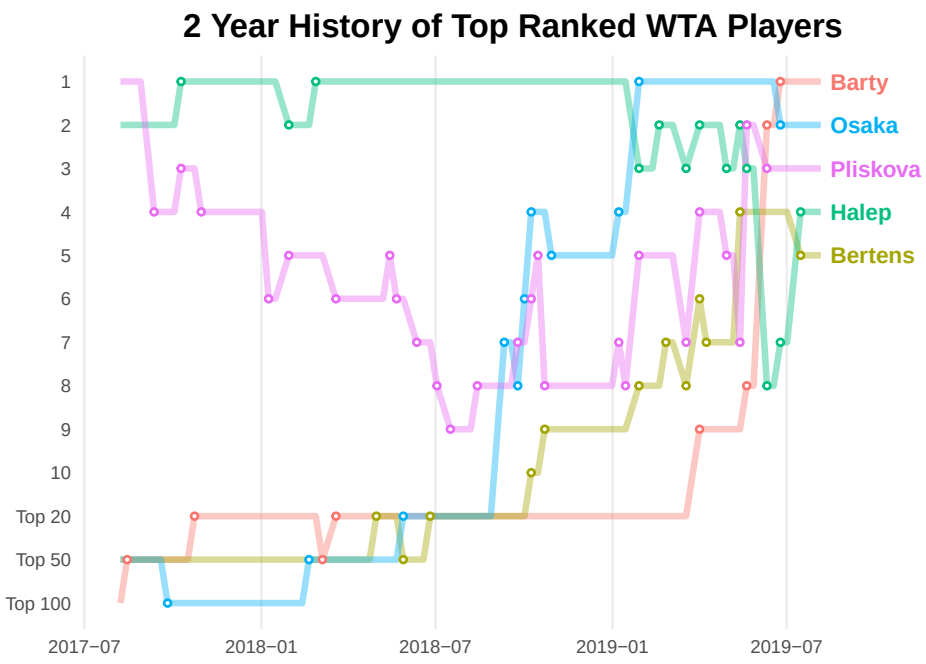
Adding some points to signify the times at which the players' rankings changed using *geom_point*. We are going to use two *geom_point* geometries to create a smaller white circle inside the coloured larger circles.

```
changes <- filter(rankings, Change != 0)


wta_plt <- wta_plt + geom_point(data = changes, aes(x = Date, y = Ranking, color = Pla
  geom_point(data = changes, color = "#FFFFFF", size = 0.25)

wta_plt
```



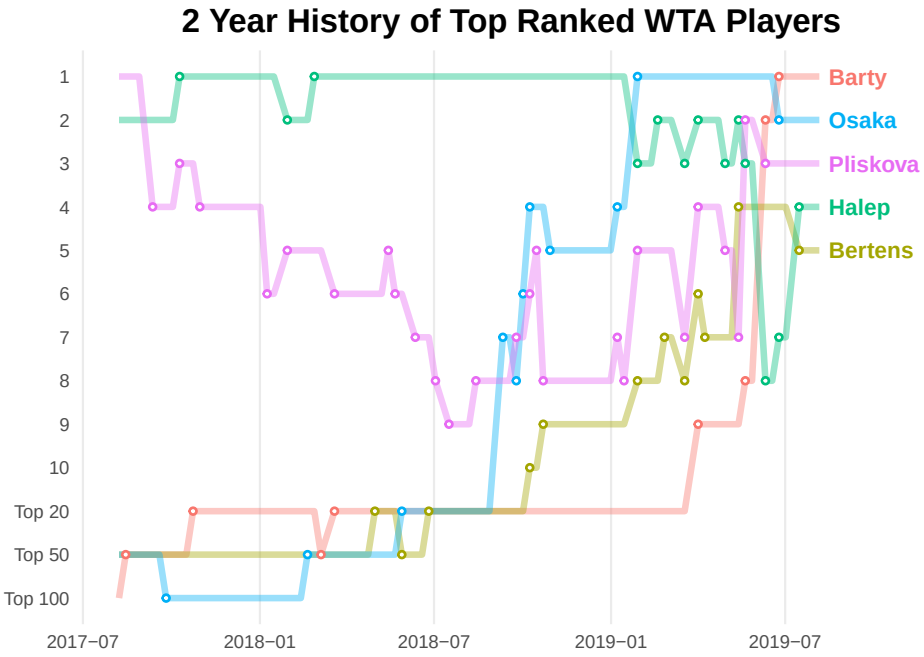
Now to add the annotation of the players' names using *geom_text*.

```
last_rankings <- rankings %>% top_n(5, Date)

last_rankings$nextd <- next_date

wta_plt <- wta_plt + geom_text(data = last_rankings,
            aes(label = Player, x = nextd,  colour = Player) , hjust = 0,
            fontface = "bold", size = 4)

wta_plt
```

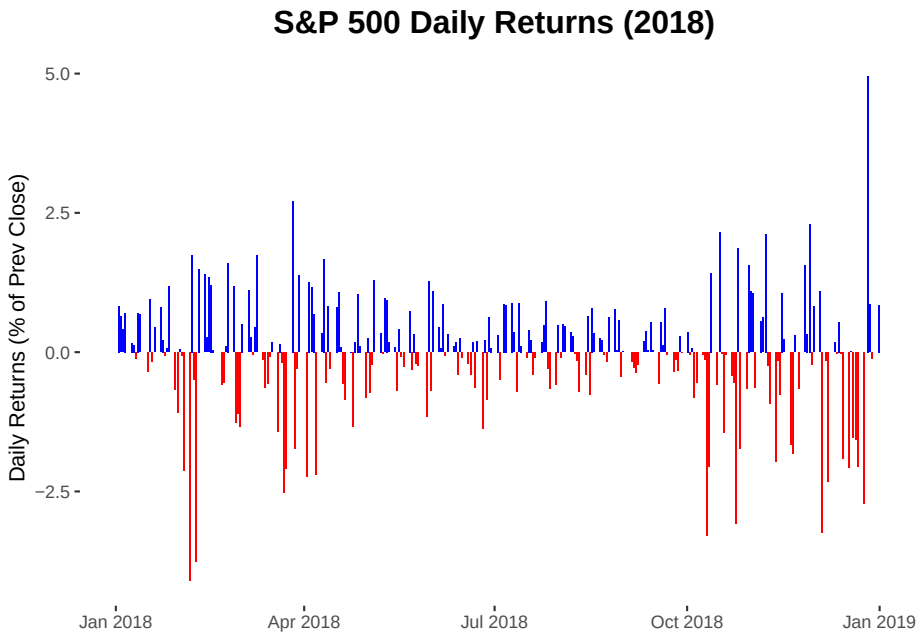


The complete code for the plot:

```
library(ggplot2)

wta_plt <- ggplot(data = rankings, aes(x = Date, y = Ranking, group = Player)) +
  geom_line(aes(color = Player), alpha = 0.4, size = 1.5) +
  scale_y_continuous(breaks = c(1:14), labels = ylabels, trans = "reverse") +
  ggtitle("2 Year History of Top Ranked WTA Players") +
  xlim(c(begin_date, show_date)) +
  theme_bw() +
  theme(panel.grid.major.y = element_blank(), panel.grid.minor.y = element_blank(),
        panel.grid.minor.x = element_blank(), axis.ticks = element_blank(),
        legend.position = "none", panel.border = element_blank(),
        axis.title.x = element_blank(), axis.title.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5)) + geom_point
  geom_point(data = changes, color = "#FFFFFF", size = 0.25) + geom_text(data = last_ra
            aes(label = Player, x = nextd,  colour = Player) , hjust = 0,
            fontface = "bold", size = 4)

wta_plt
```


2 Year History of Top Ranked WTA Players
1
2
3
4
5
6
7
8
9
10
Top 20
Top 50
Top 100
2017−07
2018−01
2018−07
2019−01
2019−07
Barty
Osaka
Pliskova
Halep
Bertens

S&P 500 daily returns in 2018



0.0.0.0.2 Data

This plot uses the *sp500* data frame of the *gcubed* package.

```
library(gcubed)
tail(sp500)
```

```
## # A tibble: 6 x 12
##   Month   Day  Year  Open  High   Low Close `Adj Close` Volume PrevClose
##   <int> <int> <int> <dbl> <dbl> <dbl> <dbl>       <dbl>  <dbl>     <dbl>
```

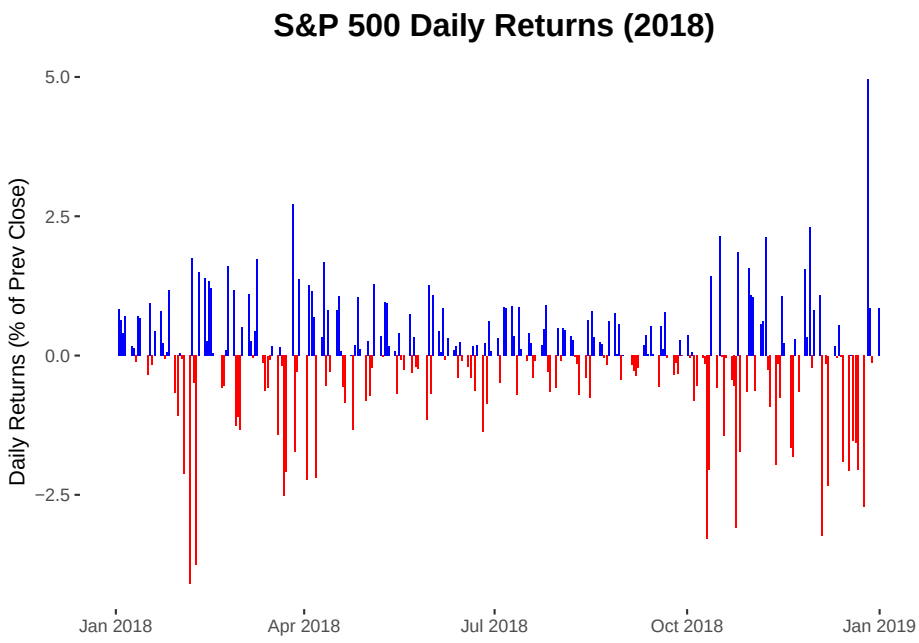
```
## 1    12    21  2018 2465. 2504. 2409. 2417.       2417. 7.61e9     2467.
## 2    12    24  2018 2401. 2410. 2351. 2351.       2351. 2.61e9     2417.
## 3    12    26  2018 2363. 2468. 2347. 2468.       2468. 4.23e9     2351.
## 4    12    27  2018 2442. 2489. 2398. 2489.       2489. 4.10e9     2468.
## 5    12    28  2018 2499. 2520. 2473. 2486.       2486. 3.70e9     2489.
## 6    12    31  2018 2499. 2509. 2483. 2507.       2507. 3.44e9     2486.
## # ... with 2 more variables: daily_return <dbl>, abs_ret <dbl>
```

First, we will restrict the data to only those entries from the year 2018. Then we will create a new column, *updown* that will simply say whether or not each day's return represented a gain or a loss. This will be used later to colour the bars of the plot.

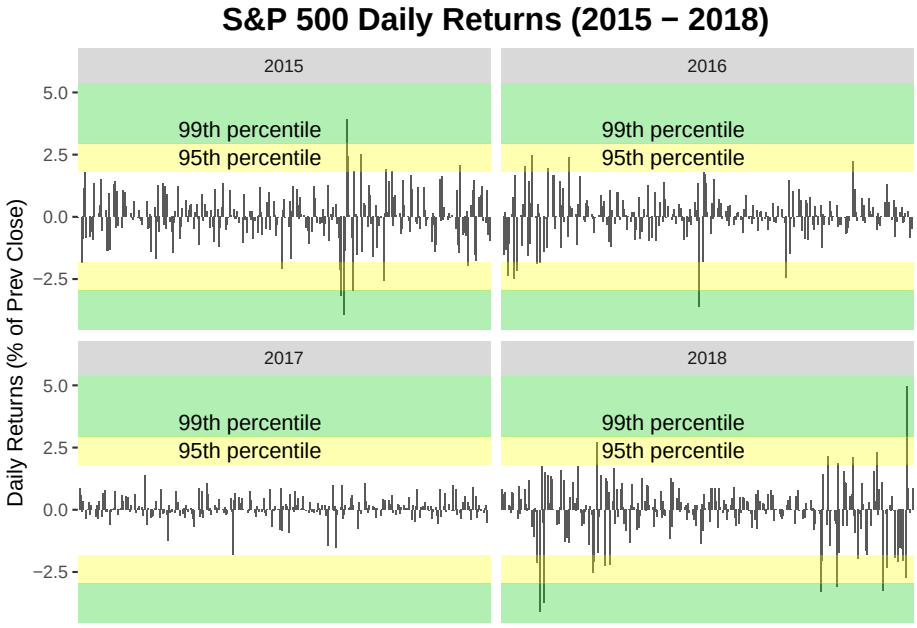
Code for plot

We will use the *geom_bar* geometry to create this plot. The *fill* aesthetic_ will be used to colour the bars appropriately for positive and negative daily returns.

```
library(ggplot2)

sp18_plt <- ggplot(data = sp18, aes(x = Date, y = daily_return, fill = updown )) +
  geom_bar(stat = "identity") +
  ylab("Daily Returns (% of Prev Close)")+
  guides(fill = guide_legend(override.aes= list(alpha = 0.2))) +
  ggtitle("S&P 500 Daily Returns (2018) ") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        panel.background = element_blank(),
        axis.title.x=element_blank(),
        legend.position = "none") +
  scale_fill_manual(values = c("blue", "red"))

sp18_plt
```


S&P 500 Daily Returns (2018)
Daily Returns (% of Prev Close)
5.0
2.5
0.0
−2.5
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019

S&P 500 daily returns 2015 - 2018



0.0.0.0.3 Data

This plot uses the *sp500* data frame of the *gcubed* package. Rows 250, 500, 750 and 1000 of the data frame are shown below.

```
library(gcubed)
sp500[c(250,500,750,1000),]
```

```
## # A tibble: 4 x 12
##   Month   Day  Year  Open  High   Low Close `Adj Close` Volume PrevClose
```

```
##   <int> <int> <int> <dbl> <dbl> <dbl> <dbl>       <dbl>  <dbl>     <dbl>
## 1    12    29  2015 2061. 2082. 2061. 2078.       2078. 2.54e9     2056.
## 2    12    23  2016 2260. 2264. 2259. 2264.       2264. 2.02e9     2261.
## 3    12    21  2017 2683. 2693. 2682. 2685.       2685. 3.27e9     2679.
## 4    12    20  2018 2497. 2510. 2441. 2467.       2467. 5.59e9     2507.
## # ... with 2 more variables: daily_return <dbl>, abs_ret <dbl>
```

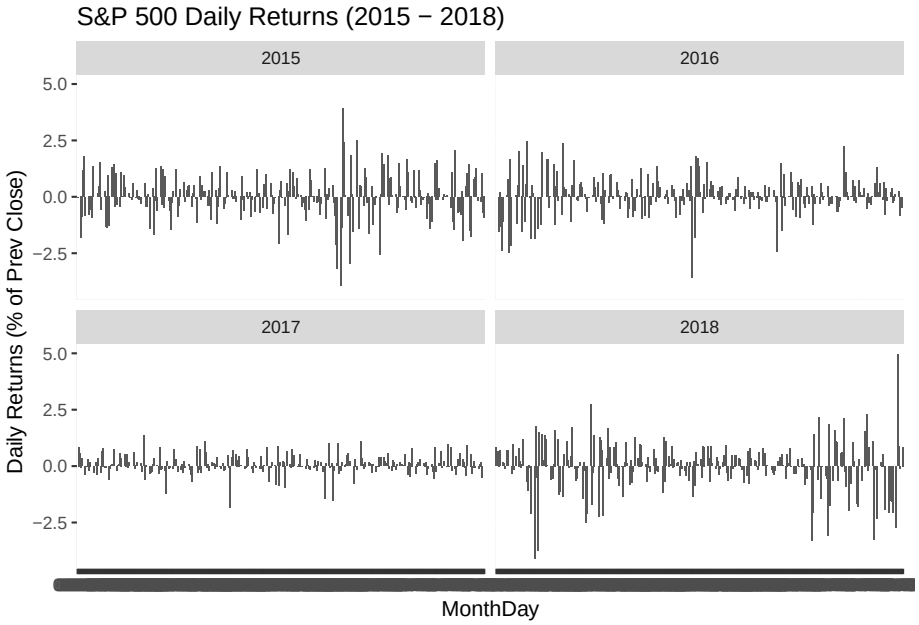
First, we will restrict the data to only those entries from the year 2018. Then we will create a new column, *updown* that will simply say whether or not each day's return represented a gain or a loss. This will be used later to colour the bars of the plot.

```
## # A tibble: 4 x 13
##   Month   Day  Year  Open  High   Low Close `Adj Close` Volume PrevClose
##   <int> <int> <int> <dbl> <dbl> <dbl> <dbl>       <dbl>  <dbl>     <dbl>
## 1    12    29  2015 2061. 2082. 2061. 2078.       2078. 2.54e9     2056.
## 2    12    23  2016 2260. 2264. 2259. 2264.       2264. 2.02e9     2261.
## 3    12    21  2017 2683. 2693. 2682. 2685.       2685. 3.27e9     2679.
## 4     5    27  2015 2105. 2126. 2105. 2123.       2123. 3.13e9     2104.
## # ... with 3 more variables: daily_return <dbl>, abs_ret <dbl>,
## #   MonthDay <chr>
```

Code for plot

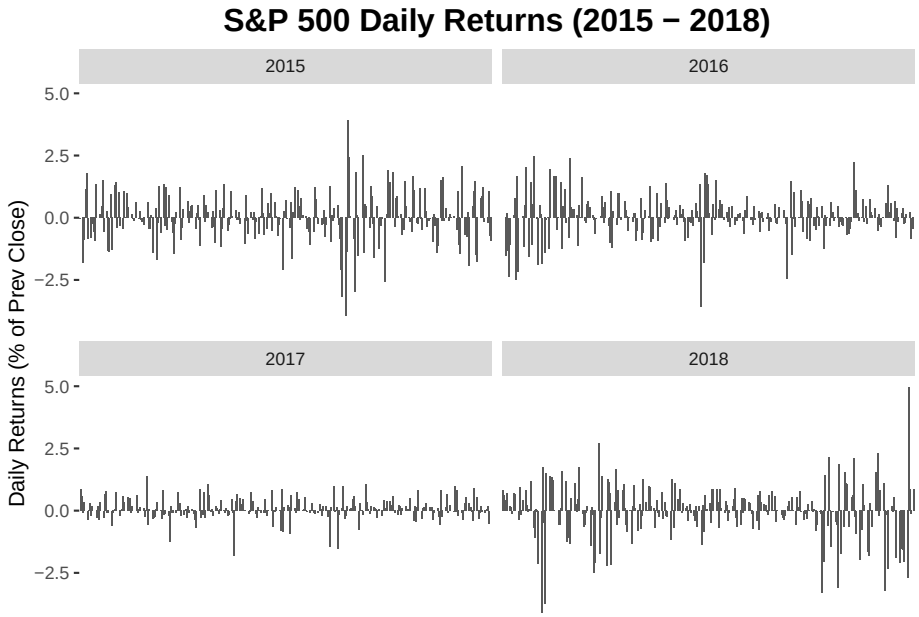
We will use the *geom_bar* geometry to create this plot. The *fill* aesthetic_ will be used to colour the bars appropriately for positive and negative daily returns.

```
library(ggplot2)

sp_plt <- ggplot(data = df, aes(x = MonthDay, y = daily_return)) +
  geom_bar(stat = "identity") +
  facet_wrap(~Year) +
  ylab("Daily Returns (% of Prev Close)") +
  ggtitle("S&P 500 Daily Returns (2015 - 2018)")


sp_plt
```



At present the x-axis labels are from a categorical variable, *MonthDay*. The hundreds of overlapping values being displayed can be removed to de-clutter the lower portion of the plot.

```
sp_plt <- sp_plt +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        panel.background = element_blank(), axis.title.x=element_blank(),
        axis.text.x =  element_blank(), axis.ticks.x = element_blank())

sp_plt
```



To add bands representing 95th and 99th percentile moves, first we use determine what the 95th and 99th percentile moves are.

```
df$abs_return <- abs(df$daily_return)
head(df)
```

```
## # A tibble: 6 x 14
##   Month   Day  Year  Open  High   Low Close `Adj Close` Volume PrevClose
##   <int> <int> <int> <dbl> <dbl> <dbl> <dbl>       <dbl>  <dbl>     <dbl>
## 1     1     2  2015 2059. 2072. 2046. 2058.       2058. 2.71e9     2059.
## 2     1     5  2015 2054. 2054. 2017. 2021.       2021. 3.80e9     2058.
## 3     1     6  2015 2022. 2030. 1992. 2003.       2003. 4.46e9     2021.
## 4     1     7  2015 2006. 2030. 2006. 2026.       2026. 3.81e9     2003.
## 5     1     8  2015 2031. 2064. 2031. 2062.       2062. 3.93e9     2026.
## 6     1     9  2015 2063. 2064. 2038. 2045.       2045. 3.36e9     2062.
## # ... with 4 more variables: daily_return <dbl>, abs_ret <dbl>,
## #   MonthDay <chr>, abs_return <dbl>
```

```
pct95 <- quantile(df$abs_return, .95)
pct95
```

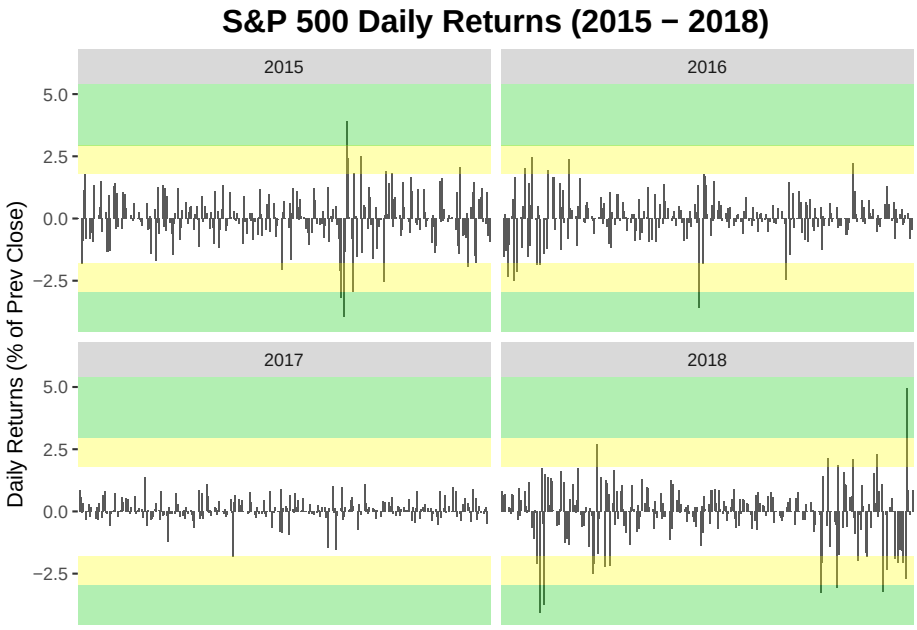
```
##      95%
## 1.817147
```

```
pct99 <- quantile(df$abs_return, .99)
pct99
```

```
##      99%
## 2.945549
```

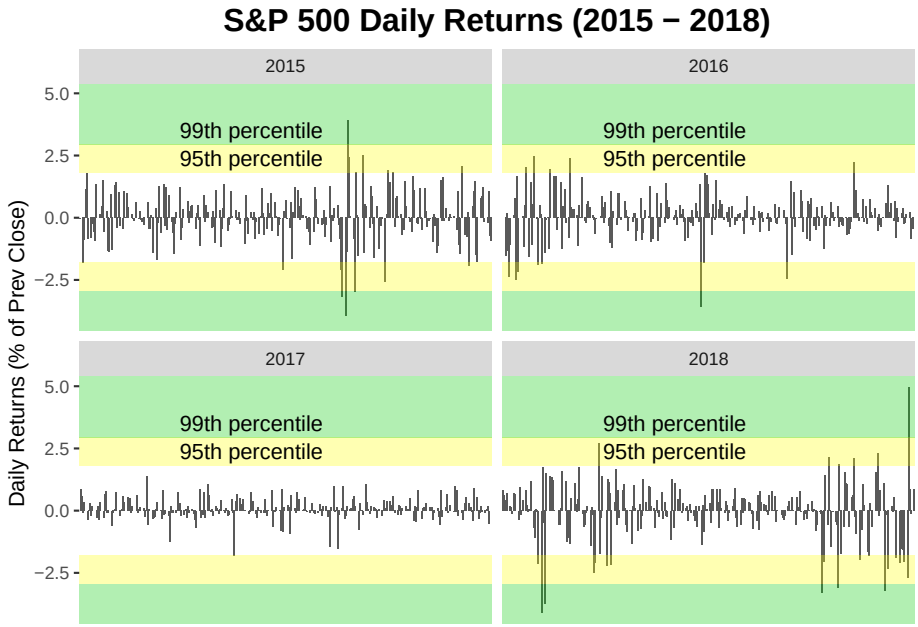
The bands can be added using *annotate* to create the ribbons.

```
sp_plt <- sp_plt +
  annotate("ribbon", ymin = pct95, ymax = pct99, x = c(-Inf,Inf), alpha = 0.3, fill = "95") +
  annotate("ribbon", ymin = pct99, ymax = Inf, x = c(-Inf, Inf), alpha = 0.3, fill = "99") +
  annotate("ribbon", ymax = -pct95, ymin = -pct99, x = c(-Inf,Inf), alpha = 0.3, fill = "95") +
  annotate("ribbon", ymax = -pct99, ymin = -Inf, x = c(-Inf, Inf), alpha = 0.3, fill = "99")

sp_plt
```



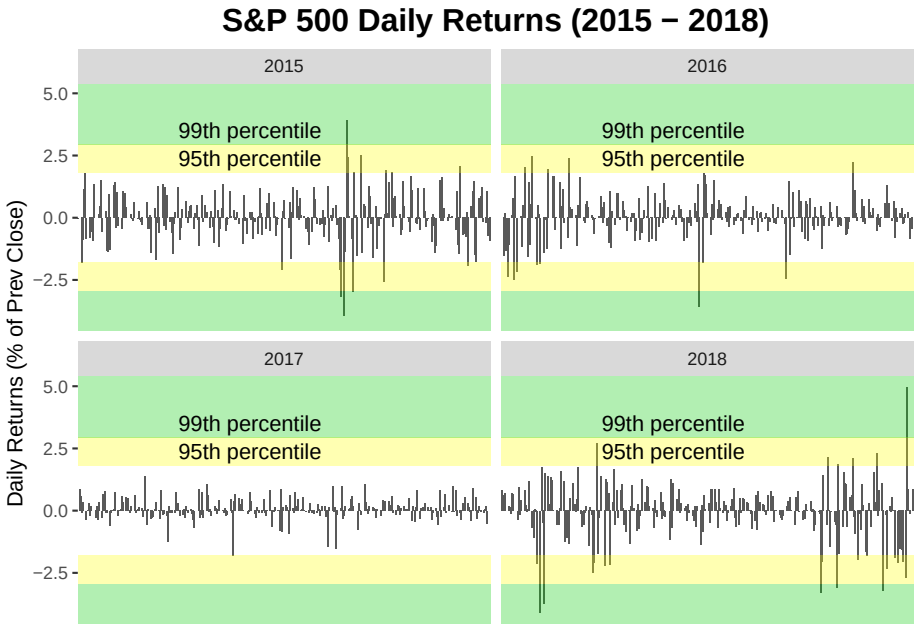
To add the text, *annotate* can be used again. This time with the *geom* argument set to "text".

```
sp_plt <- sp_plt +
  annotate("text", label = "95th percentile", y = (pct95+pct99)/2, x = "06-01" ) +
  annotate("text", label = "99th percentile", y = pct99 + (pct99-pct95)/2, x = "06-01")

sp_plt
```

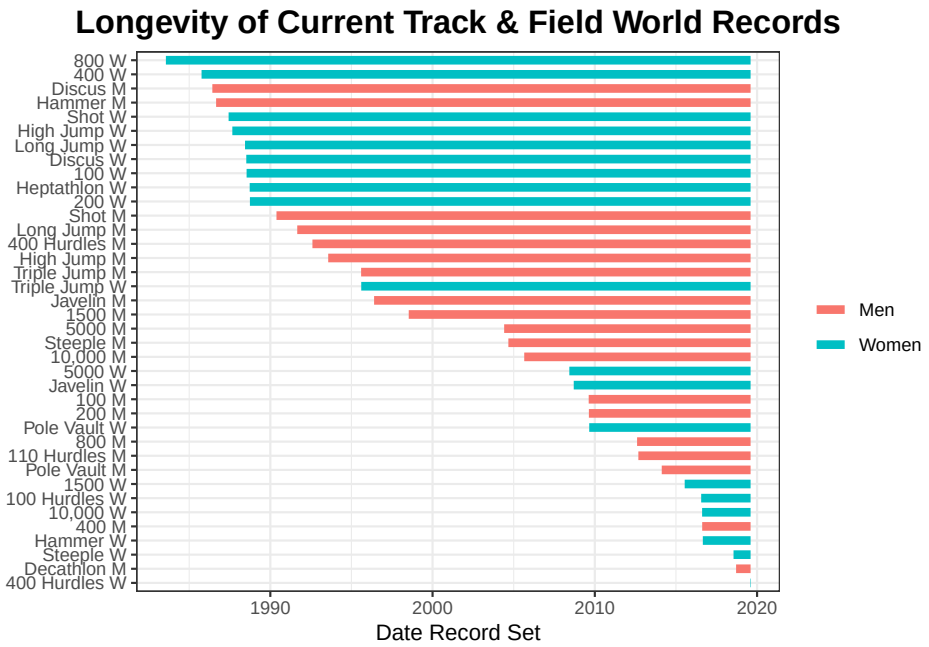


The complete code for the plot

```
sp_plt <- ggplot(data = df, aes(x = MonthDay, y = daily_return)) +
  geom_bar(stat = "identity") +
  facet_wrap(~Year) +
  ylab("Daily Returns (% of Prev Close)") +
  ggtitle("S&P 500 Daily Returns (2015 - 2018)") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        panel.background = element_blank(), axis.title.x=element_blank(),
        axis.text.x =  element_blank(), axis.ticks.x = element_blank()) +
  annotate("ribbon", ymin = pct95, ymax = pct99, x = c(-Inf,Inf), alpha = 0.3, fill =
  annotate("ribbon", ymin = pct99, ymax = Inf, x = c(-Inf, Inf), alpha = 0.3, fill = "
  annotate("ribbon", ymax = -pct95, ymin = -pct99, x = c(-Inf,Inf), alpha = 0.3, fill =
  annotate("ribbon", ymax = -pct99, ymin = -Inf, x = c(-Inf, Inf), alpha = 0.3, fill =
  annotate("text", label = "95th percentile", y = (pct95+pct99)/2, x = "06-01" ) +
  annotate("text", label = "99th percentile", y = pct99 + (pct99-pct95)/2, x = "06-01"

sp_plt
```


S&P 500 Daily Returns (2015 − 2018)
2015
2016
2017
2018
99th percentile
95th percentile
Daily Returns (% of Prev Close)
5.0
2.5
0.0
−2.5

World Record Progression



Data

This plot uses the *wr* data frame of the *gcubed* package. The data was originally obtained from Track and Field News.

```
library(gcubed)
head(wr)
```

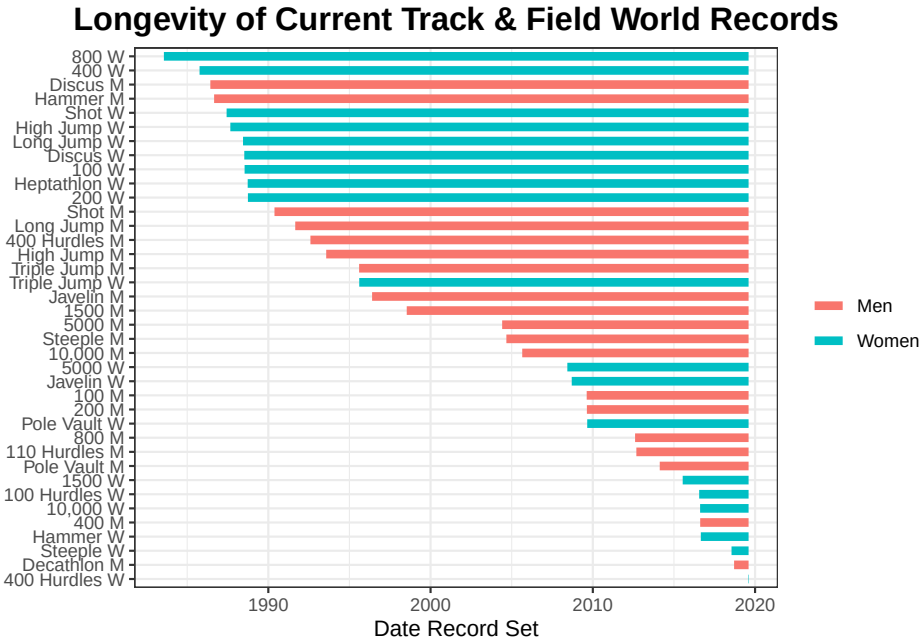
```
## # A tibble: 6 x 7
##   Event     WR       Athlete           Location      Date       MF    MF2
##   <fct>     <chr>    <chr>             <chr>         <date>     <chr> <chr>
```

```
## 1 400 Hurd~ 52.2     Dalilah Muhammad~ Des Moines, ~ 2019-07-28 W     Women
## 2 Decathlo~ 9126     Kevin Mayer (Fra~ Talence, Fra~ 2018-09-16 M     Men
## 3 Steeple W 08:44.3  Beatrice Chepkoe~ Fontvieille,~ 2018-07-20 W     Women
## 4 Hammer W  82.98 |~ Anita Włodarczyk~ Warsaw, Pola~ 2016-08-28 W     Women
## 5 400 M     43.03    Wayde van Nieker~ Rio de Janei~ 2016-08-14 M     Men
## 6 10,000 W  29:17.4  Almaz Ayana (Eth~ Rio de Janei~ 2016-08-12 W     Women
```

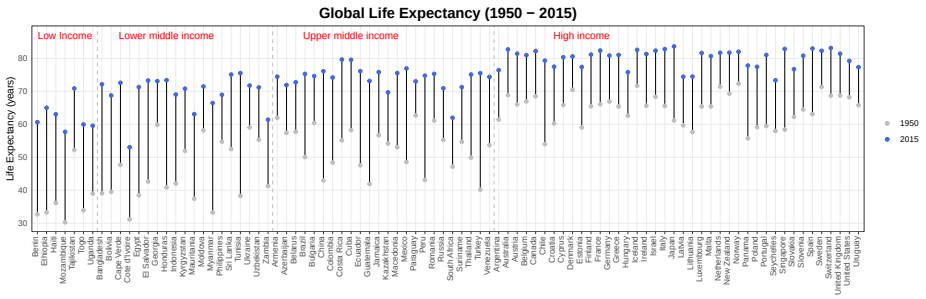
Code for plot

This plot uses *geom_segment* geometry.

```
library(ggplot2)
library(dplyr) # used for arrange which sorts data by date

today <- as.Date("2019-08-08")
wr <- arrange(wr, desc(Date))
wr$Event <- factor(wr$Event, levels = wr$Event)
wr$MF2 <- ifelse(wr$MF == "M", "Men", "Women")

wr_plt <- ggplot(wr, aes(x = Date, y = Event)) +
  geom_segment(aes(x = Date, xend = today, y = Event, yend = Event, colour = MF2), size
  ggtitle("Longevity of Current Track & Field World Records") +
  theme_bw() +
  theme(legend.title = element_blank(),
        axis.title.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5)) +
  xlab("Date Record Set")

wr_plt
```


Longevity of Current Track & Field World Records
800 W
400 W
Discus M
Hammer M
Shot W
High Jump W
Long Jump W
Discus W
100 W
Heptathlon W
200 W
Shot M
Long Jump M
400 Hurdles M
High Jump M
Triple Jump M
Triple Jump W
Javelin M
1500 M
5000 M
Steeple M
10,000 M
5000 W
Javelin W
100 M
200 M
Pole Vault W
800 M
110 Hurdles M
Pole Vault M
1500 W
100 Hurdles W
10,000 W
400 M
Hammer W
Steeple W
Decathlon M
400 Hurdles W
1990
2000
2010
2020
Date Record Set
Men
Women

Life Expectancy for Selected Countries



Data

This plot uses the *life_ex* and *regions* data frames of the *gcubed* package. This data was originally obtained from Our World in Data.

First we join the two tables and select only the life expectancy values for 1950 and 2015.

```
library(gcubed)
library(dplyr)
le <- inner_join(life_ex, regions, by = c("Entity" = "country")) %>%
  filter(Year == 1950 | Year == 2015)

head(le)
```

```
## # A tibble: 6 x 5
##   Entity    Code   Year    LE incomegroup
##   <chr>     <chr> <dbl> <dbl> <chr>
## 1 Argentina ARG    1950  61.4 High income
## 2 Argentina ARG    2015  76.4 High income
```

```
## 3 Armenia   ARM    1950  62.0 Upper middle income
## 4 Armenia   ARM    2015  74.4 Upper middle income
## 5 Australia AUS    1950  68.8 High income
## 6 Australia AUS    2015  82.7 High income
```

Then create a factor variable for the income levels and also order the countries alphabetically within each income group.

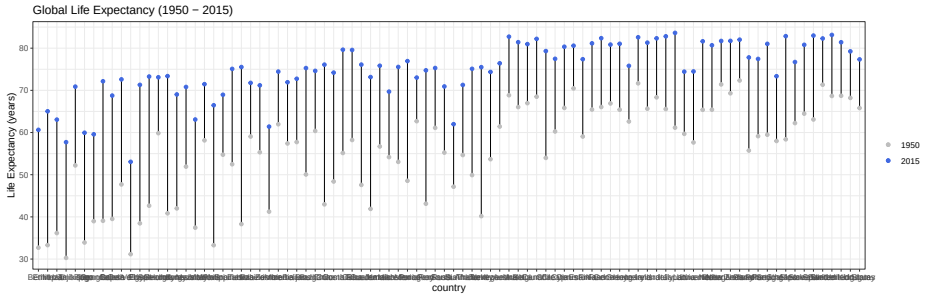
```
library(dplyr)
library(tidyr)

income_levels <- c("Low income", "Lower middle income",
                           "Upper middle income", "High income")
le$incomegroup <- factor(le$incomegroup, levels = income_levels)


le <- le %>%  spread(key = Year, value = LE) %>%
  arrange(incomegroup, Entity)

country_levels <- le$Entity
le$country <- factor(le$Entity, levels = country_levels)

head(le)
```

```
## # A tibble: 6 x 6
##   Entity     Code  incomegroup `1950` `2015` country
##   <chr>      <chr> <fct>        <dbl>  <dbl> <fct>
## 1 Benin      BEN   Low income    32.7   60.6 Benin
## 2 Ethiopia   ETH   Low income    33.3   65.0 Ethiopia
## 3 Haiti      HTI   Low income    36.2   63.1 Haiti
## 4 Mozambique MOZ   Low income    30.3   57.7 Mozambique
## 5 Tajikistan TJK   Low income    52.2   70.9 Tajikistan
## 6 Togo       TGO   Low income    33.9   59.9 Togo
```

Code for plot

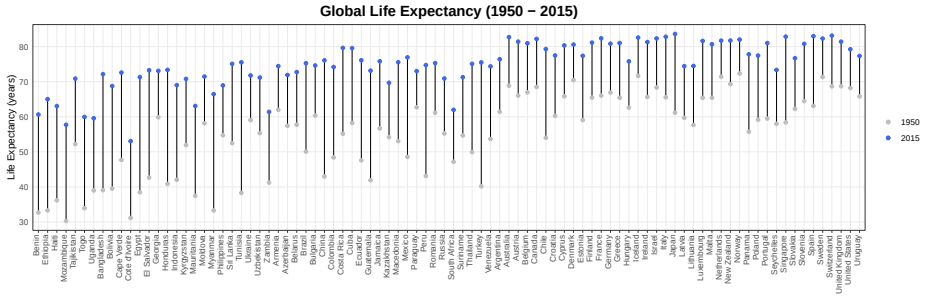
```
library(ggplot2)


le_plt <- ggplot() +
  geom_segment(data=le, mapping=aes(x=country, xend=country,
                                    y = `1950`, yend=`2015`) ) +
  geom_point(data = le, aes(x = country, y = `1950`, colour = "1950")) +
```

```
  geom_point(data = le, aes(x = country, y = `2015`, colour = "2015")) +
  theme_bw() +
  scale_x_discrete(labels=country_levels)+
  ylab("Life Expectancy (years)") +
  ggtitle("Global Life Expectancy (1950 - 2015)") +
  scale_colour_manual(values = c("grey", "royalblue"), name = "")

le_plt
```



Cleaning up the theme elements a little:

```
le_plt <- le_plt +
    theme(axis.title.x=element_blank(),
        axis.text.x = element_text(hjust = 1, angle = 90, vjust=0.1),
        axis.ticks.y = element_blank(),
        panel.grid.minor.y = element_blank(),
        legend.title = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

le_plt
```



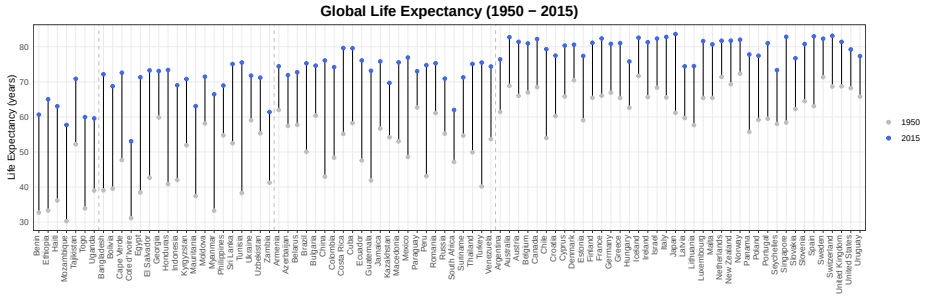
Figuring out where the income groups end (along the x-axis).

```
cumsum(table(le$incomegroup))
```

```
##          Low income Lower middle income Upper middle income
##                   7                   26                   50
##         High income
##                  90
```

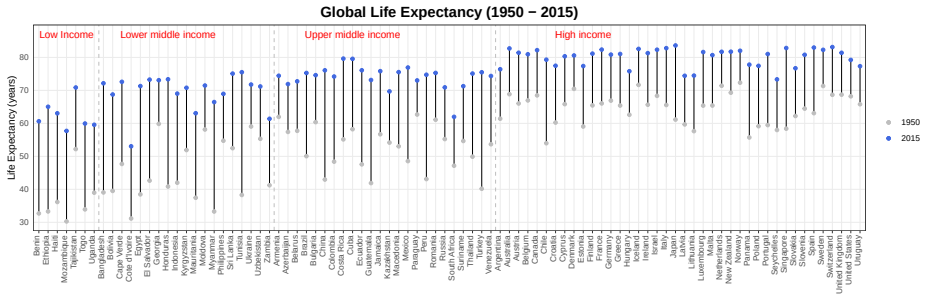
Adding dividing lines between the income groups using *geom_vline*:

```
le_plt <- le_plt +
  geom_vline(xintercept = 7.5, linetype = "dashed", colour = "grey") +
  geom_vline(xintercept = 26.5, linetype = "dashed", colour = "grey") +
  geom_vline(xintercept = 50.5, linetype = "dashed", colour = "grey")

le_plt
```

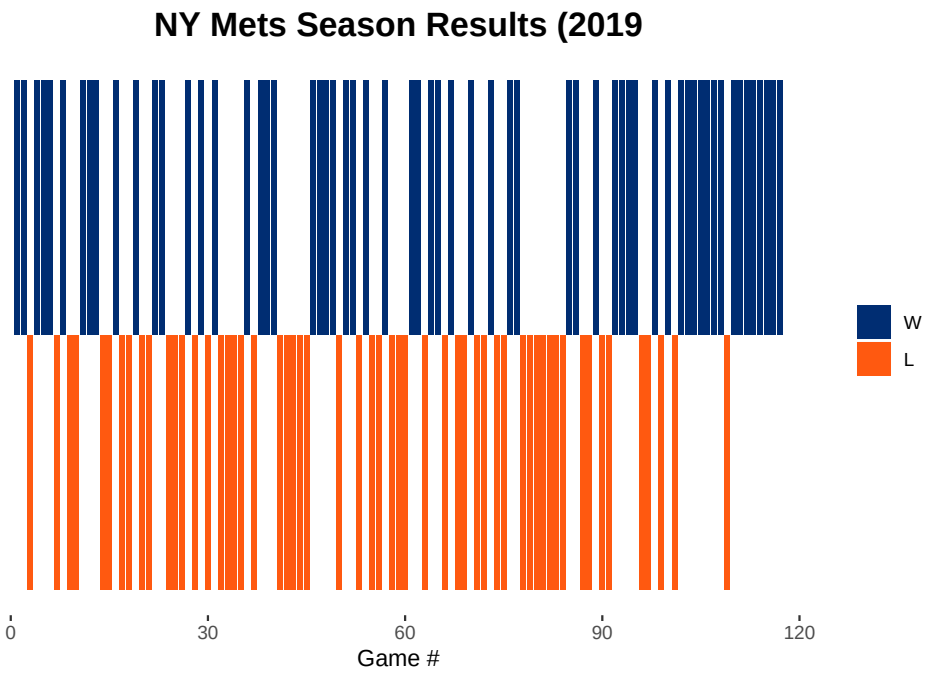


Adding the text for the income groups using *geom_text*:

```
le_plt <- le_plt +
  geom_text(aes(4,87,label = "Low Income"), colour = "red") +
  geom_text(aes(15,87,label = "Lower middle income"), colour = "red") +
  geom_text(aes(35,87,label = "Upper middle income"), colour = "red") +
  geom_text(aes(60,87,label = "High income"), colour = "red")

le_plt
```



NY Mets 2019 Season



Data

This plot uses the *nym* data frame of the *gcubed* package. The original data was obtained from Baseball Reference.

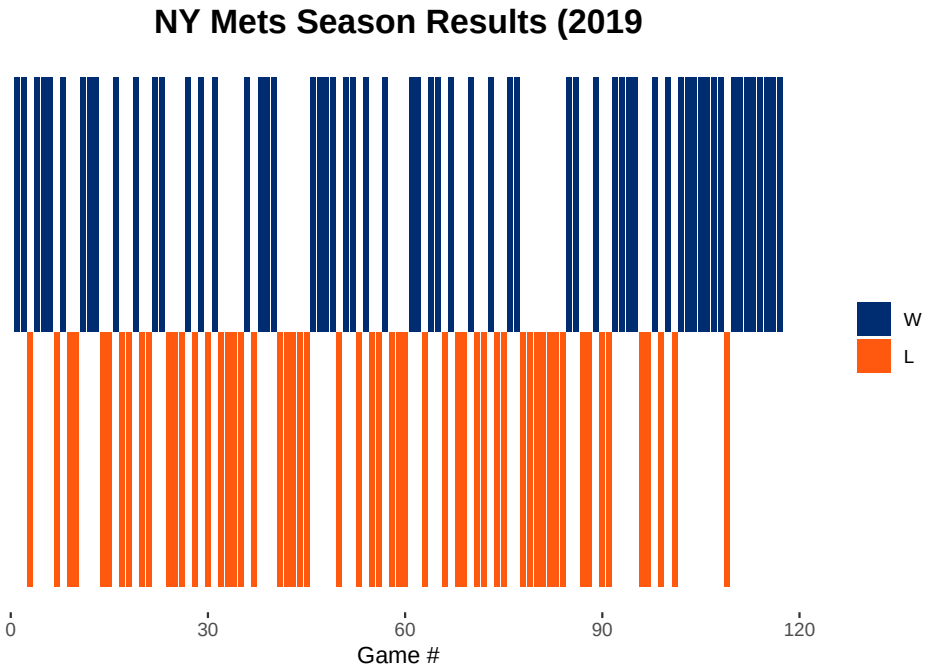
```
library(gcubed)
head(nym)
```

```
## # A tibble: 6 x 26
##   `Gm#` Date  Tm    HomeAway Opp   WL    wo        R    RA   Inn `W-L`
##   <dbl> <chr> <chr> <chr>    <chr> <chr> <chr> <dbl> <dbl> <dbl> <chr>
```

```
## 1     1 Thur~ NYM   @        WSN   W     <NA>      2     0    NA 1-0
## 2     2 Satu~ NYM   @        WSN   W     <NA>     11     8    NA 2-0
## 3     3 Sund~ NYM   @        WSN   L     wo        5     6    NA 2-1
## 4     4 Mond~ NYM   @        MIA   W     <NA>      7     3    NA 3-1
## 5     5 Tues~ NYM   @        MIA   W     <NA>      6     5    NA 4-1
## 6     6 Wedn~ NYM   @        MIA   W     <NA>      6     4    NA 5-1
## # ... with 15 more variables: Rank <dbl>, Win <chr>, Loss <chr>,
## #   Save <chr>, Time <drtn>, `D/N` <chr>, Attendance <dbl>, Streak <chr>,
## #   `Orig. Scheduled` <lgl>, win_updown <dbl>, games_updown <dbl>,
## #   played <dbl>, wins <dbl>, tot_played <dbl>, tot_wins <dbl>
```

Code for plot

```
library(ggplot2)

mets_plt <- ggplot(nym, aes(x = `Gm#`, y = ifelse(WL == "W", 1,-1), fill = WL) ) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("#FF5910", "#002D72")) +
  xlab("Game #") +
  ggtitle("NY Mets Season Results (2019") +
  theme(panel.background = element_blank(),
        axis.title.y = element_blank(),
        legend.title = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5)) +
  guides(fill = guide_legend(reverse=TRUE))

mets_plt
```


NY Mets Season Results (2019
W
L
0
30
60
90
120
Game #

NL East 2019 Season Records

Data

This plot uses the *atl*, *phi*, *was*, *nym* and *mia* data frames of the *gcubed* package. The original data was obtained from Baseball Reference.

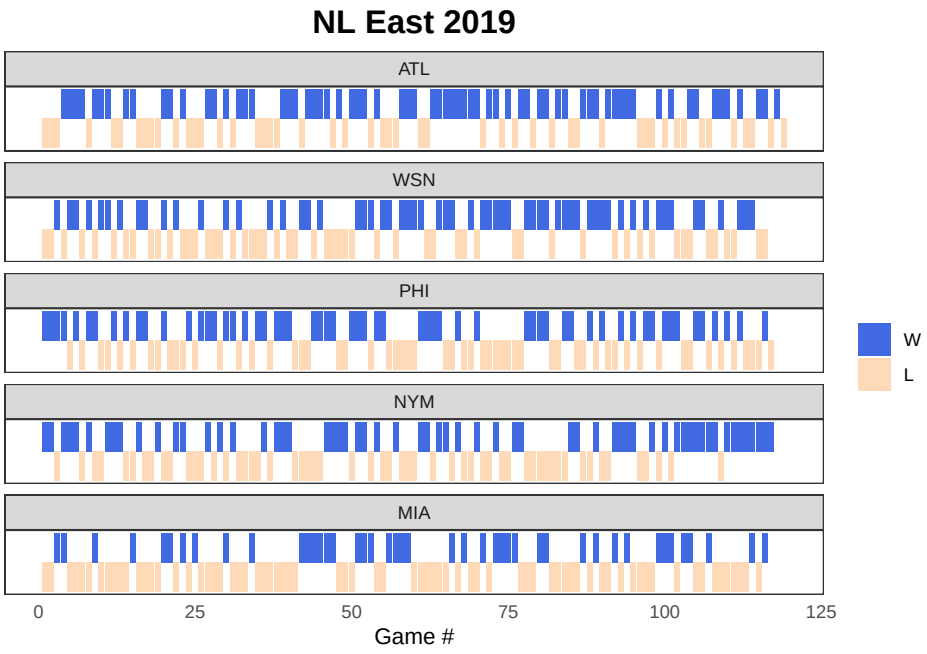
```
library(gcubed)
head(nleast)
```

```
## # A tibble: 6 x 26
##   `Gm#` Date  Tm    HomeAway Opp   WL    wo        R    RA   Inn `W-L`
##   <dbl> <chr> <chr> <chr>    <chr> <chr> <chr> <dbl> <dbl> <dbl> <chr>
## 1     1 Thur~ ATL   @        PHI   L     <NA>      4    10    NA 0-1
## 2     2 Satu~ ATL   @        PHI   L     <NA>      6     8    NA 0-2
## 3     3 Sund~ ATL   @        PHI   L     <NA>      1     5    NA 0-3
## 4     4 Mond~ ATL   <NA>     CHC   W     <NA>      8     0    NA 1-3
## 5     5 Wedn~ ATL   <NA>     CHC   W     <NA>      6     4    NA 2-3
## 6     6 Thur~ ATL   <NA>     CHC   W     <NA>      9     4    NA 3-3
## # ... with 15 more variables: Rank <dbl>, Win <chr>, Loss <chr>,
## #   Save <chr>, Time <drtn>, `D/N` <chr>, Attendance <dbl>, Streak <chr>,
## #   `Orig. Scheduled` <lgl>, win_updown <dbl>, games_updown <dbl>,
## #   played <dbl>, wins <dbl>, tot_played <dbl>, tot_wins <dbl>
```

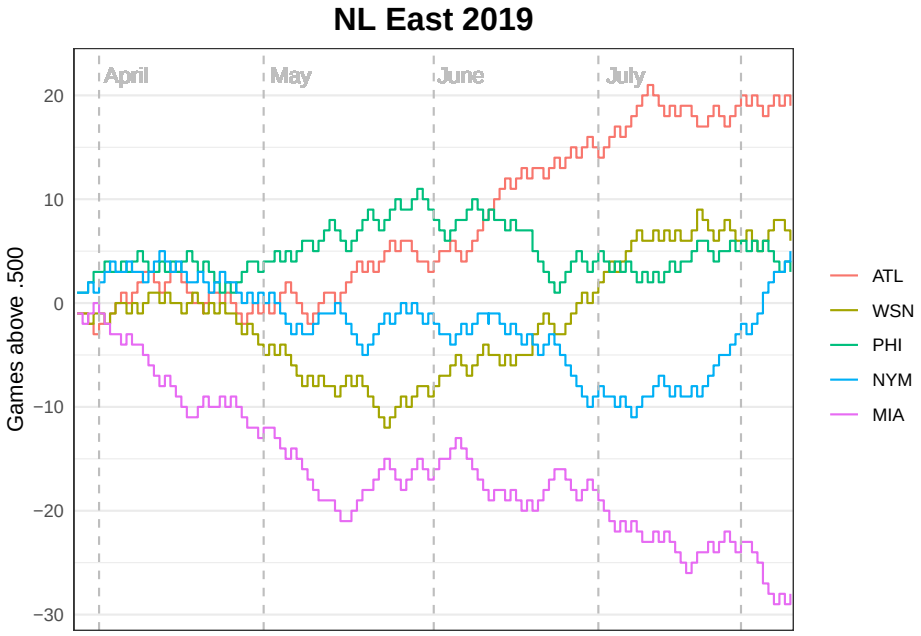
Code

```
nleast$Tm <- factor(nleast$Tm, levels = c("ATL", "WSN", "PHI", "NYM", "MIA"))
nlplot <- ggplot(data = nleast, aes(x = `Gm#`, y = win_updown,
                          fill = factor(WL, levels = c("W", "L")))) +
  geom_bar(stat = "identity") +
  facet_wrap(~Tm, ncol = 1) +
```

```
  scale_fill_manual(values = c("royalblue", "peachpuff"), name = "") +
  theme_bw() +
  ggtitle("NL East 2019") +
  xlab("Game #") +
  theme(axis.title.y=element_blank(),
        #axis.text.x=element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks.y = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

nlplot
```



NL East Teams Games above .500 in 2019



Data

This plot uses the *nleast* data frame of the *gcubed* package. The original data was obtained from Baseball Reference. Some data wrangling has been done to add the last 6 columns. In particular the *games_updown* variable is the y-variable in this plot (the number of wins - the number of losses for team in that row after the game).

A small subset of the data can be seen below:

```
library(gcubed)
nleast[c(1,125,250,400,500), c(2,3,22)]
```

```
## # A tibble: 5 x 3
##   Date             Tm    games_updown
##   <chr>            <fct>        <dbl>
## 1 Thursday Mar 28  ATL             -1
## 2 Friday Apr 5     PHI              4
## 3 Sunday Apr 14    WSN              0
## 4 Wednesday May 22 NYM             -2
## 5 Friday May 3     MIA            -13
```

Some data wrangling code

First split the date into *Weekday*, *Month* and *Day*.

```
library(tidyr)
library(dplyr)

Month = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov",

month_num_table <- data.frame(Month, MonthNum = 1:12)

df <- separate(nleast, Date, into = c("WeekDay", "Month", "Day"), sep = " ") %>%
  mutate(Day = as.integer(Day))
```

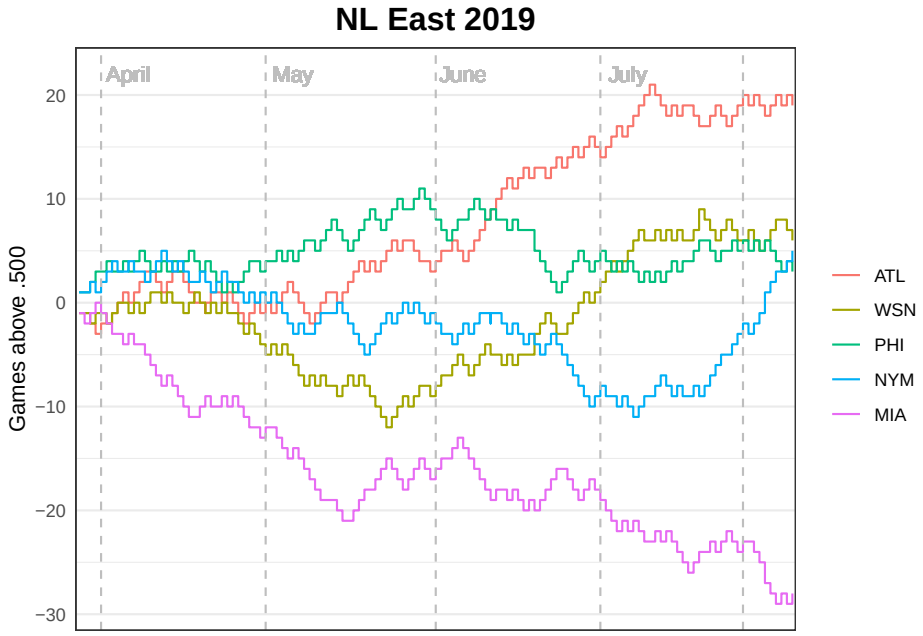
Create a variable named *MonthDay* to sort the games by:

```
df <- left_join(df, month_num_table, by = "Month") %>%
  mutate(MonthNum = sprintf("%02d", as.numeric(MonthNum)),
         Day = sprintf("%02d", as.numeric(Day)),
         MonthDay = paste(MonthNum, Day, sep = "-")) %>%
  arrange(MonthDay)
```

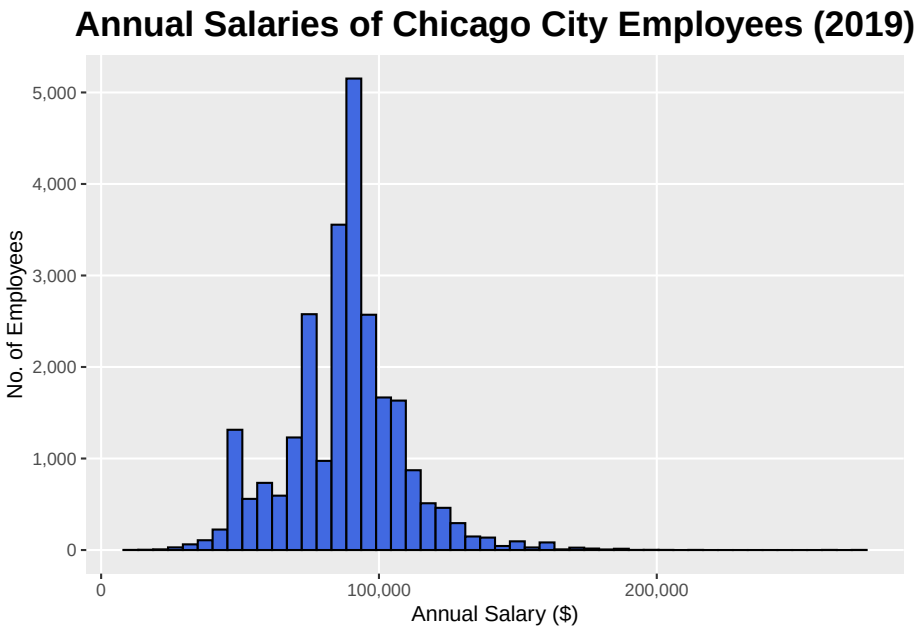
Code for plot

```
library(ggplot2)

df$MonthDay <- factor(df$MonthDay)
games_500_plot <- ggplot(data = df, aes(x = MonthDay, y = games_updown, group = Tm, col
  geom_step() +
```

```
  theme_bw() +
  geom_vline(xintercept = 5, linetype = "dashed", colour = "grey") +
  geom_vline(xintercept = 35, linetype = "dashed", colour = "grey") +
  geom_vline(xintercept = 66, linetype = "dashed", colour = "grey") +
  geom_vline(xintercept = 96, linetype = "dashed", colour = "grey") +
  geom_vline(xintercept = 122, linetype = "dashed", colour = "grey") +
  geom_text(aes(10,22,label = "April"), colour = "grey") +
  geom_text(aes(40,22,label = "May"), colour = "grey") +
  geom_text(aes(71,22,label = "June"), colour = "grey") +
  geom_text(aes(101,22,label = "July"), colour = "grey") +
  ylab("Games above .500") +
  ggtitle("NL East 2019") +
  theme(axis.title.x=element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.y = element_blank(),
        panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        legend.title = element_blank(),
        plot.title = element_text(size = 16, face = "bold", hjust = 0.5))


games_500_plot
```



Chicago Employee Salaries



Data

This plot uses the *chi_emps* data set of the *gcubed* package. The original source of the data is the City of Chicago's Data Portal[5].

First filter the data down to only the salaried employees.

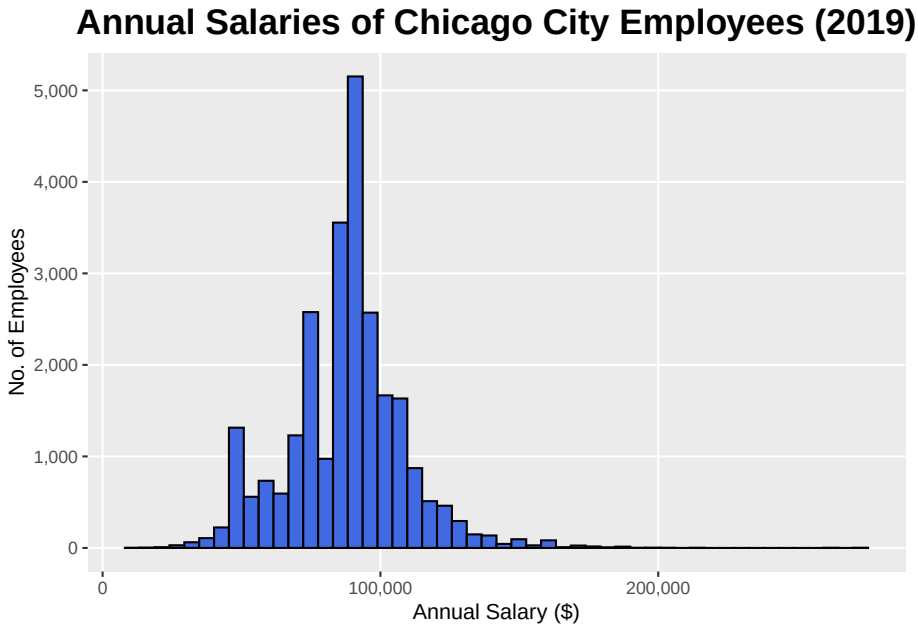
```
library(gcubed)
library(dplyr)
df <- filter(chi_emps, SalHour == "Salary")
dim(df)
```

[5]The data was current as of July 2019

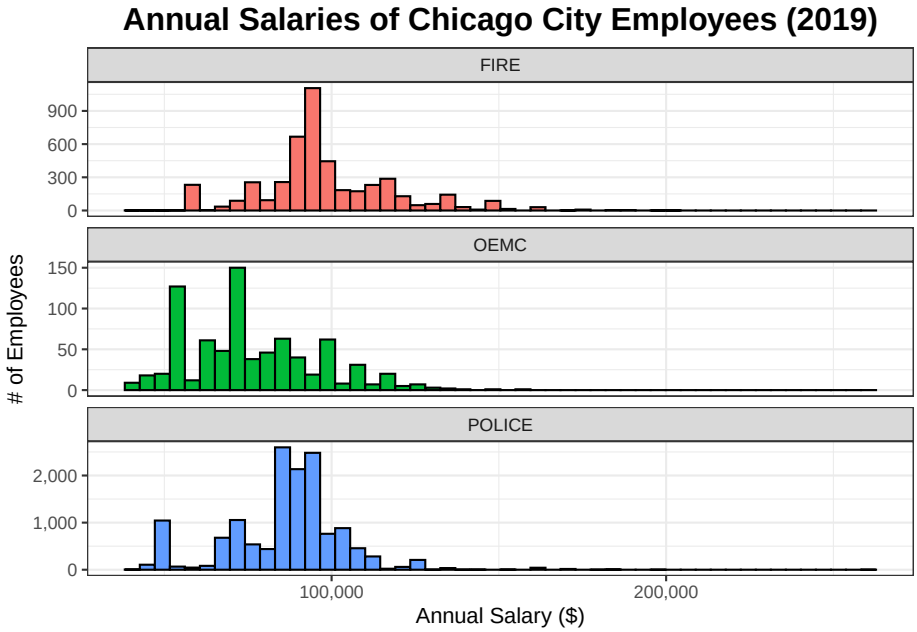
```
## [1] 25741     8
```

Code for plot

```
library(ggplot2)

chi_emps_hist <- ggplot(df, aes(x = AnnualSalary)) +
  geom_histogram(fill = I("royalblue"), colour = I("black"), bins = 50) +
  scale_y_continuous(label = comma) +
  scale_x_continuous(label = comma) +
  xlab("Annual Salary ($)") +
  ylab("No. of Employees") +
  ggtitle("Annual Salaries of Chicago City Employees (2019)") +
  theme(panel.grid.minor = element_blank(),
        plot.title = element_text(size = 18, face = "bold", hjust = 0.5))
chi_emps_hist
```



Chicago Employee Salary: Comparison using histograms



Data

This plot uses the *chi_emps* data frame from package *gcubed*. The original source of the data is the City of Chicago's Data Portal[6].

First, find the 3 departments with the most salaried employees.

[6]The data was current as of July 2019

```
library(gcubed)
library(dplyr)
df <- filter(chi_emps, SalHour == "Salary")
large_dept_names <- names(sort(table(df$Department), decreasing = TRUE))[1:3]
large_dept_names
```

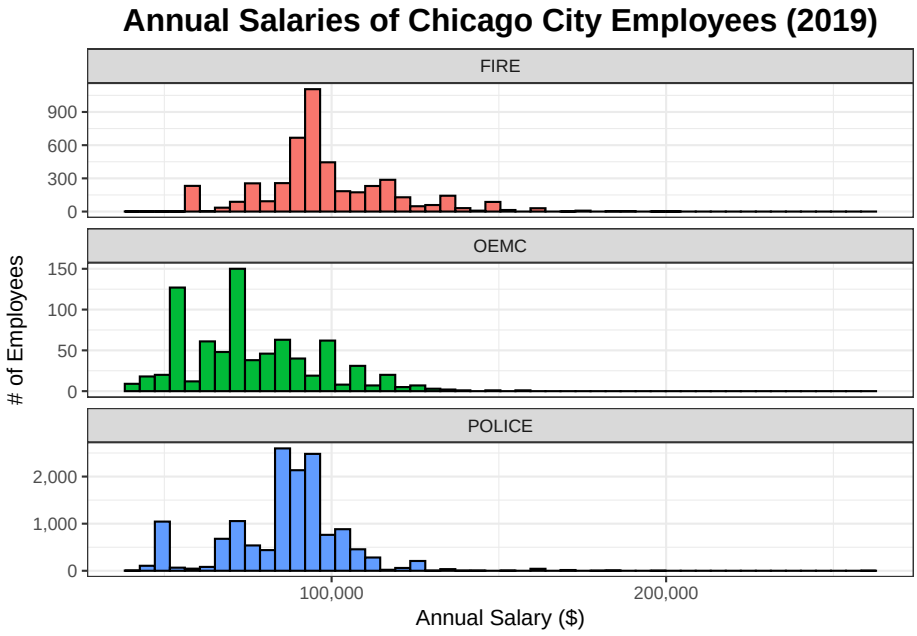
```
## [1] "POLICE" "FIRE"   "OEMC"
```

```
large_depts <- df[df$Department %in% large_dept_names, ]
head(large_depts)
```

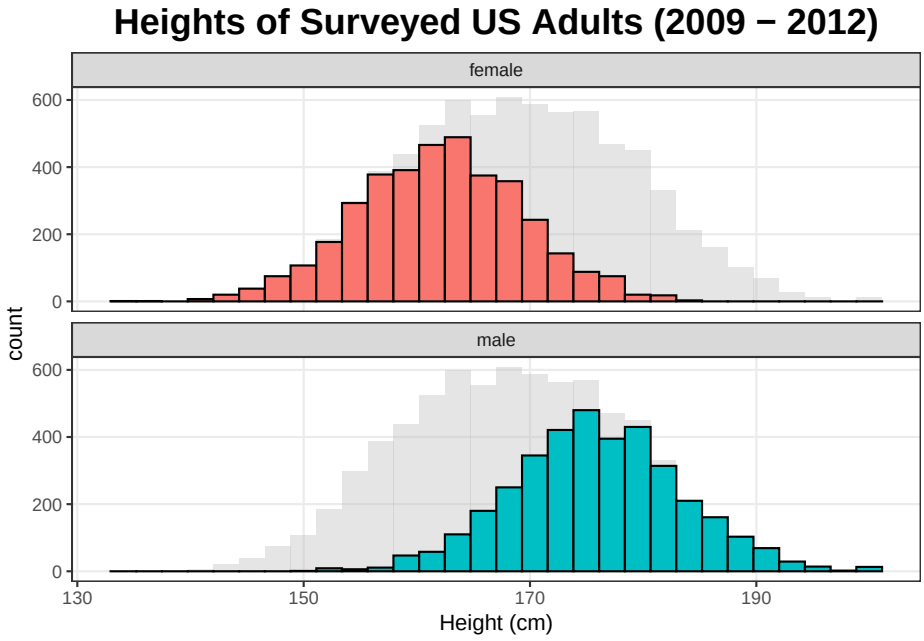
```
## # A tibble: 6 x 8
##   Name  Titles Department FullPart SalHour TypicalHours AnnualSalary
##   <chr> <chr>  <chr>      <chr>    <chr>           <dbl>        <dbl>
## 1 AARO~ SERGE~ POLICE     F        Salary             NA       101442
## 2 AARO~ POLIC~ POLICE     F        Salary             NA        94122
## 3 ABAR~ POLIC~ POLICE     F        Salary             NA        48078
## 4 ABBA~ FIRE ~ FIRE       F        Salary             NA       103350
## 5 ABBA~ POLIC~ POLICE     F        Salary             NA        93354
## 6 ABBO~ POLIC~ POLICE     F        Salary             NA        68616
## # ... with 1 more variable: HourlyRate <dbl>
```

Code for plot

```
library(ggplot2)
library(scales) # to add commas to axis values

chi_comp_plt <- ggplot(large_depts, aes(x = AnnualSalary, fill = Department)) +
         geom_histogram(bins = 50, colour = "black") +
         facet_wrap(~Department, ncol = 1, scales = "free_y") +
  theme_bw() +
  scale_x_continuous(label = comma) +
  scale_y_continuous(label = comma) +
  xlab("Annual Salary ($)") + ylab("# of Employees") +
  ggtitle("Annual Salaries of Chicago City Employees (2019)") +
  theme(legend.position = "none", plot.title = element_text(size = 16, face = "bold", l

chi_comp_plt
```


Annual Salaries of Chicago City Employees (2019)
FIRE
OEMC
POLICE
# of Employees
900
600
300
0
150
100
50
0
2,000
1,000
0
100,000
200,000
Annual Salary ($)

Histogram of NHANES Survey respondents



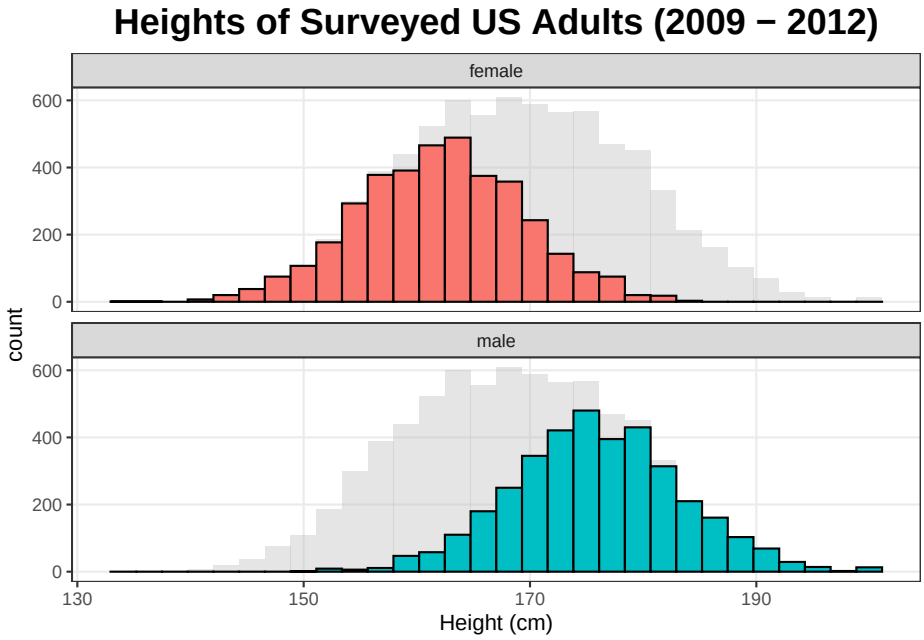
Data

This plot uses the *NHANES* data set contained in the *NHANES* package available on CRAN. Along with many other recorded variables it contains the heights of individuals surveyed between 2009 and 2012.

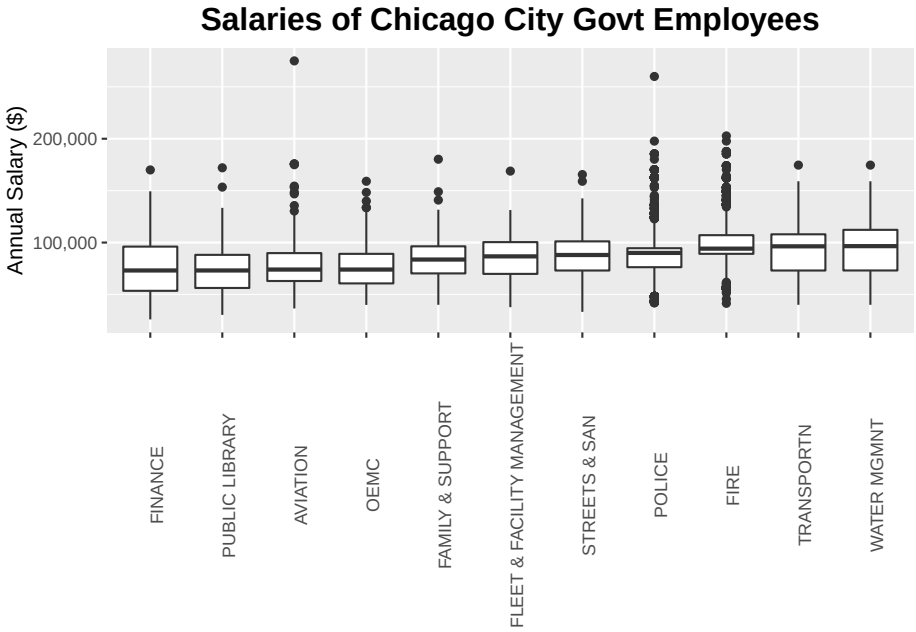
Code for plot

This code uses *geom_histogram*. To get the background histogram, for the entire population, another data frame is used which is the original data frame without the *Gender* variable. Thus there are two *geom_histogram* geometries added to this plot.

```
library(NHANES)
library(dplyr)
library(ggplot2)


NHANES_adults <- filter(NHANES, Age >= 18)
NHANES_bg <- select(NHANES_adults, -Gender)


nhanes_height_plot <- ggplot(data = NHANES_adults, aes(x = Height)) +
  geom_histogram(data = NHANES_bg, fill = "grey", alpha = .4) +
  geom_histogram(mapping = aes(fill = Gender), colour = "black") +
  facet_wrap(~ Gender, ncol = 1) +
  guides(fill = FALSE) +  # to remove the legend
  theme_bw() + xlab("Height (cm)") + ggtitle("Heights of Surveyed US Adults (2009 - 20
  theme(panel.grid.minor = element_blank(),
        plot.title = element_text(size = 18, face = "bold", hjust = 0.5))


nhanes_height_plot
```


Heights of Surveyed US Adults (2009 − 2012)
female
male
count
600
400
200
0
130
150
170
190
Height (cm)

Chicago Employee Salary Box plot



Data

This plot uses the *chi_emps* data set of the *gcubed* package. The original source of the data is the City of Chicago's Data Portal[7].

First we identify the departments with 500 or more employees. Then we restrict the data set to those employees that are salaried[8].

[7]The data was current as of July 2019

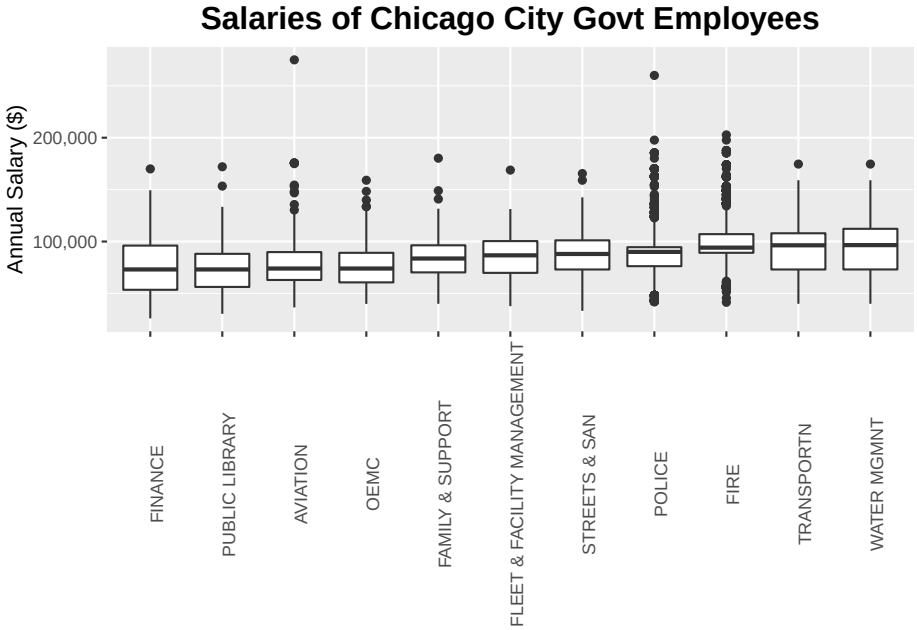
[8]This classification is not exactly the same as used in some of the other example plots using the same data set.

```
library(gcubed)
library(dplyr)

dept_counts <- table(chi_emps$Department)
large_dept_counts <- dept_counts[dept_counts >= 500 ]
large_dept_names <- names(large_dept_counts)

large_depts <- chi_emps[chi_emps$Department %in% large_dept_names & chi_emps$SalHour ==


sorted_depts <- group_by(large_depts, Department) %>%
  summarise(MedSal = median(AnnualSalary)) %>%
  arrange(MedSal)

large_depts$Department <- factor(large_depts$Department, levels = sorted_depts$Departme
```

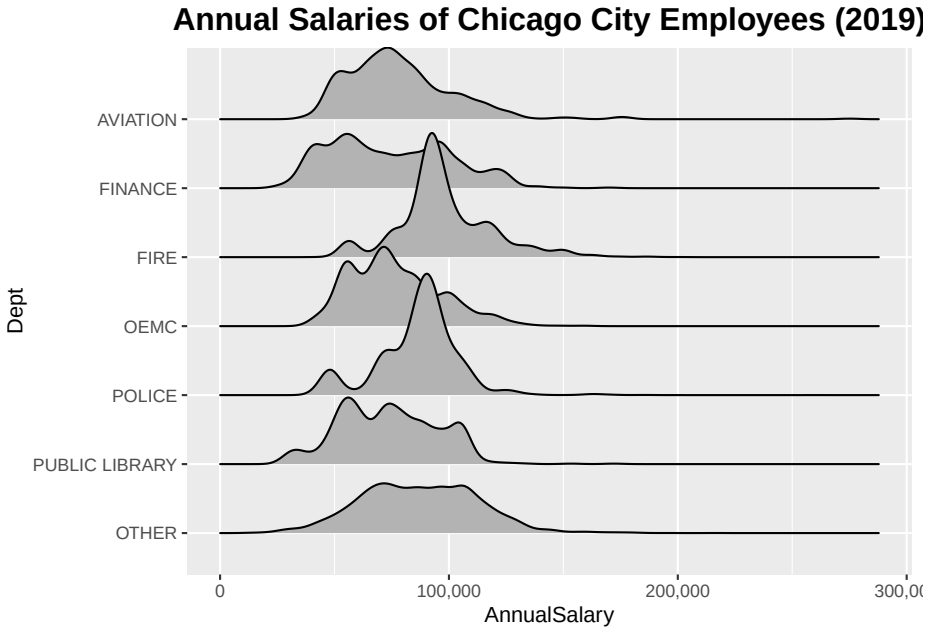
Code for plot

This plot uses the *geom_boxplot* geometry.

```
library(ggplot2)
library(scales)


chi_dept500_boxplot <- ggplot(data = large_depts, aes(x = Department, y = AnnualSalary)
  geom_boxplot() +
  ggtitle("Salaries of Chicago City Govt Employees") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5),
        axis.text.x = element_text(angle = 90),
        axis.title.x=element_blank()) +
  ylab("Annual Salary ($)") +
  scale_y_continuous(label = comma)

chi_dept500_boxplot
```


Salaries of Chicago City Govt Employees
Annual Salary ($)
200,000
100,000
FINANCE
PUBLIC LIBRARY
AVIATION
OEMC
FAMILY & SUPPORT
FLEET & FACILITY MANAGEMENT
STREETS & SAN
POLICE
FIRE
TRANSPORTN
WATER MGMNT

Chicago City Salaries Compared: Density Ridges



Data

This plot uses the *chi_emps* data set of the *gcubed* package. The original source of the data is the City of Chicago's Data Portal[9].

First, get the departments that have more than 500 salaried employees:

[9]The data was current as of July 2019

```
library(gcubed)

df <- chi_emps[chi_emps$SalHour == "Salary", ]
dept_counts <- table(df$Department)
large_dept_names <- names(dept_counts[dept_counts > 500])
large_dept_names
```

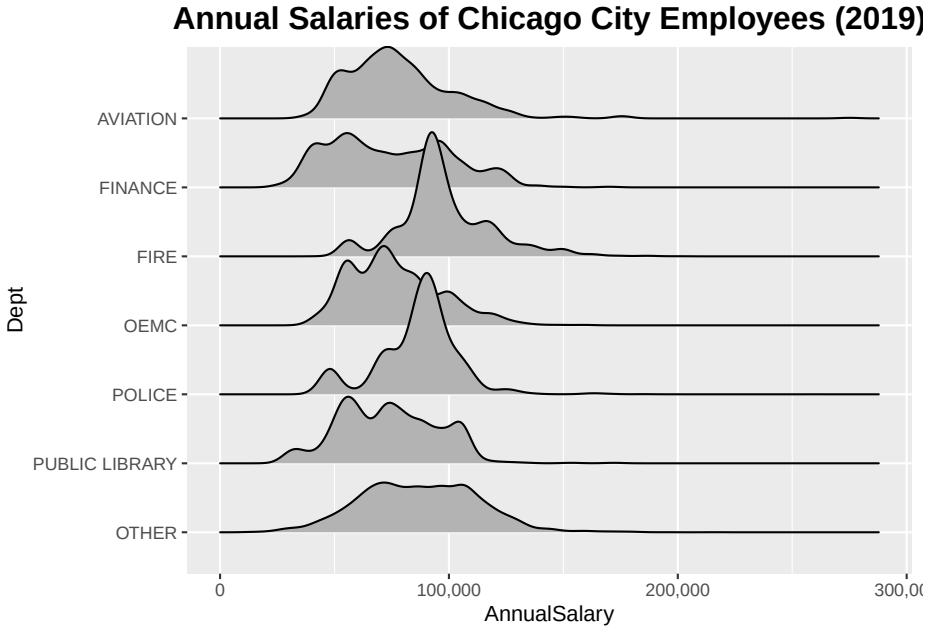
```
## [1] "AVIATION"       "FINANCE"        "FIRE"           "OEMC"
## [5] "POLICE"         "PUBLIC LIBRARY"
```

```
df$Dept <- ifelse(df$Department %in% large_dept_names, df$Department, "OTHER")
table(df$Dept)
```

```
##
##       AVIATION        FINANCE           FIRE           OEMC          OTHER
##            583            534           4631            799           4432
##         POLICE PUBLIC LIBRARY
##          14060            702
```

Code for plot

```
dept_levels <- c("OTHER", rev(large_dept_names))
df$Dept <- factor(df$Dept, levels = dept_levels)

library(ggridges) # to use geom_density_ridges
library(scales) # to format axis values with commas

chi_ridge_plt <- ggplot(data = df, aes(x = AnnualSalary, y = Dept)) +
  geom_density_ridges() +
  scale_x_continuous(label=comma) +
  ggtitle("Annual Salaries of Chicago City Employees (2019)") +
  theme(plot.title = element_text(size = 16, face = "bold", hjust = 0.5))

chi_ridge_plt
```


Annual Salaries of Chicago City Employees (2019)
Dept
AVIATION
FINANCE
FIRE
OEMC
POLICE
PUBLIC LIBRARY
OTHER
0
100,000
200,000
AnnualSalary

The gcubed package

How do I install the package?

You can install the *gcubed* package by running the following commands.

```
library(devtools)
install_github("thisisdaryn/gcubed")
```

If you don't have the *devtools* package installed, you can do so by running:

```
install.packages("devtools")
```

What's in the package?

The package contains data sets used for plots in the gallery of examples.

I strive to maintain the data in a form that does not require too much manipulation prior to plotting. However there are instances in which I have found it appropriate to include some data manipulation steps for the purpose of accurately representing the practical workflow of using *ggplot2* e.g. I often have to convert variables to factors with orders of my setting to get elements in the graph to be displayed in the right order or the shape of the data used by *ggplot2* may unlikely to be a shape that the data is kept in on an ongoing basis.

I have compiled these data sets from several sources. Presently the data in the sets are collectively sourced from:

- ATP Tour
- Baseball Reference
- Basketball Reference
- City of Chicago Data Portal
- ESPN Cricinfo
- Federal Reserve Economic Data (St. Louis Fed)
- Our World in Data

- Track and Field News
- US Bureau of Labor Statistics
- WTA Tennis

ggplot2

What is ggplot2

ggplot2 is a widely-used R package for data visualisation. I think it provides a good balance between providing aesthetically pleasing plots easily and offering fine-grained ability to control details: you can get most of the way to a great plot very easily.

I do admit, that I often spend most of my time getting the final details right.

Currently, *ggplot2* is my primary tool for producing plots that I wish to share publicly. For more exploratory data visualisation work, I use this package slightly more frequently than the graphical functions in base R.

Learning ggplot2

Introducing *ggplot2* is beyond the ambition of this site. However, there are many resources available to learners. In particular I will point to three that I have firsthand experience with.

- Data Science: Visualization on edX (HarvardX: PH125.2x)
  - I came across this course after several years of using *ggplot2*. In my opinion it gives a very good explanation of the underlying concepts. This is my first choice recommendation to someone learning to use *ggplot2*.
- R for Data Science
  - This book is well-written and easy to follow. While it addresses a wider range of topics, *ggplot2* is covered well at the very beginning of the book . This is not an encyclopedic resource but certainly provides a good platform to subsequently progress to expert-level proficiency. This is the best introduction/coverage of *ggplot2* I have come across in a general purpose R book.
- ggplot2: Elegant Graphics for Data Analysis (Use R!)

- This book is now dated and, in my opinion, has been superceded by the project's website. However it is the resource that allowed me to go from copy-pasting and tweaking code from other plots to building plots from the ground up relatively fluently. I have owned a physical copy of this book for several years and I still consult it regularly. It was written by *ggplot2*'s author, Hadley Wickham when the package was still considered new.