

Basics of ggplot2

Daniel Storzreiter

NB: The worksheet has been 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
## [15] 1865 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883
## [29] 1884 1885 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
```

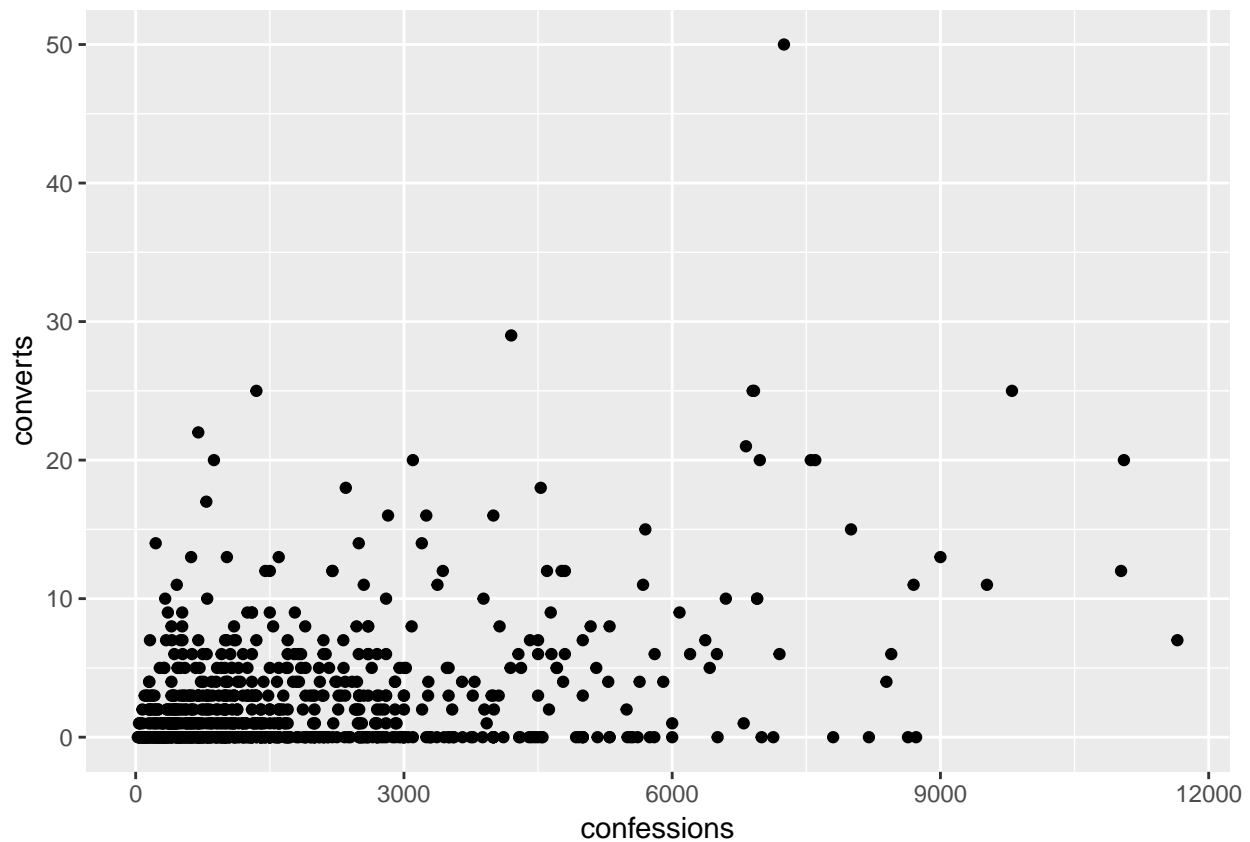
```
## # A tibble: 841 x 17
##   mission_number church city state date_start date_end year decade
##           <int> <chr> <chr> <chr> <date>    <date>    <dbl> <chr>
## 1             1 St. J~ New ~ NY  1851-04-06 1851-04-20  1851 1850s
## 2             2 St. M~ Lore~ PA   1851-04-27 1851-05-11  1851 1850s
## 3             3 St. M~ Holl~ PA   1851-05-18 1851-05-28  1851 1850s
## 4             4 Churc~ John~ PA   1851-05-31 1851-06-08  1851 1850s
## 5             5 St. P~ New ~ NY  1851-09-28 1851-10-12  1851 1850s
## 6             6 St. P~ New ~ NY  1851-10-19 1851-11-03  1851 1850s
## 7             7 St. P~ Erie  PA   1851-11-17 1851-11-28  1851 1850s
## 8             8 St. P~ Cuss~ PA   1851-12-01 1851-12-08  1851 1850s
## 9             9 St. V~ Youn~ PA   1851-12-10 1851-12-19  1851 1850s
```

```
## 10          10 St. P~ 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?

dataset, y axis and x axis incl. axis names

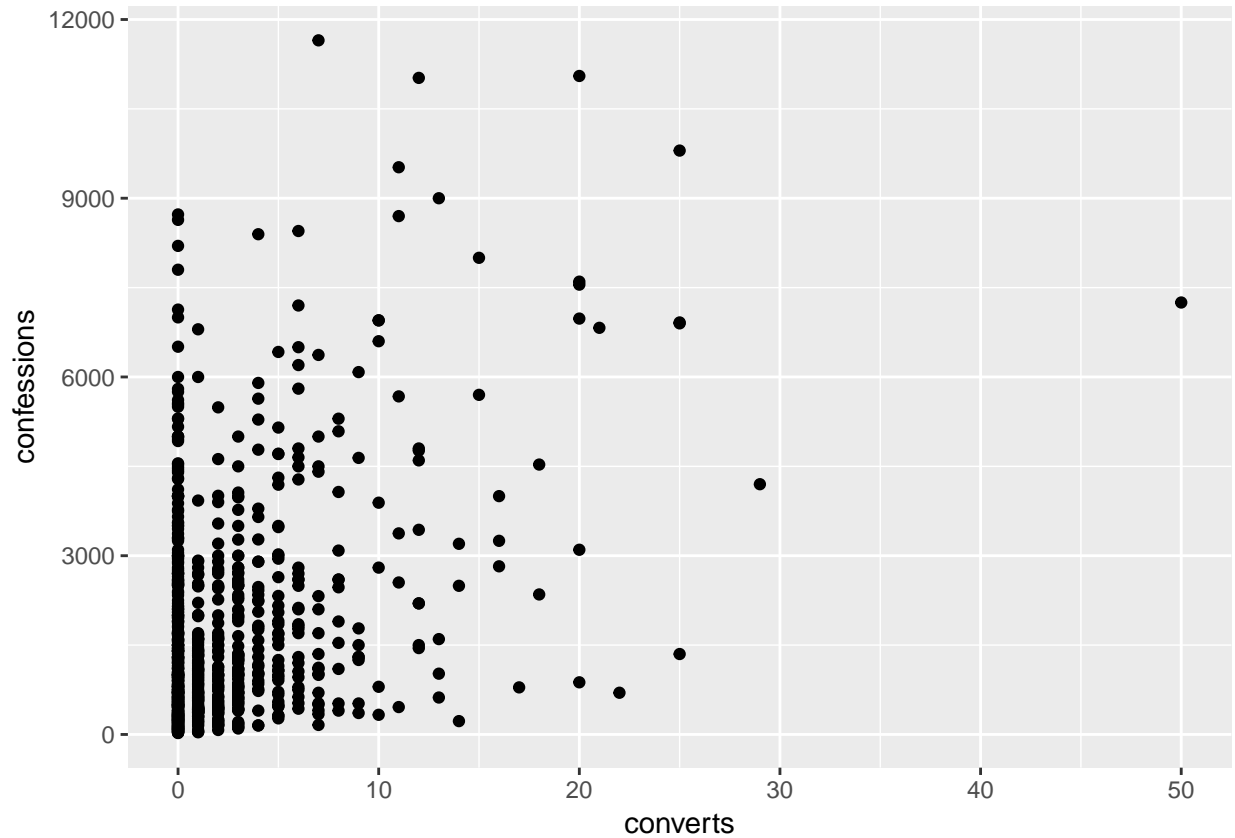
(2) What is the relationship between each row in the dataset and each glyph in the plot?

how many confessions did lead to a conversion

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

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

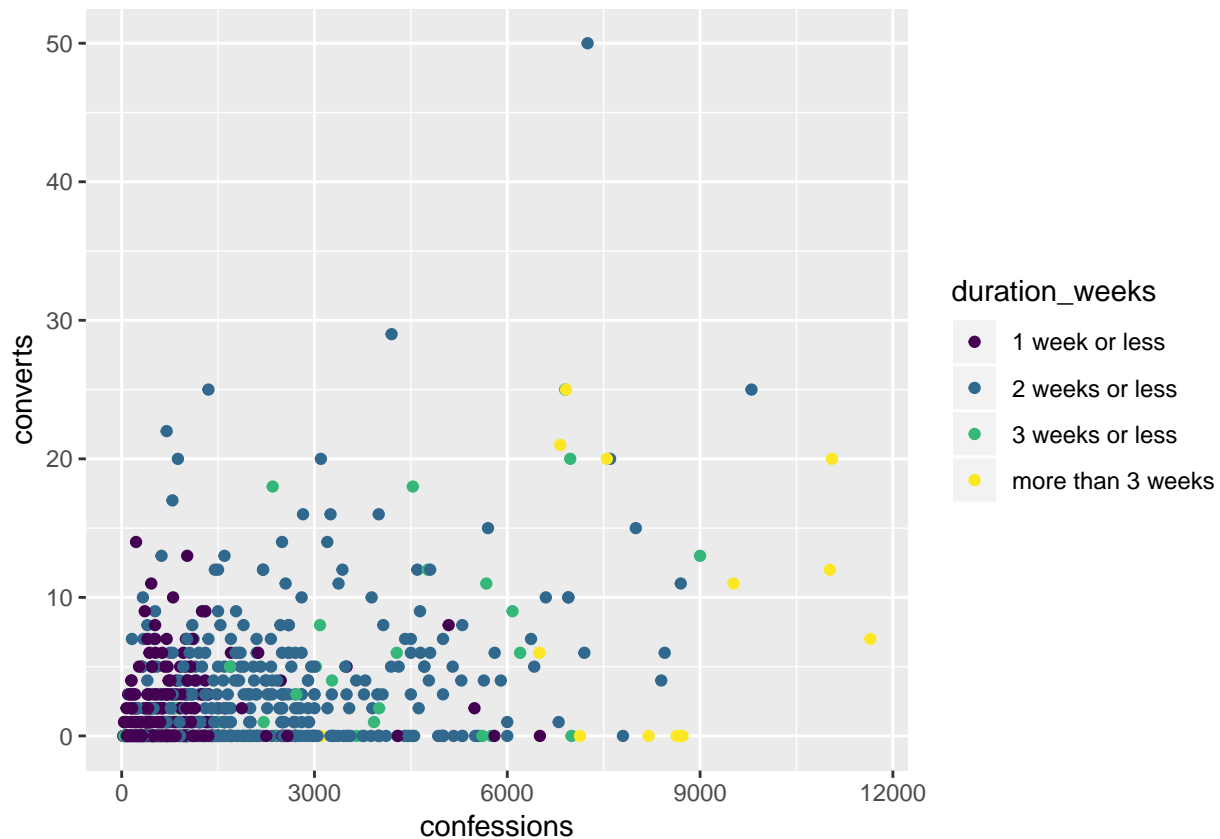
```
## Warning: Removed 6 rows containing missing values (geom_point).
```



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).
```

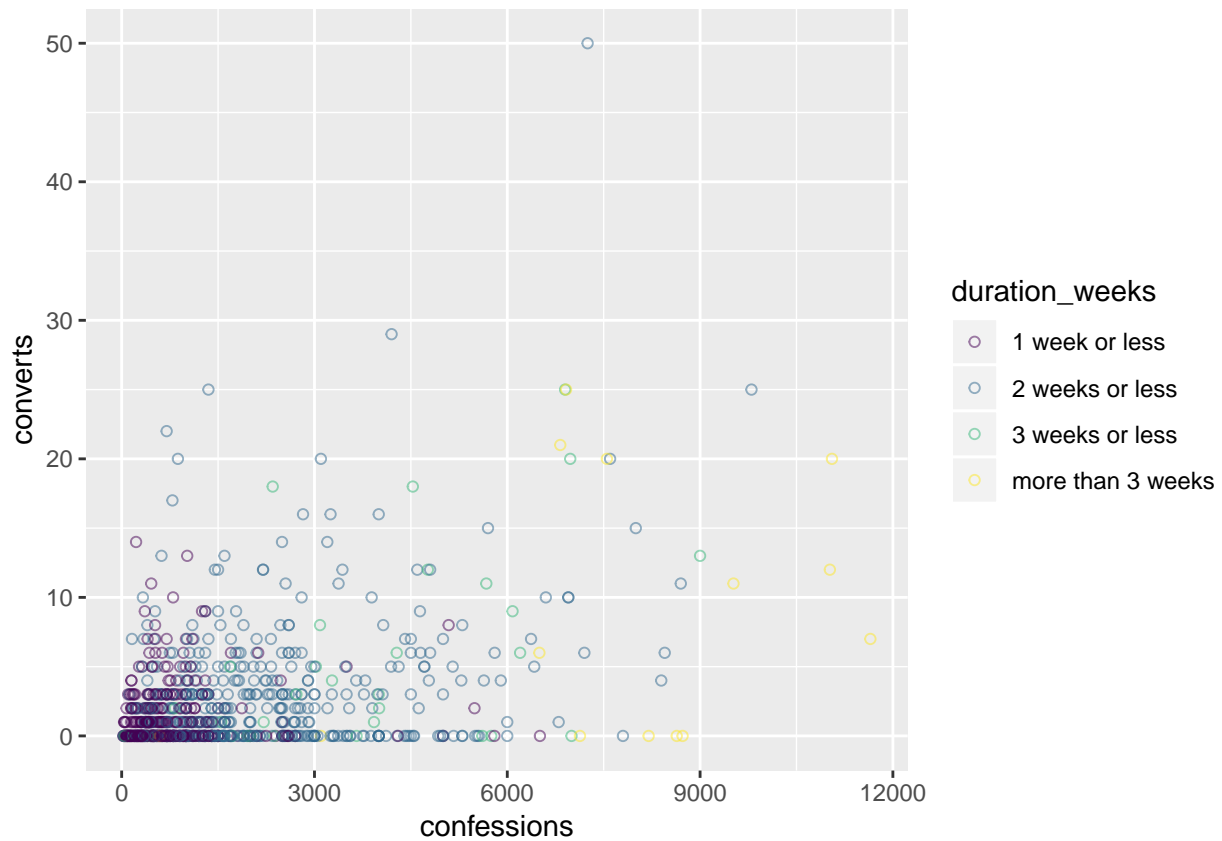


We can also specify static properties, These can go either in the call to `ggplot()` if they affect the entire plot, or in a specific layer (one of the `geom_*()` functions) if they affect just that layer.

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.

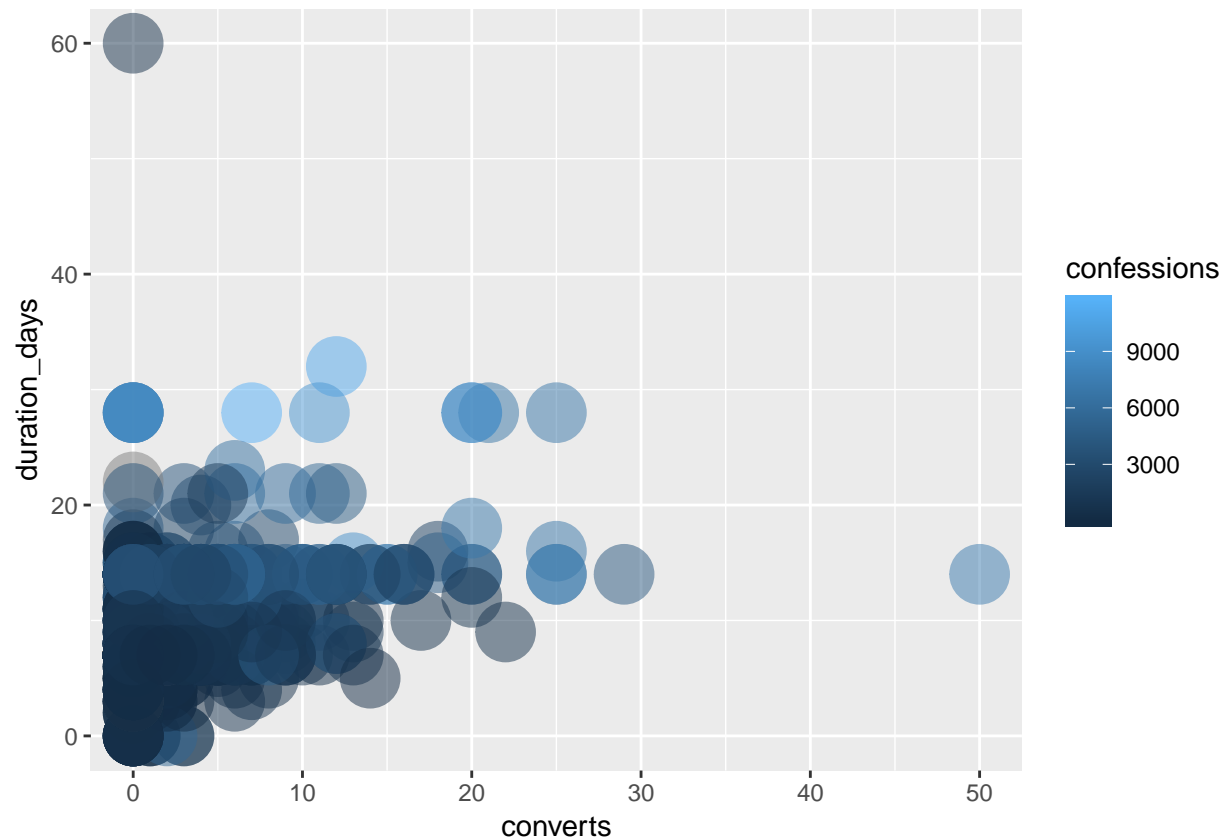
```
ggplot(paulist_missions, aes(x = confessions, y = converts, color = duration_weeks)) +  
  geom_point(alpha = 0.5, shape = 1)
```

```
## 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.

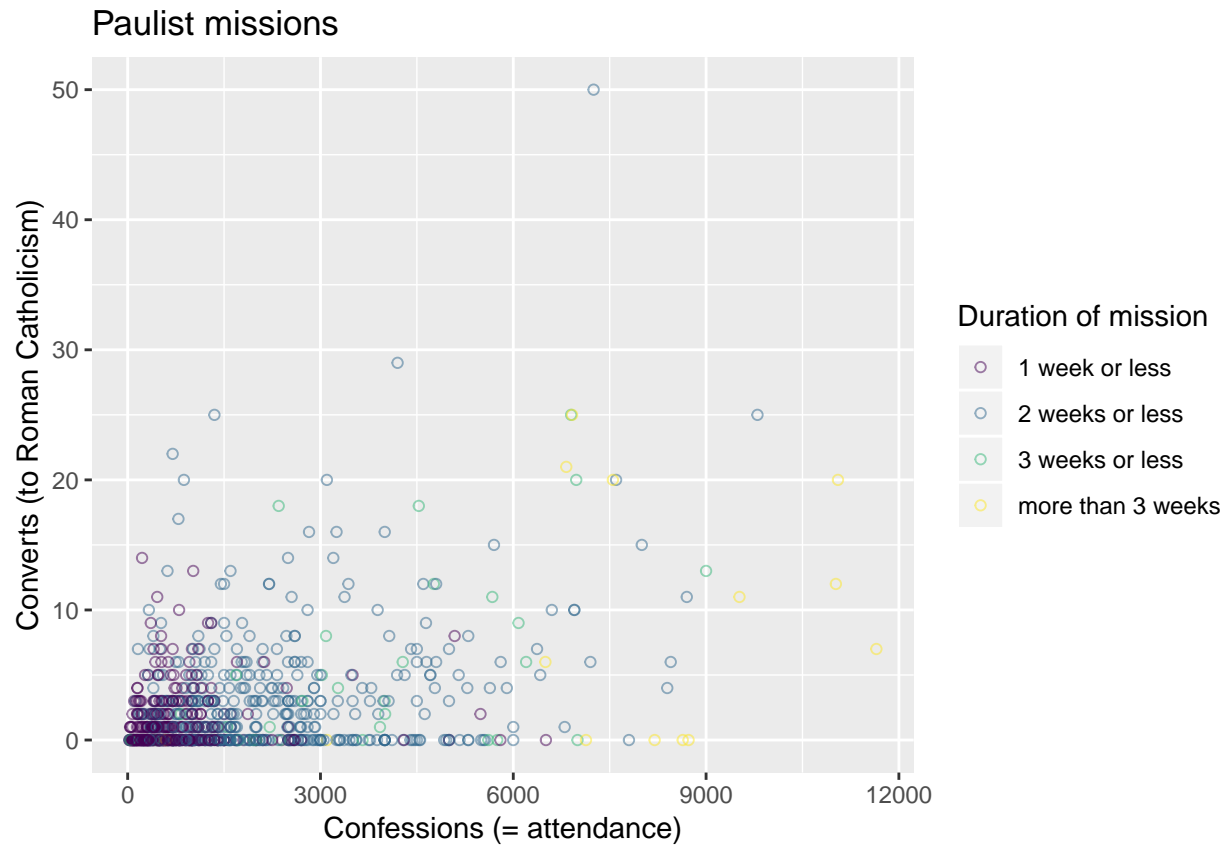
```
ggplot(paulist_missions, aes(x = converts, y = duration_days, color = confessions))+
  geom_point(alpha = 0.5, size = 10)
```



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

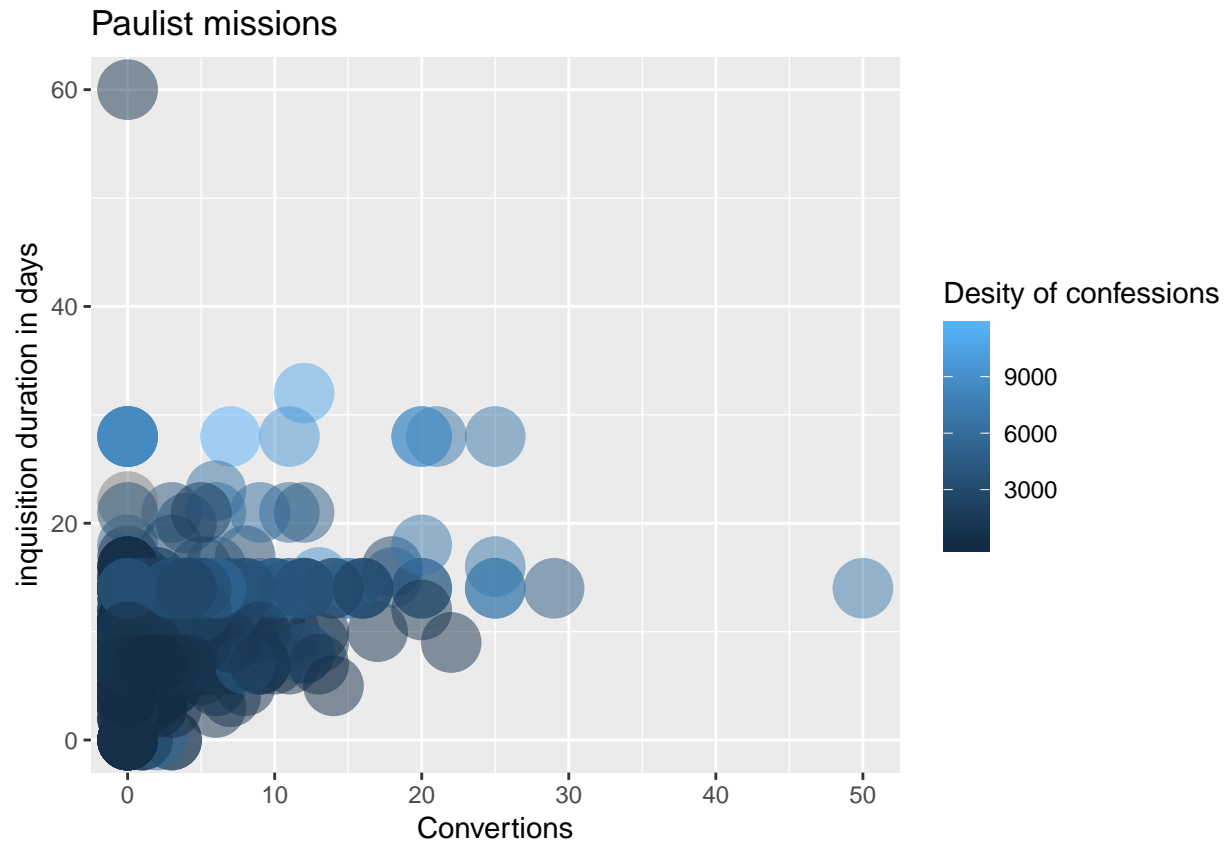
```
ggplot(paulist_missions, aes(x = confessions, y = converts,
                             color = duration_weeks)) +
  geom_point(alpha = 0.5, shape = 1) +
  labs(title = "Paulist missions",
       x = "Confessions (= attendance)",
       y = "Converts (to Roman Catholicism)",
       color = "Duration 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 = converts, y = duration_days, color = confessions))+
  geom_point(alpha = 0.5, size = 10)+
  labs(title = "Paulist missions",
       x = "Conversions",
       y = "inquisition duration in days",
       color = "Desity of confessions")
```

Basic geoms in ggplot2

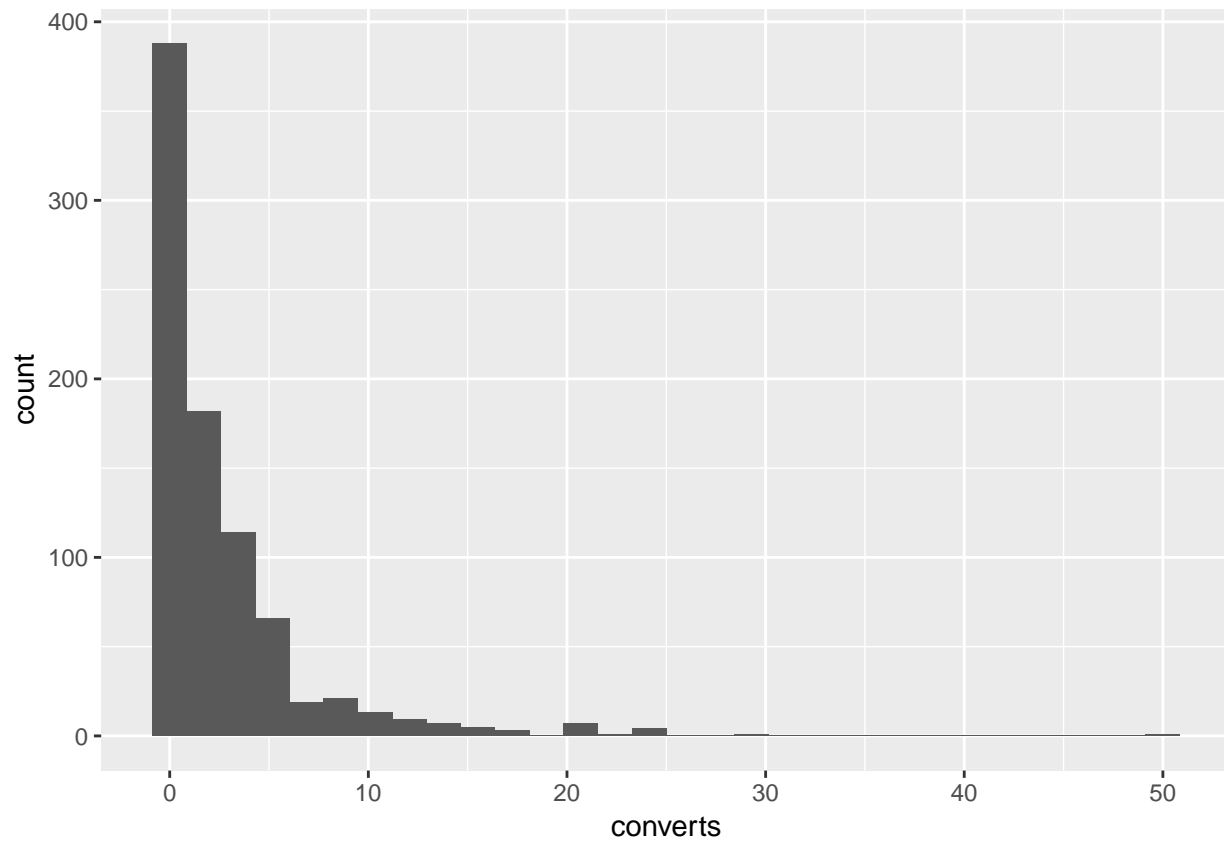
So far we have only used points (with `geom_point()`) as the meaningful glyphs in our plot. Now we will take a tour of different kinds of glyphs that are available to us in ggplot2. Not every variable is suited to every kind of glyph, and sometimes we have to aggregate our data to make certain kinds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

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

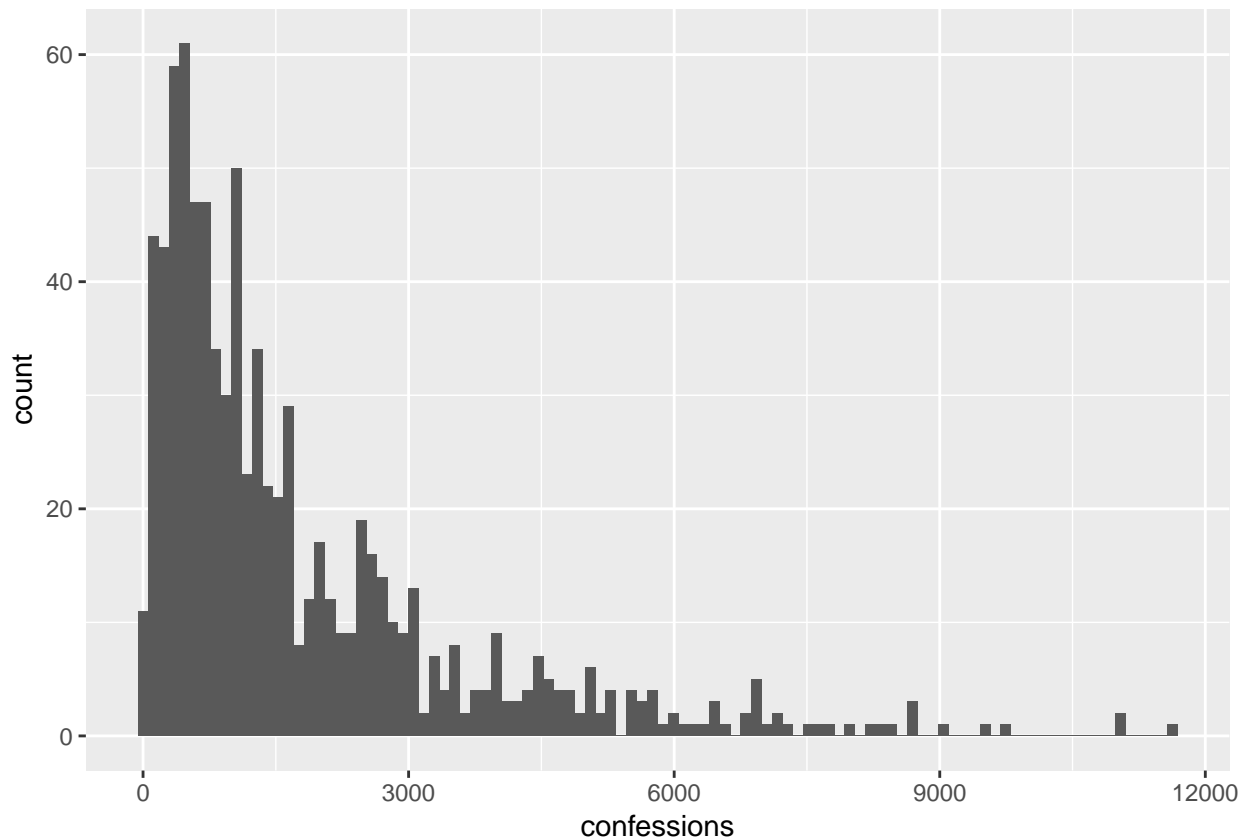
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



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

```
ggplot(paulist_missions, aes(x = confessions))+  
  geom_histogram(bins = 100)
```

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



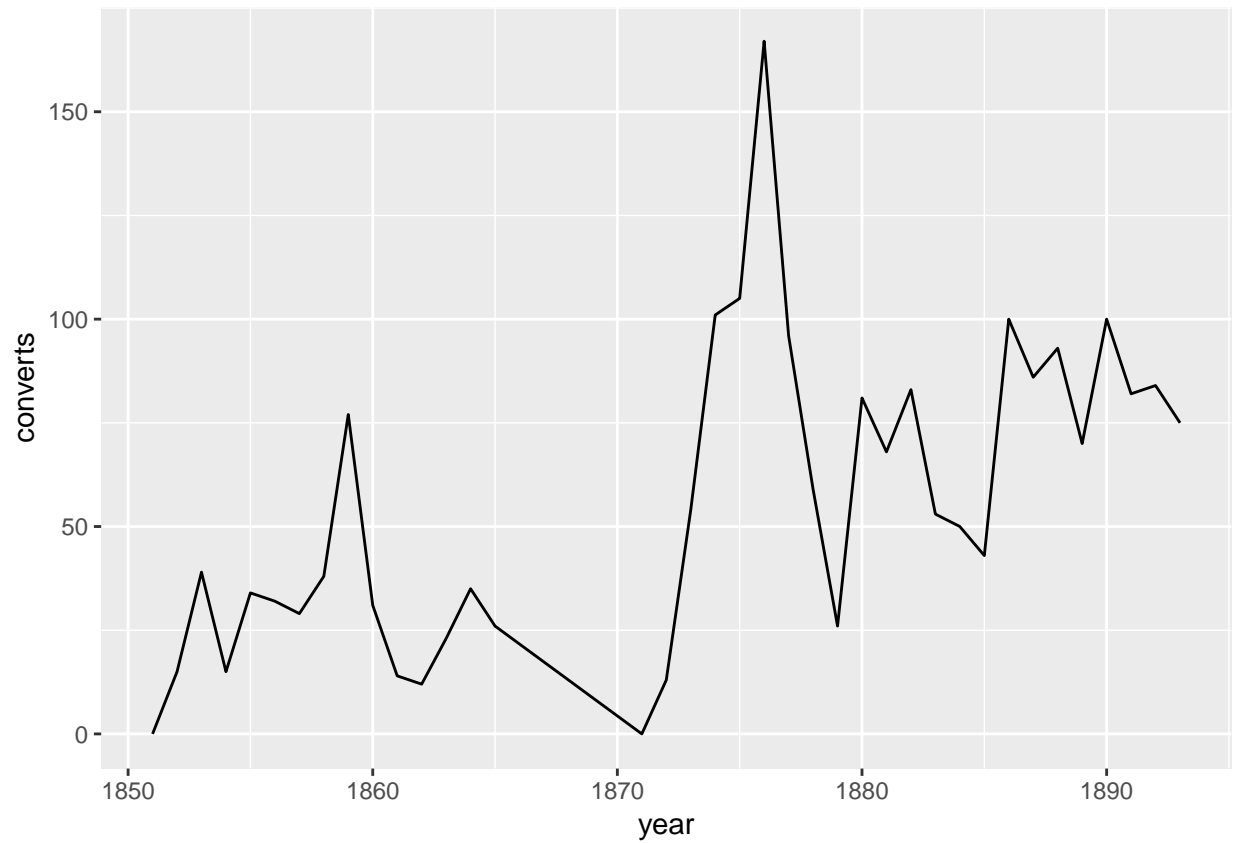
(7) Can you change the number of bins? (Hint: try `bins =` or `binwidth =`. See `?geom_histogram`.)

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

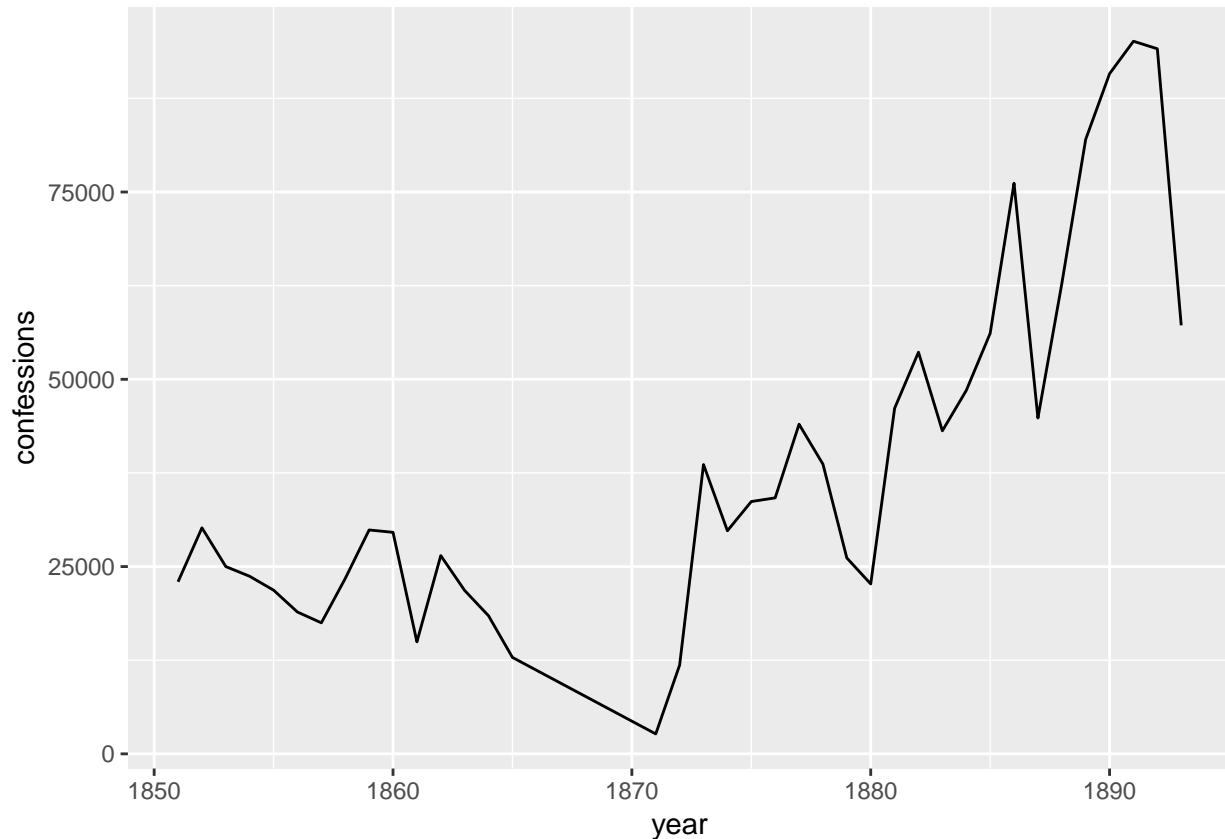
Lines are good for showing trends.

```
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(y = confessions, x = year)) +  
  geom_line()
```



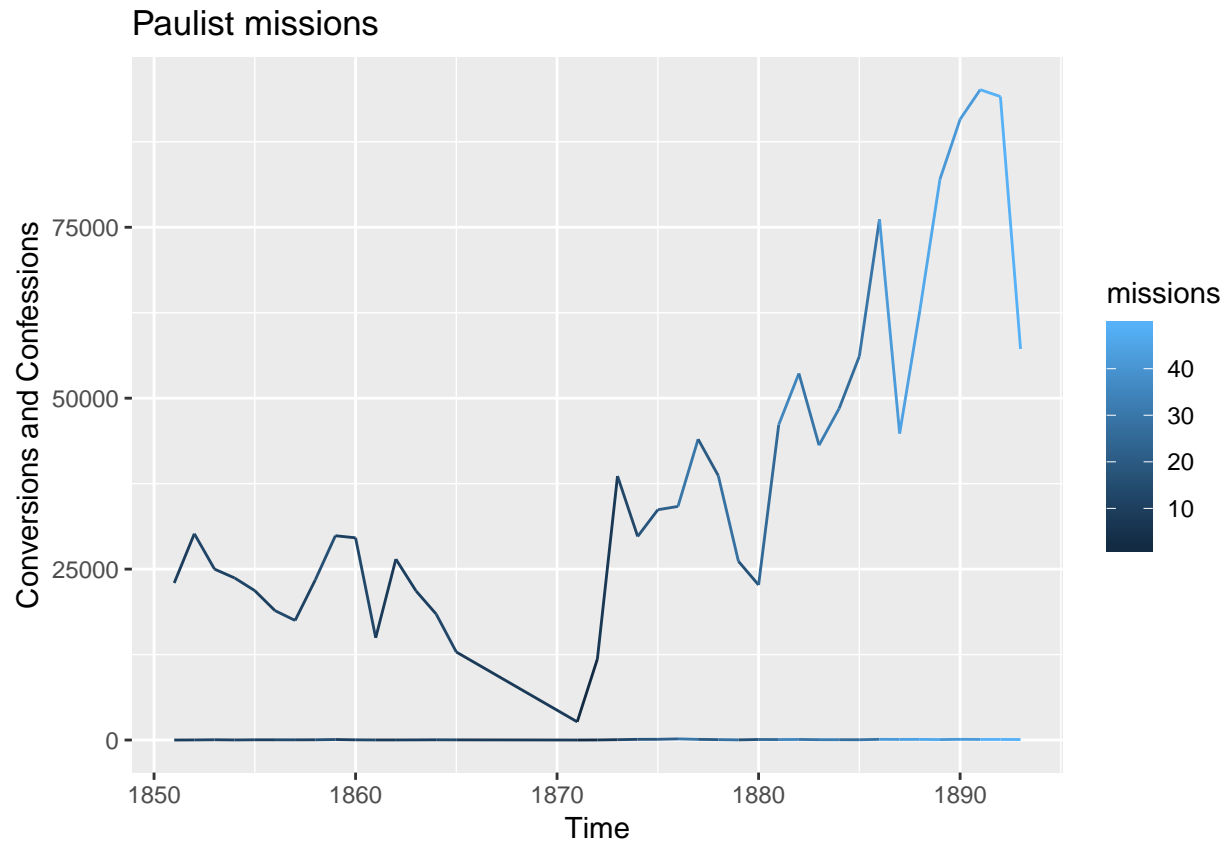
(9) Do you see any issues with the current graph? Please, explain.

The main issue with line graphs is that they do not reflect missing data which could be the case between 1865 and 1871.

(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 functions for each layer. For instance: `geom_line(aes(y = converts))`.)

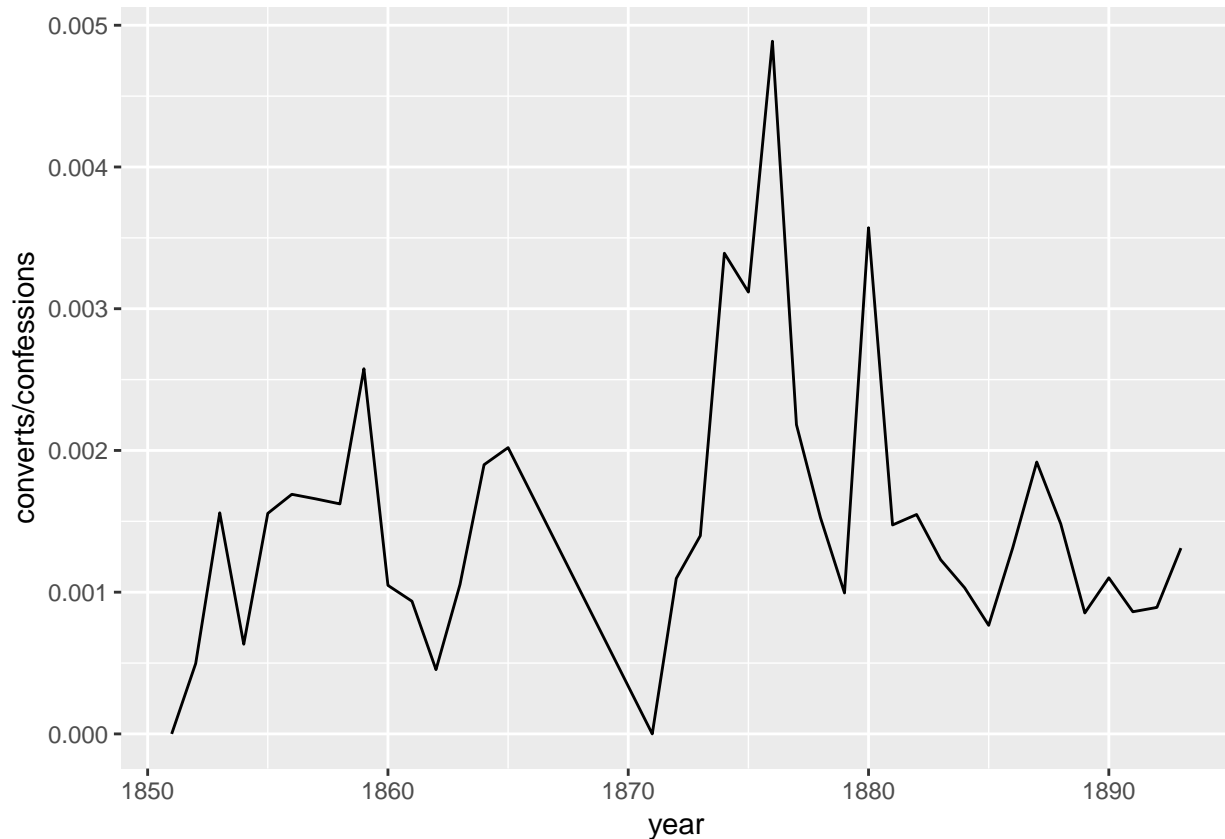
MGR Comment: We can barely see the *converts* line. Can you make these lines visually more comparable? (*Hint:* when we are looking at trends, the first thing we want to see is ups and downs).

```
library(ggplot2)
ggplot(paulists_by_year, aes(x = year, color = missions)) +
  geom_line(aes(y = confessions)) +
  geom_line(aes(y = converts)) +
  labs(title = "Paulist missions",
       x = "Time",
       y = "Conversions and Confessions")
```



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



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

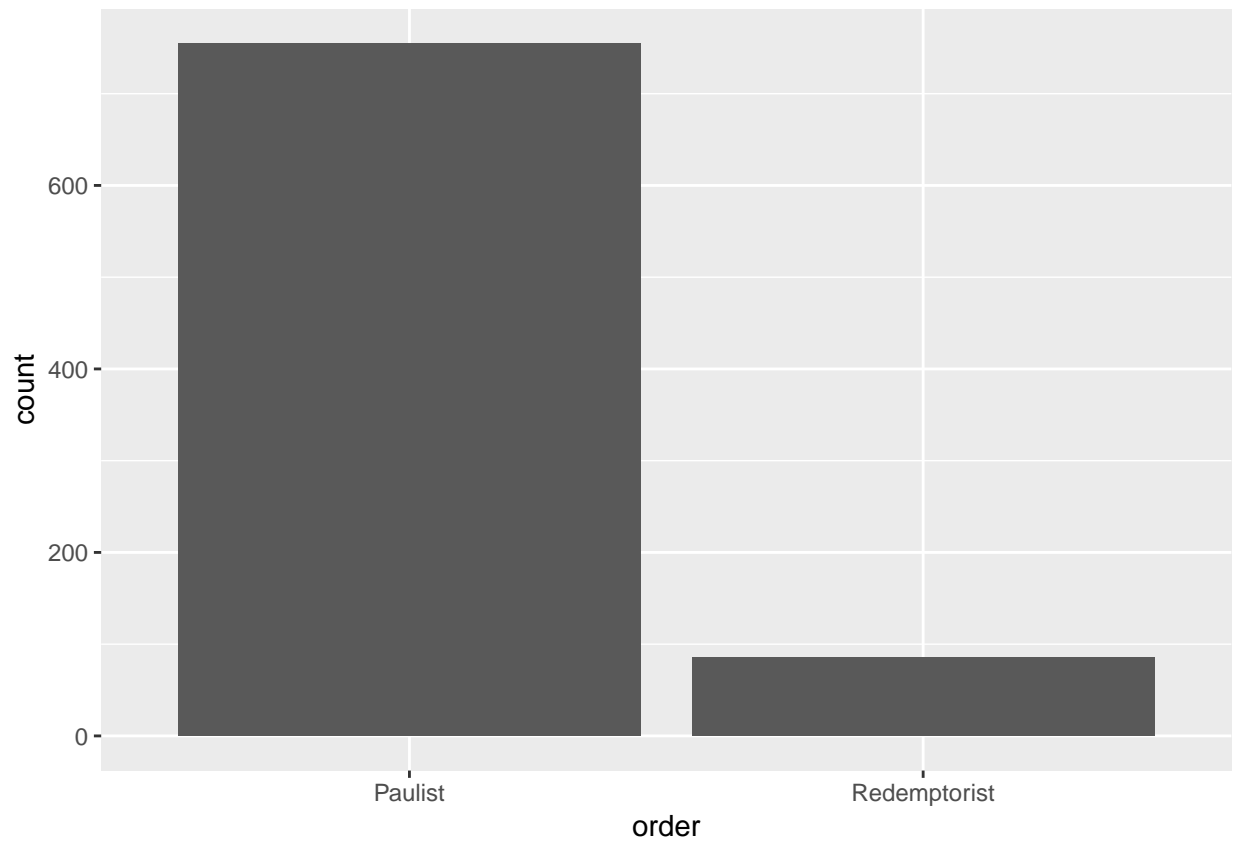
Bar plots

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.

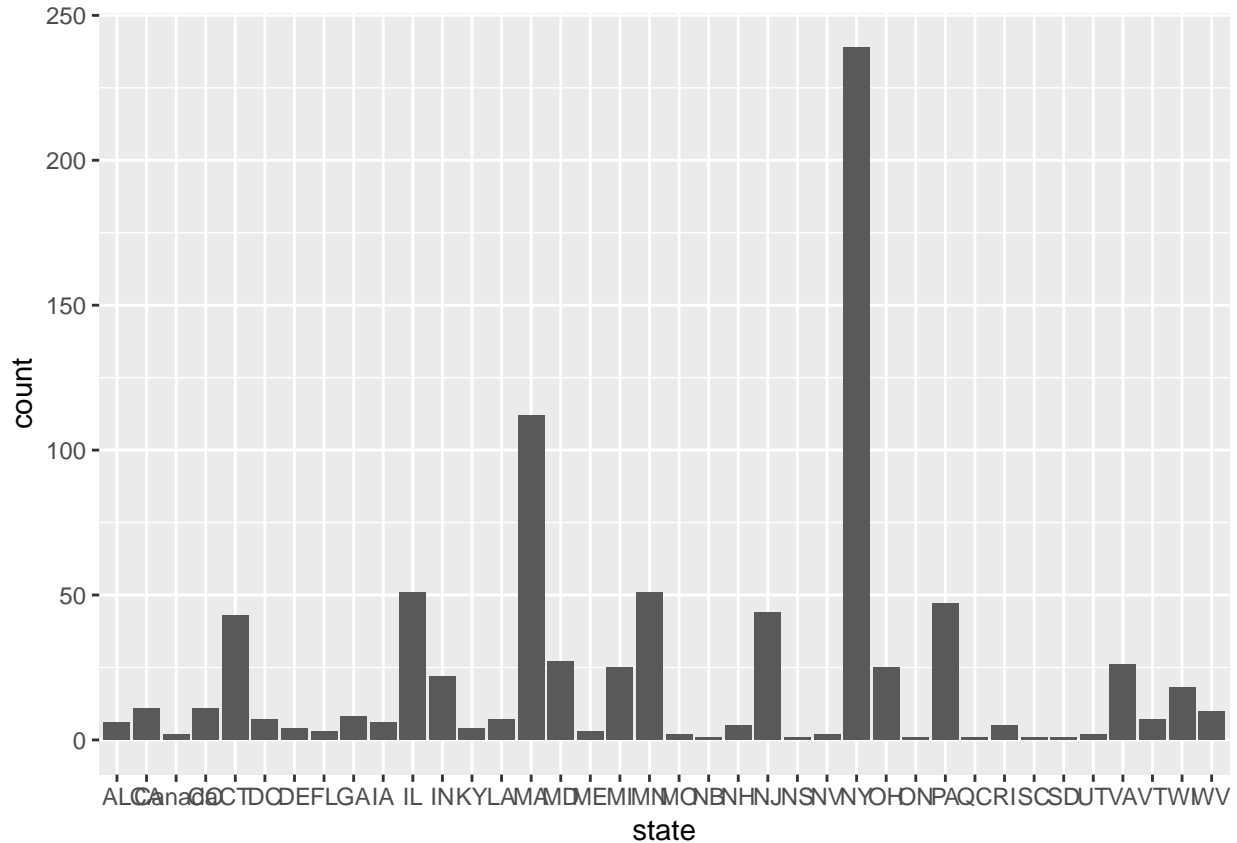
But bar plots are better used for counts of categorical variables. Here we count the number of missions done by the Paulists and the Redemptorists. Notice that is applying a statistical transformation to count the number of observations in each category.

```
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()
```

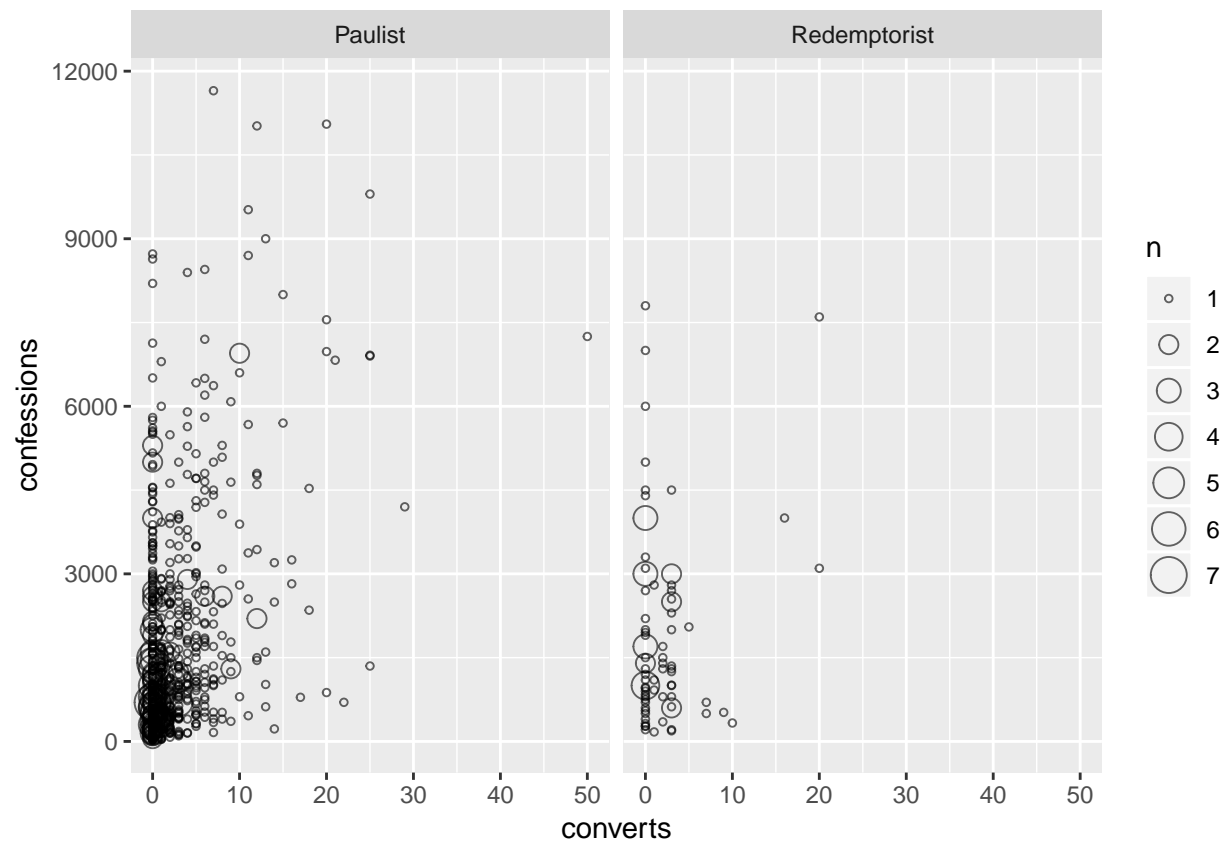
MGR Comment: Such bar graphs are rather useless because of their readability (or, better, the lack of it). Items got organized alphabetically, but some other principle — from east to west, for example — could work better for this kind of data.

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)
```

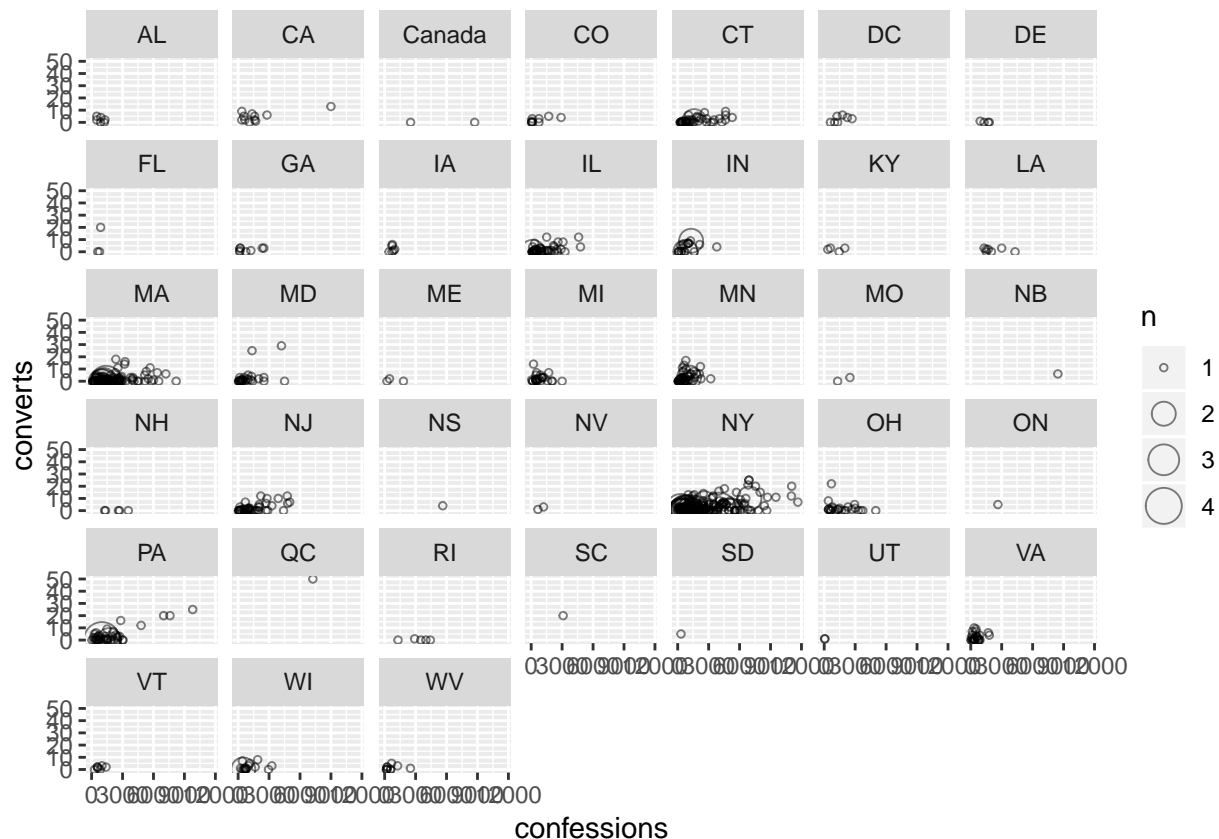
```
## Warning: Removed 6 rows containing non-finite values (stat_sum).
```



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

```
ggplot(paulist_missions, aes(x= confessions, y = converts))+
  geom_count(shape = 1, alpha= 0.5)+
  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.`)

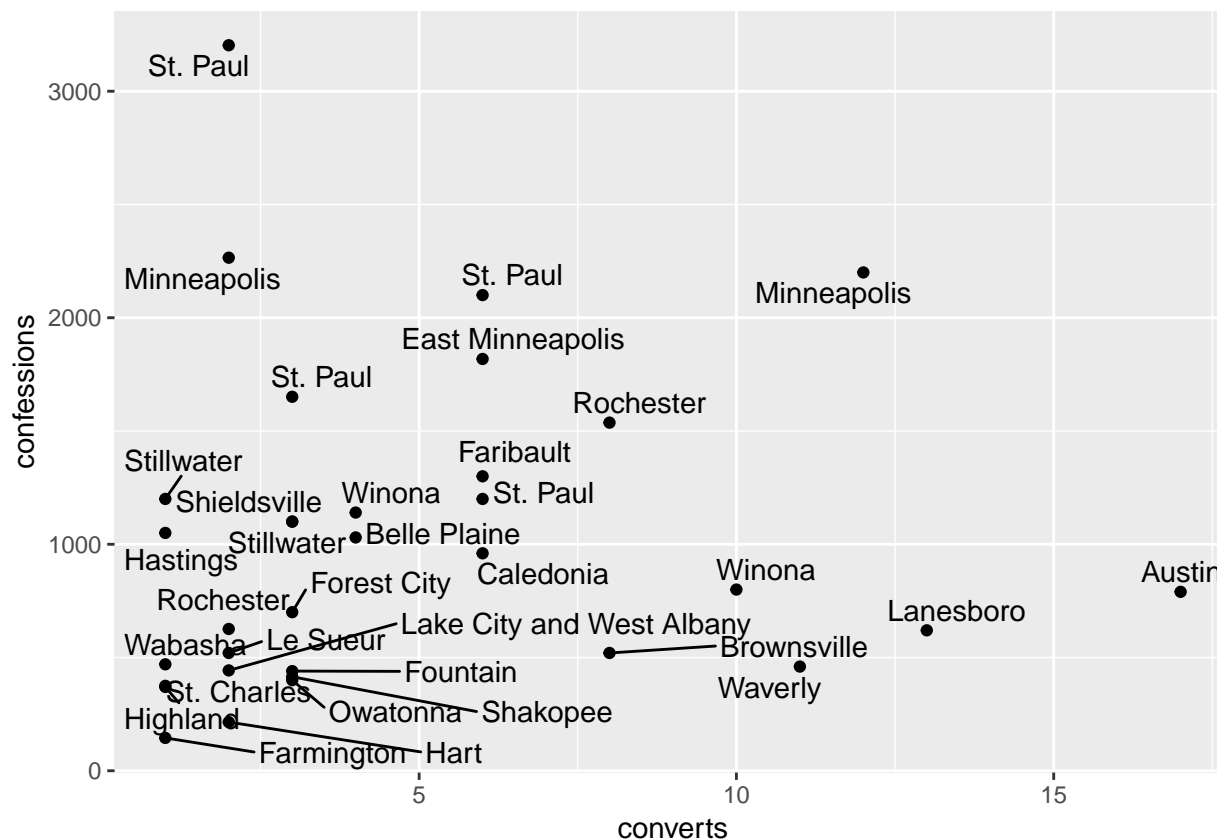
It counts/aggregates the number of observations at each location.

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))
```



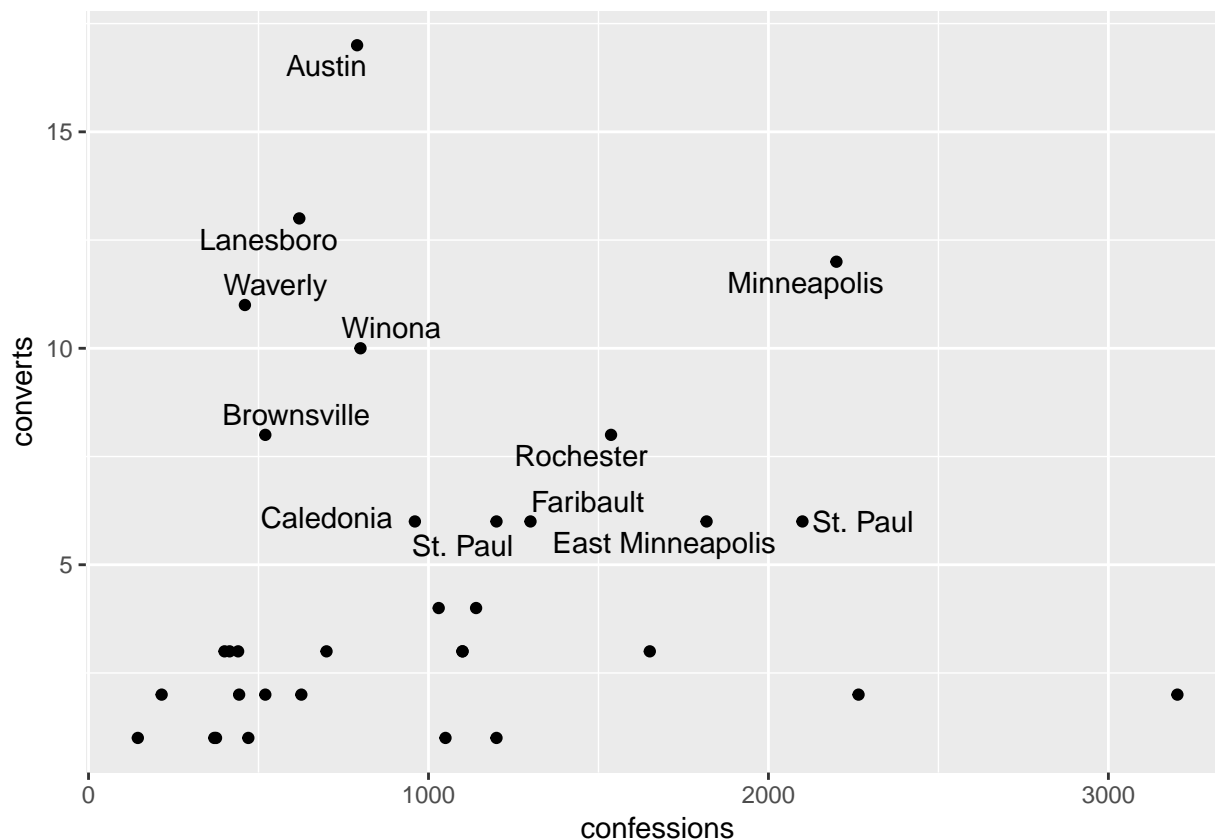
Sometimes it makes sense to add a label only for some points. In this example we create a new column, `label`, and only put the city name in it if there were five or more converts.

```
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.

```
ggplot(mn_missions_with_labels, aes(x = confessions, y = converts)) +
  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.

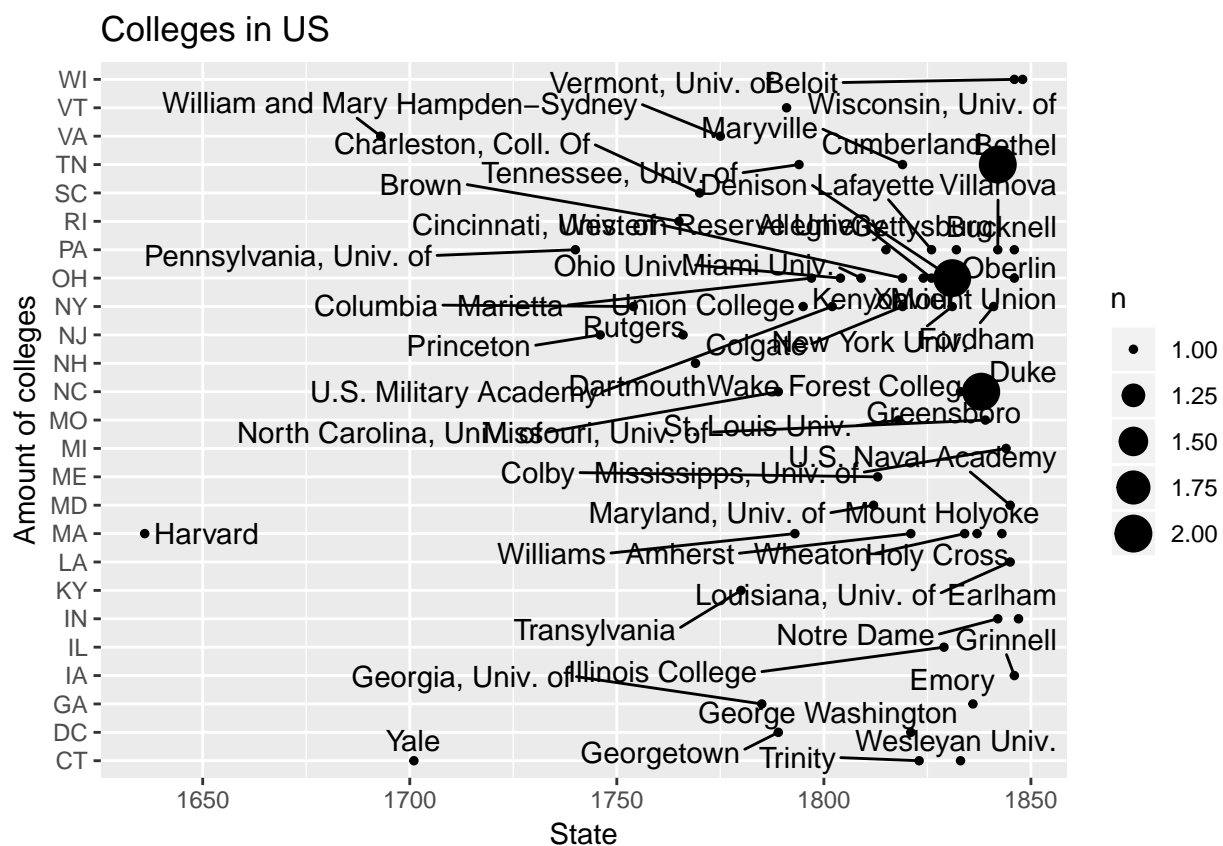
(16) Plot 1

```
library(dplyr)
library(historydata)
library(ggplot2)
head(early_colleges)
```

```
## # A tibble: 6 x 6
##   college      original_name  city    state established sponsorship
```

```
##   <chr>          <chr>          <chr> <chr>          <int> <chr>
## 1 Harvard        <NA>          Cambri~ MA          1636 Congregational; a~
## 2 William and~   <NA>          Willia~ VA          1693 Anglican
## 3 Yale           <NA>          New Ha~ CT          1701 Congregational
## 4 Pennsylvani~   <NA>          Philad~ PA          1740 Nondenominational
## 5 Princeton     College of New~ Prince~ NJ          1746 Presbyterian
## 6 Columbia       King's College New Yo~ NY          1754 Anglican
```

```
ggplot(early_colleges, aes(x = established, y = state))+
  geom_count()+
  geom_text_repel(aes(label = college))+
  labs(title="Colleges in US",
       y="Amount of colleges",
       x="State")
```



This plot has two purposes. The first one is to find out which colleges was founded first in which state and how much time passed till the next one was established. The second purpose is to see when most colleges were established and which states established several colleges in the same period of time. The text labels are not the best option here but allow to find out the name of the college that was established first.

(17) Plot 2

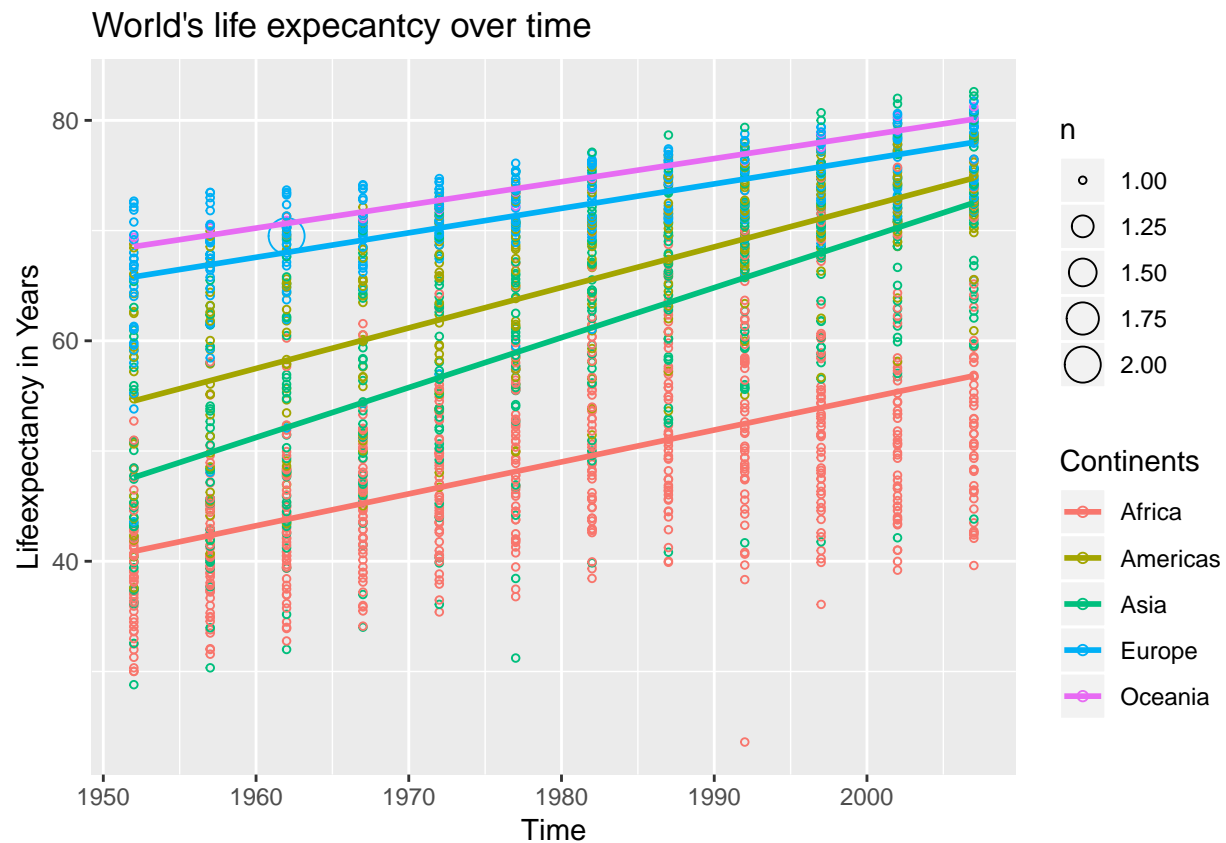
```
head(gapminder)
```

```
## # A tibble: 6 x 6
##   country    continent  year lifeExp      pop gdpPercap
##   <fct>      <fct>    <int>  <dbl>    <int>    <dbl>
## 1 Afghanistan Asia      1952   28.8  8425333    779.
## 2 Afghanistan Asia      1957   30.3  9240934    821.
## 3 Afghanistan Asia      1962   32.0 10267083    853.
## 4 Afghanistan Asia      1967   34.0 11537966    836.
## 5 Afghanistan Asia      1972   36.1 13079460    740.
## 6 Afghanistan Asia      1977   38.4 14880372    786.
```

```
unique(gapminder$continent)
```

```
## [1] Asia    Europe  Africa  Americas Oceania
## Levels: Africa Americas Asia Europe Oceania
```

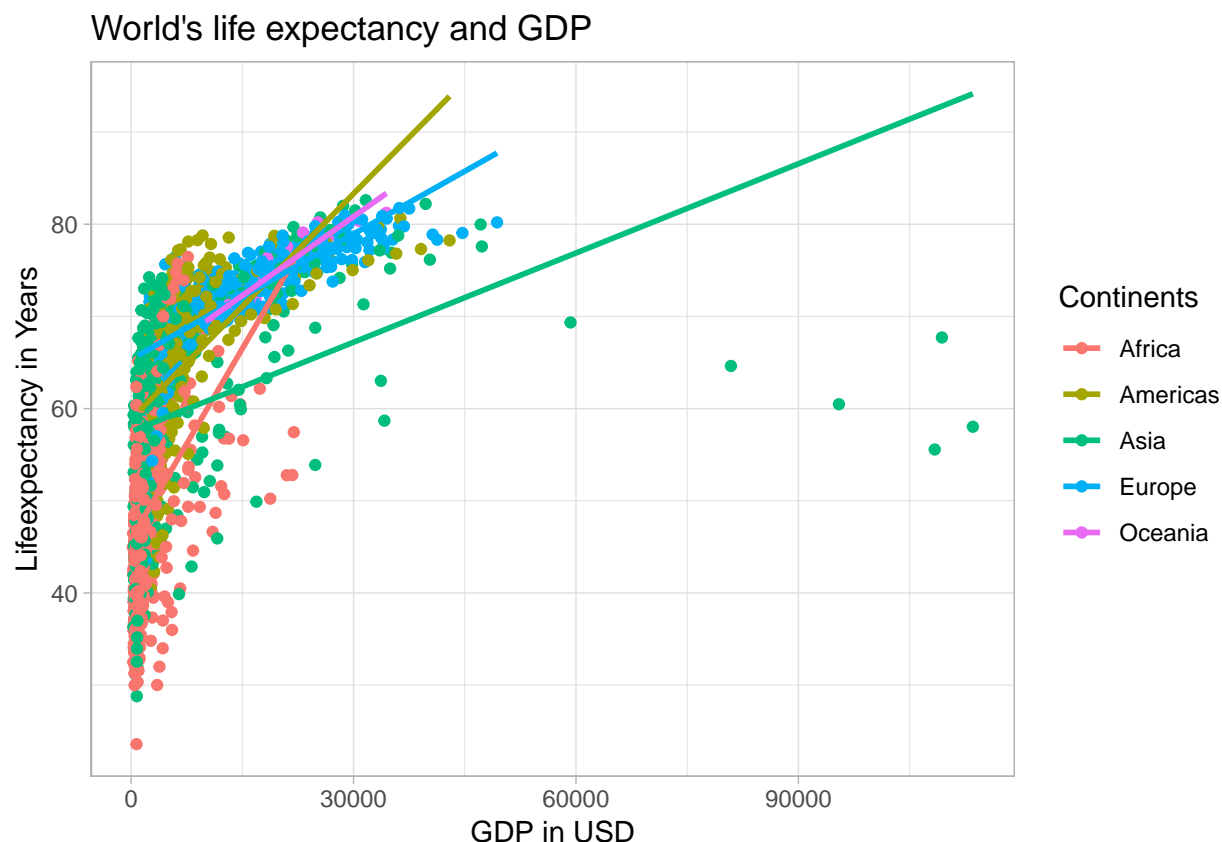
```
ggplot(gapminder, aes(x = year, y = lifeExp, color = continent))+
  geom_count(shape = 1)+
  geom_smooth(method="lm", se=F)+
  labs(title = "World's life expectancy over time",
       x = "Time",
       y = "Lifeexpectancy in Years",
       color = "Continents")
```



This plot provides information about how the life expectancy changed over time on different continents. While the plot has problems showing the dots for Oceania the average line provides better information about which continent has the highest life expectancy. I choose `geom_count` to visualize the amount of countries being part of a continent if there is a spontaneous increase in life expectancy at some time. One circle evolved around 1962 were life expectancy had been rising in several countries. The `geom_count` also visualizes that especially Africa has a great range that counts from about 30 to around 70 depending on the country. Other continents like Oceania seem to be quite homogeneous. The `geom_smooth` also indicates how Asia could catch up to other from the 1990s onwards.

(18) Plot 3

```
theme_set(theme_light())
ggplot(gapminder, aes(x = gdpPercap, y = lifeExp, color = continent)) +
  geom_point() +
  geom_smooth(method="lm", se=F) +
  labs(title = "World's life expectancy and GDP",
       x = "GDP in USD",
       y = "Lifeexpectancy in Years",
       color = "Continents")
```



A comparison of GDP per capita and country population seems to be somehow interesting. High GDP does not necessarily mean that the country has a high life expectancy as in Asian countries over 60,000 and 90,000. The trend, however, tends to a growing life expectancy with higher GDP. I used `geom_point` to receive the density of countries of a specific ratio. `geom_smooth` provides

good information with countries are above or below the average of their continent. Labeling the dots with text did not provide any usable results.