Basics of ggplot2

Completed by Leon Woltermann

Contents

NB: The worksheet has beed developed and prepared by Lincoln Mullen. Source: Lincoln A. Mullen, *Computational Historical Thinking: With Applications in R (2018–)*: http://dh-r.lincolnmullen.com. (Updated and modified by Maxim G. Romanov)

Aim of this worksheet

After completing this worksheet, you should be able to use the powerful ggplot2 to make basic plots using the grammar of graphics. You may find the ggplot2 documentation or the R Graph Catalog to be helpful.

In addition to the ggplot2 package, we will use three packages with sample data, and we will load dplyr to get nice printing of data frames. Let's load them now, and also bring some of the data frames into the global environment.

```
library(ggplot2)
library(dplyr)
library(ggrepel)

library(gapminder)
data("gapminder")

library(historydata)
data("paulist_missions")
data("naval_promotions")

# MGR Comment: `paulist_missions` table is different in downloadable package
githubURL <- "https://github.com/ropensci/historydata/raw/master/data/paulist_missions.rda"
download.file(githubURL,"myfile")
load("myfile")

library(europop)
data("europop")

# Make a summarized dataset of Paulist missions by year
paulists_by_year <- paulist_missions %>%
  #mutate(year = format(as.Date(end_date, format="%m/%d/%Y"), "%Y")) %>% # the table in the downloadabl
  # one item does not convert properly > turns into NA (although the date looks fine)
  group_by(year) %>%
  summarize(missions = n(),
            converts = sum(converts, na.rm = TRUE),
            confessions = sum(confessions, na.rm = TRUE))

unique(paulists_by_year$year)
```

```
##  [1] 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865
## [16] 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885
## [31] 1886 1887 1888 1889 1890 1891 1892 1893
```

```
paulists_by_year
```

```
## # A tibble: 38 x 4
##     year missions converts confessions
##    <dbl>    <int>    <int>       <int>
##  1  1851        9        0       22970
##  2  1852       12       15       30170
##  3  1853       11       39       25000
##  4  1854       12       15       23700
##  5  1855       10       34       21850
##  6  1856       14       32       18930
##  7  1857       11       29       17490
##  8  1858       16       38       23413
##  9  1859       10       77       29885
## 10  1860       13       31       29575
## # ... with 28 more rows
```

Basics of using ggplot2

The fundamental insight of the grammar of graphics is the variables in the data can be mapped to aesthetics in the visualization. A variable in a data frame will be found in a column. An aesthetic in ggplot2 can take many forms, depending on the kinds of marks (glyphs) that you are going to make in the plot. But the most common aesthetics are `x` and `y` position, `size`, `color`, `fill`, `shape` and `weight`. Some less common but still useful are `label` and `linetype`. The ggplot2 package lets us explicitly set which variables are mapped to which marks using the `aes()` function.

The three basic parts of a call to ggplot2 are these:

1. The specification of which dataset you are using by passing a variable to the `ggplot()` function as its first argument.
2. The specification of which variables map to which aesthetics, using arguments to the `aes()` function. The `aes()` function is normally passed as the second argument to `ggplot()` (though it can also be specified in the various geoms).
3. At least one layer in the plot that makes a mark (or glyph). This is specified by one of the geoms, such as `geom_point()`.

Consider this basic plot. First, let's look at the data.

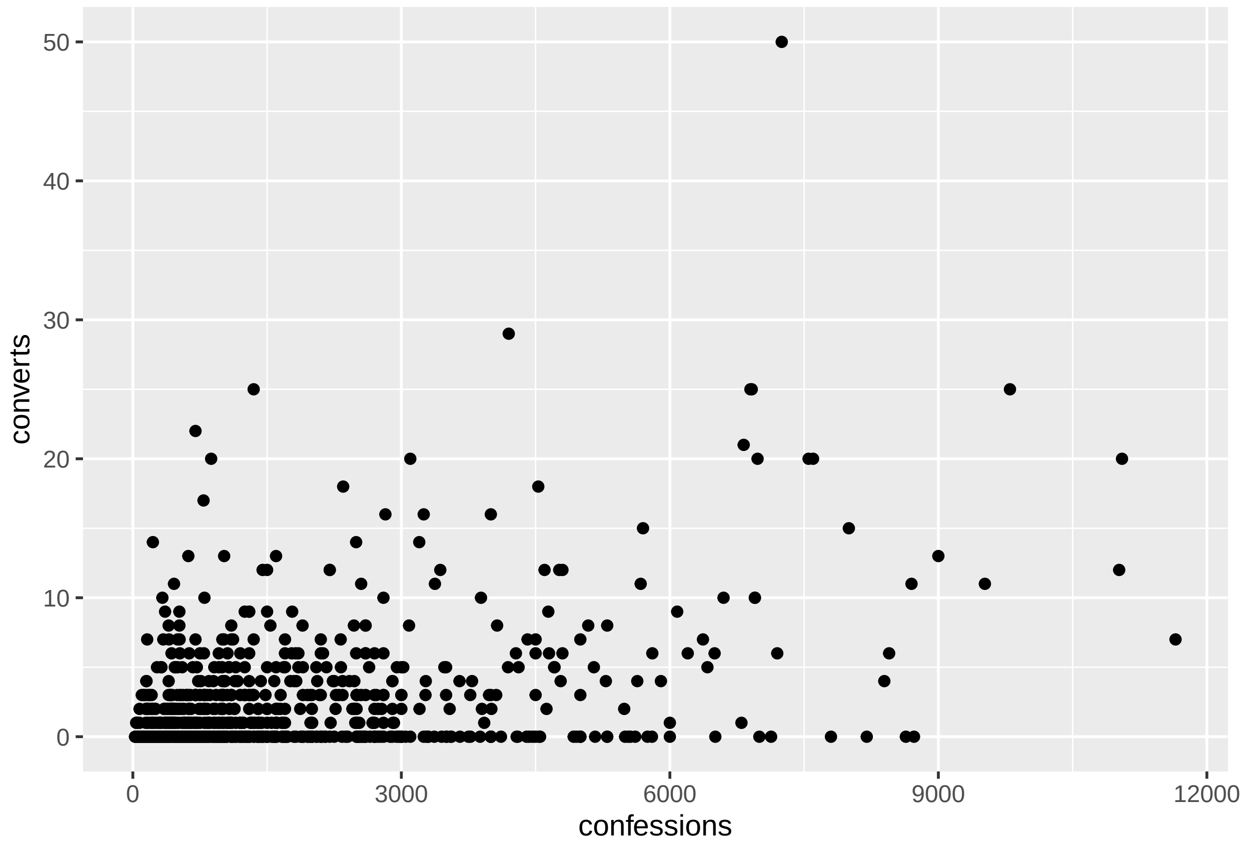
```
paulist_missions
```

```
## # A tibble: 841 x 17
##    mission_number church          city  state date_start date_end    year decade
##             <int> <chr>           <chr> <chr> <date>     <date>     <dbl> <chr>
##  1              1 St. Joseph's C~ New ~ NY    1851-04-06 1851-04-20  1851 1850s
##  2              2 St. Michael's ~ Lore~ PA    1851-04-27 1851-05-11  1851 1850s
##  3              3 St. Mary's Chu~ Holl~ PA    1851-05-18 1851-05-28  1851 1850s
##  4              4 Church of St. ~ John~ PA    1851-05-31 1851-06-08  1851 1850s
##  5              5 St. Peter's Ch~ New ~ NY    1851-09-28 1851-10-12  1851 1850s
##  6              6 St. Patrick's ~ New ~ NY    1851-10-19 1851-11-03  1851 1850s
##  7              7 St. Patrick's ~ Erie  PA    1851-11-17 1851-11-28  1851 1850s
##  8              8 St. Philip Ben~ Cuss~ PA    1851-12-01 1851-12-08  1851 1850s
##  9              9 St. Vincent's ~ Youn~ PA    1851-12-10 1851-12-19  1851 1850s
## 10             10 St. Peter's Ch~ Sara~ NY    1852-01-11 1852-01-22  1852 1850s
## # ... with 831 more rows, and 9 more variables: duration_days <int>,
## #   duration_weeks <ord>, confessions <int>, converts <int>, order <chr>,
## #   lat <dbl>, long <dbl>, volume <int>, page <int>
```

Now let's make a scatter plot.

```
ggplot(paulist_missions, aes(x = confessions, y = converts)) +
  geom_point()
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```



(1) What are the three parts of the plot, as listed above?

1. The dataset: paulist_missions
2. The specification of variables: confessions and converts
3. The glyph: geom_point()

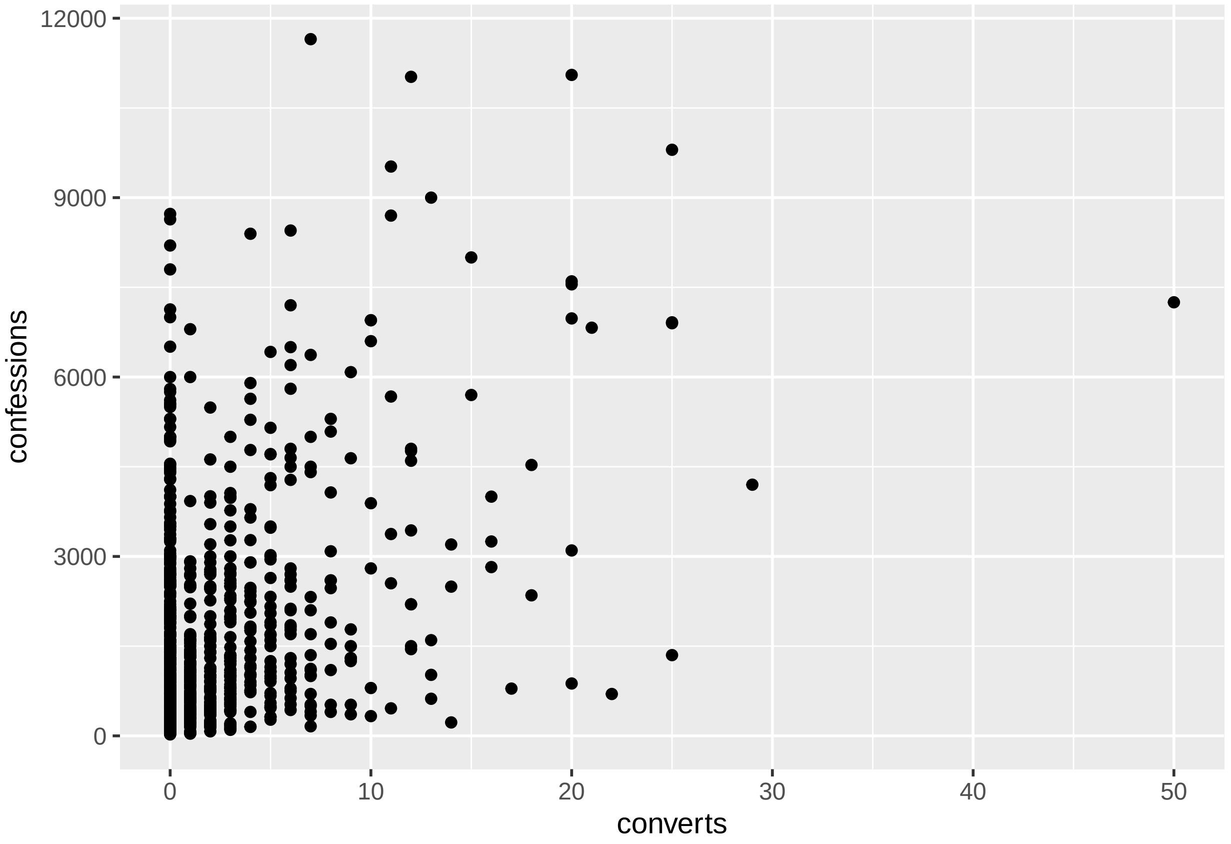
(2) What is the relationship between each row in the dataset and each glyph in the plot?

A glyph in the plot represents one line. It is positioned according to the quantity of confessions (x axis) and converts (y axis).

(3) Modify that plot so that converts are on the x-axis and confessions are y-axis.

```
ggplot(paulist_missions, aes(y = confessions, x = converts)) +
  geom_point()
```

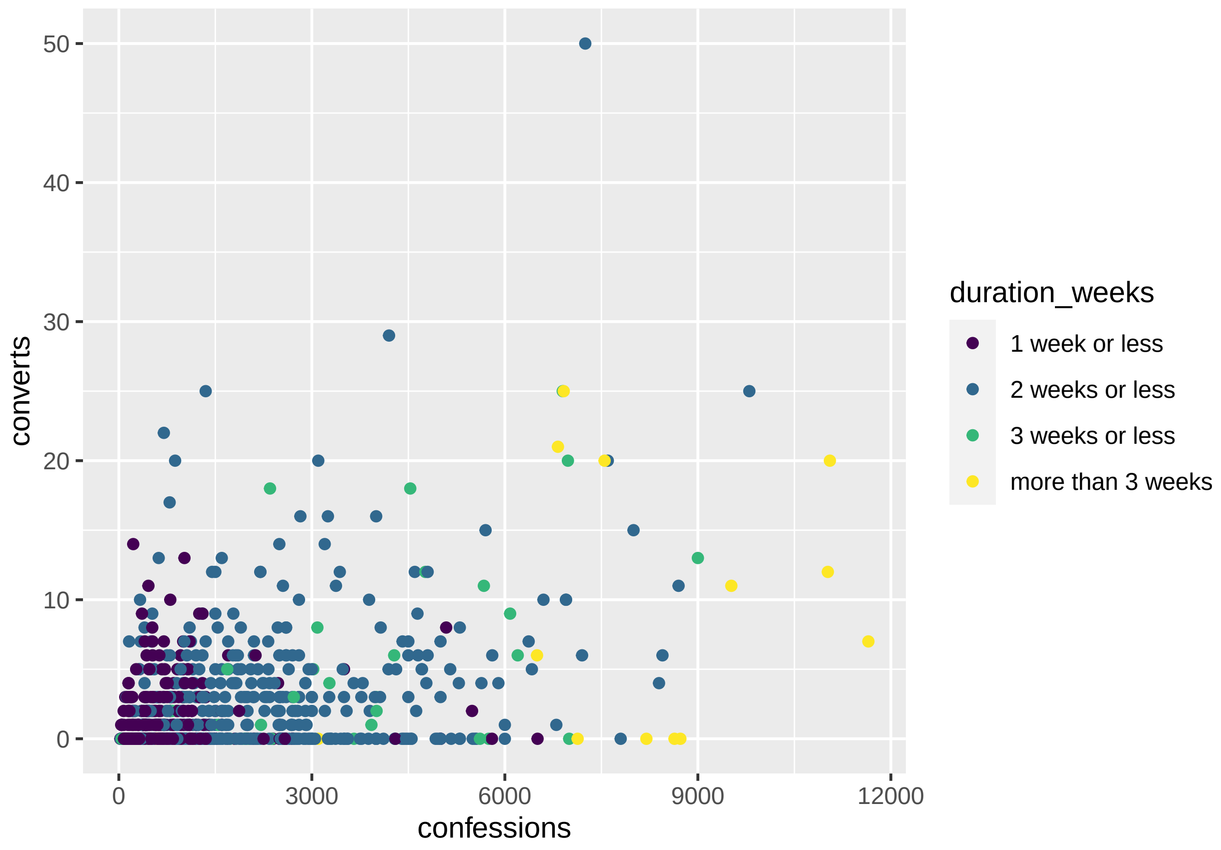
```
## Warning: Removed 6 rows containing missing values (geom_point).
```



We can specify more than two variables and aesthetics if we wish. Here we map the duration (notice: a categorical variable) to color.

```
ggplot(paulist_missions, aes(x = confessions, y = converts, color = duration_weeks)) +
  geom_point()
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```

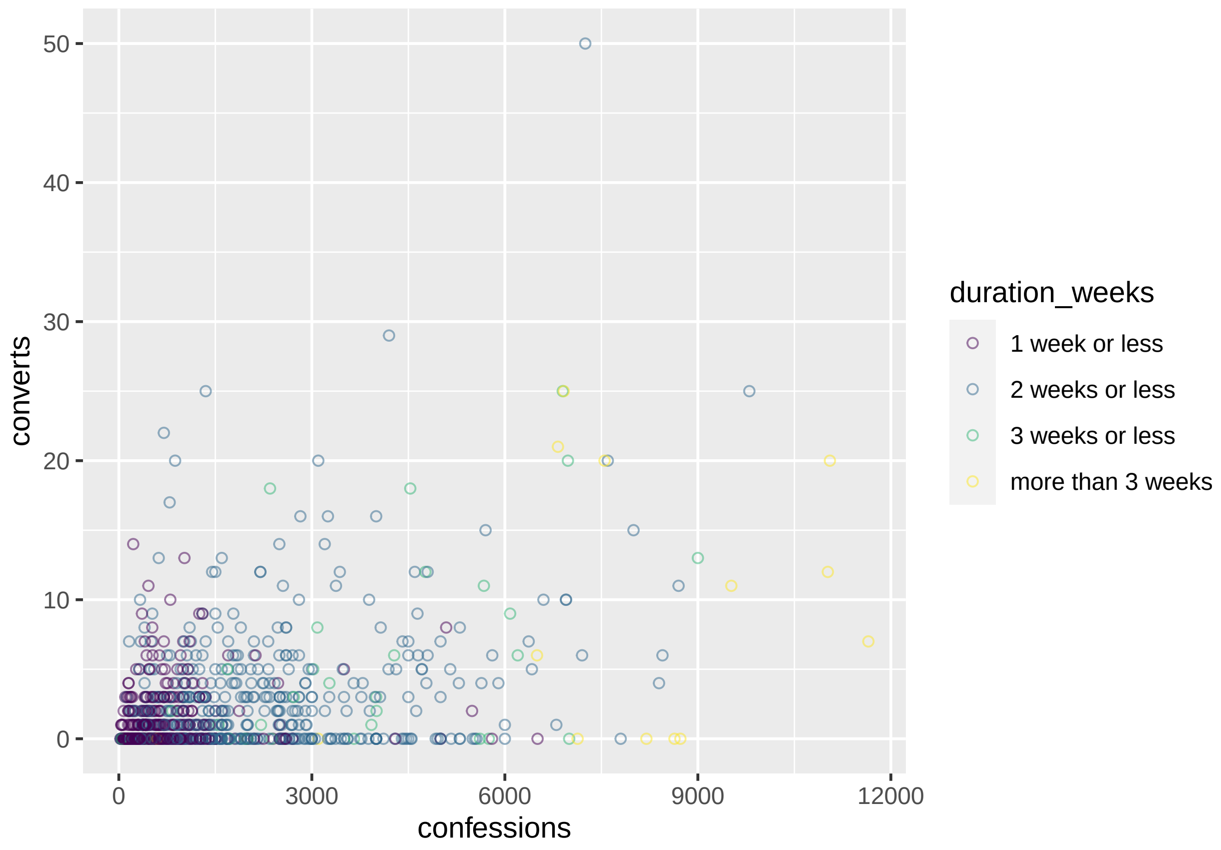


We can also specify static properties, These can go either in the call to `ggplot()` if they affect the entire plot, or in a specific layer (one of the `geom_*()` functions) if they affect just that layer.

We might notice that our chart suffers from overplotting: the points are on top of each other and we can't distinguish between them. Let try changing the shape of each point, and try making each point slightly transparent to see if this helps. Notice that in the code below, those properties are specified with static values *outside of* the `aes()` function.

```r
ggplot(paulist_missions, aes(x = confessions, y = converts, color = duration_weeks)) +
  geom_point(alpha = 0.5, shape = 1)
```

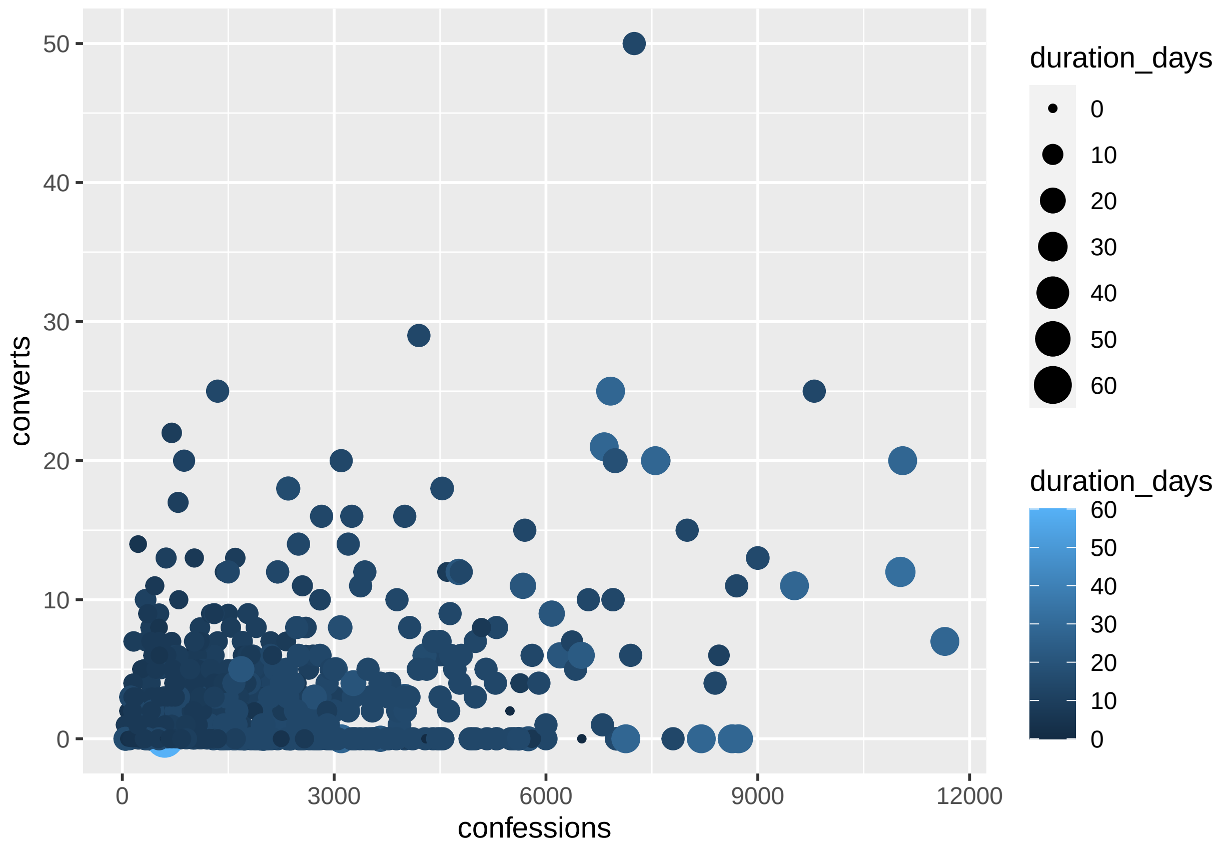
```
## Warning: Removed 6 rows containing missing values (geom_point).
```



(4) Make a different plot from any of the above using `days`, `converts`, and `confessions` variables. Try using the `x`, `y`, and `size` properties.

```
ggplot(paulist_missions, aes(x = confessions, y = converts, size=duration_days, color=duration_days)) +
  geom_point()
```

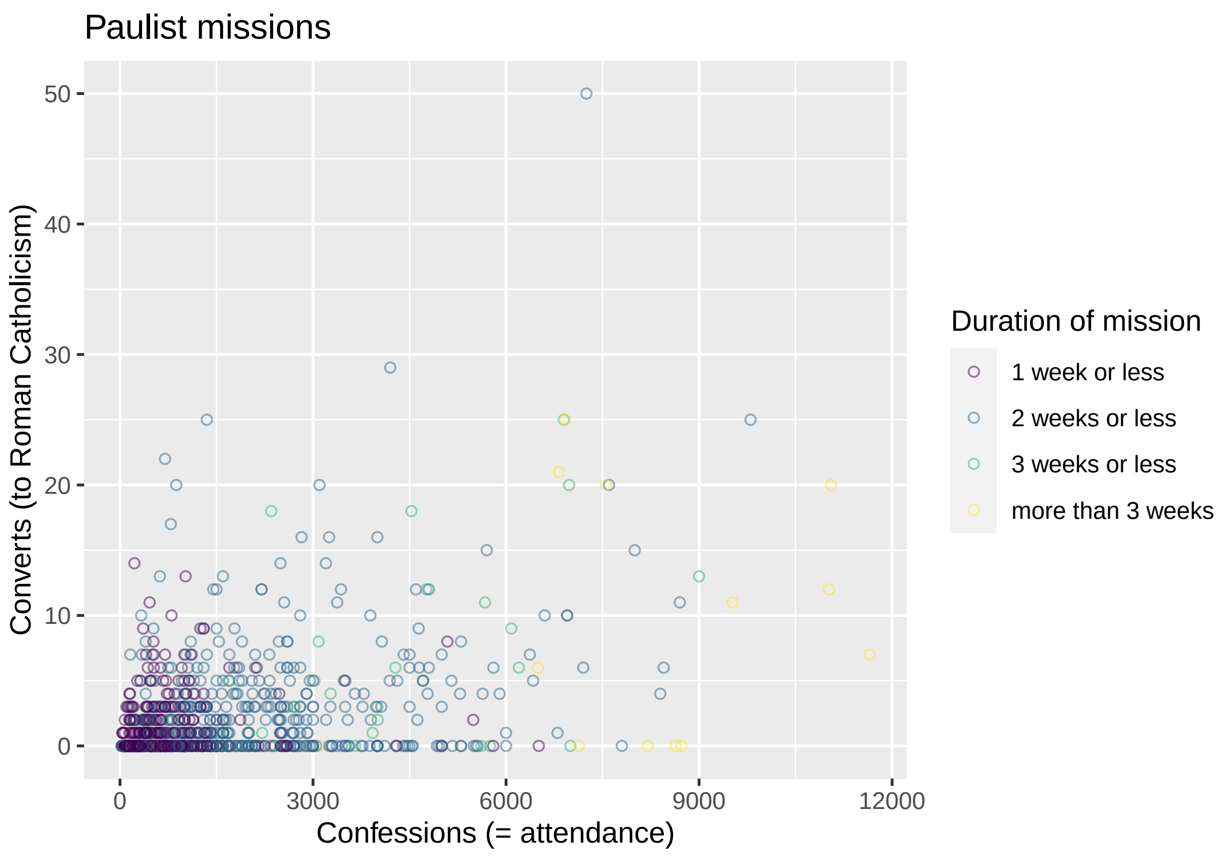
```
## Warning: Removed 6 rows containing missing values (geom_point).
```



We can change the labels of the plot using the `labs()` function as below. (Alternatively, you can use the `xlab()`, `ylab()`, and `ggtitle()` functions.)

```
ggplot(paulist_missions, aes(x = confessions, y = converts,
                             color = duration_weeks)) +
  geom_point(alpha = 0.5, shape = 1) +
  labs(title = "Paulist missions",
       x = "Confessions (= attendance)",
       y = "Converts (to Roman Catholicism)",
       color = "Duration of mission")
```

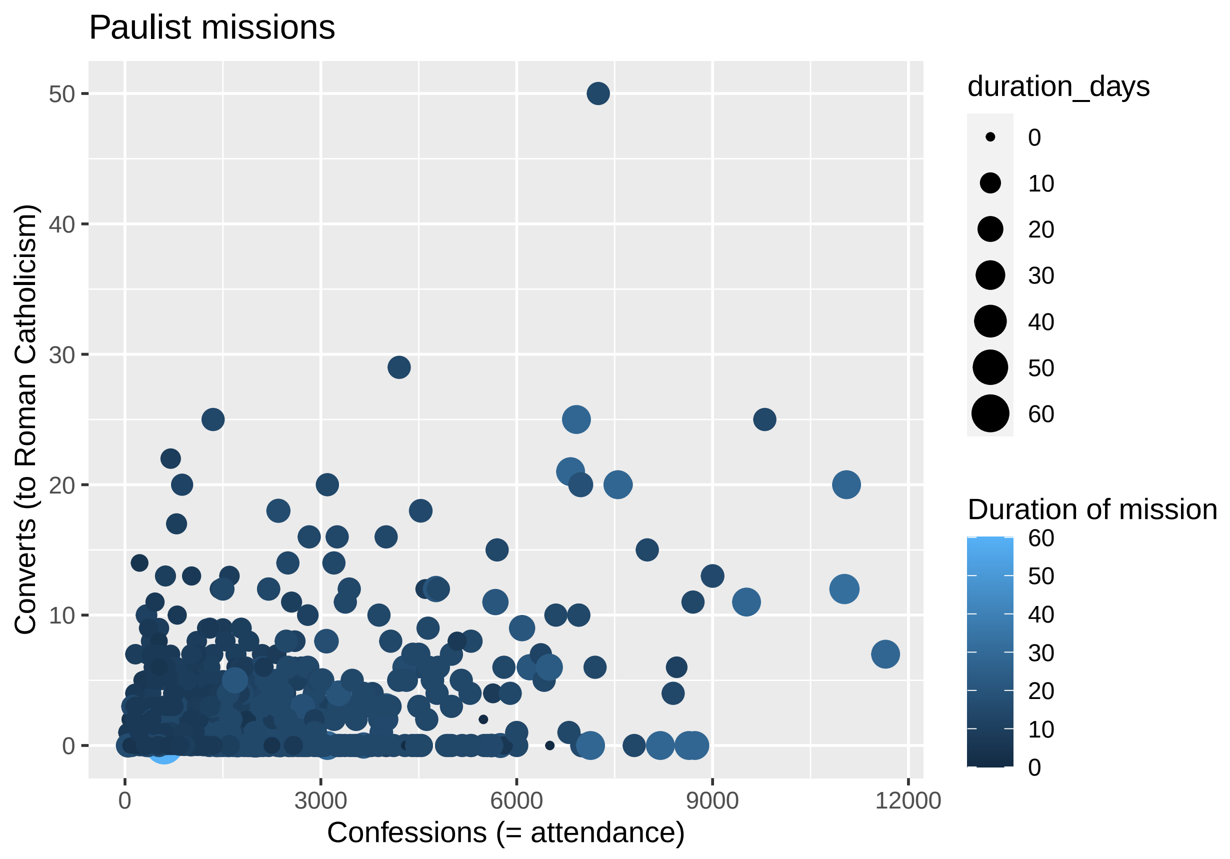
```
## Warning: Removed 6 rows containing missing values (geom_point).
```



(5) Copy your plot above and add informative labels.

```
ggplot(paulist_missions, aes(x = confessions, y = converts, size=duration_days, color=duration_days)) +
  geom_point() +
labs(title = "Paulist missions",
       x = "Confessions (= attendance)",
       y = "Converts (to Roman Catholicism)",
       color = "Duration of mission")
```

```
## Warning: Removed 6 rows containing missing values (geom_point).
```



Basic geoms in ggplot2

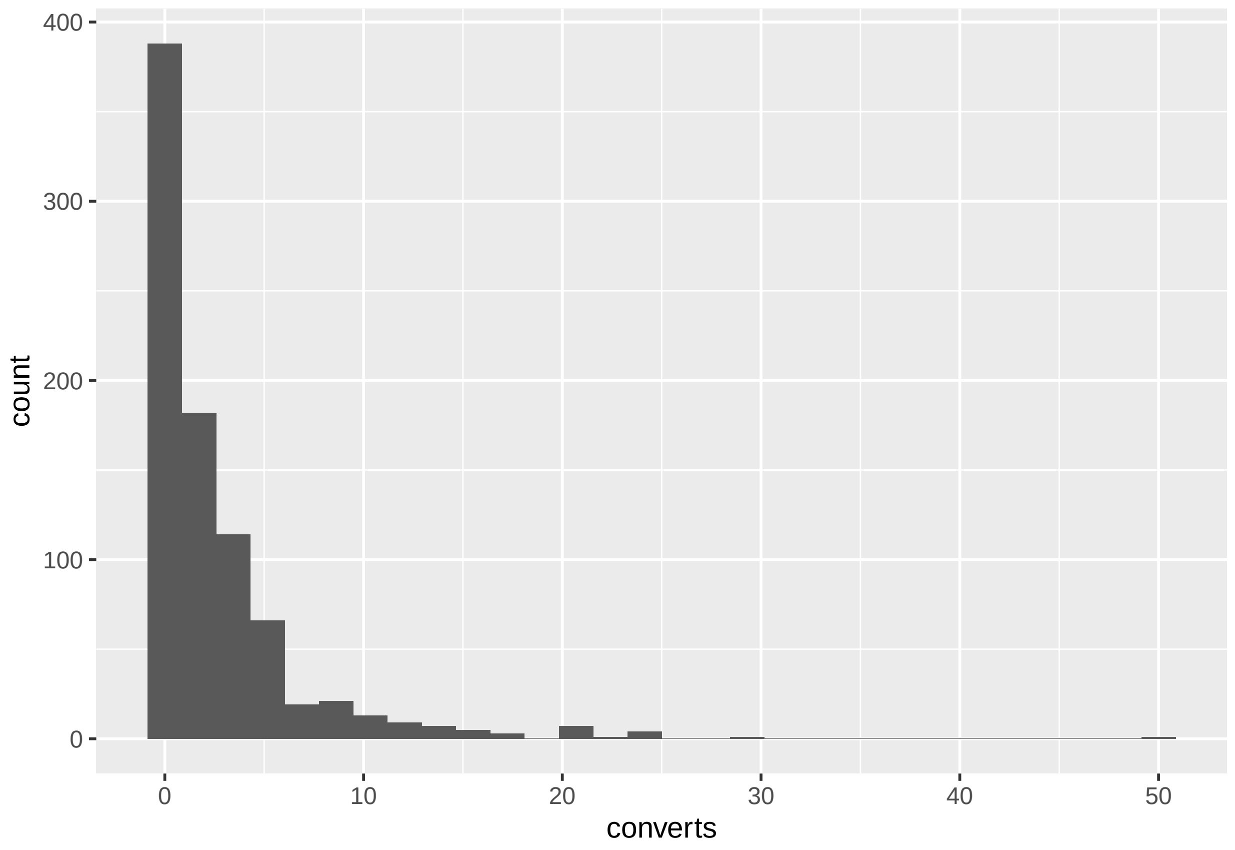
So far we have only used points (with `geom_point()`) as the meaningful glyphs in our plot. Now we will take a tour of different kinds of glyphs that are available to us in ggplot2. Not every variable is suited to every kind of glyph, and sometimes we have to aggregate our data to make certain kinds of plots. (The data aggregation will be covered in a later worksheet.)

Histogram

A histogram shows the distribution of values in a dataset by "binning" the data: in other words, it takes the domain of the data, splits it into different bins, then counts how many values fall into each bin. One bar is drawn for each bin. The height of the bin represents the number of items, whose values fall into the range of a given bin.

```r
ggplot(paulist_missions, aes(x = converts)) +
  geom_histogram()
```

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

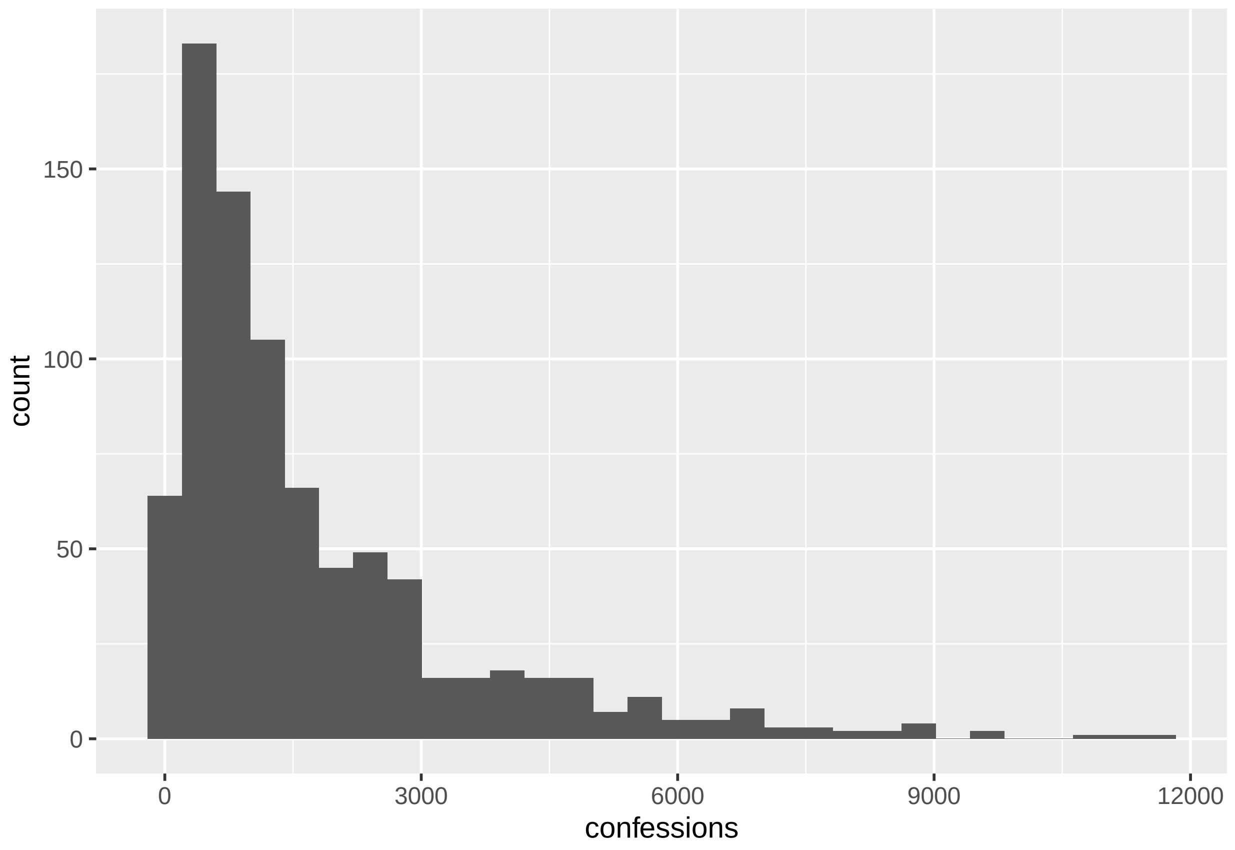


(6) Create a histogram of the number of confessions.

```
ggplot(paulist_missions, aes(x = confessions)) +
  geom_histogram()
```

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

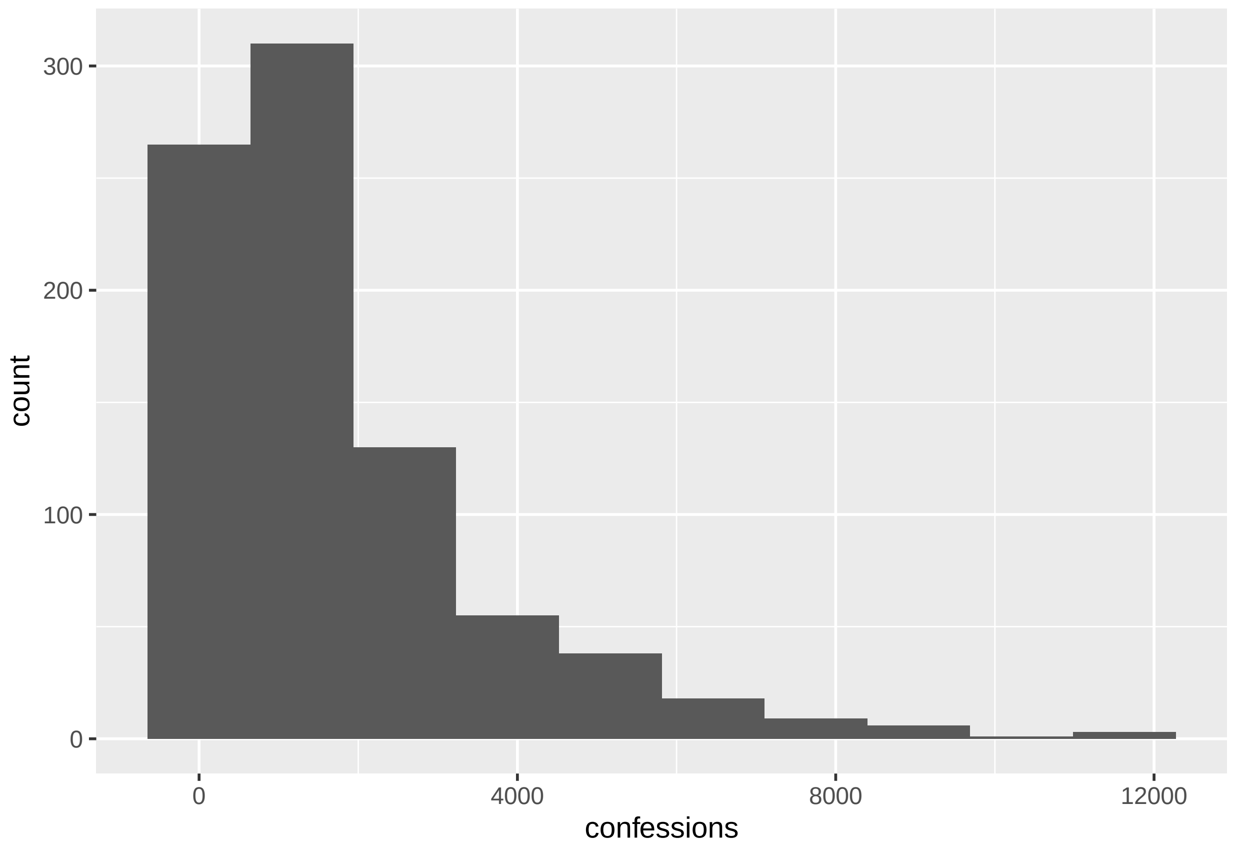
```
## Warning: Removed 6 rows containing non-finite values (stat_bin).
```



(7) Can you change the number of bins? (Hint: try `bins =` or `binwidth =`. See `?geom_histogram`.)

```
ggplot(paulist_missions, aes(x = confessions)) +
  geom_histogram(bins = 10)
```

```
## Warning: Removed 6 rows containing non-finite values (stat_bin).
```

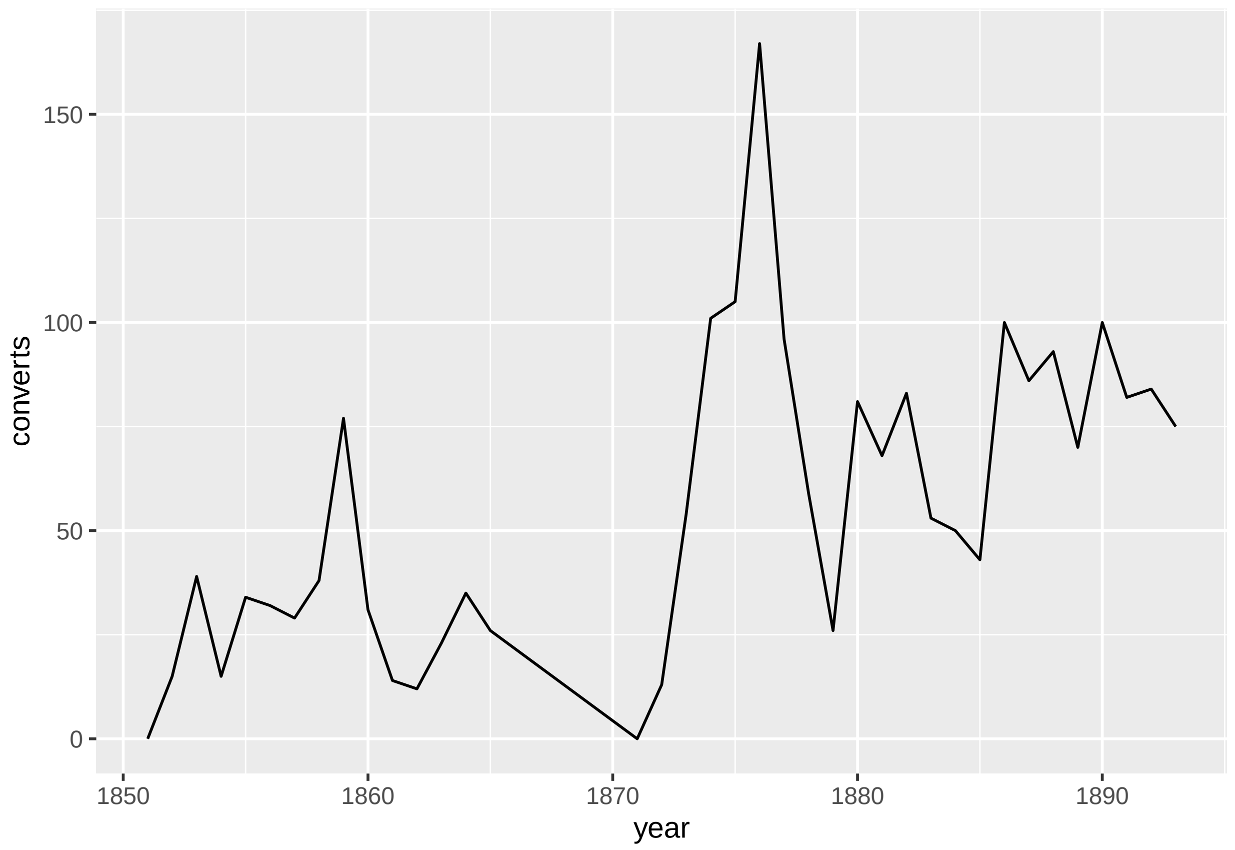


Bin parameters can be used to modify bin properties. For example, `bins=30` (example: `geom_histogram(bins=20)`) will distribute all data into 30 bins; `binwidth=20` (example: `geom_histogram(binwidth=20)`) will distribute data into bins of width 20. For other parameters, check ggplot2.tidyverse.org/reference/geom_histogram.

Lines

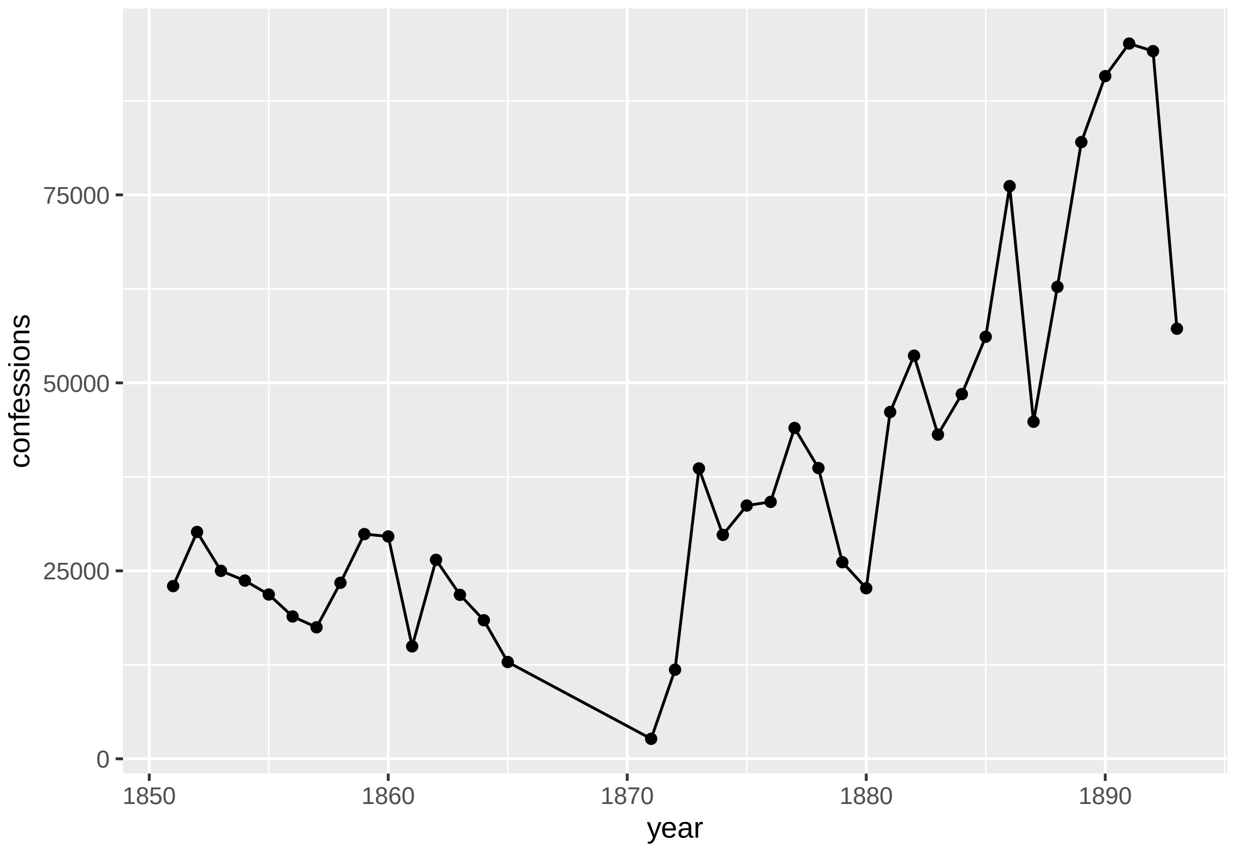
Lines are good for showing trends.

```r
ggplot(paulists_by_year, aes(x = year, y = converts)) +
  geom_line()
```



(8) Create a line chart of the number of confessions. Can you also add a layer of points in addition to the line?

```
ggplot(paulists_by_year, aes(x = year, y = confessions)) +
  geom_line() + geom_point()
```



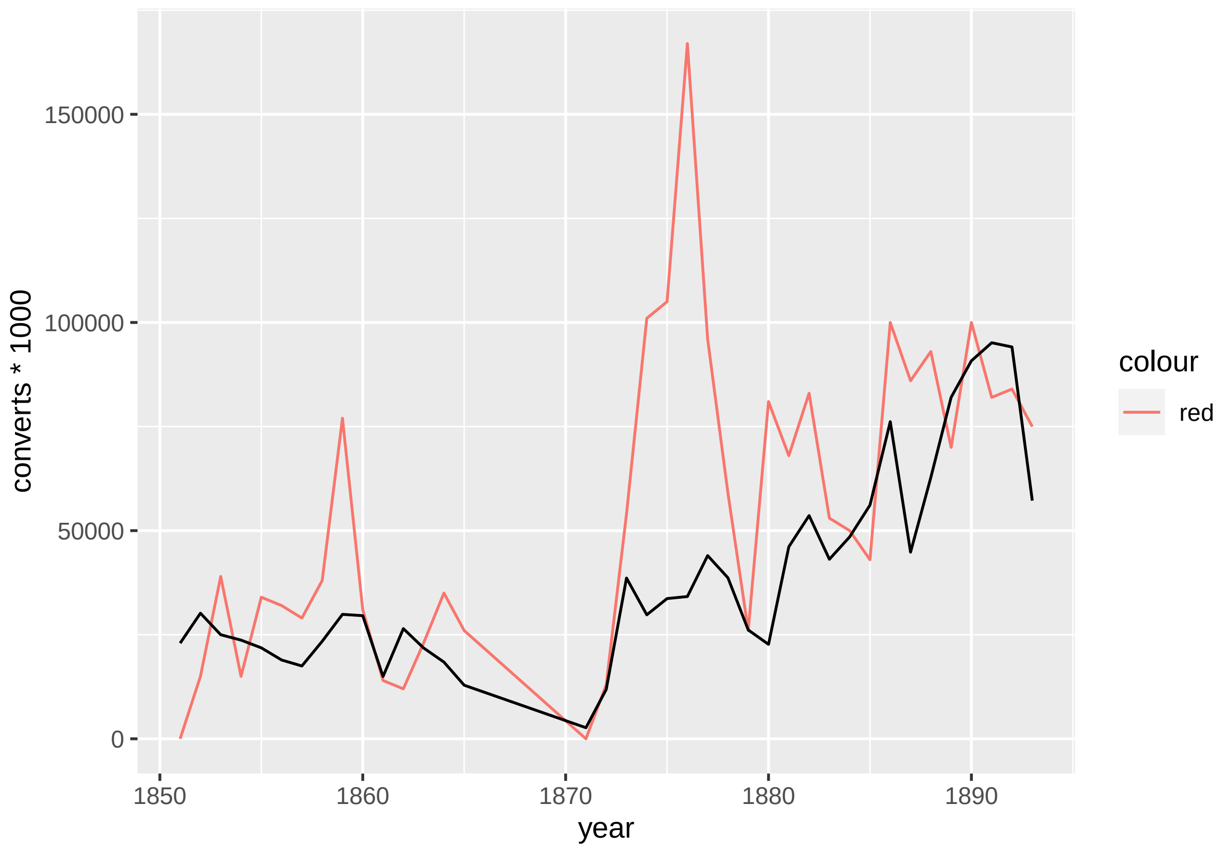
(9) Do you see any issues with the current graph? Please, explain.

There are quite big jumps between data. The graph makes the data look complete and connects two dots even if they are far away.

(10) Can you create a line chart with a line for the number of converts and a line for the number of confessions? (Hint: you will need two calls to `geom_line()`. And instead of specifying the `y` value in the call to `ggplot()` you will do it in the functons for each layer. For instance: `geom_line(aes(y = converts))`.)

MGR Comment: We can barely se the *converts* line. Can you make these lines visually more comparable? (*Hint*: when we are looking at trends, the first thing we want to see is ups and downs).

```
ggplot(paulists_by_year) + geom_line(aes(x = year, y = converts * 1000, color="red")) +
  geom_line(aes(x = year, y = confessions))
```

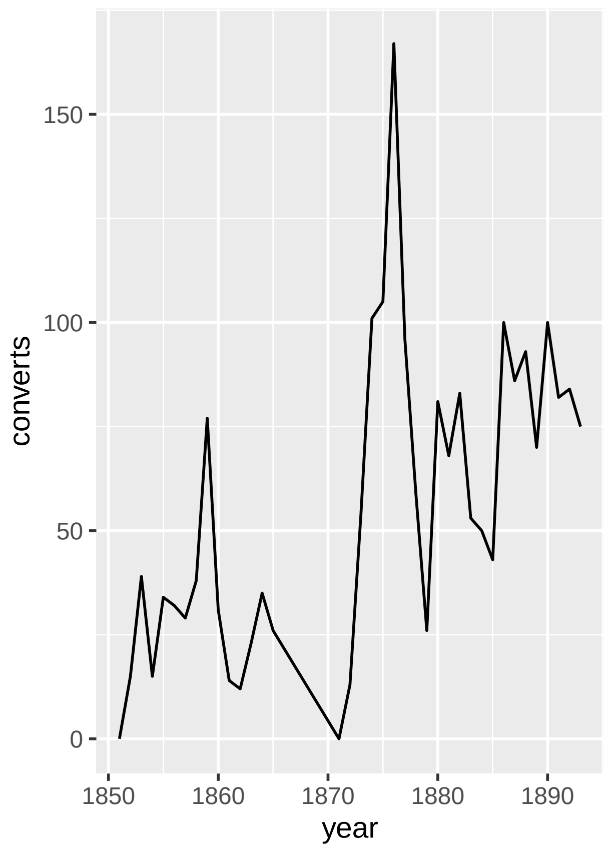


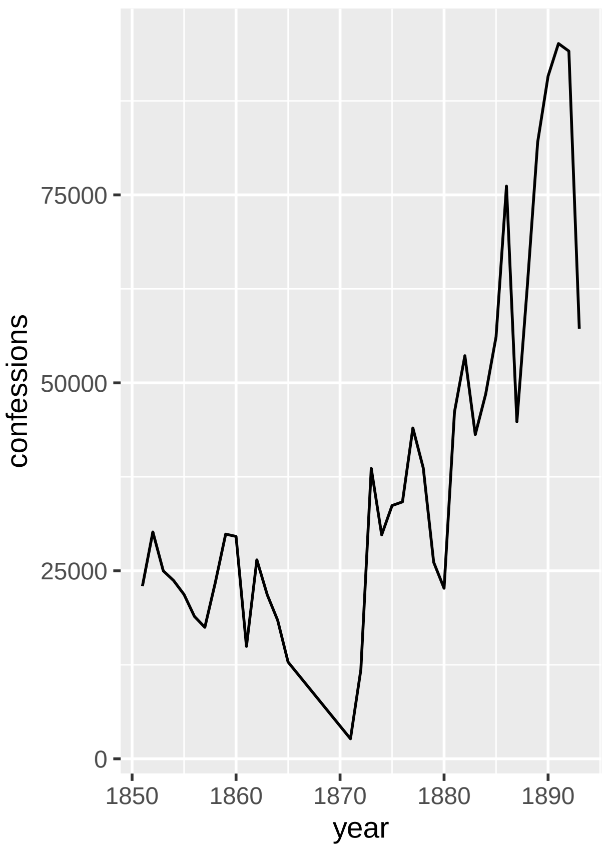
```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

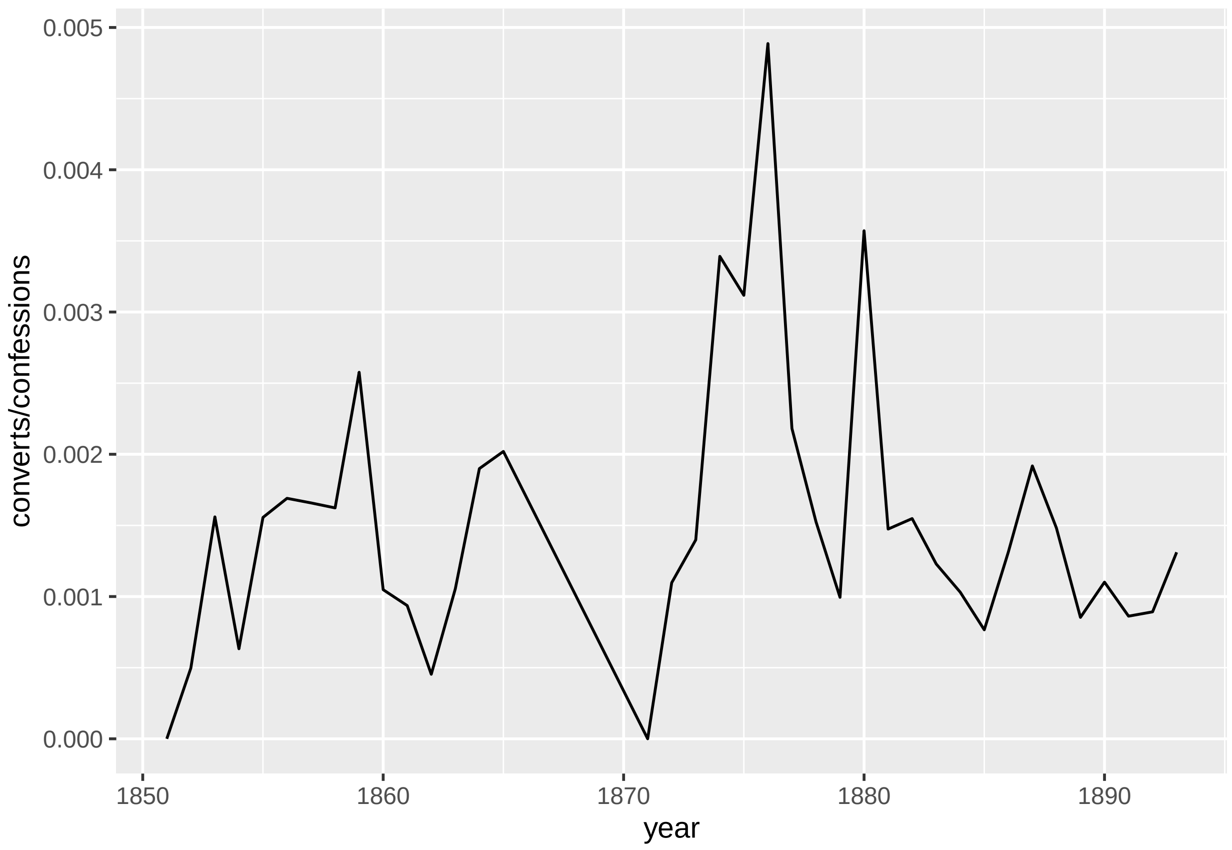
```
plot1 <- ggplot(paulists_by_year) + geom_line(aes(x = year, y = converts))
plot2 <- ggplot(paulists_by_year) + geom_line(aes(x = year, y = confessions))
grid.arrange(plot1, plot2, ncol=2)
```





(11) Can you create a plot with a single line for the ratio of converts to confessions? (Hint: the ratio of converts to confessions is given by `converts / confessions`.)

```
ggplot(data = paulists_by_year) +
  geom_line(aes(x=year, y = converts/confessions))
```



If you map `color =` to a categorical value, you will get a different colored line for each category.

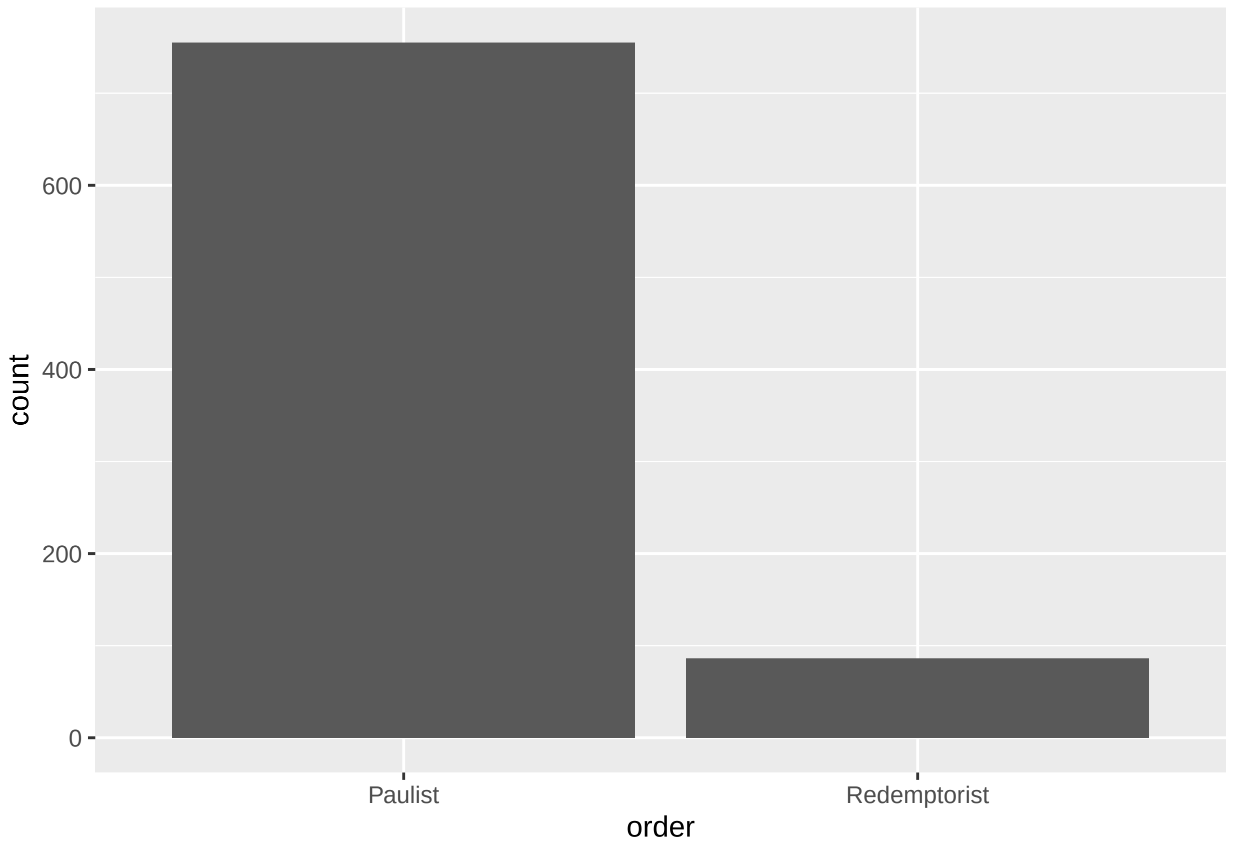
Bar plots

Bar plots can be used in much the same way as a line plot if you use `geom_col()`. That geom tells ggplot to use a `y` value that is present in the data.

MGR Comment: bar plots are better particularly for cases when values in temporal data are missing and earlier values have no effect on later values; the line that connects values implies that influence. Take a close look at the period between 1865 and 1875 on the graph where values are connected with the line and on the graph below where tyour values should be displayed with bars.

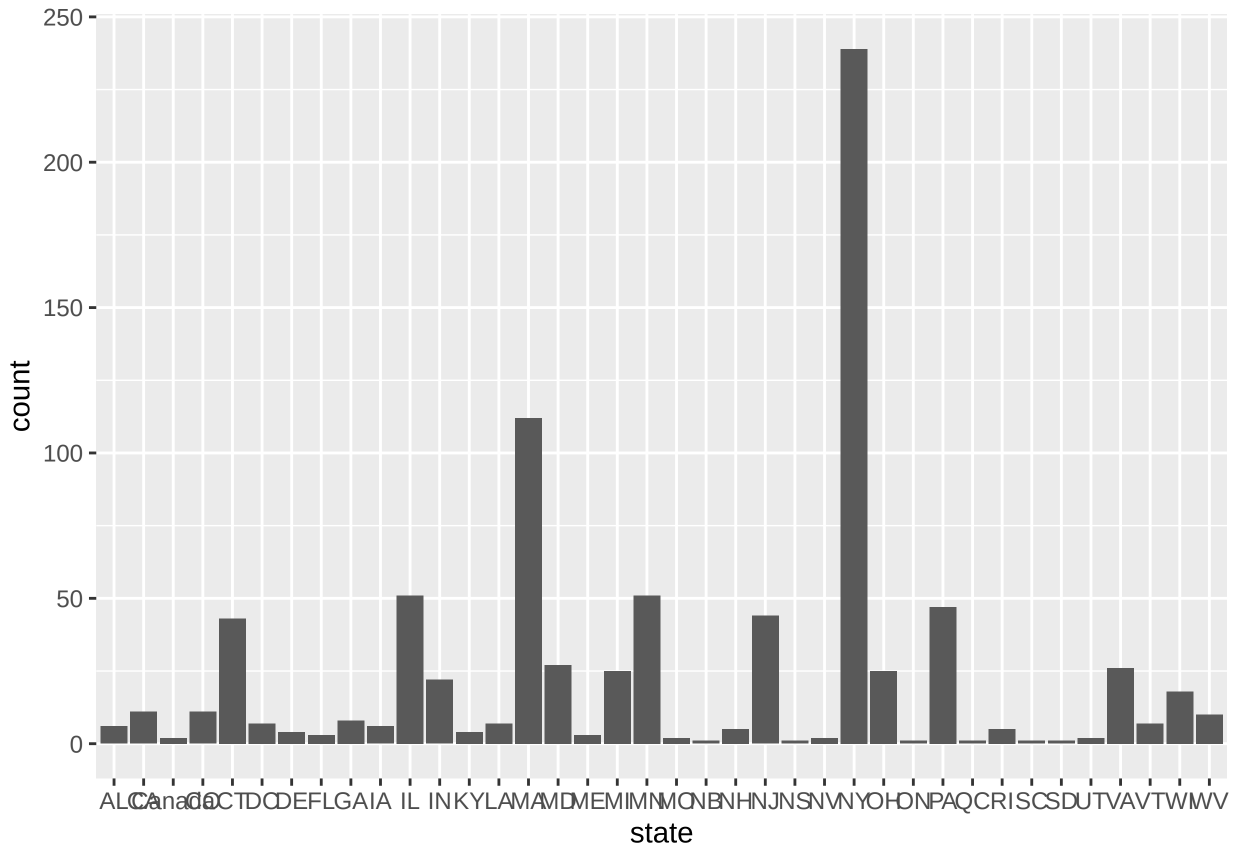
But bar plots are better used for counts of categorical variables. Here we count the number of missions done by the Paulists and the Redemptorists. Notice that is applying a statistical transformation to count the number of observations in each category.

```
ggplot(paulist_missions, aes(x = order)) +
  geom_bar()
```



(12) Create a plot with a count of the number of missions in each state.

```
ggplot(paulist_missions, aes(x = state)) +
  geom_bar()
```



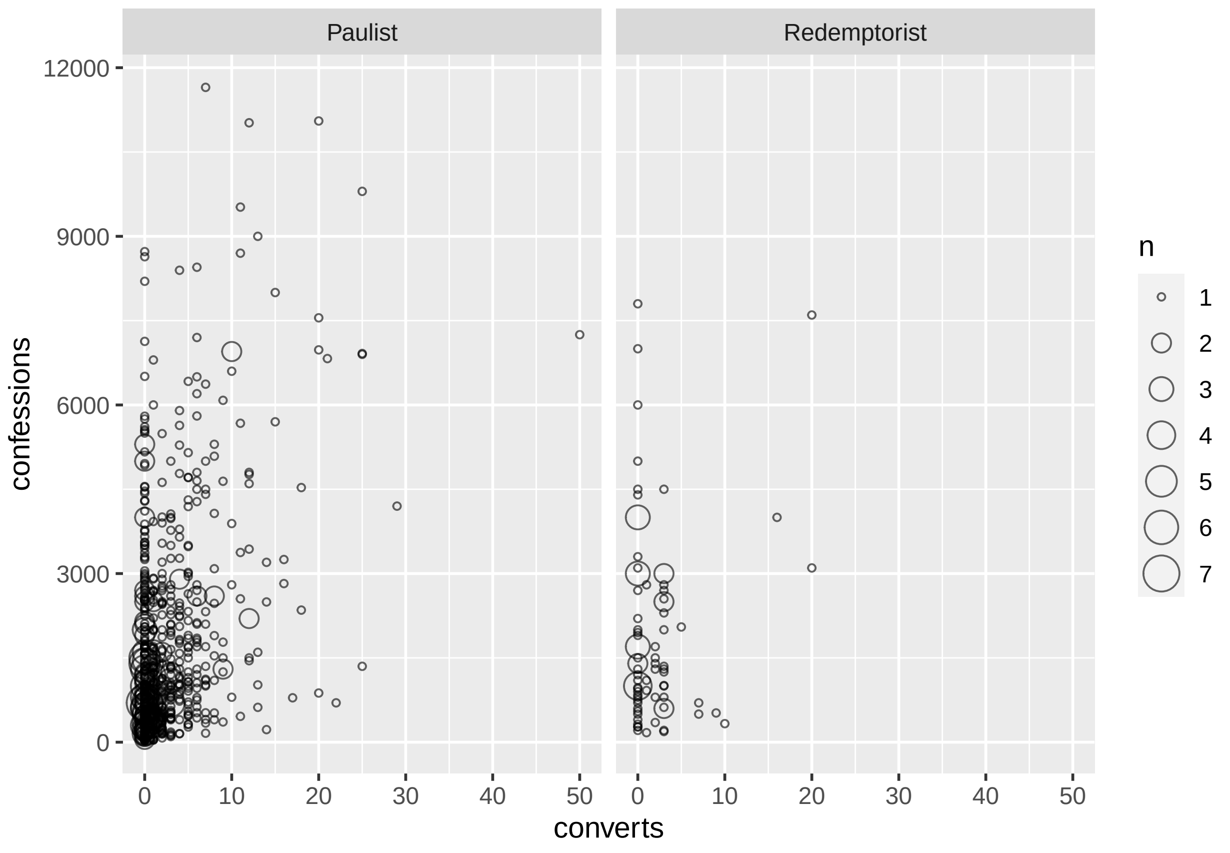
MGR Comment: Such bar graphs are rather useless because of their readability (or, better, the lack of it). Items got ograized alphabetically, but some other principle — from east to west, for example — could work better for this kind of data.

Faceting

Faceting is not a geom like the examples above, but it can create a separate panel in a plot for different categories in the data. For instance, in the plot below, we have created a separate panel for each "order" (i.e., missionary group).

```
ggplot(paulist_missions, aes(x = converts, y = confessions)) +
  geom_count(shape = 1, alpha = 0.6) +
  facet_wrap(~ order)
```

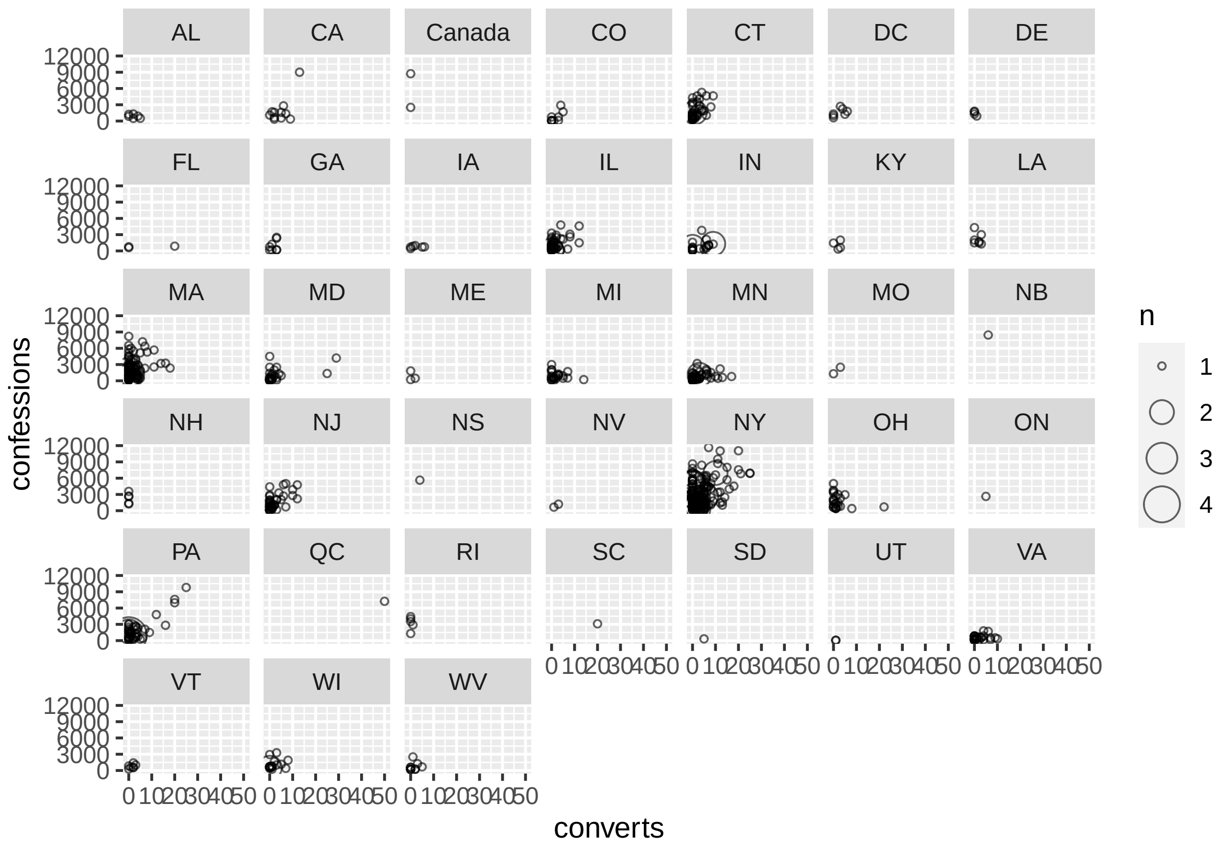
```
## Warning: Removed 6 rows containing non-finite values (stat_sum).
```



(13) Create a plot with facets for each state.

```
ggplot(paulist_missions, aes(x = converts, y = confessions)) +
  geom_count(shape = 1, alpha = 0.6) +
  facet_wrap(~ state)
```

```
## Warning: Removed 6 rows containing non-finite values (stat_sum).
```



(14) Notice that we are using `geom_count()`. What does it do? (Hint: `?geom_count`.)

“counts the number of observations at each given location”: basically it counts the occurences of data in certain columns and plots them.

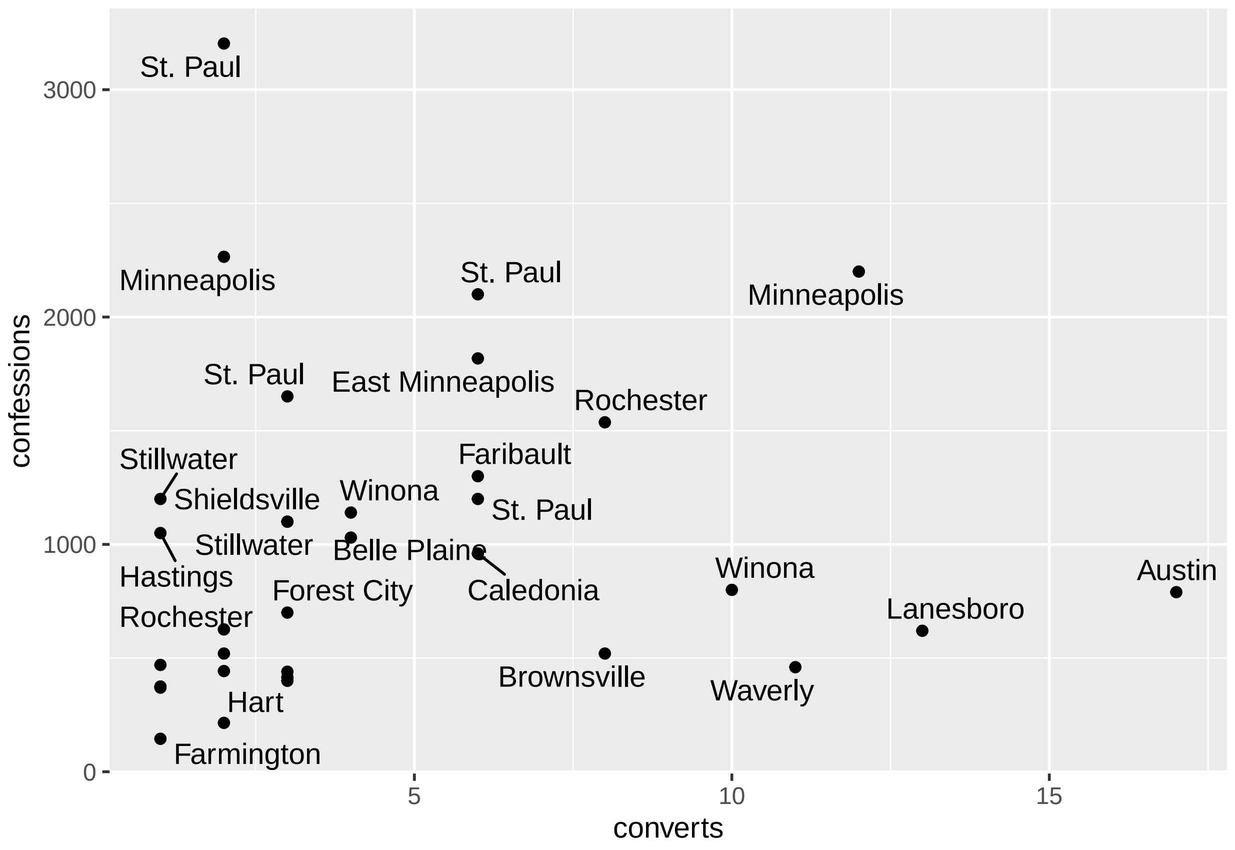
Labels

You can add labels to a plot using `geom_text()`, but it is likely that the labels will overlap with one another. The `ggrepel` package adds a function `geom_text_repel()` which will try to make the labels not overlap, as in this example below. Use this function to label your plots when it makes sense to do so.

```
mn_missions <- paulist_missions %>%
  filter(state == "MN", converts > 0)

ggplot(mn_missions, aes(x = converts, y = confessions)) +
 geom_point() +
  geom_text_repel(aes(label = city))
```

```
## Warning: ggrepel: 8 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



Sometimes it makes sense to add a label only for some points. In this example we create a new column, `label`, and only put the city name in it if there were five or more converts.

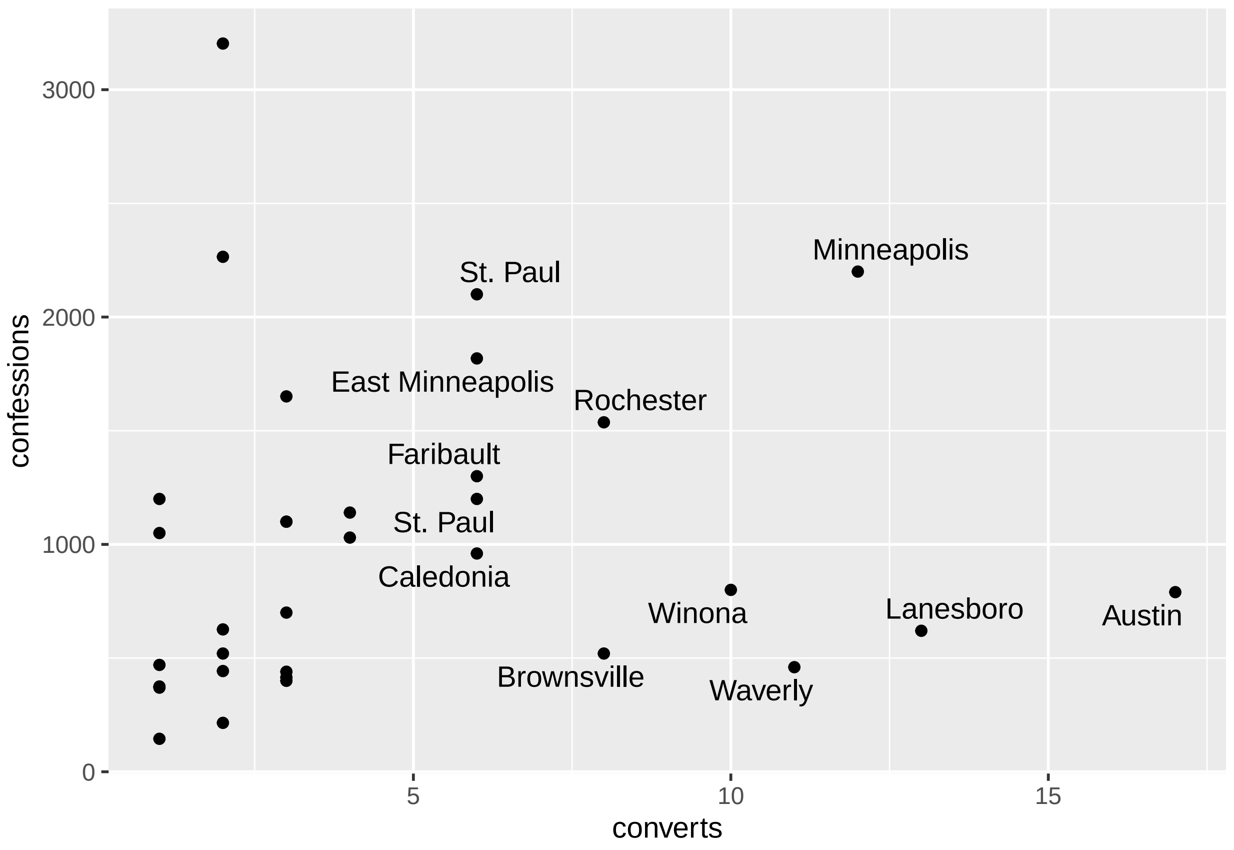
```
mn_missions_with_labels <- paulist_missions %>%
  filter(state == "MN", converts > 0) %>%
  mutate(label = ifelse(converts > 5, city, NA))
```

(15) Make a plot like the one immediately above but using the `mn_missions_with_labels` data frame and the labels in `label` column.

```
mn_missions_label <- mn_missions_with_labels %>%
  filter(state == "MN", converts > 0)

ggplot(mn_missions_label, aes(x = converts, y = confessions)) +
 geom_point() +
  geom_text_repel(aes(label = label))
```

```
## Warning: Removed 21 rows containing missing values (geom_text_repel).
```



Create your own plots

There are a number of data sets available to you. You may try using `early_colleges`, `catholic_dioceses`, `naval_promotions`, `quasi_war`, `sarna`, `us_national_population`, or `us_state_populations` (all from the historydata package), `gapminder` (from the gapminder package), or `europop` (from the europop package).

Create three plots below, using any one or more than one of those datasets. Your three plots should try to make some kind of historical observation. For each plot, include no more than four sentences explaining what you think the plot means. You should try to make each plot as informative as possible by using different geoms and including as many variables as is reasonable in each plot. Be sure to add good titles and labels. Use a different theme if you wish. Explain why the type of variables that you chose (e.g., continuous numeric, categorical) are an appropriate fit to the aesthetics you mapped them to.

You may wish to look at Jenny Bryan's R Graph Catalog or Selva Prabhakaran's "Top 50 ggplot2 Visualizations" to find examples of what you can do with ggplot along with sample code.

(16) Plot 1

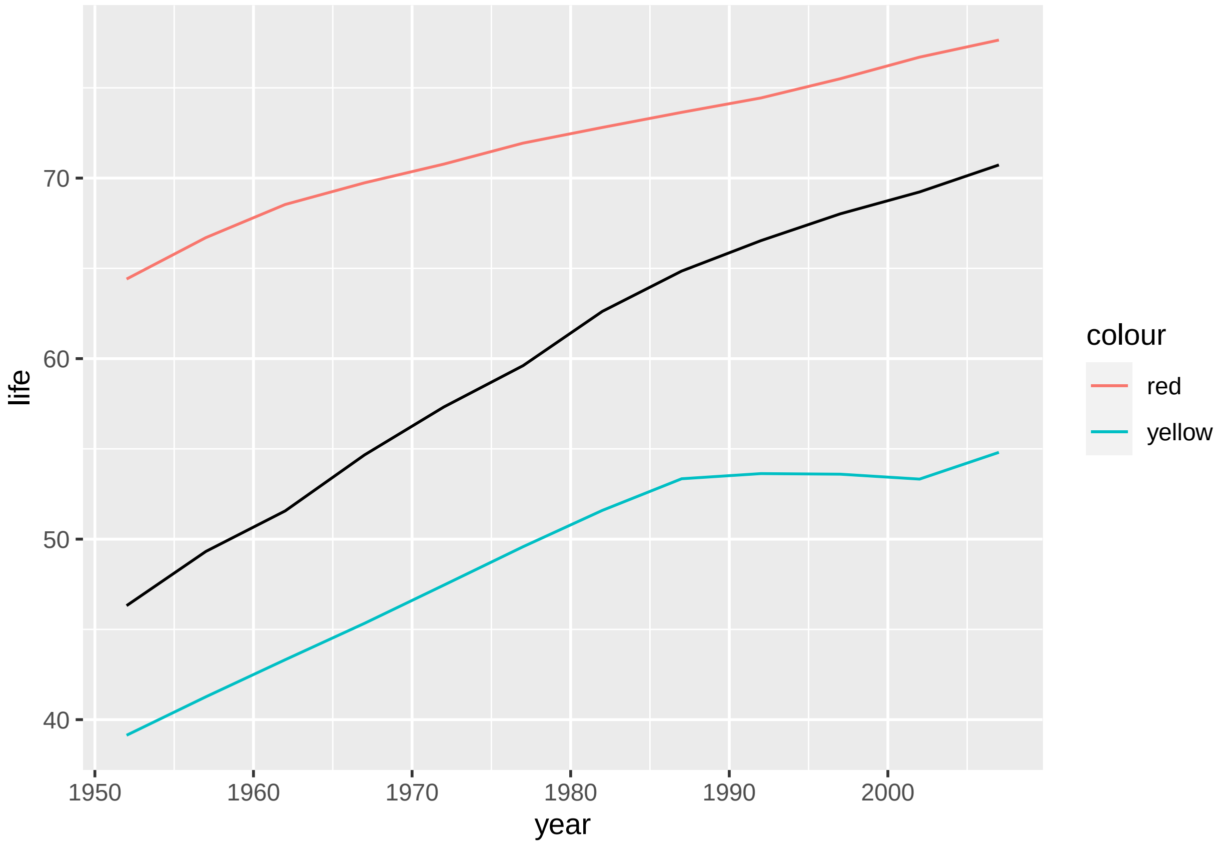
```
europe <- gapminder %>%
  filter(continent == "Europe") %>%
  group_by(year) %>%
  summarise(life = mean(lifeExp))



africa <- gapminder %>%
  filter(continent == "Africa")%>%
  group_by(year) %>%
  summarise(life = mean(lifeExp))
```

```
asia <- gapminder %>%
  filter(continent == "Asia")%>%
  group_by(year) %>%
  summarise(life = mean(lifeExp))


ggplot() + geom_line(data=europe, aes(x = year, y = life, color="red")) +
  geom_line(data= africa, aes(x = year, y = life, color= "yellow")) +
  geom_line(data= asia, aes(x = year, y = life))
```

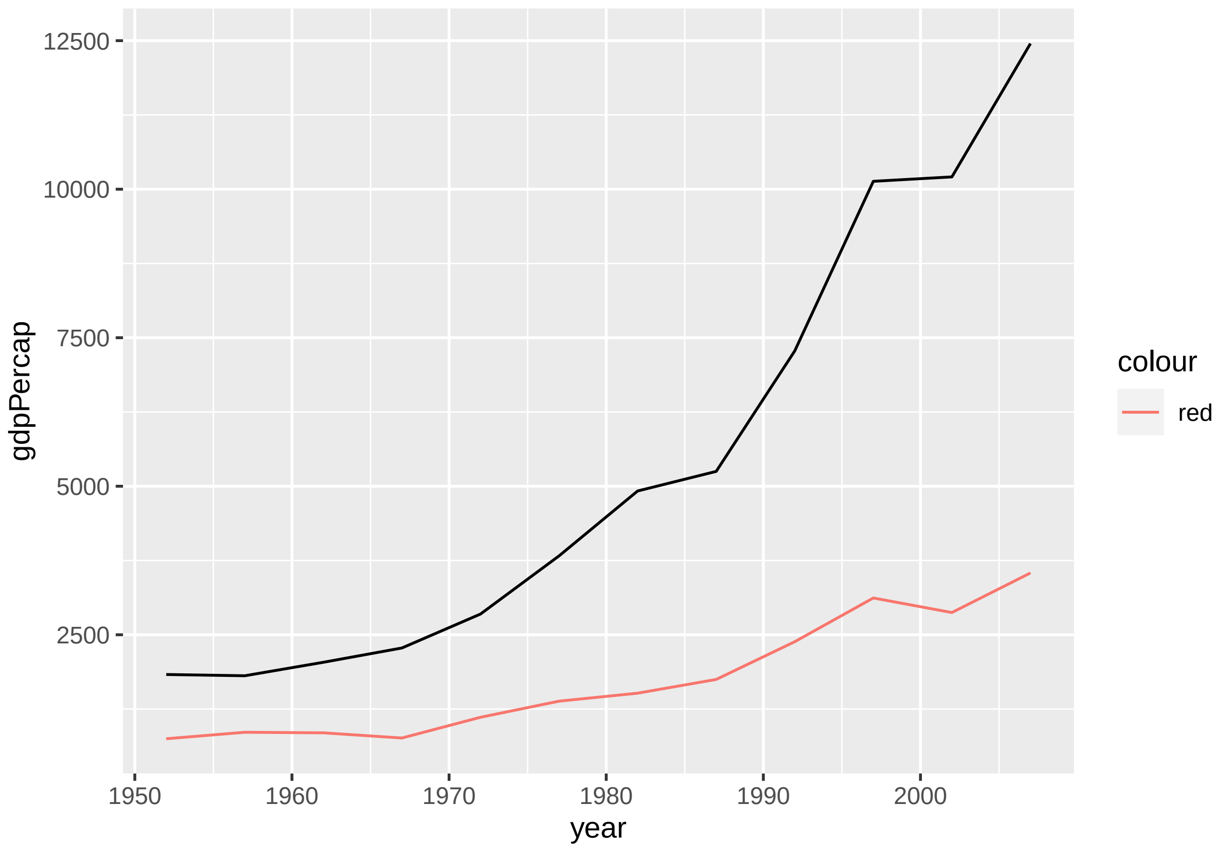


This graph compares the development of life expactancy on the continents of europe, africa, and asia.

(17) Plot 2

```
indo <- gapminder %>%
  filter(country == "Indonesia")
malay <- gapminder %>%
  filter(country == "Malaysia")

ggplot() + geom_line(data=indo, aes(x = year, y = gdpPercap, color="red")) +
  geom_line(data= malay, aes(x = year, y = gdpPercap))
```



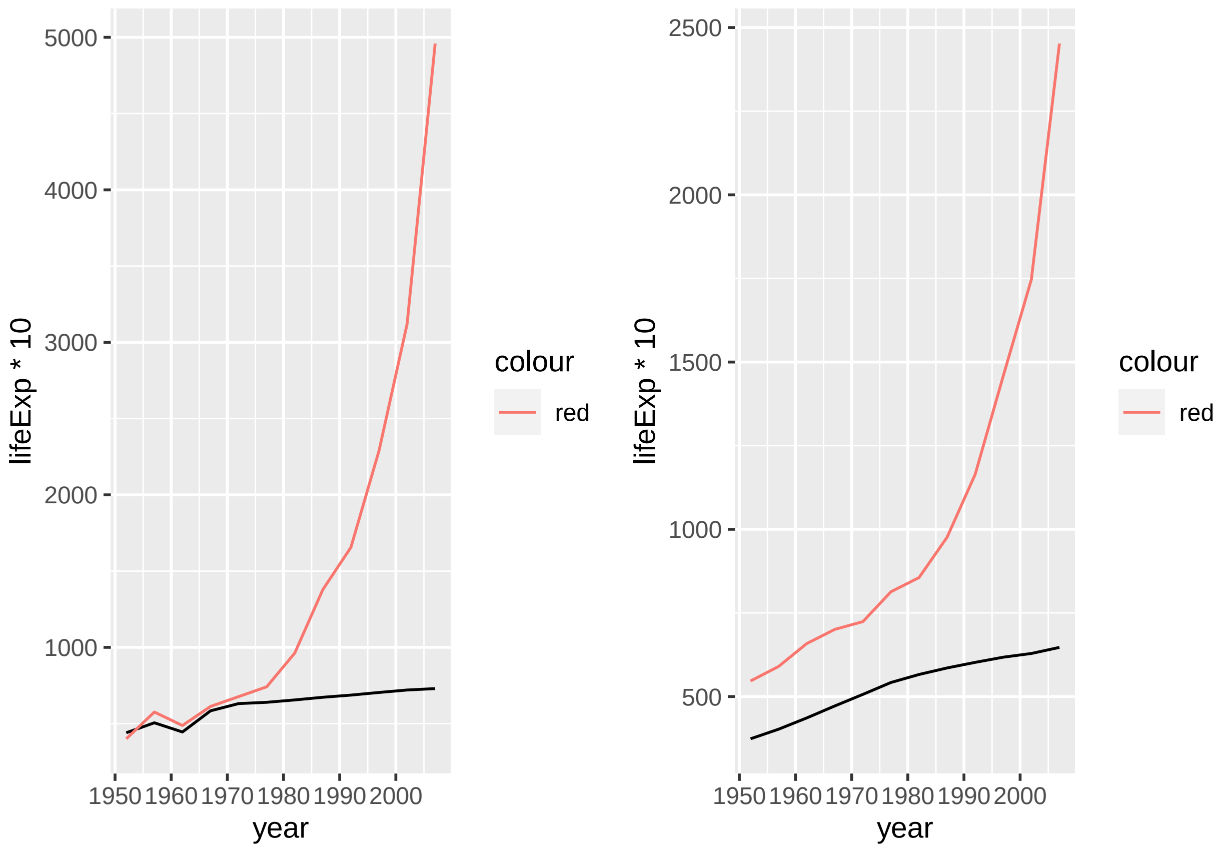
This graph compares the development of growth of the gdp per capita between Indonesia and Malaysia.

(18) Plot 3

```
china <- gapminder %>%
  filter(country == "China")

india <- gapminder %>%
    filter(country == "India")

library(gridExtra)
plot1 <- ggplot(china) + geom_line(aes(x = year, y = lifeExp * 10)) + geom_line(aes(x = year, y = gdpPe
plot2 <- ggplot(india) + geom_line(aes(x = year, y =  lifeExp * 10)) + geom_line(aes(x = year, y = gdpP
grid.arrange(plot1, plot2, ncol=2)
```



This graph compares the connection of life expectancy and gdp per capita in china and india.