Introduction to Visualization in R

LTC Melanie Vinton

2017-10-05

Contents

Chapter 1

Background

This module is an introduction to visualization in R using the ggplot2 package. These relatively simple examples should provide a baseline of knowledge to allow the user to create far more interesting and informative graphics than spreadsheet based graphing functionality. Many of the most impressive visualizations seen today are made in R with packages like ggplot2.

Chapter 2

Introduction

This module is about creating visualizations in R, using the **ggplot2** package. We will add to the basic information about visualization covered in the pre-course work but continue to focus on bar charts, line charts, and histograms.

- **ggplot2** is among the most useful and widely used for visualization in R.
- Plots are constructed in layers, which can be more intuitive.
- This package is among the most useful and widely used for visualization in R, created by Hadley Wickham.
- There are plotting functions available in base R but **ggplot2** provides a powerful, intuitive framework for creating and customizing visualizations.

Here's some examples: http://www.r-graph-gallery.com/portfolio/ggplot2-package

2.1 ggplot2 Package

The **ggplot2** package is based on the grammar of graphics, which breaks graphics into parts which are controlled separately and combined. Every graph has the same basic components - data, a coordinate system, and geoms that represent data points visually.

First, lets install the packages we will use in this module.

```
library(ggplot2)
library(RColorBrewer)
library(plyr)
library(dplyr)
library(scales)
```

Also, clear your Global Environment.

```
rm(list = ls())
```

2.2 References

Some useful references include:

- http://www.cookbook-r.com
- http://ggplot2.tidyverse.org/reference/

- https://www.rstudio.com/resources/cheatsheets/
- "R Graphics Cookbook" by Winston Chang (PDF can be found online at http://ase.tufts.edu/bugs/guide/assets/R%20Graphics%20Cookbook.pdf)
- http://www.google.com

2.3 Layering

Every graph has the same basic components:

1. Data

- usually a data frame

2. Aesthetic mapping

- how variables in the data frame map to visual objects on the graph
- objects (or aesthetics) to map to include x-axis, y-axis, size, shape, color, transparency
- specified with the `aes()` function

3. Geoms that represent data points visually.

- specifies the type of plot
- geom = geometric object
- examples (see the cheatsheet!): `geom_line()`, `geom_point()`, `geom_bar()`, `geom_histogram()`

Chapter 3

Bar Charts

Bar charts are generally used to display numeric values on the y-axis for different categories or discrete values on the x-axis. Bar charts are not well suited for continuous data on the x-axis because try to make a bar for every possible value of the continuous range.

The heights of the bar on the y-axis can represent counts or values from your data set. The code for the graphic changes depending on which is the case.

3.1 GDELT Data

For the first part of this module we will use the data from the earlier modules from GDELT, http://www.gdeltproject.com. The dataframe is events in Nigeria that involve Chinese actors in January 2017.

If you don't already have it in your environment, open the "subsahara_jan17.csv" file.

```
subsahara_jan17 <- read.csv("subsahara_jan17.csv")
```

Lets explore the QuadClass variable, which indicates one of four types of events.

1. Verbal Cooperation
2. Material Cooperation
3. Verbal Conflict
4. Material Conflict

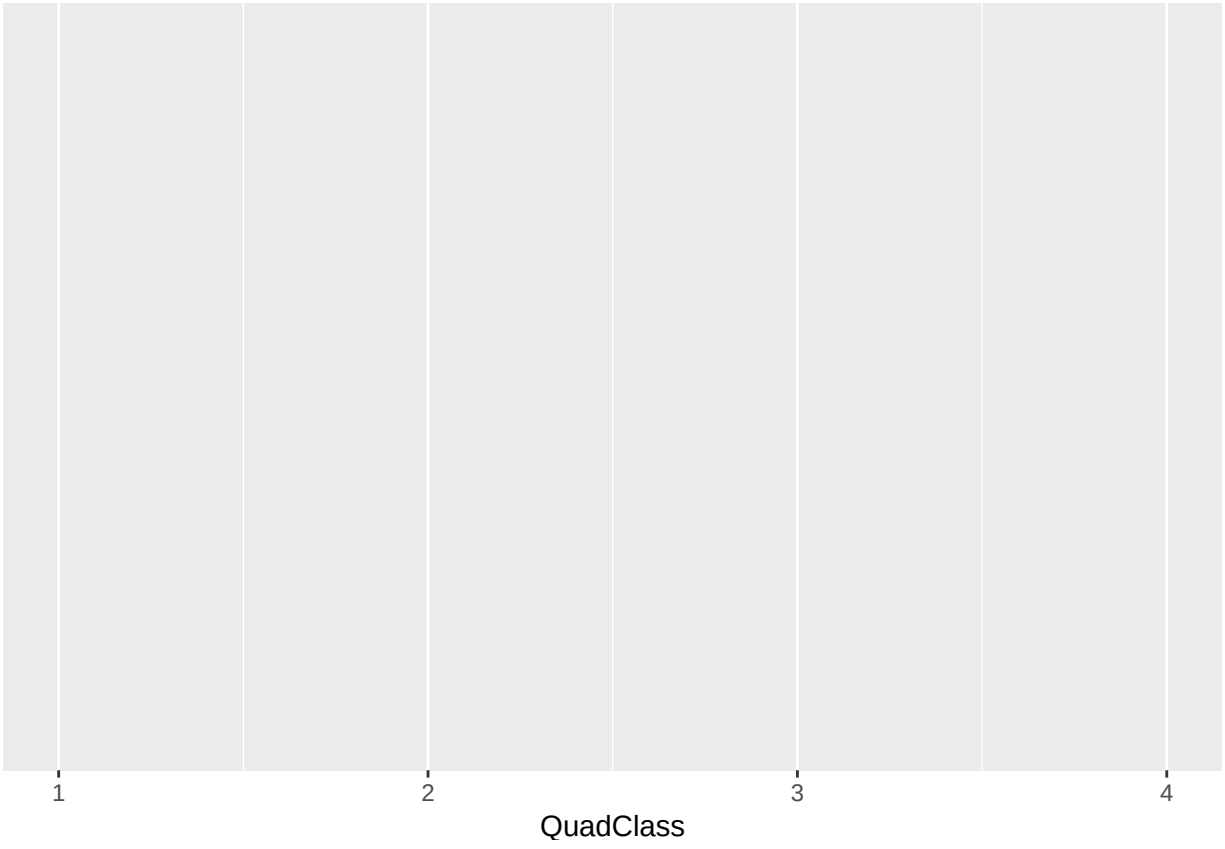
```
summary(subsahara_jan17$QuadClass)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.523   2.000   4.000
```

3.2 Charts with Counts

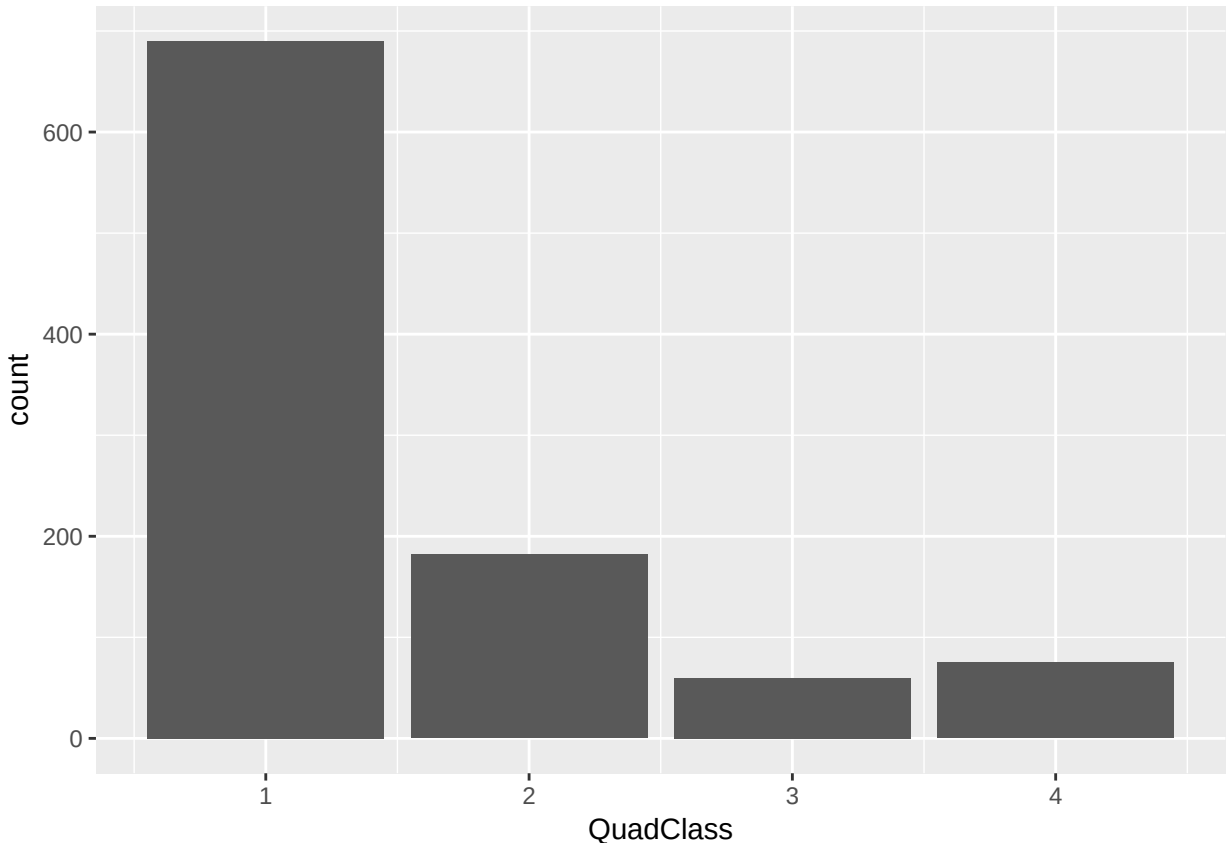
Let's create a bar chart the shows the number of events for each level of the QuadClass variable. First, we start with the *ggplot* function to define the data and basic aesthetics, in this case which variable we want to use. Aesthetics can also include visual properties like size and shape, which we will cover later.

```
ggplot(data = subsahara_jan17, aes(x = QuadClass))
```



Next we need to add a geom function, in this case *geom_bar()*. This determines the basic graphical look.

```
ggplot(data = subsahara_jan17, aes(x = QuadClass)) +
  geom_bar()
```


600
400
200
0
count
1
2
3
4
QuadClass

Chapter 4

Change the Look

Now we want to adjust the look of the chart by applying more aesthetic controls inside the geom function and using themes and scales.

We will change:

- Color of the bars
- Width of the bars.
- Axis labels.
- Chart title.

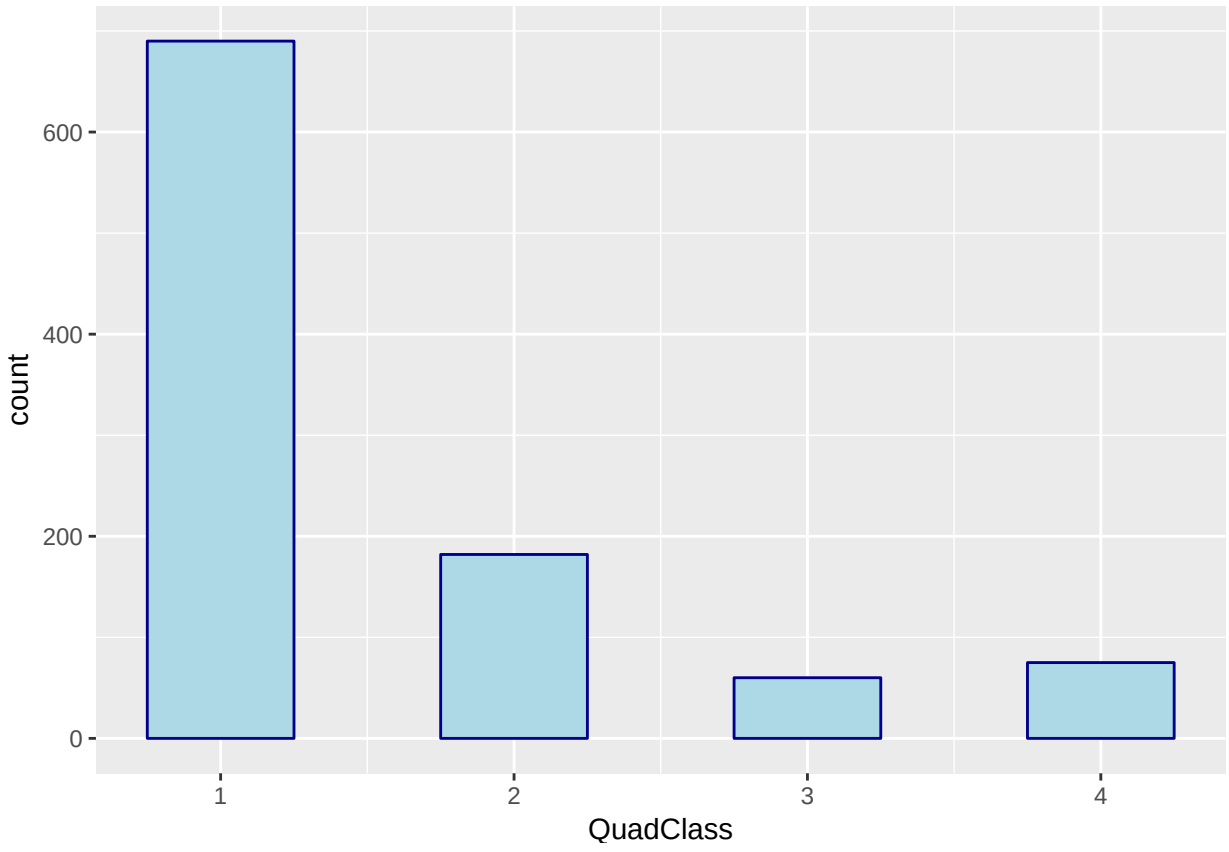
4.1 Colors

- R has over 600 colors built in which you can refer to by name.
- The *colors()* function provides a list of the names.
- You can also select colors by hexadecimal codes, such as "#990000" for deep red.
- R Cookbook Graphs section has a very useful section on colors and palettes.
- Another good reference for color palettes is http://colorbrewer2.org
- The *RColorBrewer* package provides a set of useful color palettes that can be used to customize the color ranges on a graph. Use *display.brewer.all()* to see the set of palettes.
- Note: Sometimes you will see the word "color" spelled "colour" - they are interchangeable in R.

4.2 Changing Bar Colors

- Change the fill color of the bars with `fill = "lightblue"`
- Change the color of the outline of the bars with `color = "darkblue"`
- Change with width of the bars with `width = .5`

```
ggplot(subsahara_jan17, aes(x = QuadClass)) +
  geom_bar(fill = "lightblue", color = "darkblue", width = .5)
```



4.3 Themes

There are many more options for customizing a graph. Adjusting the look involves of two different types of elements, themes and scales.

- Theme elements are non-data elements in the plot such as the title, legend, and axes.
- To change the appearance of a theme element, use the *theme()* function and a corresponding element object.
- Common element objects are text or lines.
- R Graphics Cookbook covers this in section 9.4 very clearly.

Here's an example, changing the color of the y-axis title to red:

```
theme(axis.title.y = element_text(color = "red"))
```

A few other useful theme functions:

- Remove an axis label: `theme(axis.title.x = element_blank())`
- Remove grid lines: `theme(panel.grid.major = element_blank())`
- Rotate x-axis text labels by 90 degrees: `theme(axis.text.x = element_text(angle = 90))`
- Move the position of a legend: `theme(legend.position = "bottom")`

4.4 Scales

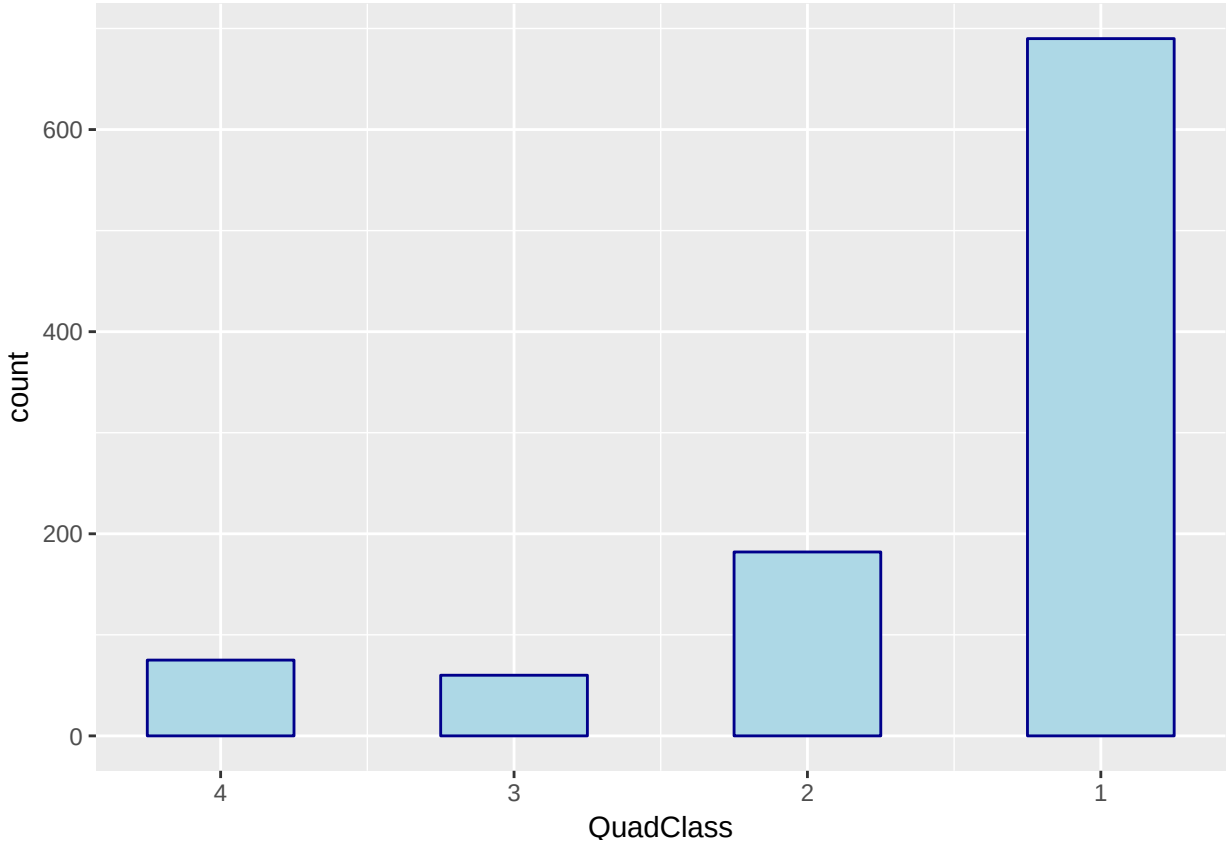
Scale elements map data values to visual values.

- To change the default mapping, a new scale is added.
- The scale function identifies the aesthetic you want to adjust, a pre-packaged scale to use, and the specific arguments for what you are trying to change.
- Different types of data use different types of aesthetics and scales - it is important to match them up correctly.
- The Data Visualization Cheatsheet are a good reference for different types of scales.

Here's an example, using `scale_x_reverse()`:

- "scale" indicates that we want to customize a data element.
- "x" indicates which data element.
- "reverse" is used to override the default linear mapping and reverse the order on the x-axis.

```
ggplot(subsahara_jan17, aes(x = QuadClass)) +
  geom_bar(fill = "lightblue",color = "darkblue", width = .5)+
  scale_x_reverse()
```



4.5 Labels and Titles

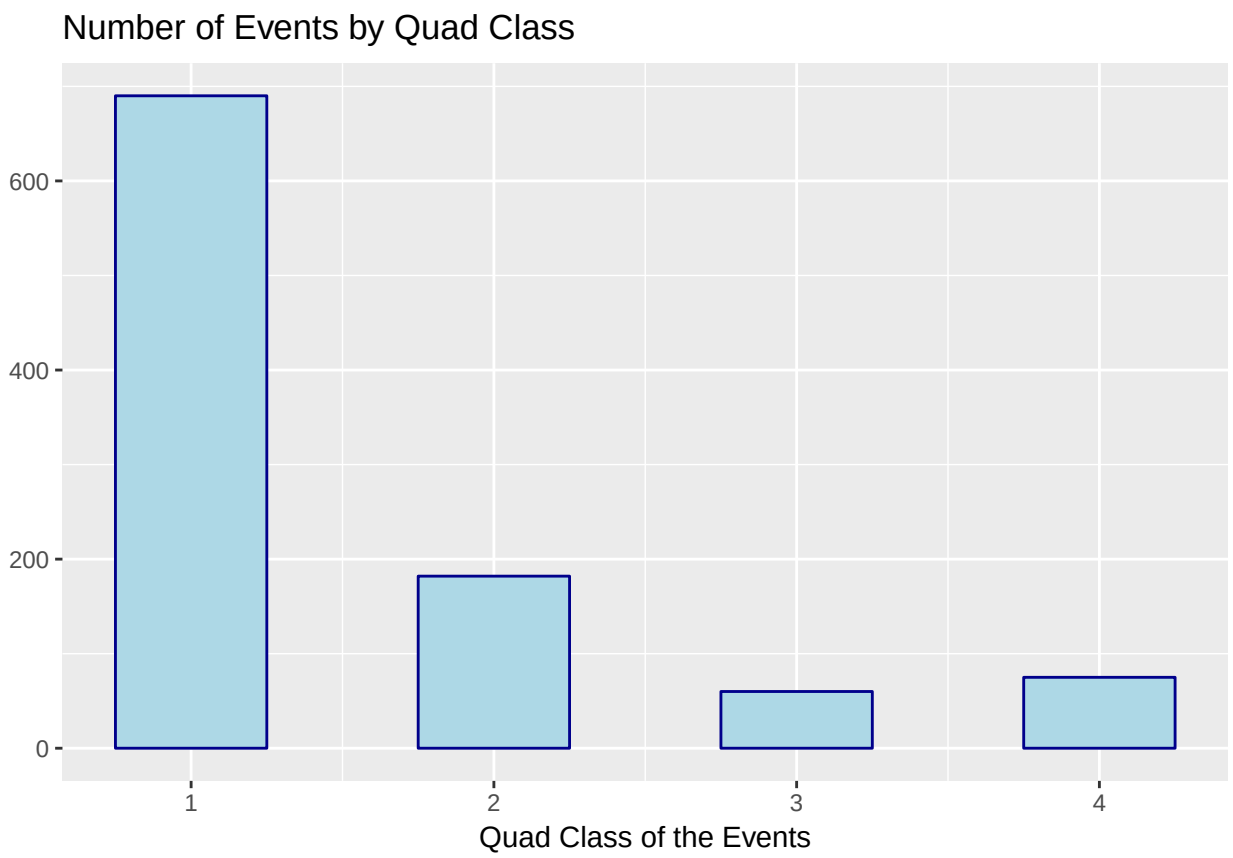
Going back to our bar chart of the Quad Class varaible, next we will adjust the labels and add a chart title.

- Change the label on the x-axis with `xlab("Quad Class of Events")`

- Delete the label on the y-axis (currently "count") with `theme(axis.title.y = element_blank())`

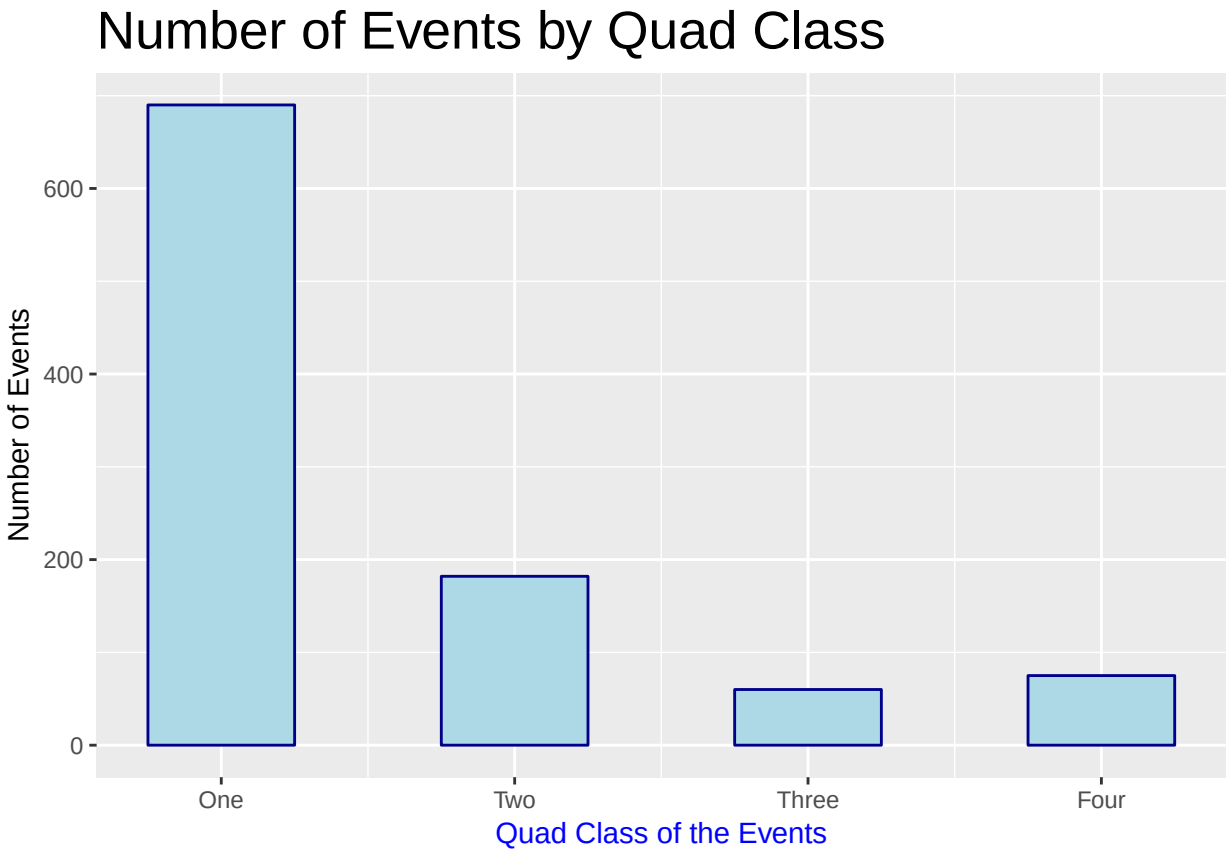
- Add a chart title with `ggtitle("Number of Events by Quad Class")`

```
ggplot(subsahara_jan17, aes(x = QuadClass)) +
  geom_bar(fill = "lightblue", colour = "darkblue", width = .5) +
  xlab("Quad Class of the Events") +
  theme(axis.title.y = element_blank()) +
  ggtitle("Number of Events by Quad Class")
```



4.6 Change the Look Exercise

1. Change the color of the x-axis label. (hint: use a theme function)
2. Increase the size of the chart title. (hint: use a theme function)
3. Change the labels on the x-axis to One, Two,Three, and Four. (hint: `scale_x_continuous()`)
4. Change the y-axis label to "Number of Events".


Number of Events by Quad Class
600
400
200
0
Number of Events
One
Two
Three
Four
Quad Class of the Events

Chapter 5

Bar Charts with Y Values

Remember that there's a difference in how to make a bar chart with values from a variable in the data on the y-axis vice counts for a single variable. The variable used for the y-axis should be continuous.

5.1 Values on the Y-axis

We want to get a sense of the tone of articles associated with each Quad Class. First we need to summarize the data to find the mean of the AvgTone variable for each QuadClass, using the *summarize* and *group_by* functions in **dplyr**.

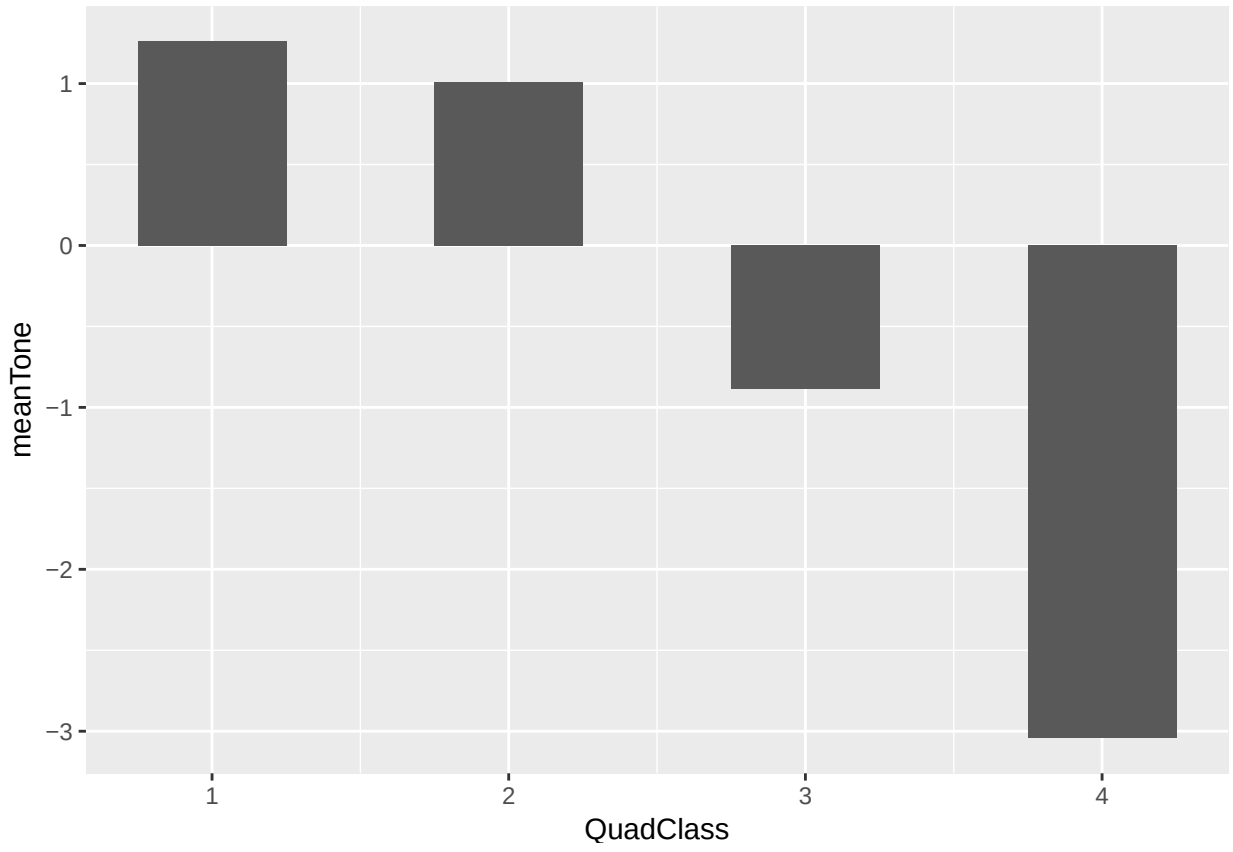
```
quadTone <- summarise(group_by(subsahara_jan17, QuadClass), meanTone = mean(AvgTone))

quadTone
```

```
## # A tibble: 4 x 2
##   QuadClass   meanTone
##       <int>      <dbl>
## 1         1  1.2619367
## 2         2  1.0109206
## 3         3 -0.8859623
## 4         4 -3.0444109
```

To chart the new dataframe with values on the y-axis, we must add `stat = "identity"` to the *geom_bar()* function.

```
ggplot(quadTone, aes(x = QuadClass, y = meanTone)) +
  geom_bar(stat = "identity", width = .5)
```



5.2 Multiple Variables

You may want to map multiple variables to the x-axis, to create a stacked or grouped bar chart where the bars are colored based on the second variable. In **ggplot** this is done by adding an aesthetic mapping called *fill*.

For this we will use a new data set that captures Chinese government activity in Nigeria, Zambia, and Zimbabwe in the first 2 weeks of January 2017. If you don't have the "china_2weeks" data in your environment, then read in the CSV file.

```
china_2weeks <- read.csv("china_2weeks.csv", stringsAsFactors = FALSE)

head(china_2weeks)
```

```
##         days country n
## 1 2017-01-03      NI 1
## 2 2017-01-03      ZA 0
## 3 2017-01-03      ZI 1
## 4 2017-01-04      NI 0
## 5 2017-01-04      ZA 0
## 6 2017-01-04      ZI 0
```

```
str(china_2weeks)
```

```
## 'data.frame':    42 obs. of  3 variables:
##  $ days   : chr  "2017-01-03" "2017-01-03" "2017-01-03" "2017-01-04" ...
##  $ country: chr  "NI" "ZA" "ZI" "NI" ...
```

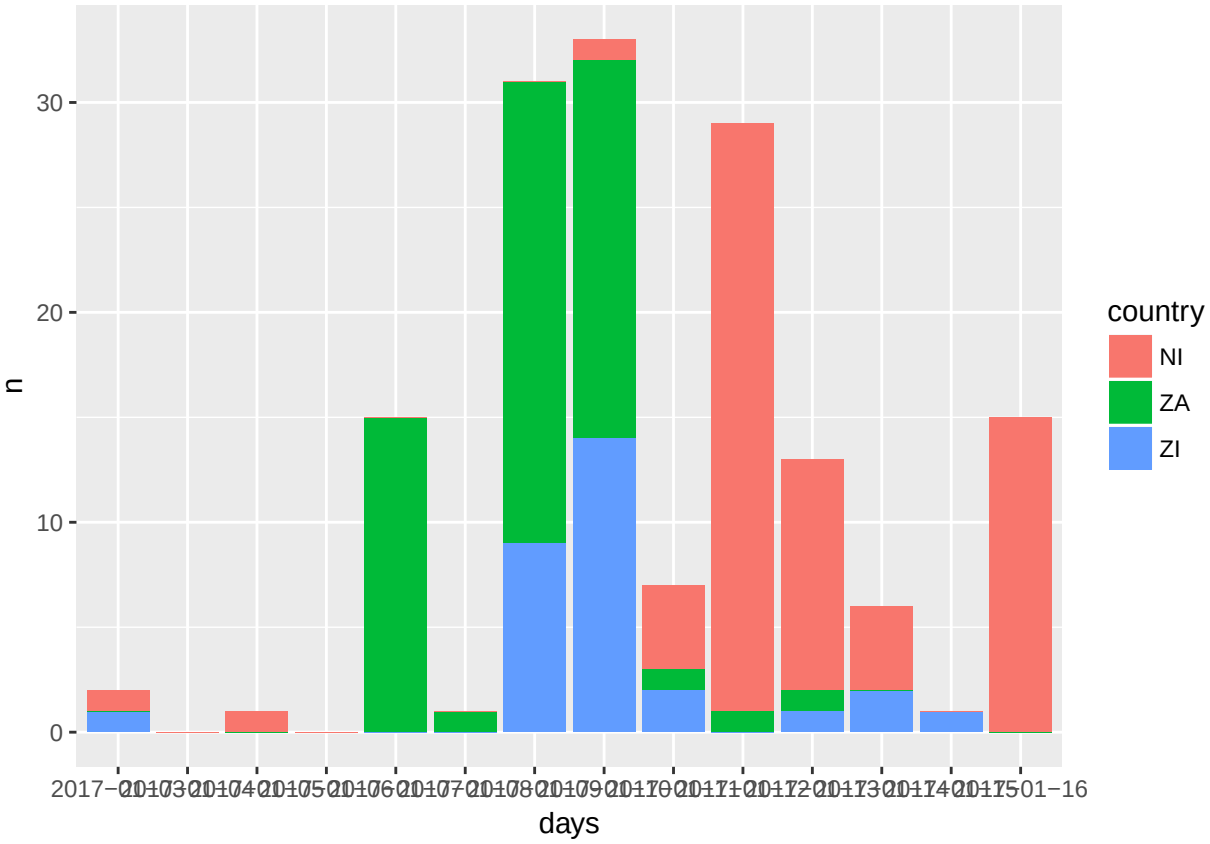
```
##  $ n       : int  1 0 1 0 0 0 1 0 0 0 ...
```

When you look at the variables in the new data set, notice that the "days" variable is a Character. This is important to be aware of because we will have to transform it into a Date later.

To compare the activity of the Chinese government across the same time period in each of the countries, use these mappings:

- Days on the x-axis.
- Number of events on the y-axis.
- Bars for each of the countries, stacked.

```
ggplot(china_2weeks, aes(x = days, y = n, fill = country))+
  geom_bar(stat = "identity")
```



5.3 Date Sequence

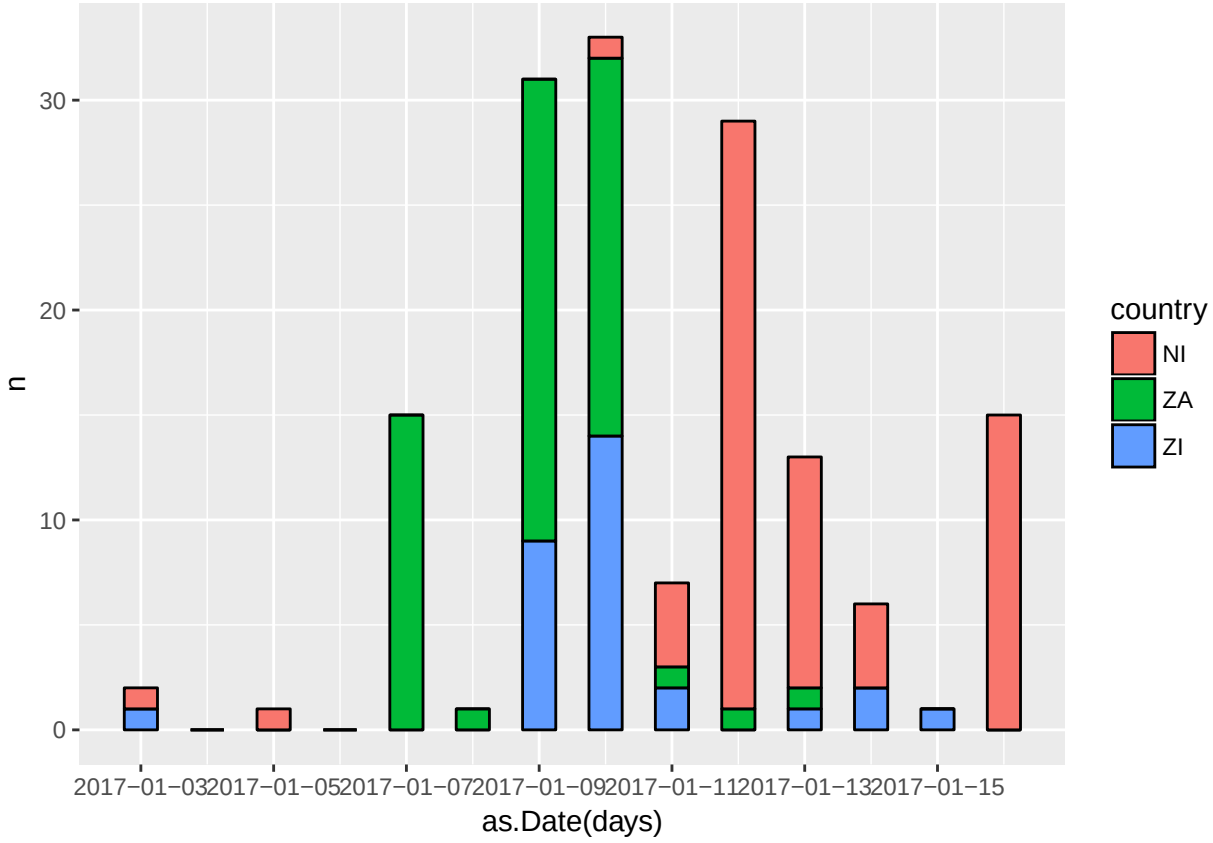
You can see that this chart needs some formatting to make it more readable.

- Adjust the labels on the x-axis.
  - Create a sequence of dates with the *seq()* function that covers every other day in the range of dates in our data.
  - That will make the labels easier to read.
  - Notice the use of *as.Date()* function.

```
dailybreaks <- seq(min(as.Date(china_2weeks$days)),
                   max(as.Date(china_2weeks$days)),
                   by="2 day")
```

- Use that sequence in a *scale_x_date()* function to set the date breaks to that sequence.
- Use the *as.Date()* function in the aesthetics for the x-axis to make sure the data types match between the data and the scale.

```
ggplot(china_2weeks, aes(x = as.Date(days), y = n, fill = country)) +
  geom_bar(stat = "identity", width = .5, color = "black") +
  scale_x_date(breaks = dailybreaks)
```



5.4 hjust and vjust

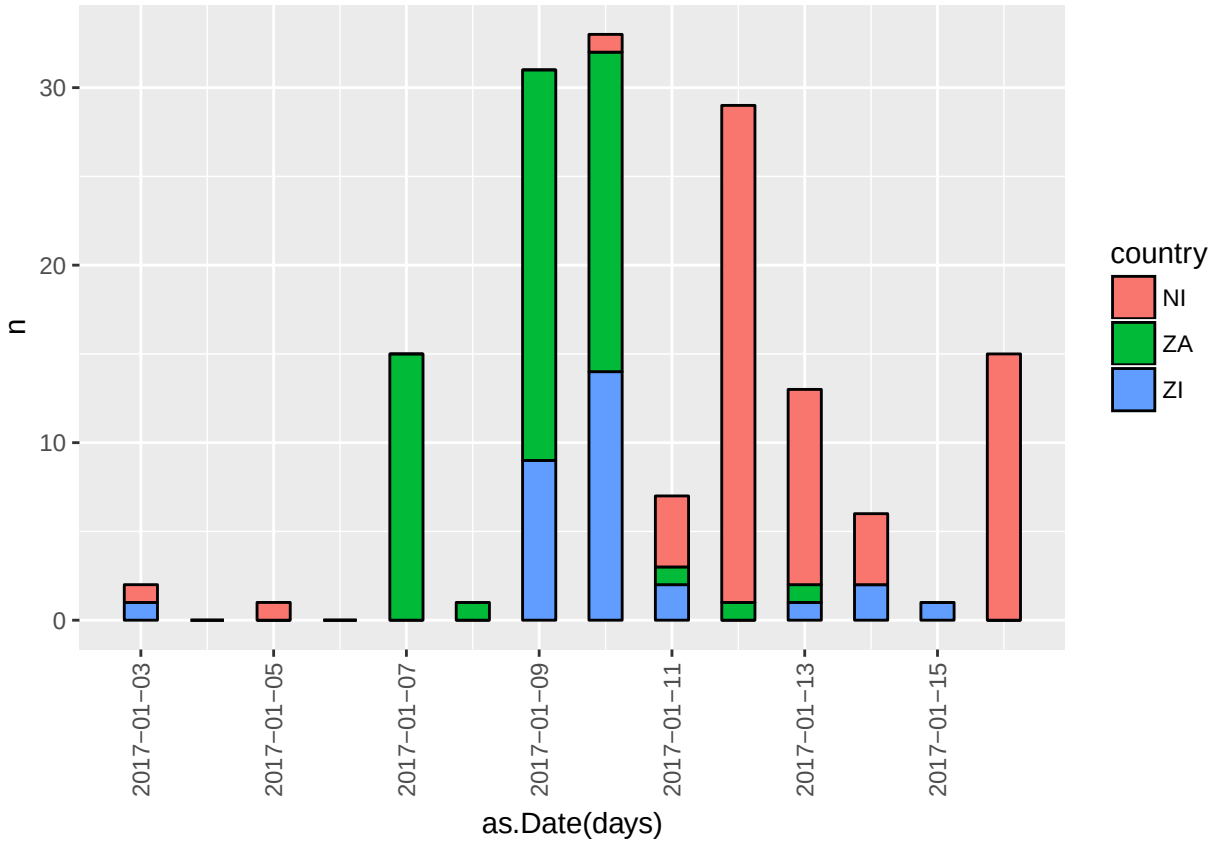
To adjust the position, vertically and horizontally, of text you may need to use the vjust and hjust commands in conjunction with the angle command. Note that each command can only take certain values and the combination results in different positions.

The source of the reference for this is https://www.r-bloggers.com/hjust-and-vjust/

- Adjust the angle of the labels on the x-axis with another *theme()* function, including the variables:
  - `angle`
  - `hjust` (horizontal justification)
  - `vjust` (vertical justification)

```
ggplot(china_2weeks, aes(x = as.Date(days), y = n, fill = country)) +
  geom_bar(stat = "identity", width = .5, color = "black") +
```

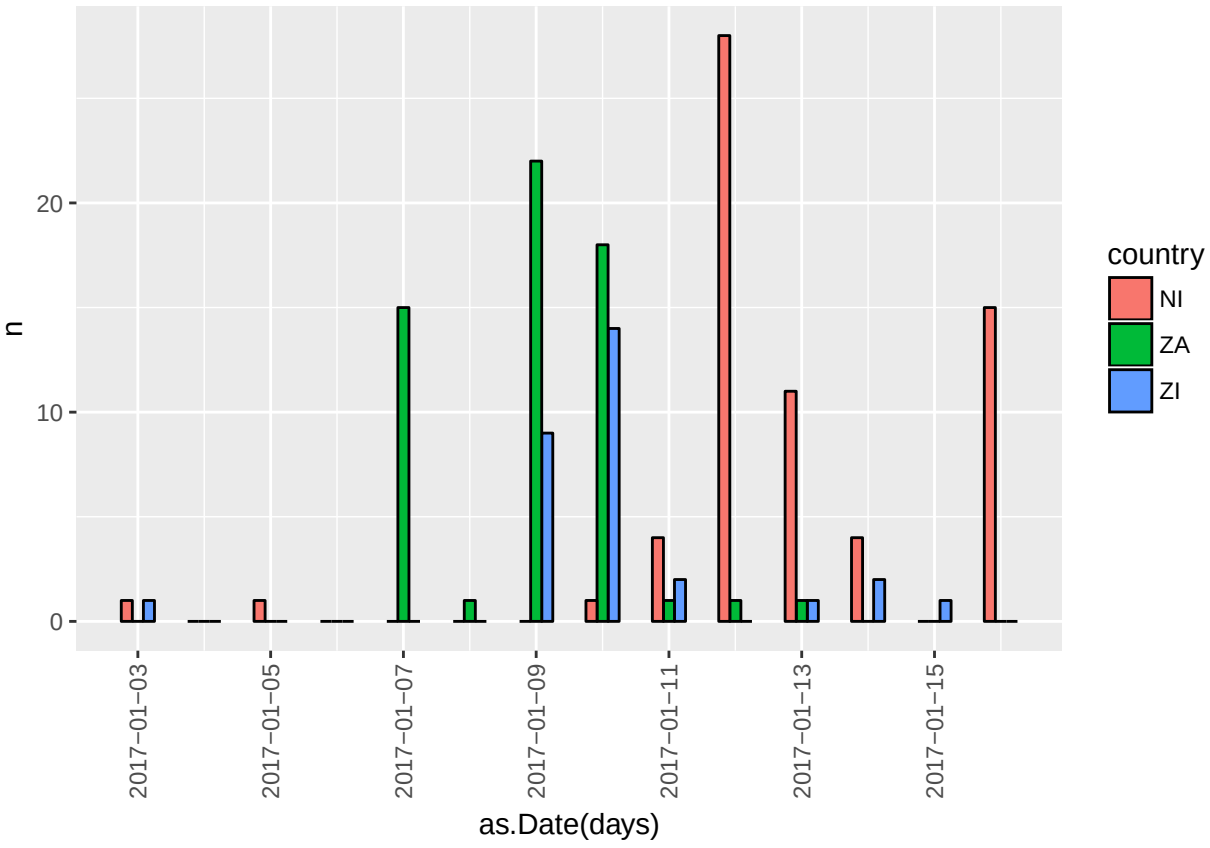
```
  scale_x_date(breaks = dailybreaks) +
  theme(axis.text.x = element_text(angle = 90, hjust= .5, vjust= .5))
```



5.5 Stacked to Grouped Bars

- Change from stacked bars to grouped bars that are side by side using `position = "dodge"` in the *geom_bar()* function.

```
ggplot(china_2weeks, aes(x = as.Date(days), y = n, fill = country)) +
  geom_bar(stat = "identity", width = .5, color = "black", position = "dodge") +
  scale_x_date(breaks = dailybreaks) +
  theme(axis.text.x = element_text(angle = 90, hjust= .5, vjust= .5))
```

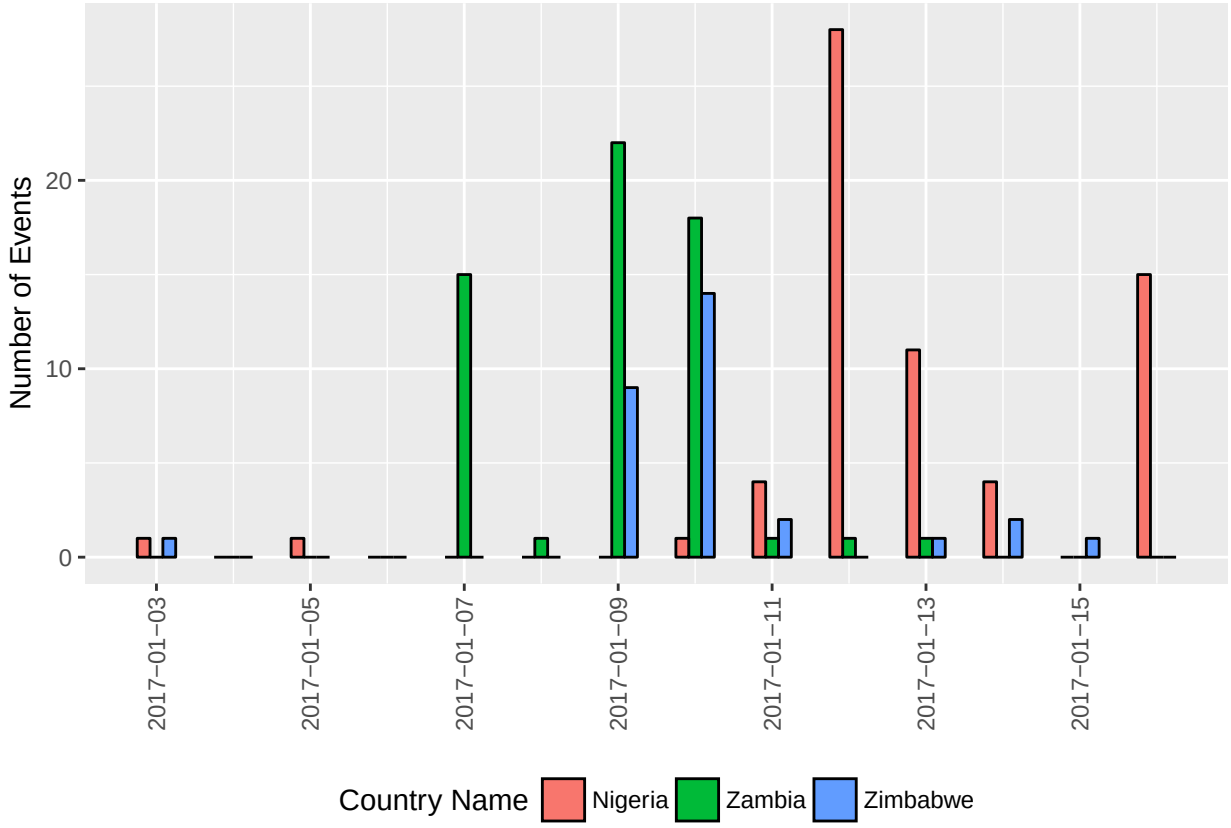


5.6 Legends

We want to make some changes to the legend on our chart.

- The legend position is controlled by the *theme()* function: `theme(legend.position = "bottom")`
- Change the legend title with `labs(fill = "Country Name")`
- Change the text of the labels with `scale_fill_discrete(labels = c("Nigeria","Zambia","Zimbabwe"))`
- Remove the x-axis label and change the y-axis label.

```
ggplot(china_2weeks, aes(x = as.Date(days), y = n, fill = country))+
  geom_bar(stat = "identity", position = "dodge", width = .5, color = "black")+
  scale_x_date(breaks = dailybreaks) +
  theme(axis.text.x = element_text(angle = 90, hjust= .5, vjust= .5)) +
  theme(axis.title.x = element_blank()) +
  ylab("Number of Events") +
  theme(legend.position = "bottom")+
  labs(fill = "Country Name") +
  scale_fill_discrete(labels = c("Nigeria","Zambia","Zimbabwe"))
```



5.7 Color Palette

The colors used for each bar are based on a default color palette which can be changed.

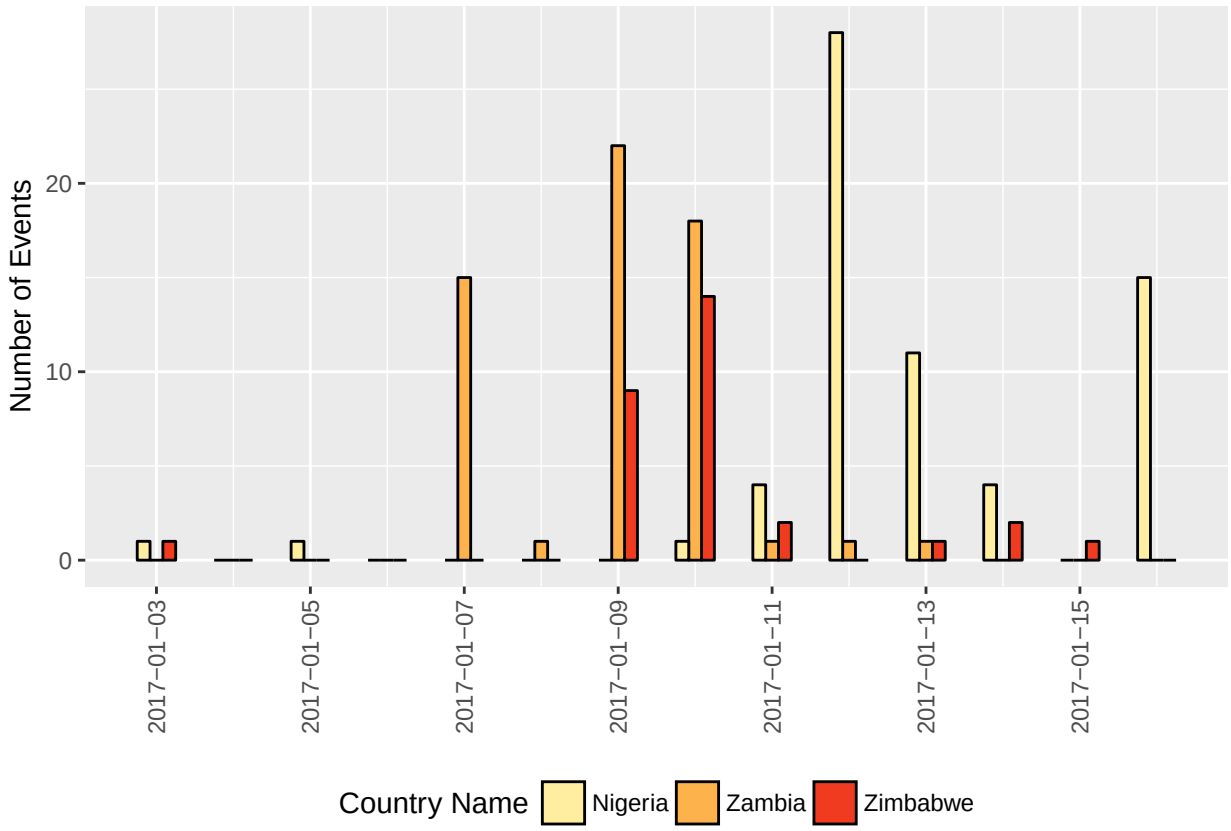
Remember this reference: http://www.cookbook-r.com/Graphs/Colors_(ggplot2)/

The **RColorBrewer** package offeres pre-defined color scales.

- Use *display.brewer.all()* to see the set of palettes.
- To use the RColorBrewer palettes, you need to use the `scale_fill_brewer()` function as the scale for the "fill" variable, instead of `scale_fill_discrete()`.
- Include the parameters for "palette" and "labels".
- See section 12.3 in the "R Graphics Cookbook".

Change the color palette of the bars with `scale_fill_brewer(palette = "YlOrRd", labels = c("Nigeria","Zambia","Zimbabwe"))`

```
ggplot(china_2weeks, aes(x = as.Date(days), y = n, fill = country))+
  geom_bar(stat = "identity", position = "dodge", width = .5, color = "black")+
  scale_x_date(breaks = dailybreaks) +
  theme(axis.text.x = element_text(angle = 90, hjust= .5, vjust= .5)) +
  theme(axis.title.x = element_blank()) +
  ylab("Number of Events") +
  theme(legend.position = "bottom")+
  labs(fill = "Country Name") +
  scale_fill_brewer(palette = "YlOrRd",labels = c("Nigeria","Zambia","Zimbabwe"))
```


Number of Events
20
10
0
2017−01−03
2017−01−05
2017−01−07
2017−01−09
2017−01−11
2017−01−13
2017−01−15
Country Name
Nigeria
Zambia
Zimbabwe

Chapter 6

Line Charts

Line charts are typically used when all variable are continuous. Often, line charts are better for graphics involving dates and/or times (often called time series charts) because trends are more visible.

Let's create some simple time series data with the "china_2weeks" dataframe.

```
china_all <- summarize(group_by(china_2weeks, days), total = sum(n))
china_all$days <- as.Date(china_all$days)
```

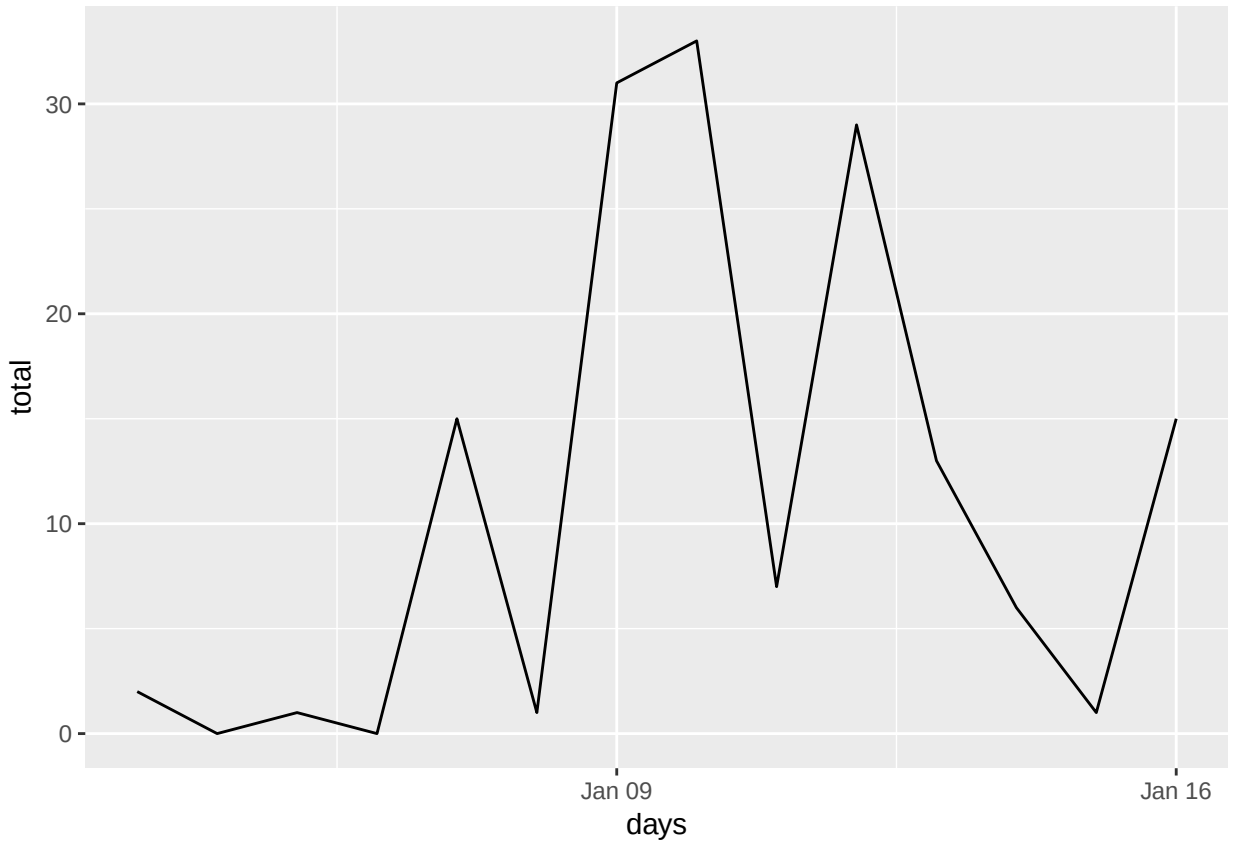
Notice that there are no dates that are skipped - we have data for every day. That's great because it means we don't need to fill in the gaps.

6.1 Basic Line Charts

In **ggplot**, line charts use geom function *geom_line()*.

Let's create a plot of the total events by day.

```
ggplot(china_all, aes(x = days, y = total))+
  geom_line()
```



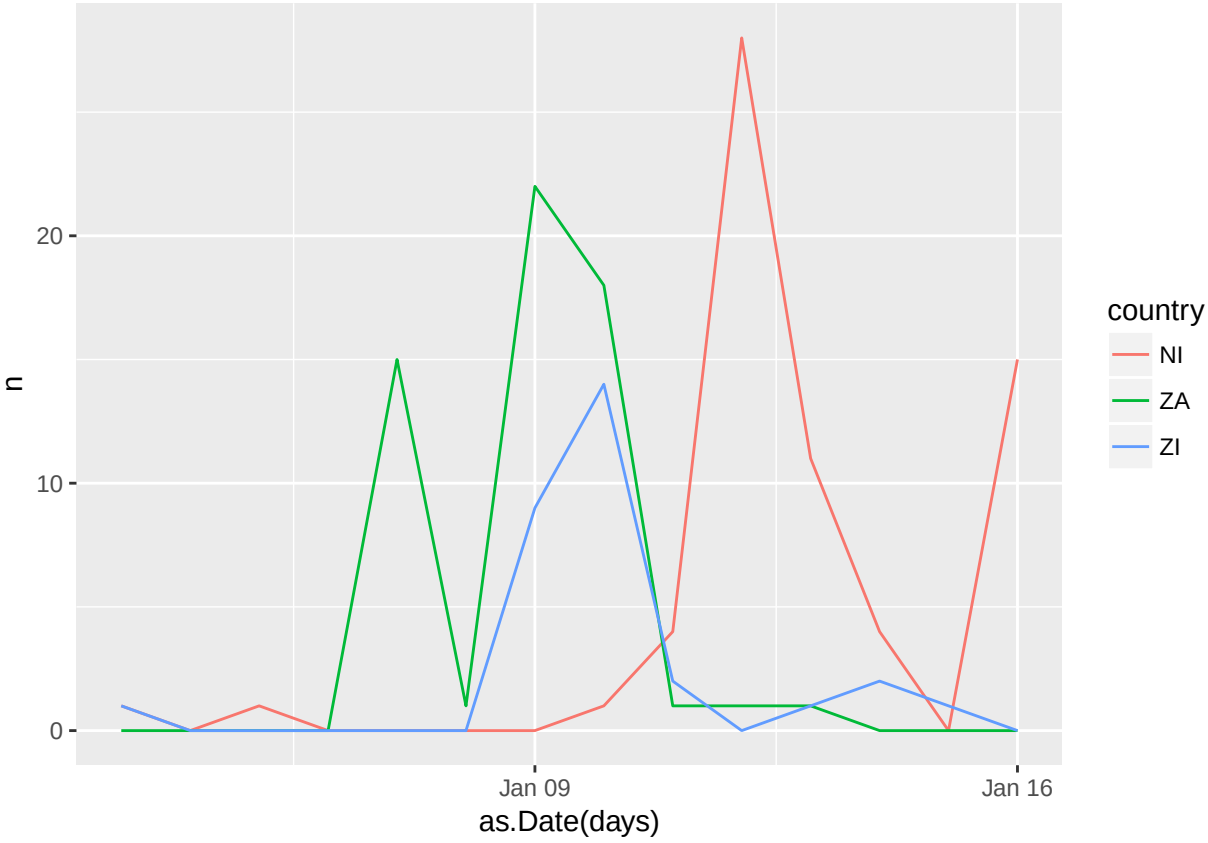
6.2 Multi-Variable Line Charts

You can also plot multiple lines on a chart. We will use the china_2weeks dataframe and plot a line for each country.

- To change the color of the line based on country, we need to change the `fill` aesthetic to `color`.

- You must have another aesthetic besides x and y so that *ggplot* knows how to group the data.

```
ggplot(china_2weeks, aes(x = as.Date(days), y = n, color = country))+
  geom_line()
```



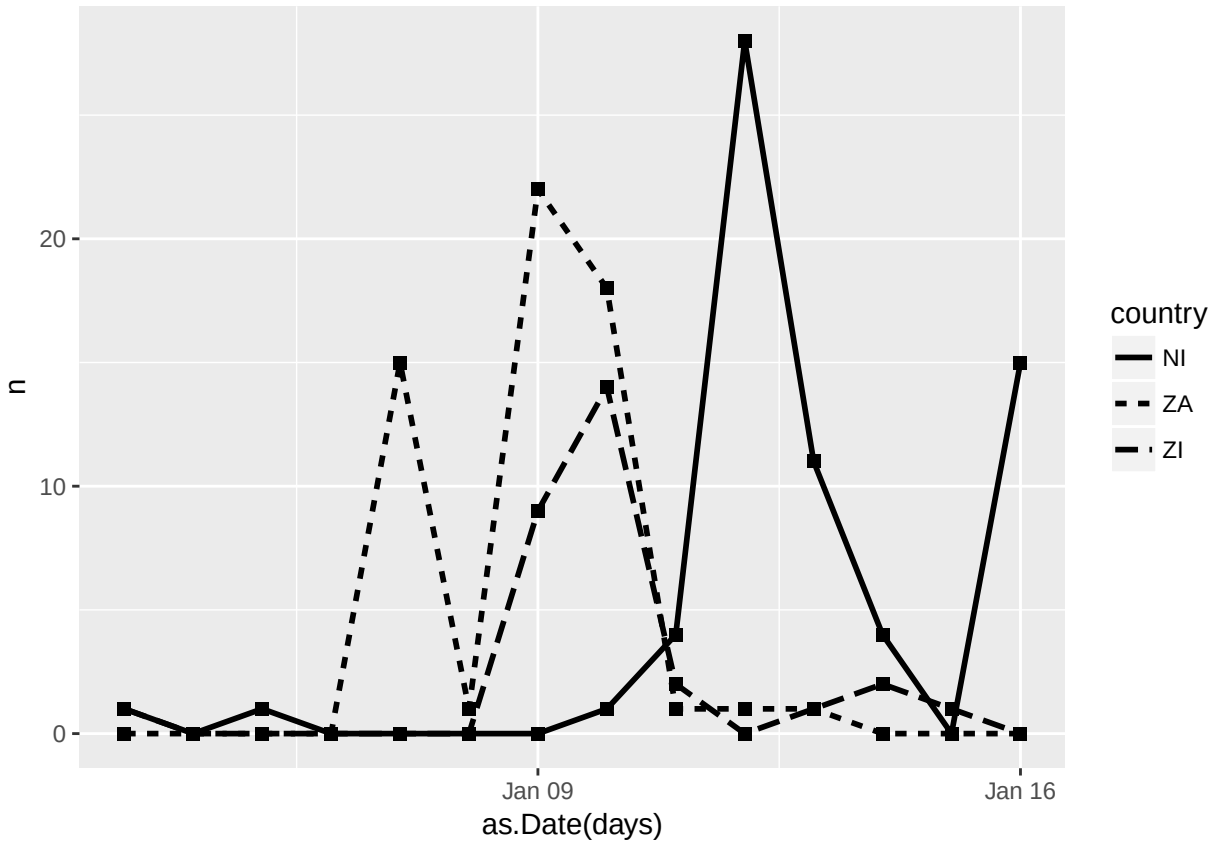
6.3 Adjust the Look

We can make some adjustments to the look of the line chart.

A good reference for points and lines: http://www.cookbook-r.com/Graphs/Shapes_and_line_types/

- Map the country to line type instead of color with `linetype = country`
- Increase the width of the line with `geom_line(size = 1)`
- Add points and change the shape and size of the point with `geom_point(shape = 15, size = 2)`

```
ggplot(china_2weeks, aes(x = as.Date(days), y = n, linetype = country))+
  geom_line(size = 1) +
  geom_point(shape = 15, size = 2)
```


20
10
0
n
Jan 09
Jan 16
as.Date(days)
country
NI
ZA
ZI

Chapter 7

Histograms

A histogram is used to map a continuous variable to the x-axis and use bins to depict the distribution of that variable.

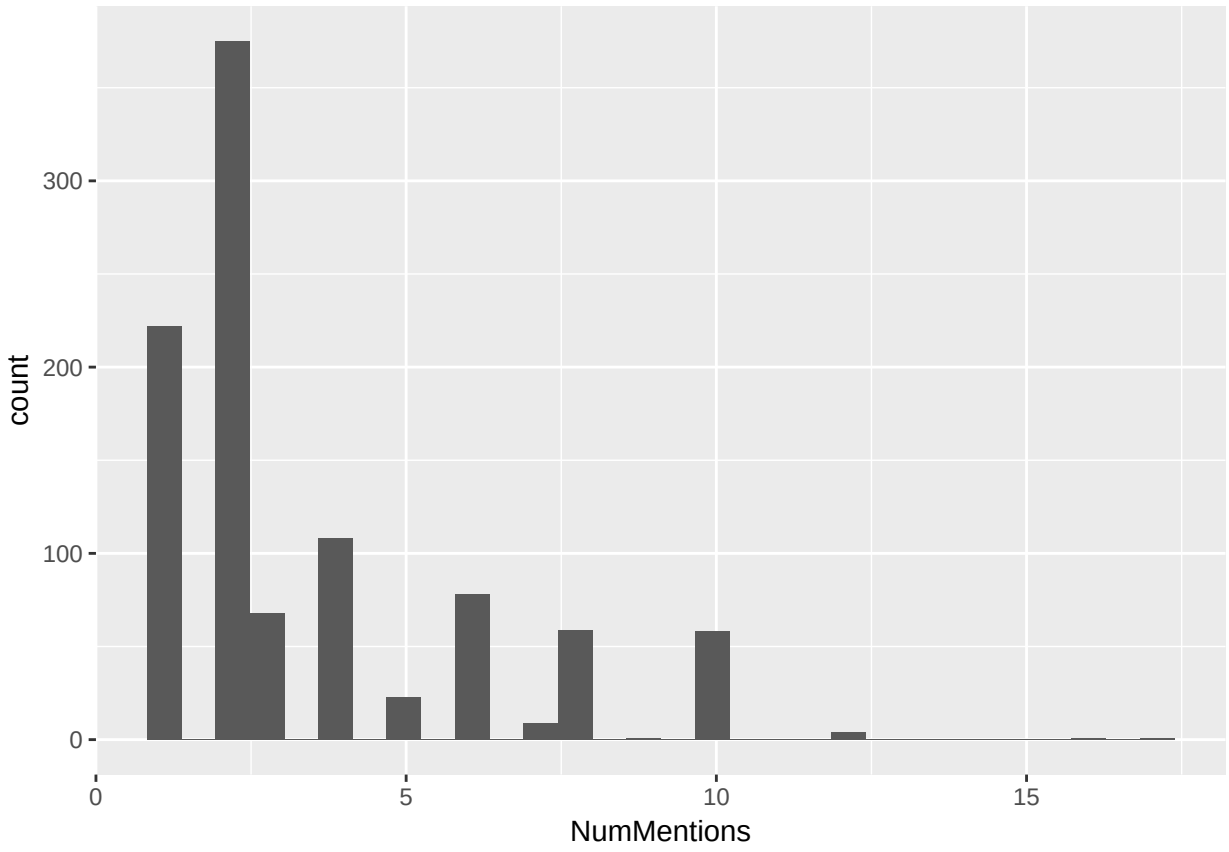
- In **ggplot**, histograms use geom function *geom_histogram()*.

- In the subsahara_jan17 dataset, a continuous variable is NumMentions, which is the number of mentions of the event across all source documents, an indicator of the importance of the event.

- We want to see the distribution of the NumMentions variable.

```
ggplot(subsahara_jan17, aes(x=NumMentions))+
  geom_histogram()
```

```
FALSE `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



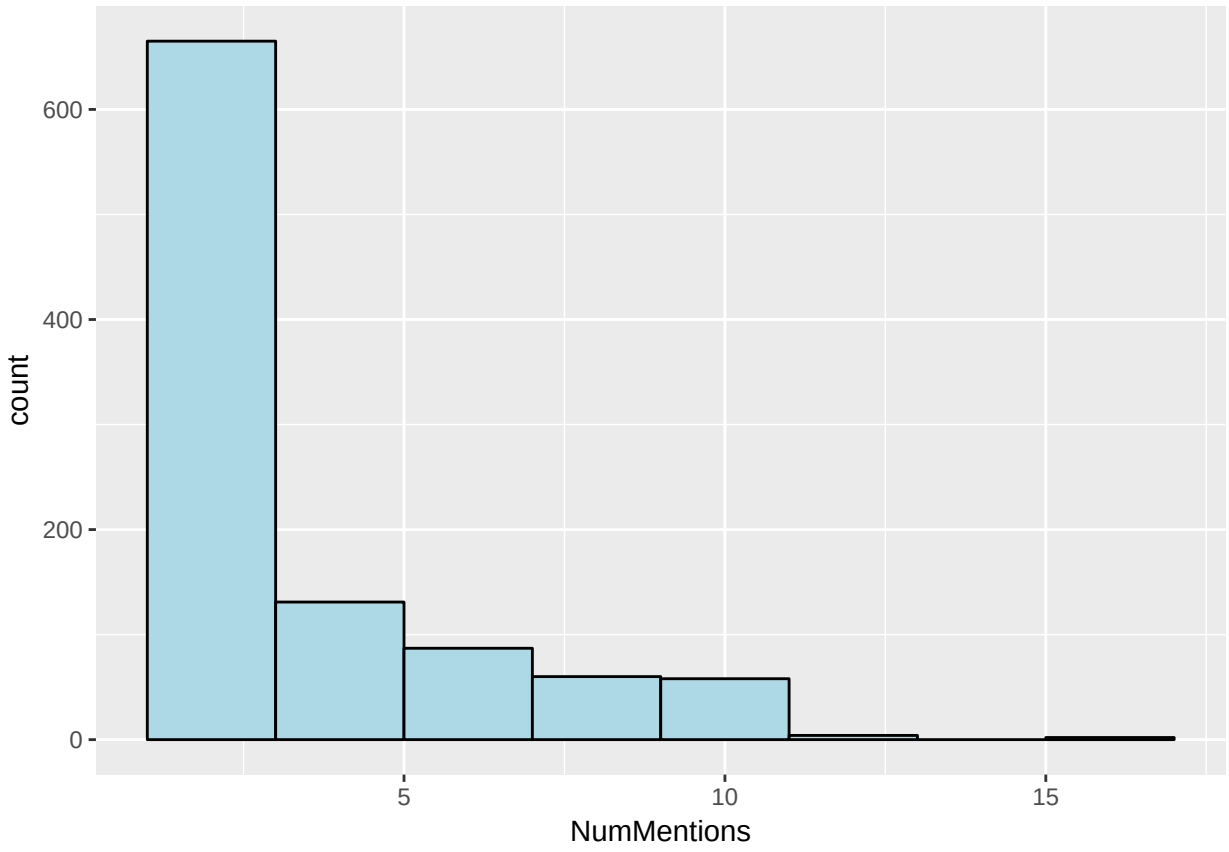
The default in **ggplot** is 30 bins. But we can set the number of bins in two ways. First we want to find the max and min of the variable.

```
summary(subsahara_jan17$NumMentions)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   2.000   3.372   4.000  17.000
```

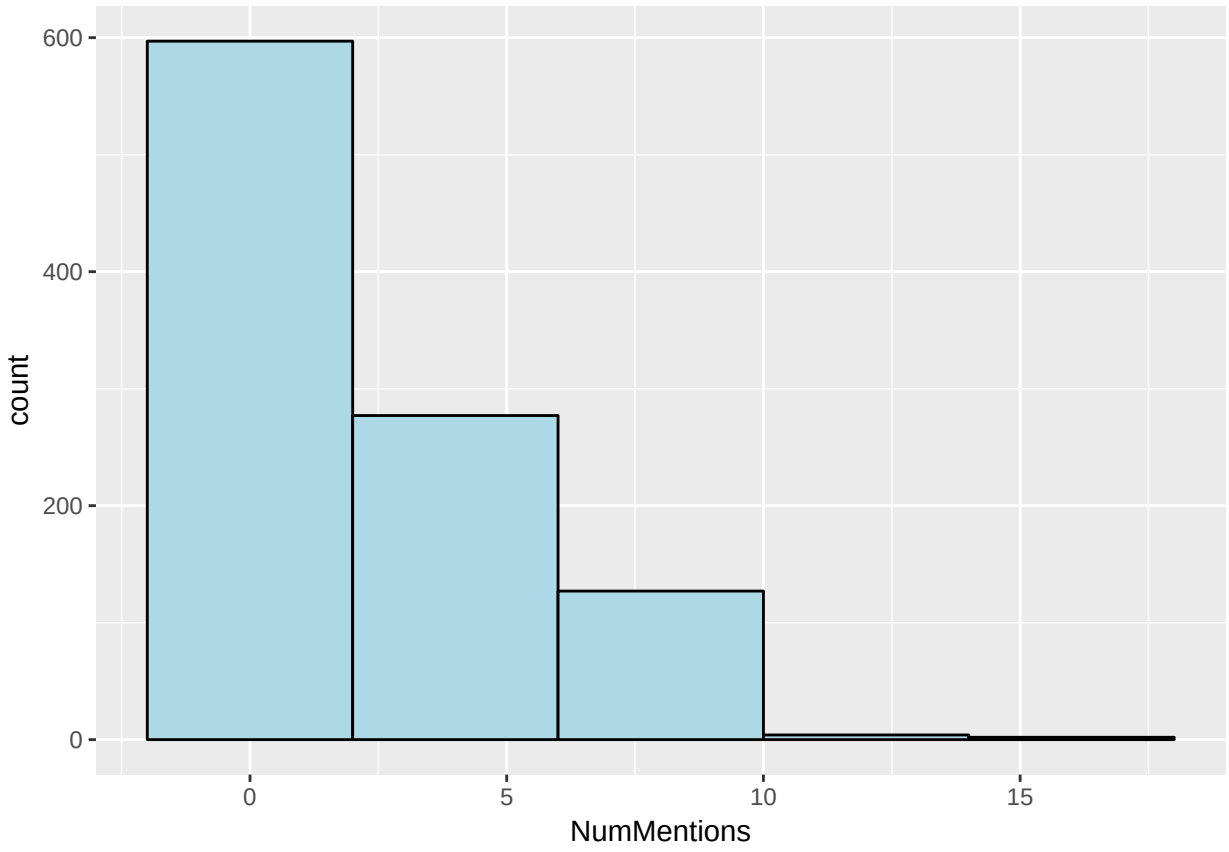
To set the width of the bins use `binwidth = 2` This will create bins from 1 up to but not including 3, 3 up to but not including 5, and so on.

```
ggplot(subsahara_jan17, aes(x = NumMentions)) +
  geom_histogram(binwidth = 2, fill = "lightblue", color = "black")
```



To set the number of bins use `bins = 5`

```
ggplot(subsahara_jan17, aes(x = NumMentions)) +
  geom_histogram(bins =5, fill = "lightblue", color = "black")
```


600
400
200
0
count
0
5
10
15
NumMentions

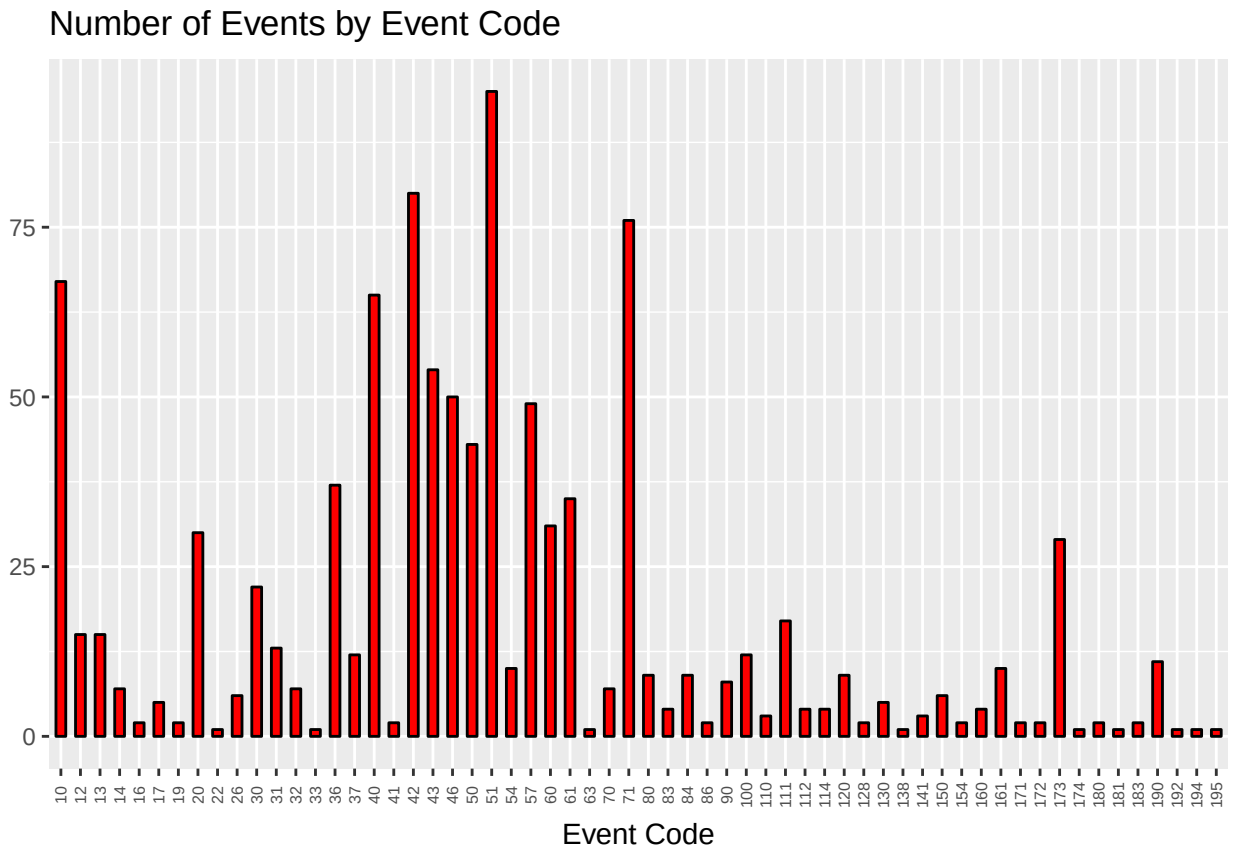
Chapter 8

Practical Exercise

Try to recreate the following charts. Use the "subsahara_jan17.csv" data.

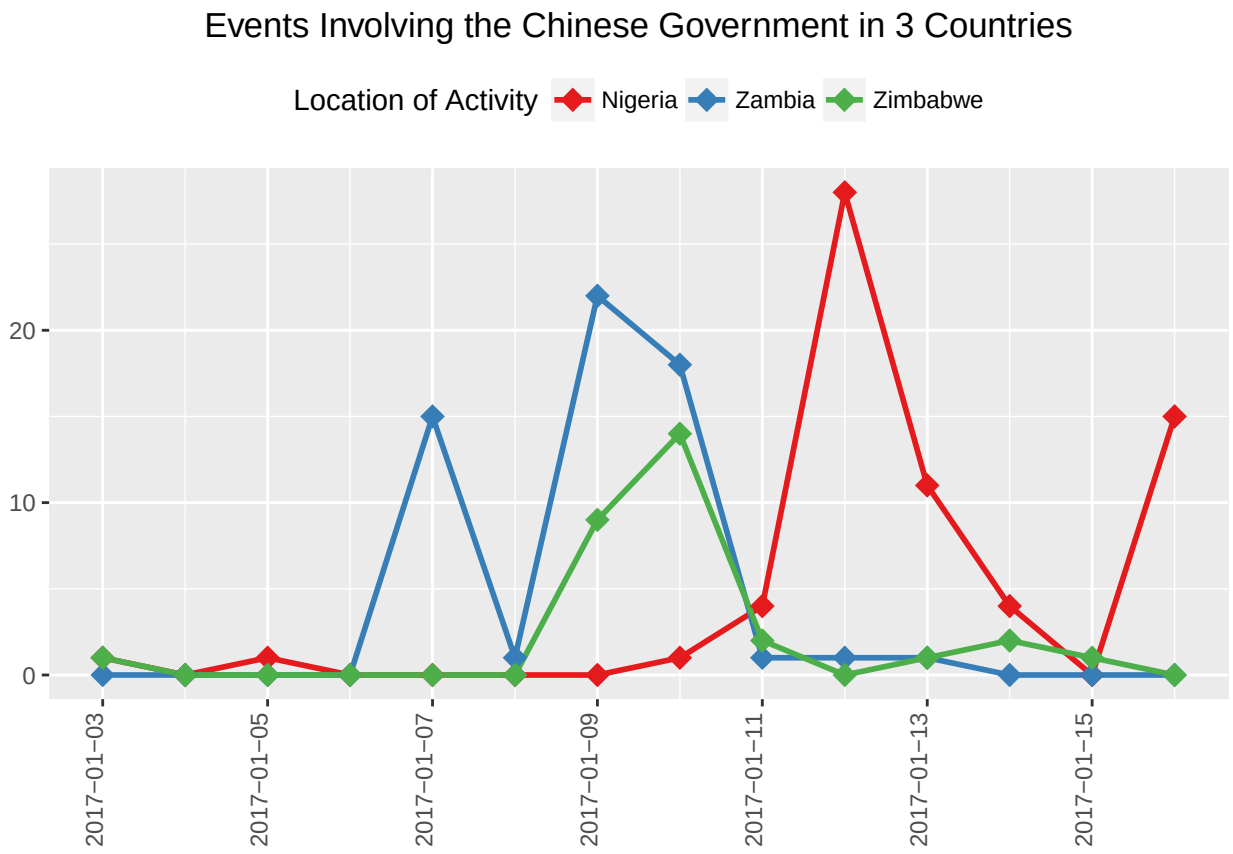
8.1 Bar: Count of Events by Event Base Code

Hint: convert EventBaseCode to a factor.



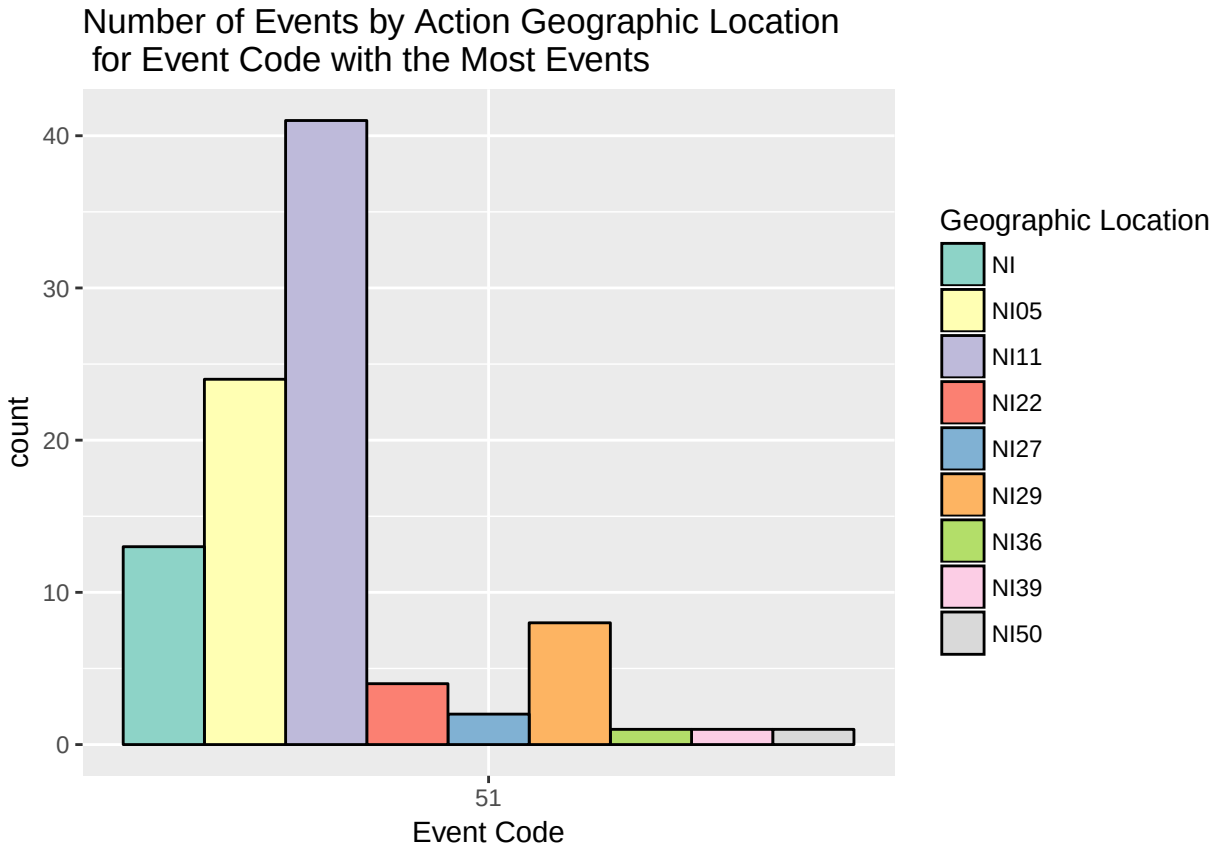
8.2 Line: Chinese Activity by Location

Change the look of the chart of Chinese government activity.



8.3 Bar: Events by Geographic Location

Hint: *filter()*

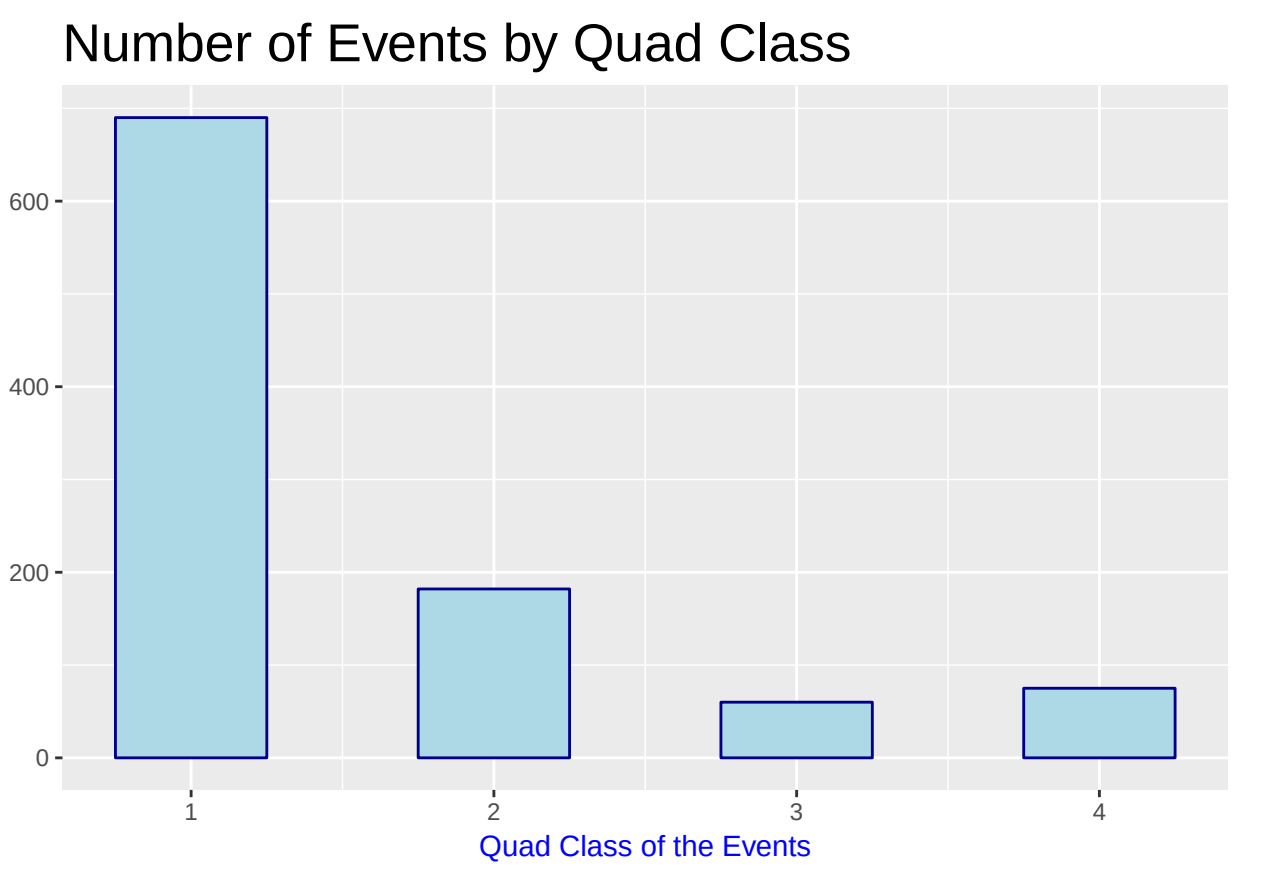

Number of Events by Action Geographic Location
for Event Code with the Most Events
count
40
30
20
10
0
51
Event Code
Geographic Location
NI
NI05
NI11
NI22
NI27
NI29
NI36
NI39
NI50

Chapter 9

Exercise Solutions

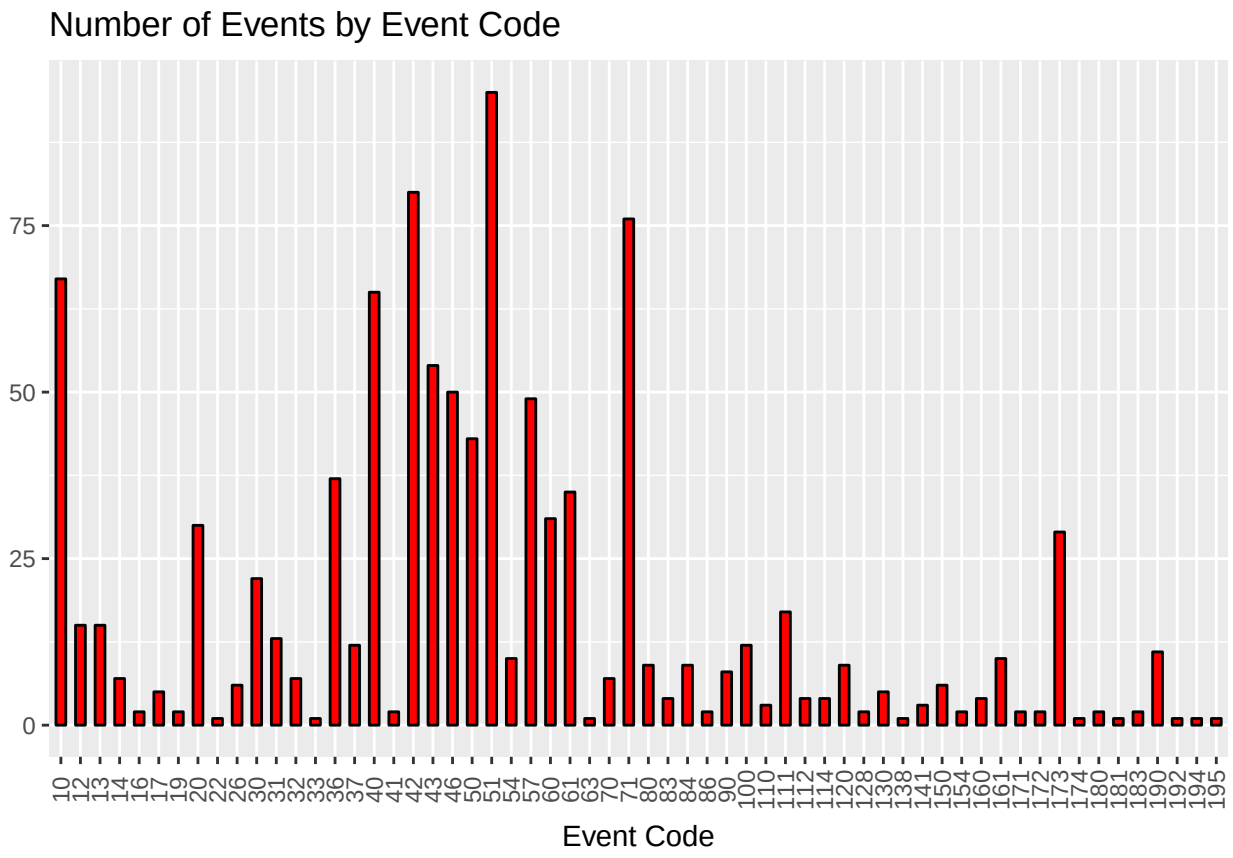
9.1 Change the Look Exercise

```
ggplot(subsahara_jan17, aes(x = QuadClass)) +
  geom_bar(fill = "lightblue", colour = "darkblue", width = .5) +
  xlab("Quad Class of the Events") +
  theme(axis.title.y = element_blank()) +
  ggtitle("Number of Events by Quad Class") +
  theme(axis.title.x = element_text(color = 'blue')) +
  theme(plot.title = element_text(size = 20))
```



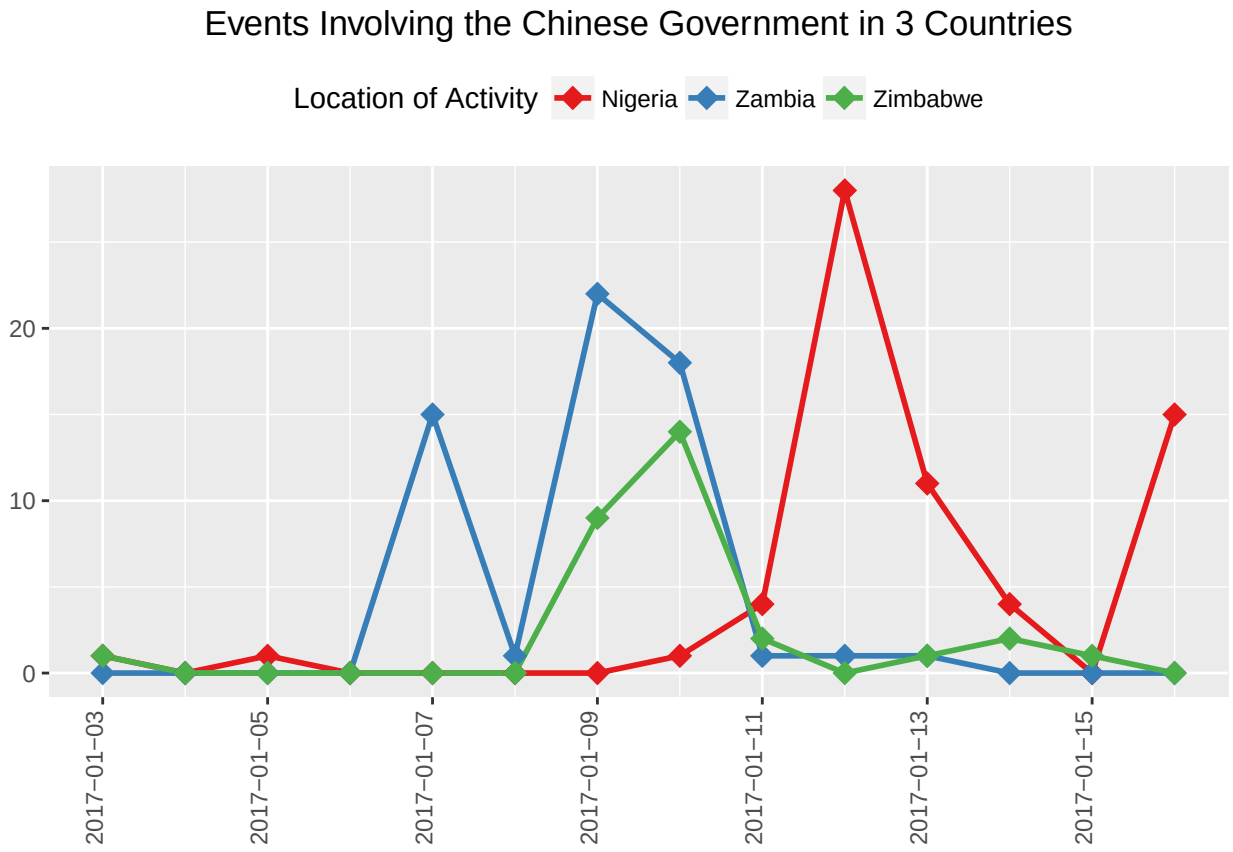
9.2 PE Part 1

```
subsahara_jan17$EventBaseCode <- as.factor(subsahara_jan17$EventBaseCode)
ggplot(subsahara_jan17, aes(x = EventBaseCode)) +
  geom_bar(fill = "red", colour = "black", width = .5) +
  theme(axis.text.x = element_text(angle = 90, hjust = .5, vjust = .5)) +
  xlab("Event Code") +
  theme(axis.title.y = element_blank()) +
  ggtitle("Number of Events by Event Code")
```



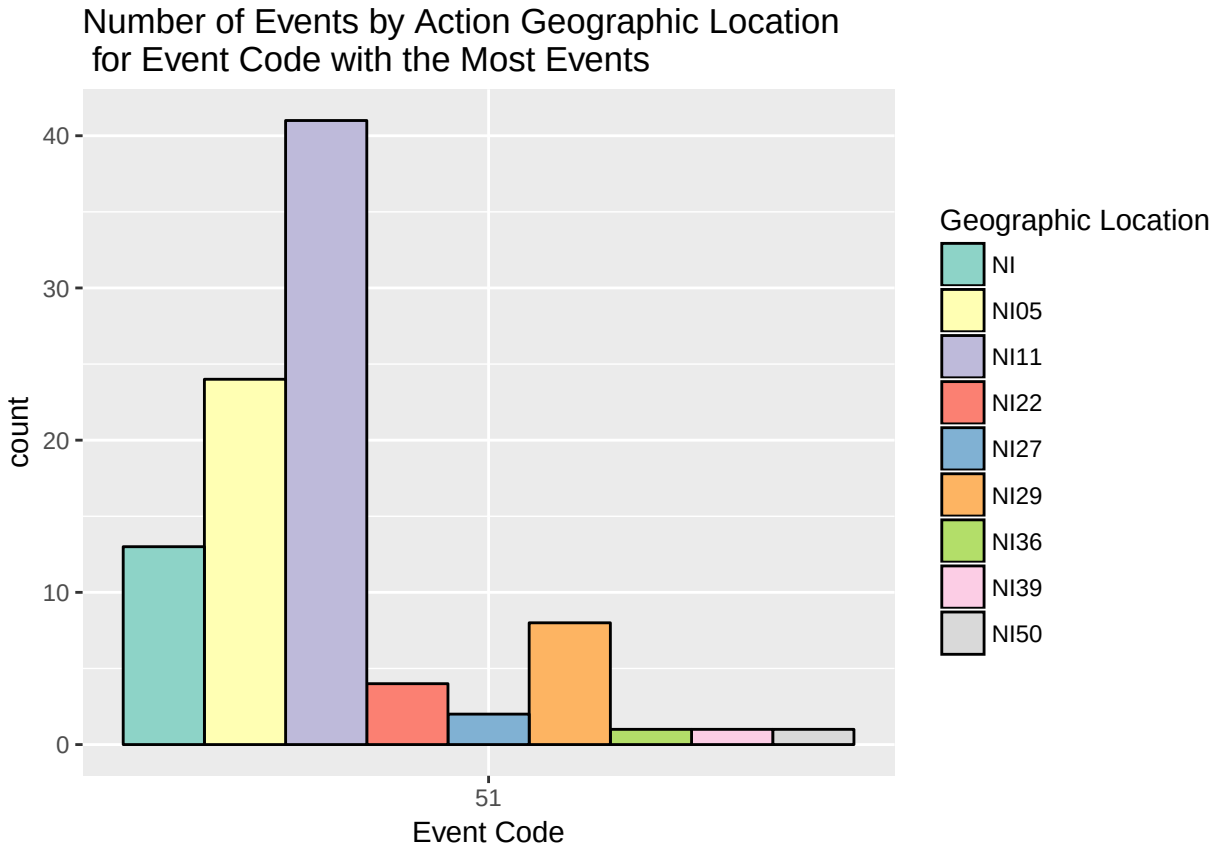
9.3 PE Part 2

```
ggplot(china_2weeks, aes(x = as.Date(days), y = n, color = country))+
  geom_line(size = 1) +
  ggtitle("Events Involving the Chinese Government in 3 Countries")+
  theme(plot.title = element_text(hjust = .5)) +  #Center the plot title
  theme(axis.title.y = element_blank()) +
  theme(axis.title.x = element_blank()) +
  scale_x_date(breaks = dailybreaks) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0)) +  #change angle of axis text
  theme(legend.position = "top")+  #legend to top of chart
  labs(color = "Location of Activity") +
  geom_point(shape = 18, size = 4) +
  scale_color_brewer(palette = "Set1", labels = c("Nigeria","Zambia","Zimbabwe"))
```



9.4 PE Part 3

```
sub51 <- dplyr::filter(subsahara_jan17, EventBaseCode == 51)
ggplot(sub51, aes(x= EventBaseCode, fill = ActionGeo_ADM1Code))+
  geom_bar(position = 'dodge', color = "black")+
  ggtitle("Number of Events by Action Geographic Location\n for Event Code with the Most Events") +
  xlab("Event Code") +
  labs(fill = "Geographic Location")+
  scale_fill_brewer(palette = "Set3")
```


Number of Events by Action Geographic Location
for Event Code with the Most Events
count
40
30
20
10
0
51
Event Code
Geographic Location
NI
NI05
NI11
NI22
NI27
NI29
NI36
NI39
NI50