SML201 Chapter 2.4

Graphing with the `ggplot2` package

Daisy Huang

Fall 2019

Contents

R functions covered

- Functions in the `ggplot2` package
- `ggpairs()` in the `GGally` package

Introduction

In the last chapter we saw that we can answer very interesting questions about the passengers with plots that compare different groups of the passengers. Unfortunately, for creating more complex graphs R's basic graphic functions become more verbose; also, personally I think graphs produced by the basic `graphics` functions are not very appealing aesthetically unless you are willing to spending a lot of time on making manual adjustments to the graphs by changing the values of the graphic variables (e.g., we showed you how to make some of these adjustments with `cex`, `cex.main` and `cex.lab` etc. in this week's precept). When it comes to making complex graphs the packages `ggplot2` and `GGally` (an extension of `ggplot2`) have a lot more advantages over R's basic `graphics` package.

About the `ggplot2` package

The package `ggplot2` was built based on the concepts in the *The Grammar of Graphics* by Leland Wilkinson (1999/2005).

The package has a lot of built-in features that make nice graphs and are more suitable for complex large datasets.

Read in the dataset

```
t.ship =
  read.csv(file =
  '/Users/billhaarlow/Desktop/SML201/titanic.csv')
```

We should refresh our memory on the properties of the dataset.

```
class(t.ship)  # this tells me what kind of R object this dataset is
[1] "data.frame"
dim(t.ship)  # 891 rows by 12 columns
[1] 891  12
```

```
head(t.ship)  # look at the first 6 (by default) rows of an object
  PassengerId Survived Pclass
1           1        0      3
2           2        1      1
3           3        1      3
4           4        1      1
5           5        0      3
6           6        0      3
                                                 Name    Sex
```

```
1                            Braund, Mr. Owen Harris   male
2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female
3                             Heikkinen, Miss. Laina female
4       Futrelle, Mrs. Jacques Heath (Lily May Peel) female
5                           Allen, Mr. William Henry   male
6                                   Moran, Mr. James   male
  Age SibSp Parch           Ticket    Fare Cabin Embarked
1  22     1     0        A/5 21171  7.2500              S
2  38     1     0         PC 17599 71.2833   C85        C
3  26     0     0 STON/O2. 3101282  7.9250              S
4  35     1     0           113803 53.1000  C123        S
5  35     0     0           373450  8.0500              S
6  NA     0     0           330877  8.4583              Q
```

```
str(t.ship, strict.width = "cut")
'data.frame':   891 obs. of  12 variables:
 $ PassengerId: int  1 2 3 4 5 6 7 8 9 10 ...
 $ Survived   : int  0 1 1 1 0 0 0 0 1 1 ...
 $ Pclass     : int  3 1 3 1 3 3 1 3 3 2 ...
 $ Name       : Factor w/ 891 levels "Abbing, Mr. Anthony",....
 $ Sex        : Factor w/ 2 levels "female","male": 2 1 1 1 2..
 $ Age        : num  22 38 26 35 35 NA 54 2 27 14 ...
 $ SibSp      : int  1 1 0 1 0 0 0 3 0 1 ...
 $ Parch      : int  0 0 0 0 0 0 0 1 2 0 ...
 $ Ticket     : Factor w/ 681 levels "110152","110413",..: 52..
 $ Fare       : num  7.25 71.28 7.92 53.1 8.05 ...
 $ Cabin      : Factor w/ 148 levels "","A10","A14",..: 1 83 ..
 $ Embarked   : Factor w/ 4 levels "","C","Q","S": 4 2 4 4 4 ..
```

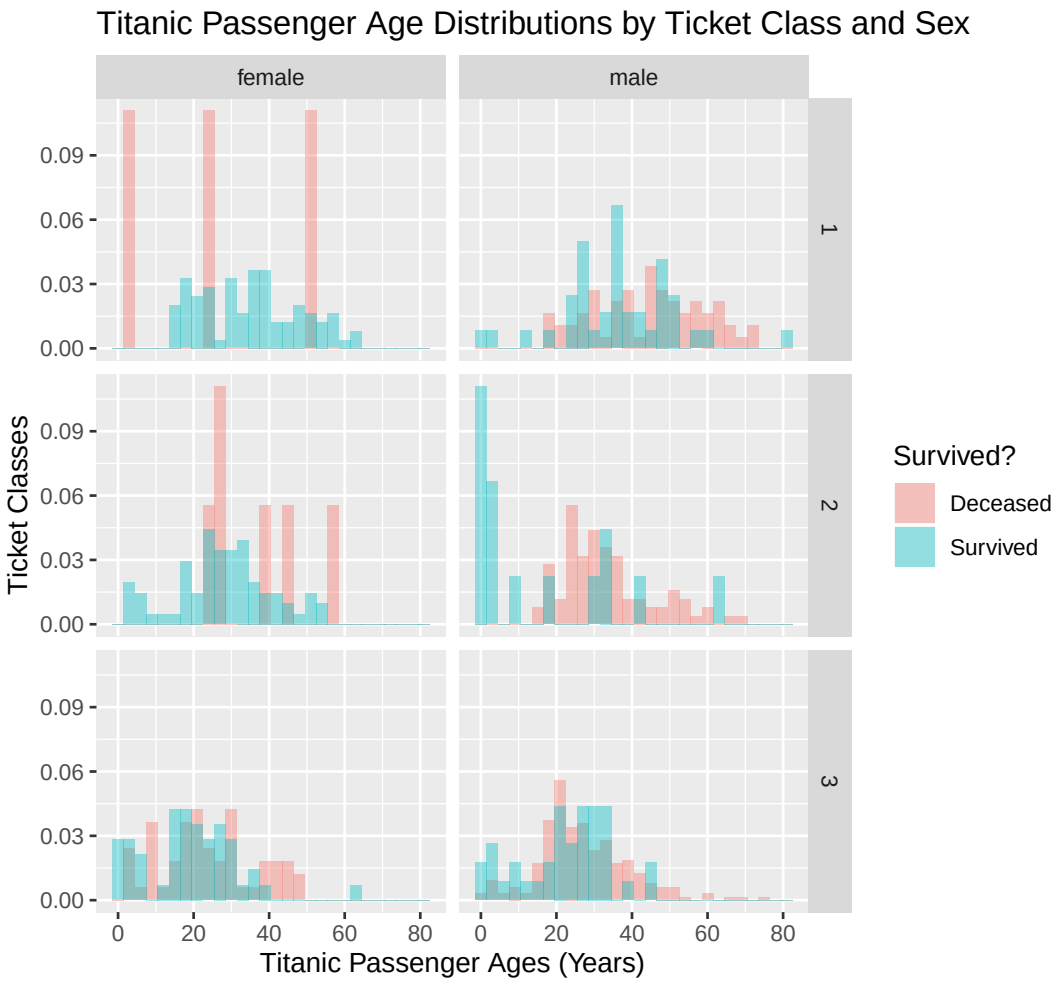
An example of a graph produced by the ggplot2 package

```
library(ggplot2)
```

Here is a set of histograms that produced by the `geom_histogram()` function in the `ggplot2` package. Note that if we had to used the basic graphic functions in R to produce this set of plots, we would have to write the code separately for 6 histograms, not to mention that we would also need to write extra lines of code to give different colors to the data points for the people in the survived and deceased groups.

```
ggplot(t.ship) +
   geom_histogram(mapping = aes(x = Age, y = ..density..,
                                fill = as.factor(Survived)),
                  position="identity", binwidth = 3,
                  alpha = .4) +
```

```
  facet_grid(as.factor(Pclass)~Sex) +
  labs(x='Titanic Passenger Ages (Years)',
       y='Ticket Classes',
       title = 'Titanic Passenger Age Distributions by Ticket Class and Sex', fill = "Survived?"
scale_fill_discrete(labels = c("Deceased", "Survived"))
```



How does the syntax work for functions in the `ggplot2` package?

The basic idea

You can combine/layer several graphical components in `ggplot2` to build and customize your graph. Some of the graphical components include:

- data in data.frame format

- aesthetic mapping: e.g., x- and y- variables, color, size and shape related to the variables
- geometric object: what kind of plot you would like to make?
- statistical transformations
- position adjustments
- faceting: for conditional plots

Main structure of the grammar

The main structure of the grammar has the form of

ggplot(data.frame, ...) + geom_FunctionType()

Some commonly used geom_ functions are

- geom_boxplot()
- geom_histogram()
- geom_line()
- geom_point()
- geom_smooth()
- geom_hex()

Steps

Creating a plot using the `ggplot2` package follows the following steps:

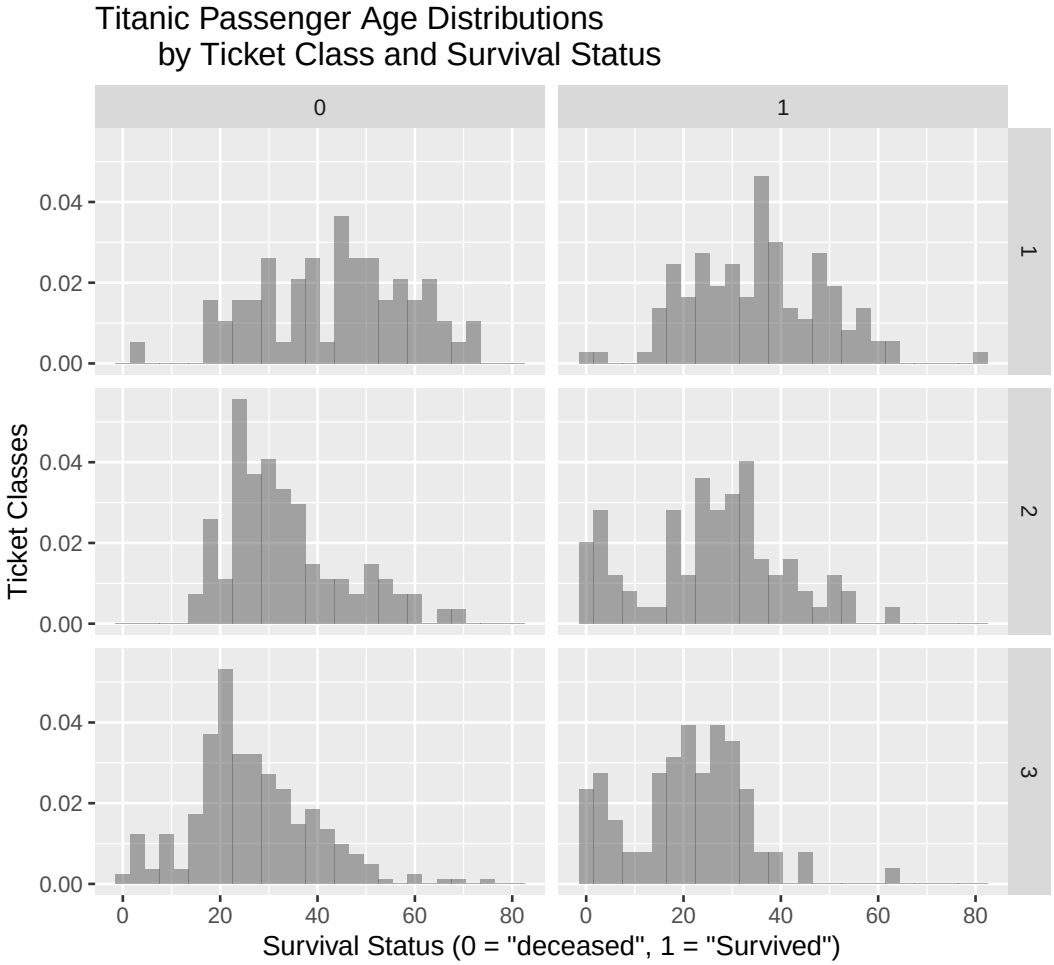
1. Dataset - use the function `ggplot()` to specify what dataset we will work with to create the plot; the dataset needs to be a data frame.

2. Type of graph to make - use `geom_FunctionTpye()` to specify what kinds of graphs (such as scatterplots, histograms, or boxplots) we want to make.

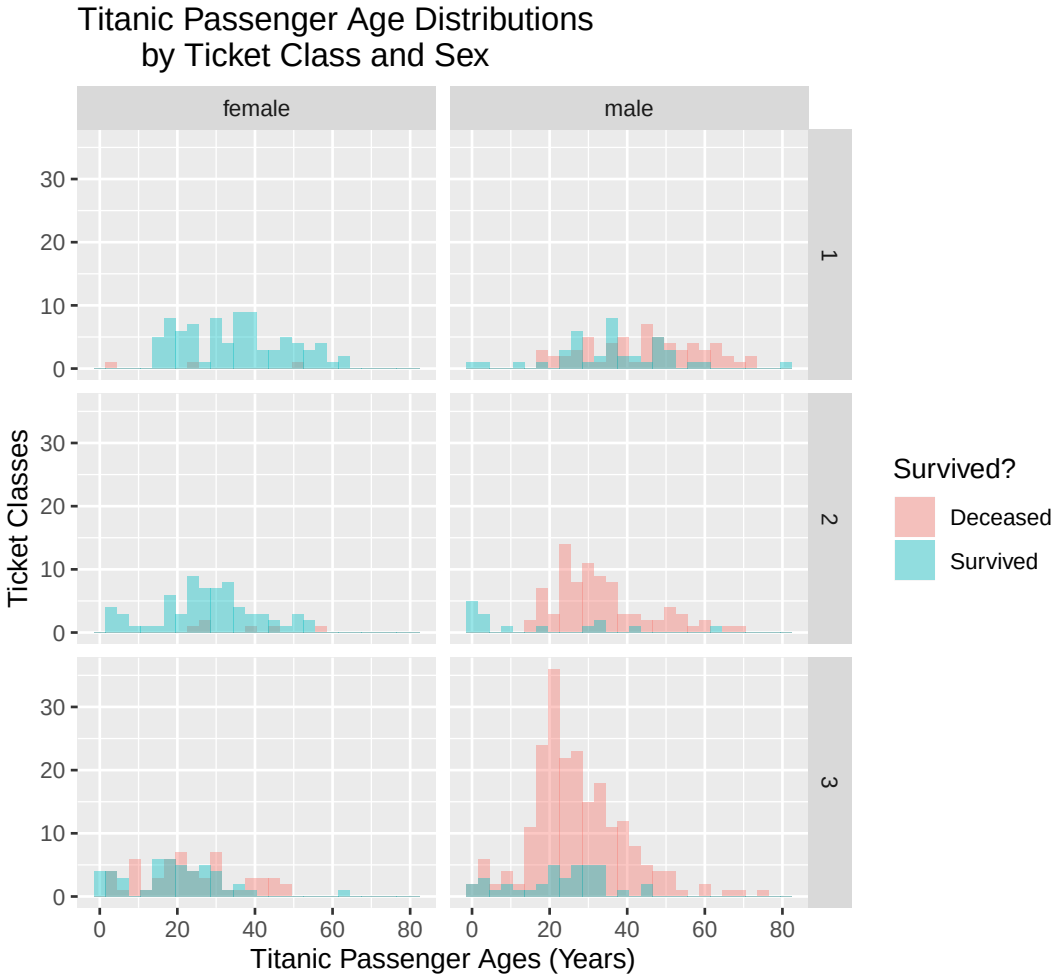
3. Aesthetic mappings - specify the variables to use for the plot(s) and how they appear on the plot.

4. Appearance - using the various parameters to change the display of the plot, such as specifying the labels, colors or showing the data by groups.

```
ggplot(t.ship) + geom_histogram(mapping = aes(x = Age, y = ..density..),
    binwidth = 3, alpha = 0.5) + facet_grid(as.factor(Pclass) ~
    as.factor(Survived)) + labs(x = "Survival Status (0 = \"deceased\", 1 = \"Survived\")",
    y = "Ticket Classes", title = "Titanic Passenger Age Distributions
       by Ticket Class and Survival Status")
```




```
ggplot(t.ship) + geom_histogram(mapping = aes(x = Age, fill = as.factor(Survived)),
    position = "identity", binwidth = 3, alpha = 0.4) + facet_grid(as.factor(Pclass) ~
    Sex) + labs(x = "Titanic Passenger Ages (Years)", y = "Ticket Classes",
    title = "Titanic Passenger Age Distributions
       by Ticket Class and Sex",
    fill = "Survived?") + scale_fill_discrete(labels = c("Deceased",
    "Survived"))
```




```
iceberg = data.frame(Exists = c(F, T), Target = c("Titanic",
    "Titanic 2"), Destroy = c("100%", "0%"), Mission = c("Passed",
    "Failed"), Respect = c("+", "N/A"), Reason = c("N/A", "Climate"))
iceberg
  Exists    Target Destroy Mission Respect  Reason
1  FALSE   Titanic    100%  Passed       +     N/A
2   TRUE Titanic 2      0%  Failed     N/A Climate
```

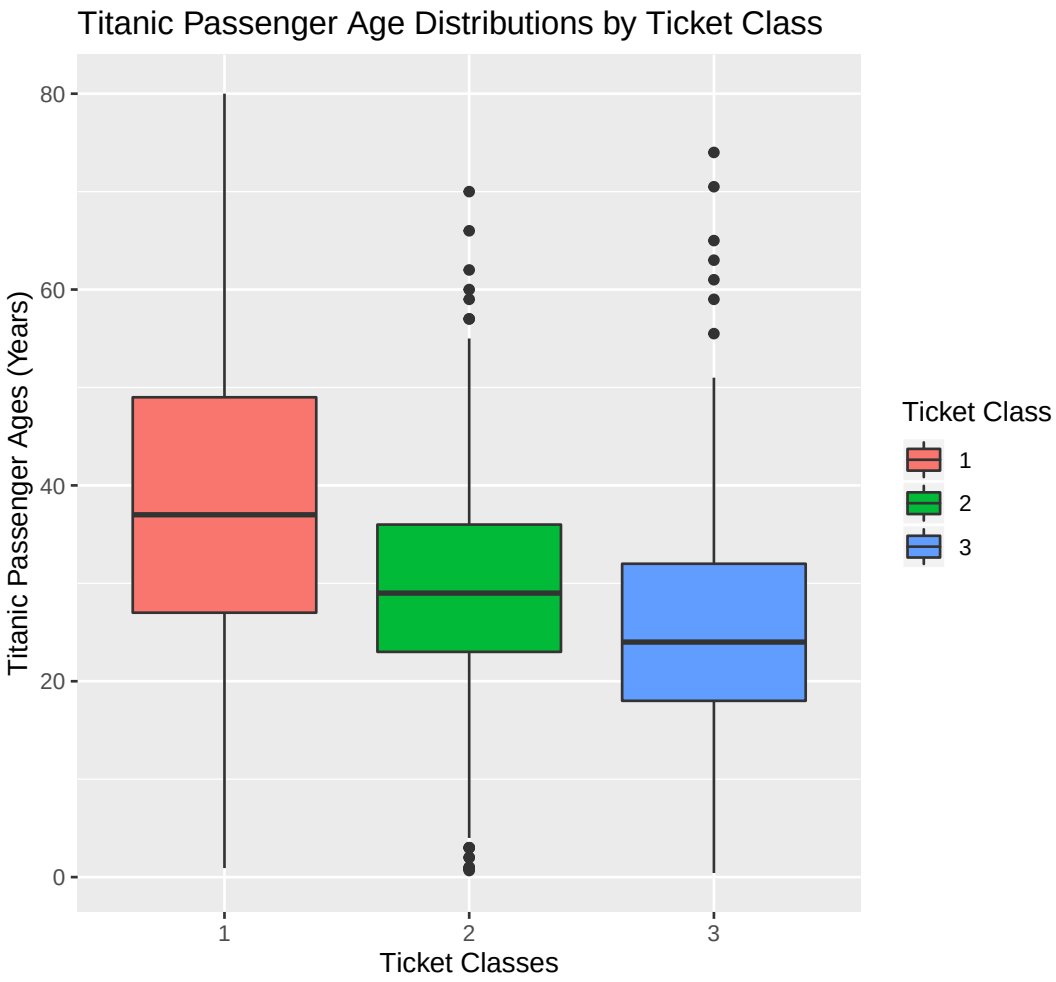
Examples

A simple example

Suppose that we would like to use a side-by-side boxplot to compare the age distributions of the passengers by ticket class.

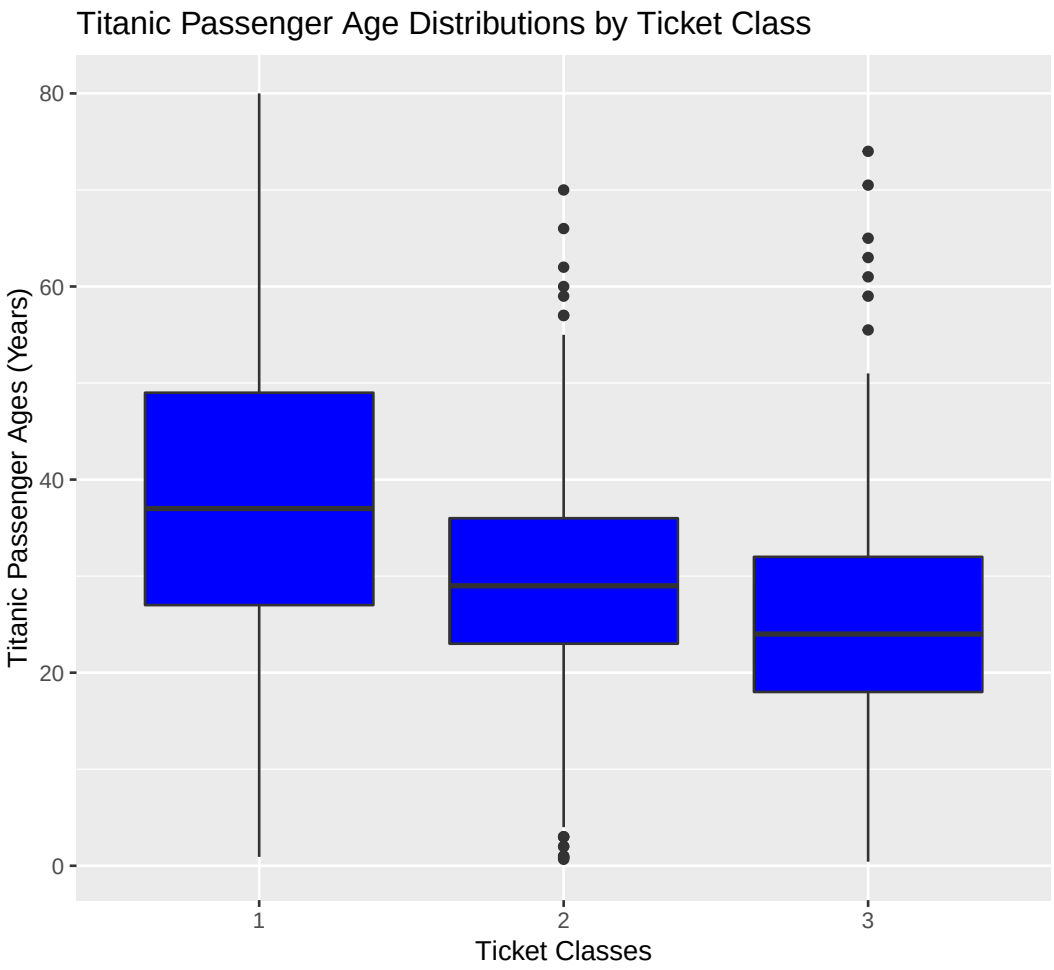
```
class(t.ship)
[1] "data.frame"
```

```
# the dataset `t.ship` is already in data.frame format

ggplot(t.ship) + geom_boxplot(mapping = aes(x = as.factor(Pclass),
    y = Age, fill = as.factor(Pclass))) + labs(y = "Titanic Passenger Ages (Years)",
    x = "Ticket Classes", title = "Titanic Passenger Age Distributions by Ticket Class",
    fill = "Ticket Class")
```



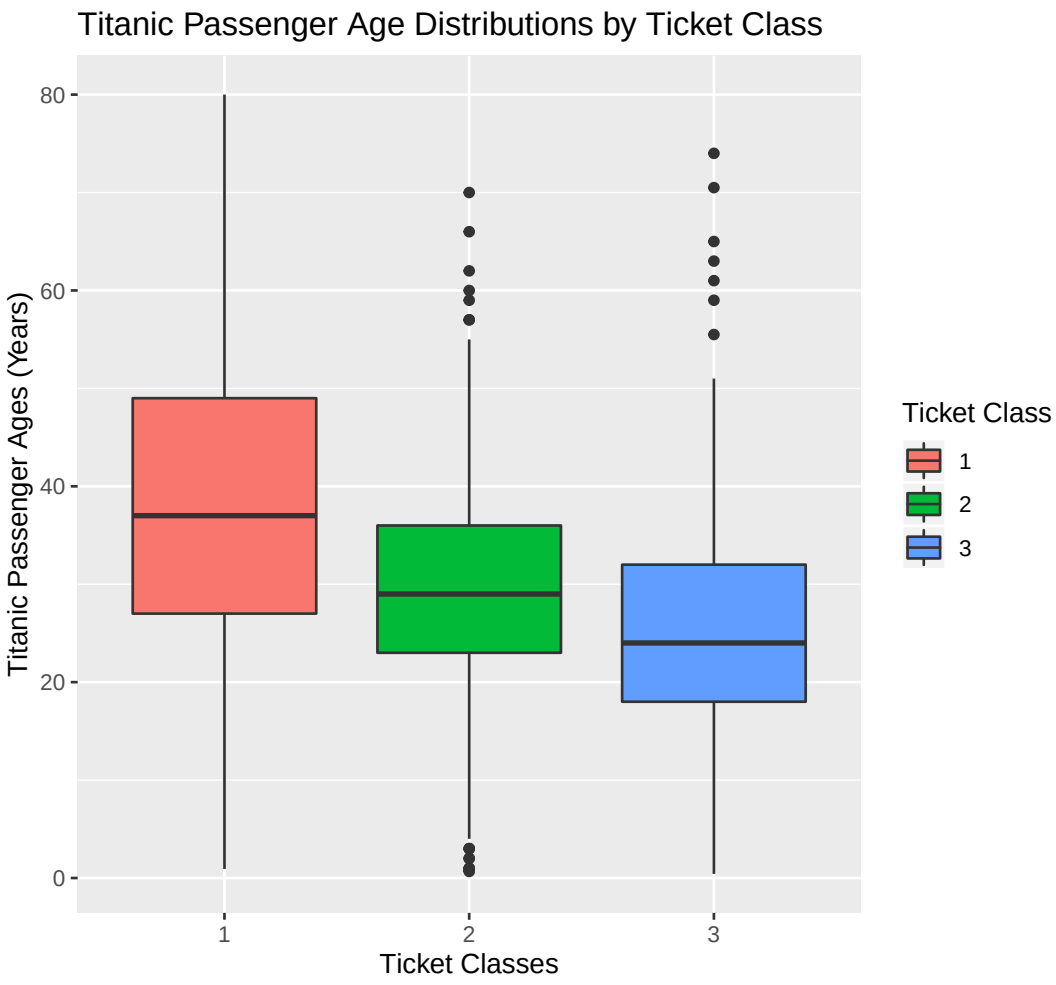
Notice that the argument `fill = as.factor(Pclass)` was placed inside of the `aes()` function. If we use a particular color for `fill` the argument should be placed outside of the `aes()` function.

```
ggplot(t.ship) + geom_boxplot(mapping = aes(x = as.factor(Pclass),
    y = Age), fill = "blue") + labs(y = "Titanic Passenger Ages (Years)",
    x = "Ticket Classes", title = "Titanic Passenger Age Distributions by Ticket Class",
    fill = "Ticket Class")
```



Also, notice that you can also place the mapping argument in the `ggplot()` function. Can you think of a situation where it will be good to do this?

```
ggplot(t.ship, mapping = aes(x = as.factor(Pclass), y = Age,
    fill = as.factor(Pclass))) + geom_boxplot() + labs(y = "Titanic Passenger Ages (Years)",
    x = "Ticket Classes", title = "Titanic Passenger Age Distributions by Ticket Class",
    fill = "Ticket Class")
```

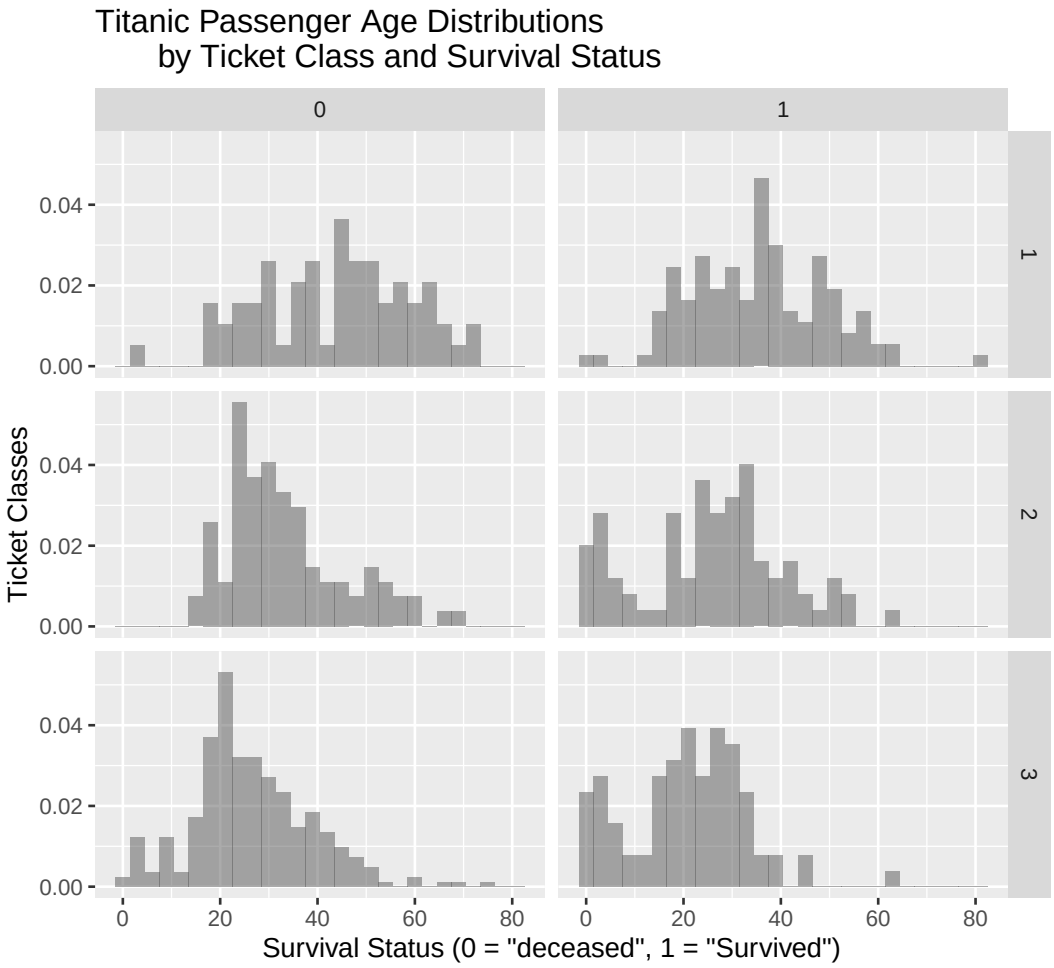


A more complex graph

Eventually we will make the graph shown at the beginning of the chapter but let's start with something simpler first.

```
ggplot(t.ship) + # `..density..` allows me to use density for the y-axis
# `binwidth` sets the bin width `alpha` sets the transparency
# of the graph
geom_histogram(mapping = aes(x = Age, y = ..density..), binwidth = 3,
    alpha = 0.5) + # `facet_grid` allows me to make a different plot for each
# subset of the data note that the variable on either side of
# `~` has to be a factor
facet_grid(as.factor(Pclass) ~ as.factor(Survived)) + # this tells R to create panels for the un
# of the `Pclass` values and the `Survived` values: the
# `Pclass` values correspond to the rows and the `Survived`
# value correspond to the columns
labs(x = "Survival Status (0 = \"deceased\", 1 = \"Survived\")",
```

```
    y = "Ticket Classes", title = "Titanic Passenger Age Distributions
       by Ticket Class and Survival Status")
```

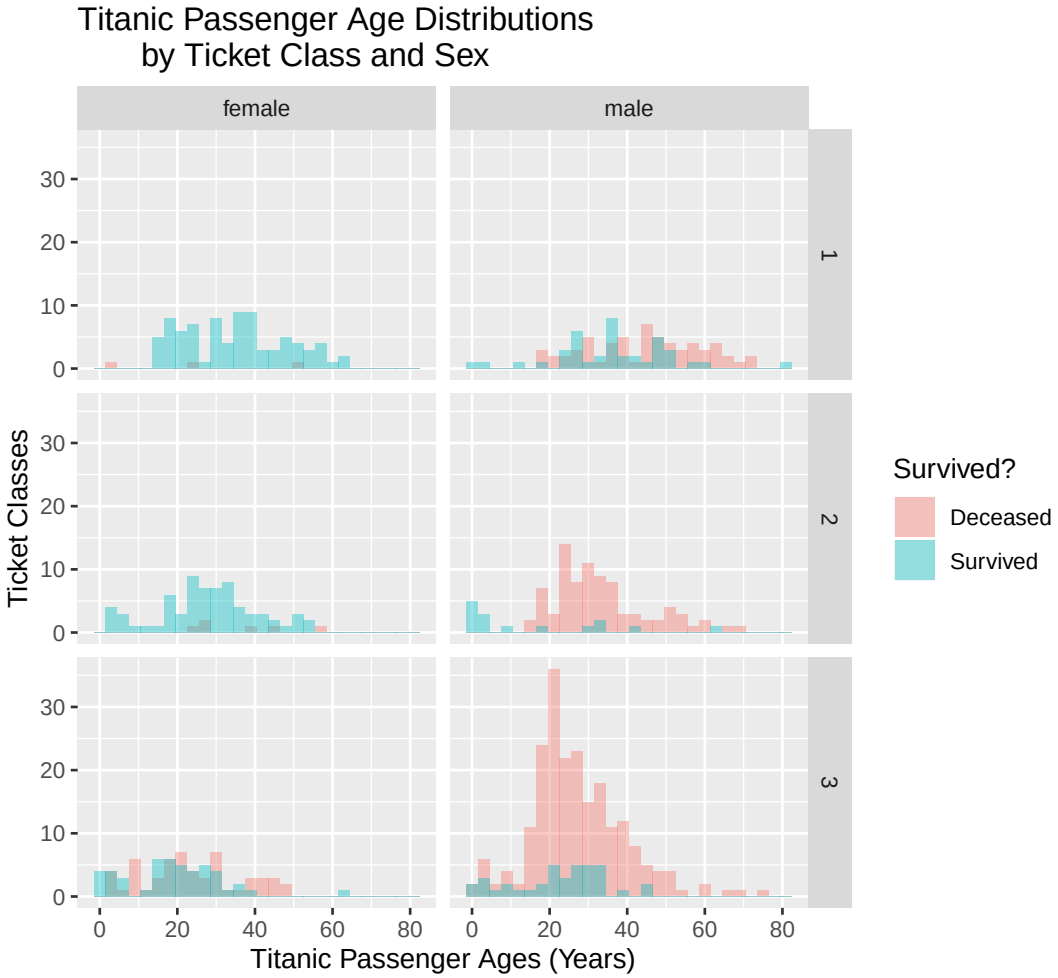


Now, let's produce the graph shown at the beginning of the chapter!

Exercise 1.: Produce a set of histograms for the passenger ages by ticket classes (rows) and gender (columns).

```
ggplot(t.ship) + # ggplot(t.ship) tells R that you're using
                 # the data in `t.ship` to make the graph
   geom_histogram(mapping = aes(x = Age,
                                fill = as.factor(Survived)),
                  # features related to the data in the
                  # dataset should be included in
                  # mapping = aes().
                  # fill = as.factor(Survived) gives different color to
                  # the graph for different the values in `Survived`
```

```
                position="identity",
                # the default is position="stack" which
                # stacks the bins
                binwidth = 3, alpha = .4) +
                # alpha controls the transparency of the bins
facet_grid(as.factor(Pclass)~Sex) +
# this tells R to create panels for the unique combinations of
# the `Pclass` values and the `Sex` values:
# the `Pclass` values correspond to the rows and
# the `Sex` value correspond to the columns
labs(x='Titanic Passenger Ages (Years)',
     y='Ticket Classes',
     title = 'Titanic Passenger Age Distributions
     by Ticket Class and Sex',
     fill = "Survived?" ) +
     # fill = "Survived?" is the header for the legend
scale_fill_discrete(labels = c("Deceased", "Survived"))
```




```
  # this labels the categories in the legend

# Warning message:
# Removed 177 rows containing non-finite values (stat_bin).
# This Warning is due to the fact that there are 177 missing
# values in `Age`
```

How to read the plots

In the set of the histograms above the area of a bin (relative to the total area of all the bins) represents the proportion of the passengers in a particular category relative to *all the Titanic passengers*. This is because R plots the absolute counts in the set of the histograms above. You can check this with the data:

```
with(t.ship, table(Pclass[Survived == 0], Sex[Survived == 0]))

    female male
```

```
  1      3   77
  2      6   91
  3     72  300
# this is the equivalent to
# table(t.ship$Pclass[t.ship$Survived == 0], t.ship$Sex[t.ship$Survived == 0]).
# the `with()` function allows you to tell R that all the data that you
# are using are from the dataset `t.ship`.
```

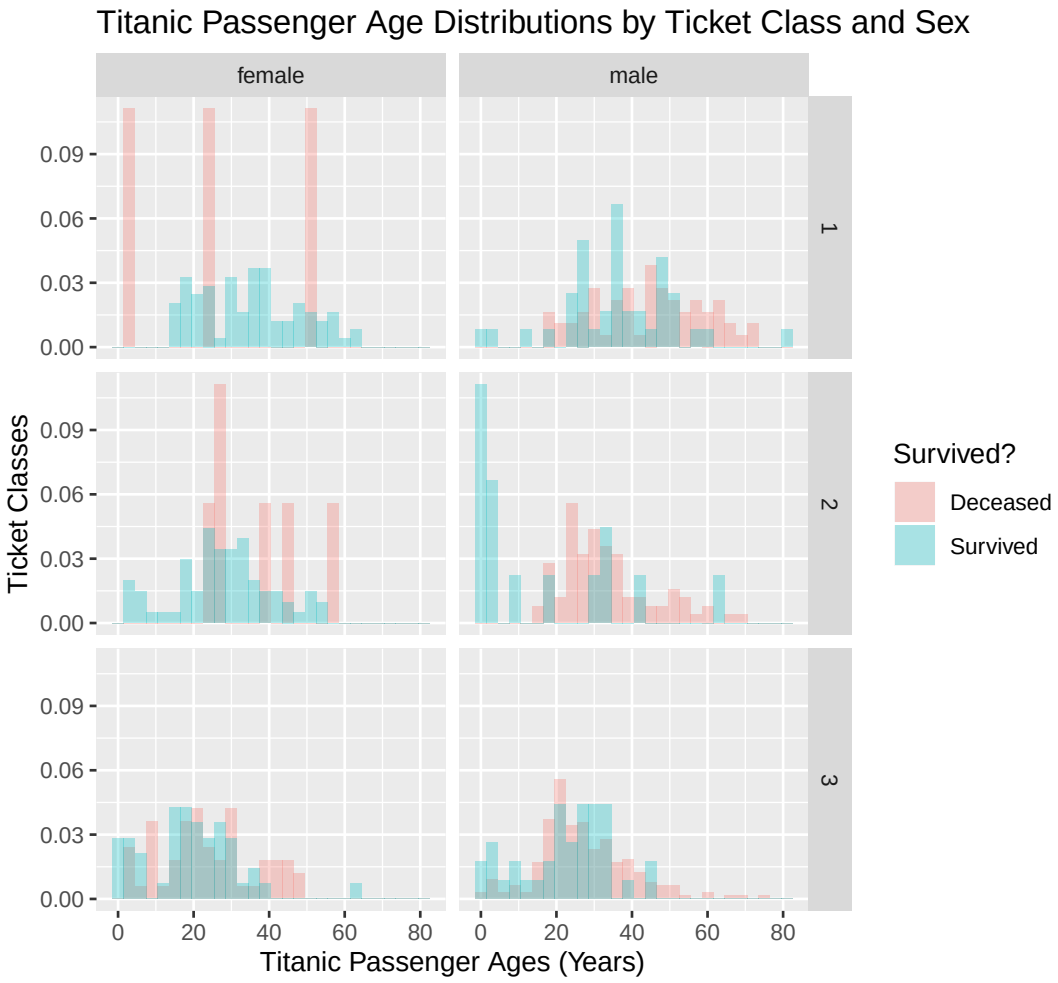
```
with(t.ship, table(Pclass[Survived == 1], Sex[Survived == 1]))
```

```
    female male
  1     91   45
  2     70   17
  3     72   47
```

Note that if we add in the `y = ..density..` argument the graph will give a different interpretation of the data:

In this case the area of a bin represents the proportion of the passengers in a particular category relative to all the passengers that fall on *a particular grid with a particular color*.

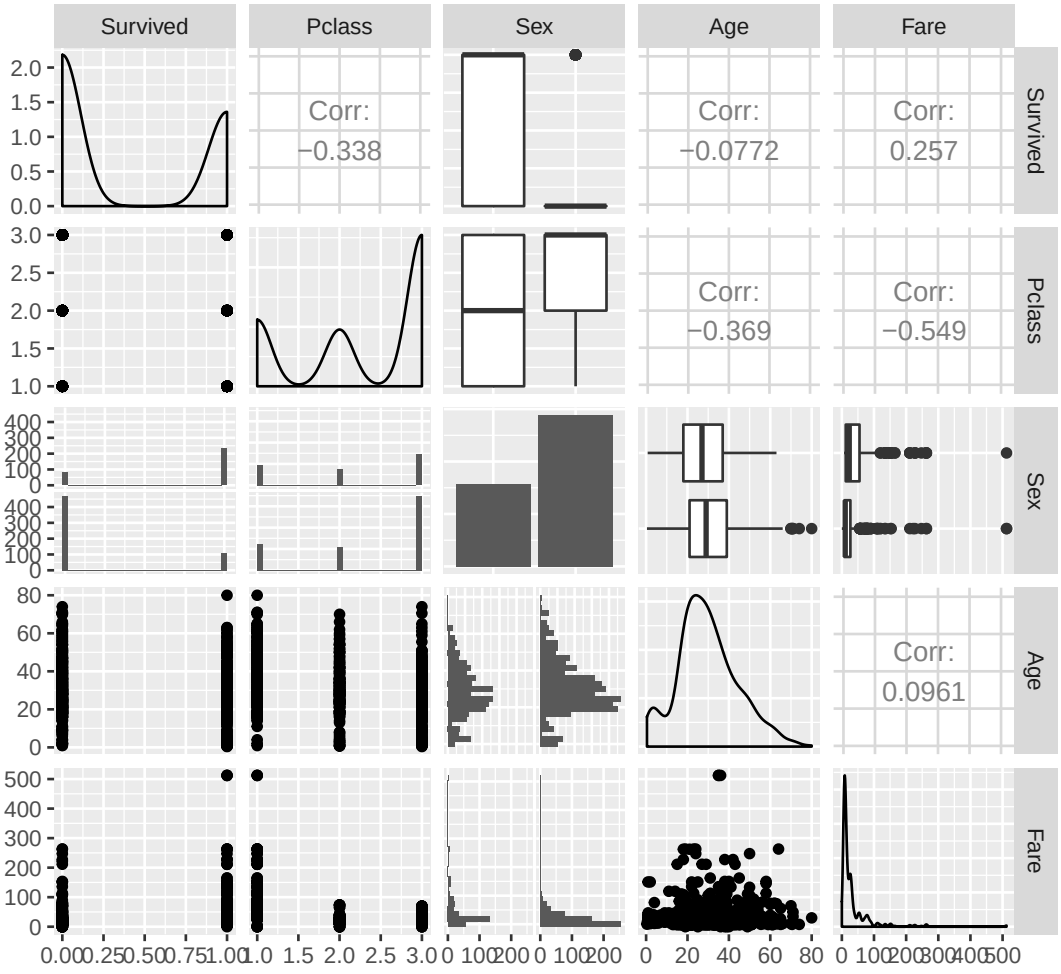
```
ggplot(t.ship) +
   geom_histogram(mapping = aes(x = Age, y = ..density..,
                                fill = as.factor(Survived)),
                  position="identity", binwidth = 3,
                  alpha = .3) +
  facet_grid(as.factor(Pclass)~Sex) +
  labs(x='Titanic Passenger Ages (Years)',
       y='Ticket Classes',
       title = 'Titanic Passenger Age Distributions by Ticket Class and Sex',
       fill = "Survived?" ) +
 scale_fill_discrete(labels = c("Deceased", "Survived"))
```



`ggpairs()`: investigating the overall pairwise relationship between variables.

We will first extract out the variables that we need for the graph.

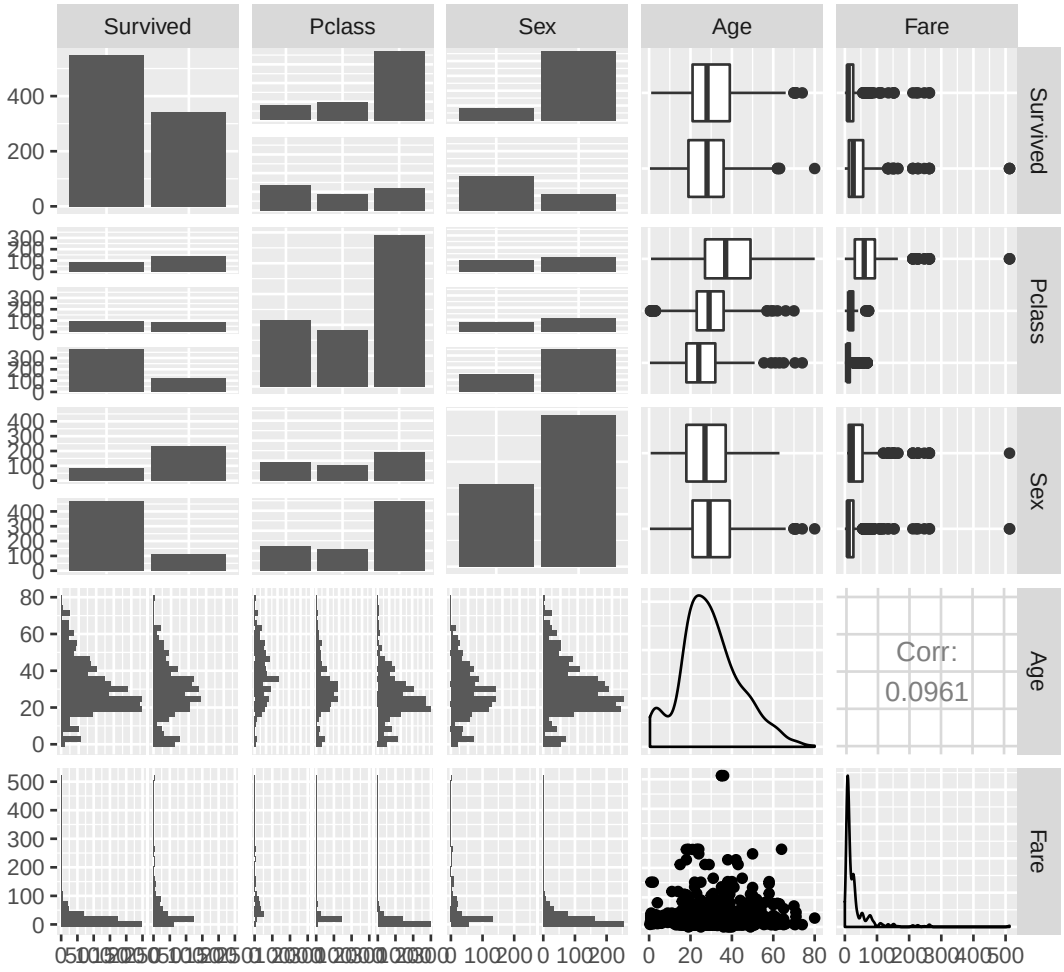
```
library(GGally)
# you will always need the `ggplot2` package when you are
# using the `GGally` package so make sure that you already
# load the `ggplot2` package too
small.t.ship = t.ship[, c("Survived", "Pclass", "Sex", "Age",
    "Fare")]
ggpairs(small.t.ship)
```



This is called a matrix plot.

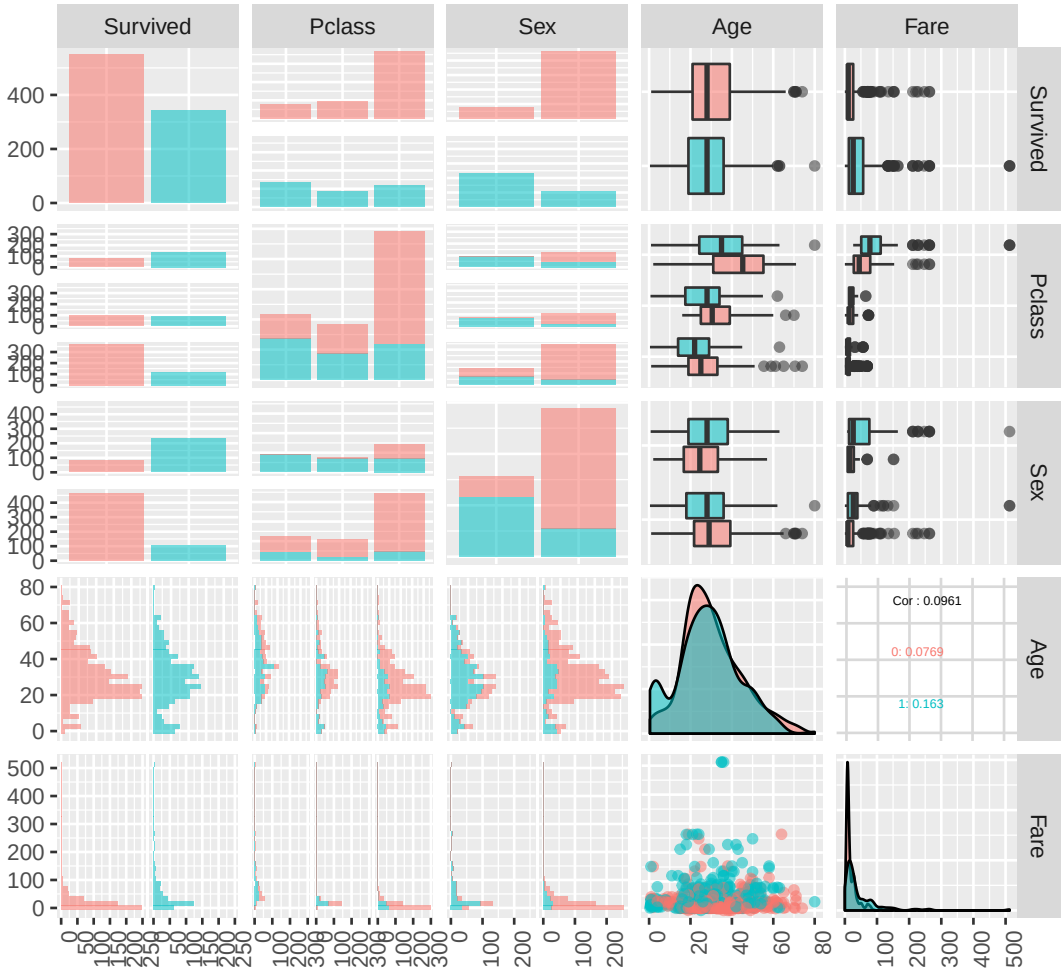
Some of the graphs on the matrix plot don't look right because they are not in the correct data types.

```
str(small.t.ship)
'data.frame':   891 obs. of  5 variables:
 $ Survived: int  0 1 1 1 0 0 0 0 1 1 ...
 $ Pclass  : int  3 1 3 1 3 3 1 3 3 2 ...
 $ Sex     : Factor w/ 2 levels "female","male": 2 1 1 1 2 2 2 2 1 1 ...
 $ Age     : num  22 38 26 35 35 NA 54 2 27 14 ...
 $ Fare    : num  7.25 71.28 7.92 53.1 8.05 ...

# change the categorical variables to factors
small.t.ship$Survived = as.factor(t.ship$Survived)
small.t.ship$Pclass = as.factor(t.ship$Pclass)

ggpairs(small.t.ship)
```



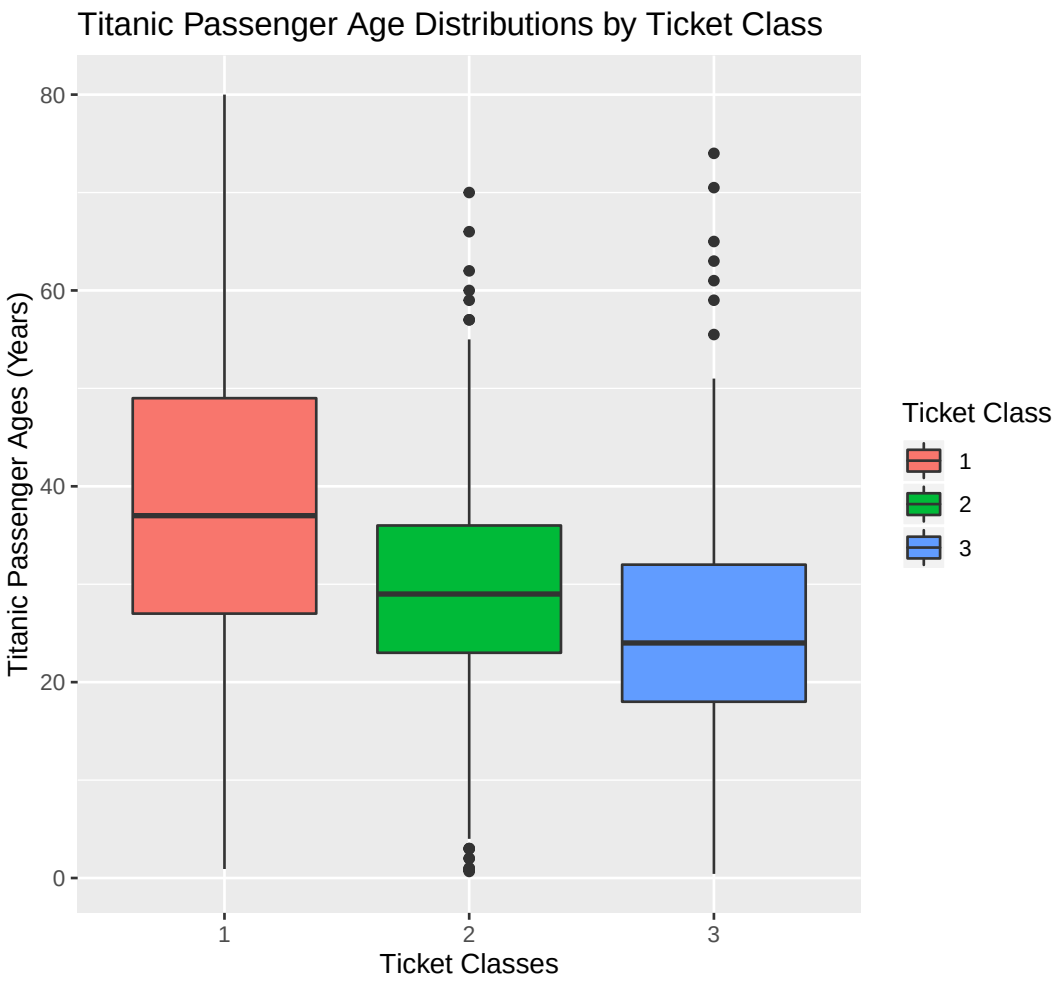
Now, we can also compare the pairwise relationship of the variables within each of the survived and deceased groups.

```
ggpairs(small.t.ship, aes(colour = Survived, alpha = 0.4), upper = list(continuous = wrap("cor",
    size = 1.9))) + theme(axis.text.x = element_text(angle = 90,
    hjust = 1))
```



Technical details related to concepts and functions covered in this chapter

```
ggplot(t.ship) + geom_boxplot(mapping = aes(x = as.factor(Pclass),
    y = Age, fill = as.factor(Pclass))) + labs(y = "Titanic Passenger Ages (Years)",
    x = "Ticket Classes", title = "Titanic Passenger Age Distributions by Ticket Class",
    fill = "Ticket Class")
```



We will create the dataset needed for this section.

```
compl.air = na.omit(airquality)
# remove NAs to prepare a dataset for the demo

g = ggplot(data = compl.air)
# tells R that we are going to use the dataset `compl.air` to
# make plots
```

Tidy data format

A dataset is in **tidy data** format[^1] if the observations are arranged as the rows and the variables are arranged as the columns of the dataset. For each row/observation the values of the variables are measured on the same object.

Question: Is the dataset `t.ship` in the tidy data format? How about the `airquality` dataset?

```
head(t.ship)
  PassengerId Survived Pclass
1           1        0      3
2           2        1      1
3           3        1      3
4           4        1      1
5           5        0      3
6           6        0      3
                                                 Name    Sex
1                             Braund, Mr. Owen Harris   male
2 Cumings, Mrs. John Bradley (Florence Briggs Thayer) female
3                              Heikkinen, Miss. Laina female
4        Futrelle, Mrs. Jacques Heath (Lily May Peel) female
5                            Allen, Mr. William Henry   male
6                                    Moran, Mr. James   male
  Age SibSp Parch           Ticket    Fare Cabin Embarked
1  22     1     0        A/5 21171  7.2500              S
2  38     1     0         PC 17599 71.2833   C85        C
3  26     0     0 STON/O2. 3101282  7.9250              S
4  35     1     0           113803 53.1000  C123        S
5  35     0     0           373450  8.0500              S
6  NA     0     0           330877  8.4583              Q

head(airquality)
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

From: H. Wickham (2014), *Tidy Data*, *Journal of Statistical Software*

Restructure a dataset for ggplot

The tidy format is not necessarily the correct form for making a plot with ggplot function. Here we will show you how to prepare the data to be used for making ggplot graphs.

We will first create a dataset for the example. This dataset consisting of historical average scores for 5 students when they were at different grade levels.

```
set.seed(8902)  # this sets the seed for random number generation.
# by setting the seed, everybody running the line of code
# below will get the same numbers.
grade = matrix(c(78, 90, 84, 95, 87) + rnorm(n = 25, mean = 0,
    sd = 2), nrow = 5)
rownames(grade) = paste(rep("student", 5), 1:5, sep = "")
colnames(grade) = paste(rep("scores.grade", 5), 1:5, sep = "")
```

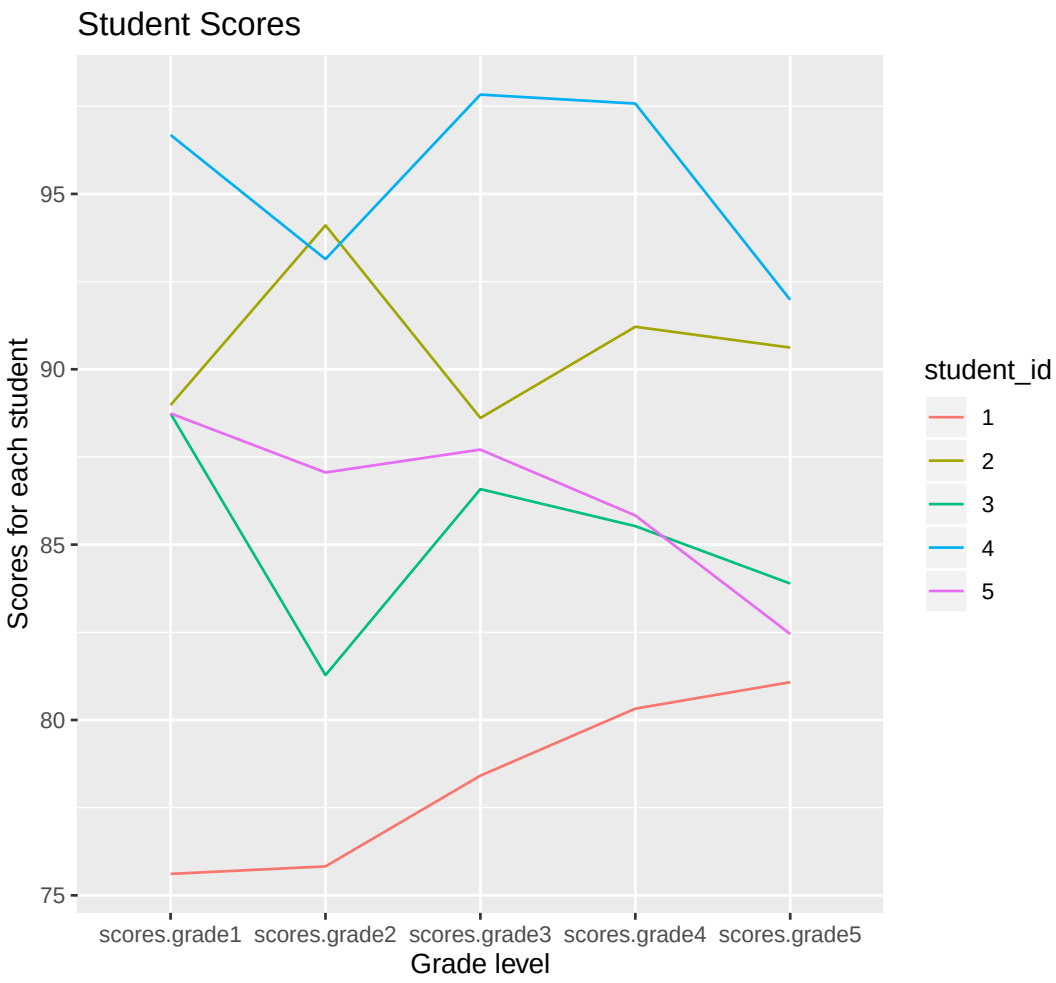
```
# take a look at the dataset:
grade
         scores.grade1 scores.grade2 scores.grade3
student1      75.61129      75.82484      78.41237
student2      88.97910      94.10457      88.61161
student3      88.73358      81.27816      86.58292
student4      96.68129      93.14097      97.83177
student5      88.74068      87.05681      87.70862
         scores.grade4 scores.grade5
student1      80.32333      81.07747
student2      91.21141      90.61836
student3      85.52735      83.89072
student4      97.57671      91.98019
student5      85.83057      82.45199
```

We would like to make a line plot for each student's scores.

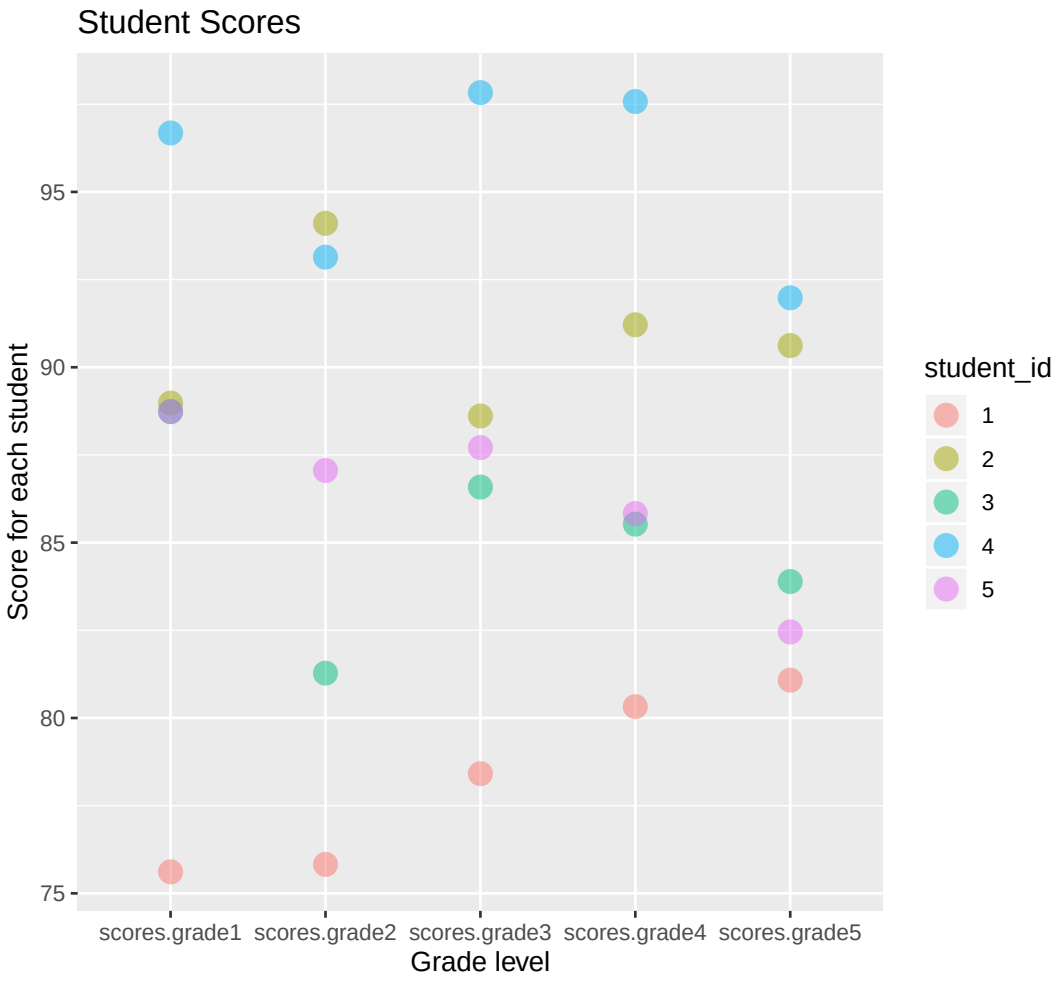
As it is now the dataset is not in the suitable object type or structure for `ggplot()`.

We want to restructure the data and the resulting data frame should have 3 variables: `scores`, `student_id` and `grade_levels`; this way we can supply the data frame to `ggplot()`. We will first show you how to do this manually since doing so will help you understand the structure of the data frame needed for ggplot functions better.

```
per.student = data.frame(score = as.vector(grade),
                         student_id = as.factor(rep(1:nrow(grade), times = ncol(grade))),
                         grade_level = rep(colnames(grade), each = nrow(grade))
                         )
head(per.student, n=10)
      score student_id   grade_level
1  75.61129          1 scores.grade1
2  88.97910          2 scores.grade1
3  88.73358          3 scores.grade1
4  96.68129          4 scores.grade1
5  88.74068          5 scores.grade1
6  75.82484          1 scores.grade2
7  94.10457          2 scores.grade2
8  81.27816          3 scores.grade2
9  93.14097          4 scores.grade2
10 87.05681          5 scores.grade2

ggplot(per.student) + geom_line(mapping = aes(x = grade_level,
y = score, color = student_id, group = student_id)) +
labs(x = "Grade level", y = "Scores for each student", title='Student Scores')
```




```
# If there were no temporal relationship between the x-values
# (e.g., if the scores were for different subjects), you can
# also display the scores with dots
ggplot(per.student) + geom_point(mapping = aes(x = grade_level,
y = score, color = student_id), alpha=.5, size=4) +
labs(x = "Grade level", y = "Score for each student", title='Student Scores')
```



Now you understand what you are supposed to do to transform the dataset into a new data frame that is suitable for `ggplot()` we will show you how to use the function in the `reshape` package to do this automatically:

```
library(reshape2)
long.grade <- melt(grade)
head(long.grade, n = 10)
       Var1          Var2    value
1  student1 scores.grade1 75.61129
2  student2 scores.grade1 88.97910
3  student3 scores.grade1 88.73358
4  student4 scores.grade1 96.68129
5  student5 scores.grade1 88.74068
6  student1 scores.grade2 75.82484
7  student2 scores.grade2 94.10457
8  student3 scores.grade2 81.27816
9  student4 scores.grade2 93.14097
```

```
10 student5 scores.grade2 87.05681
class(long.grade)
[1] "data.frame"
names(long.grade) = c("student_id", "grade_level", "score")
head(long.grade)  # same as the data frame that we created manually
  student_id   grade_level    score
1   student1 scores.grade1 75.61129
2   student2 scores.grade1 88.97910
3   student3 scores.grade1 88.73358
4   student4 scores.grade1 96.68129
5   student5 scores.grade1 88.74068
6   student1 scores.grade2 75.82484
```

To put the data back to the original tidy format we can use the function `dcast()` in the `reshape` package.

```
# This displays the values of one variable for the dataset
# and the info from the other two variables are shown in the
# row and column names
wide.grade <- dcast(long.grade, student_id ~ grade_level)
head(wide.grade)
  student_id scores.grade1 scores.grade2 scores.grade3
1   student1      75.61129      75.82484      78.41237
2   student2      88.97910      94.10457      88.61161
3   student3      88.73358      81.27816      86.58292
4   student4      96.68129      93.14097      97.83177
5   student5      88.74068      87.05681      87.70862
  scores.grade4 scores.grade5
1      80.32333      81.07747
2      91.21141      90.61836
3      85.52735      83.89072
4      97.57671      91.98019
5      85.83057      82.45199

# note that the output of `dcast()` is still a data frame
class(wide.grade)
[1] "data.frame"
```

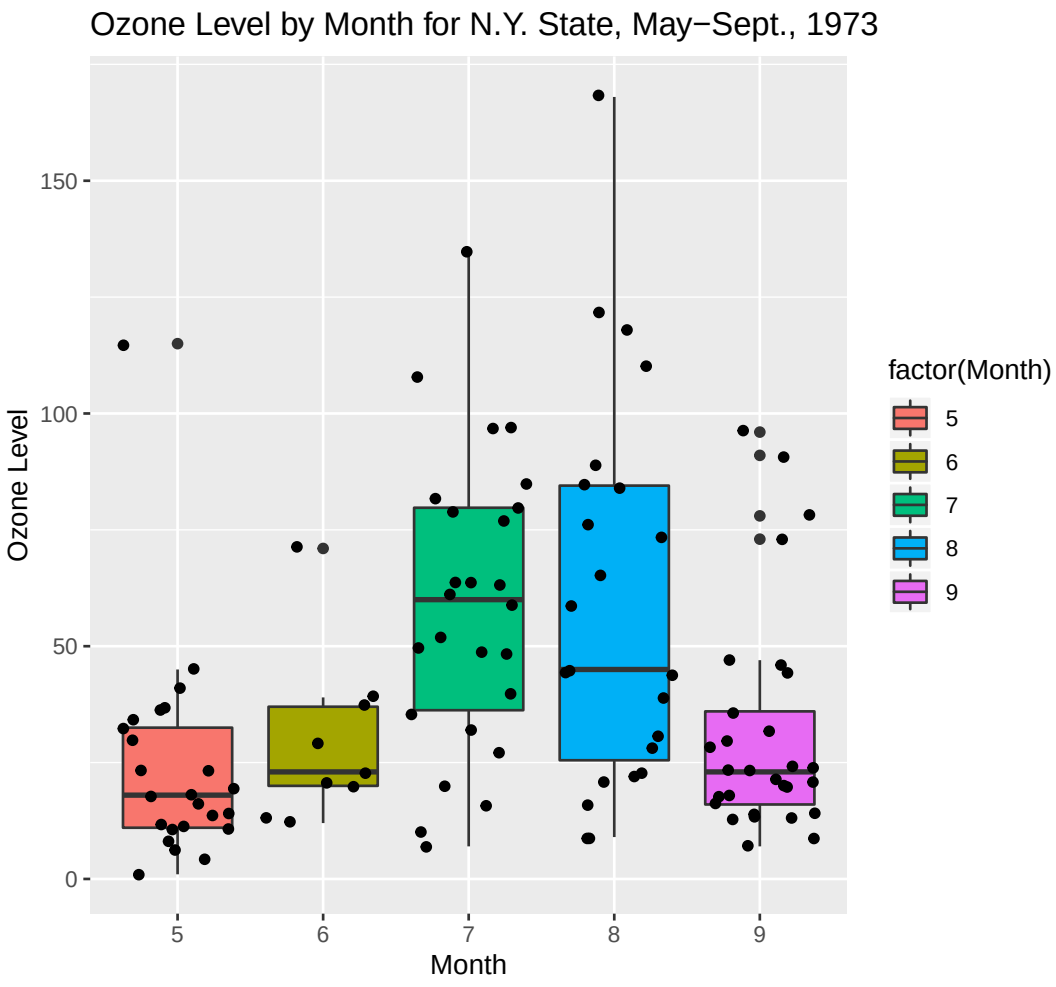
Setting the x- and y- variables for your graphs globally

```
ggplot(data = compl.air) + geom_boxplot(mapping = aes(x = factor(Month),
    y = Ozone, fill = factor(Month))) + geom_point(mapping = aes(x = factor(Month),
    y = Ozone)) + labs(title = "Ozone Level by Month for N.Y. State, May-Sept., 1973",
    x = "Month", y = "Ozone Level")


# The following line returns the same result as the line
```

```
# above
ggplot(data = compl.air, mapping = aes(x = factor(Month), y = Ozone)) +
    geom_boxplot(mapping = aes(fill = factor(Month))) + geom_point() +
    labs(title = "Ozone Level by Month for N.Y. State, May-Sept., 1973",
        x = "Month", y = "Ozone Level")
```

Using the jitter function

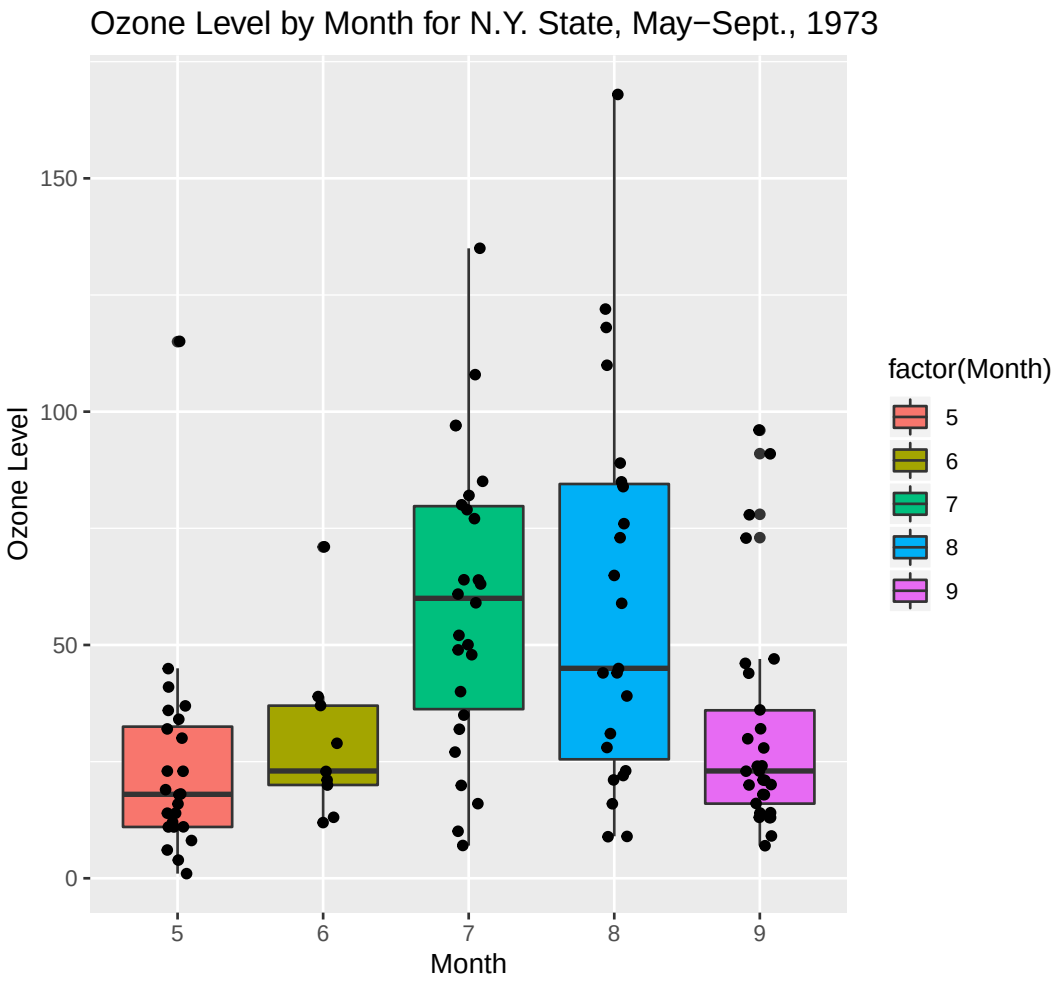
```
g + geom_boxplot(mapping = aes(x = factor(Month), y = Ozone,
    fill = factor(Month))) + geom_jitter(mapping = aes(x = factor(Month),
    y = Ozone)) + labs(title = "Ozone Level by Month for N.Y. State, May-Sept., 1973",
    x = "Month", y = "Ozone Level")
```



We use `jitter()` to distinguish overlapping points; however, the spread for jittered data in

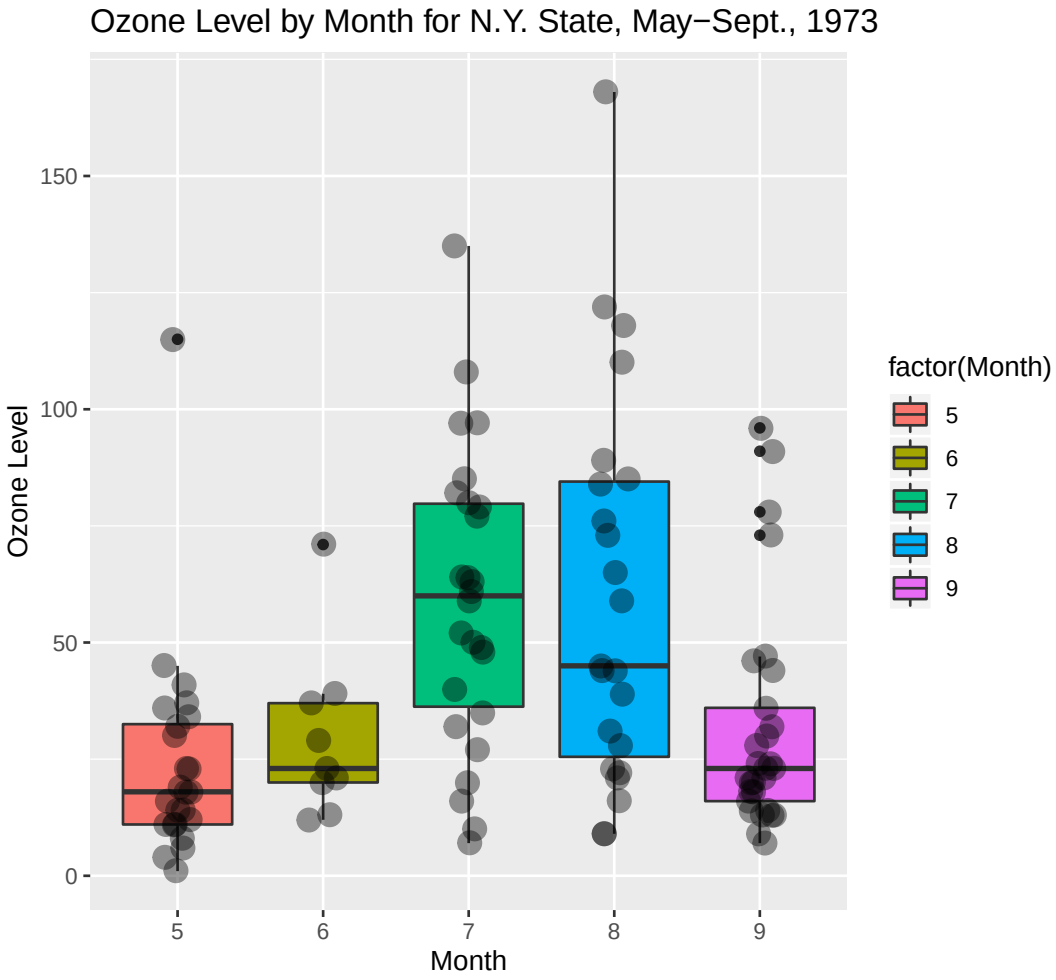
the previous graph is too big.

```
g + geom_boxplot(mapping = aes(x = factor(Month), y = Ozone,
    fill = factor(Month))) + geom_jitter(mapping = aes(x = factor(Month),
    y = Ozone), width = 0.1, height = 0.1) + labs(title = "Ozone Level by Month for N.Y. State, M
    x = "Month", y = "Ozone Level")
```



`width` and `height` adjust the amount of jitter.

```
ggplot(data = compl.air, mapping = aes(x = factor(Month), y = Ozone)) +
    geom_boxplot(mapping = aes(fill = factor(Month))) + geom_jitter(width = 0.1,
    height = 0.1, alpha = 0.4, size = 4) + labs(title = "Ozone Level by Month for N.Y. State, May
    x = "Month", y = "Ozone Level")
```

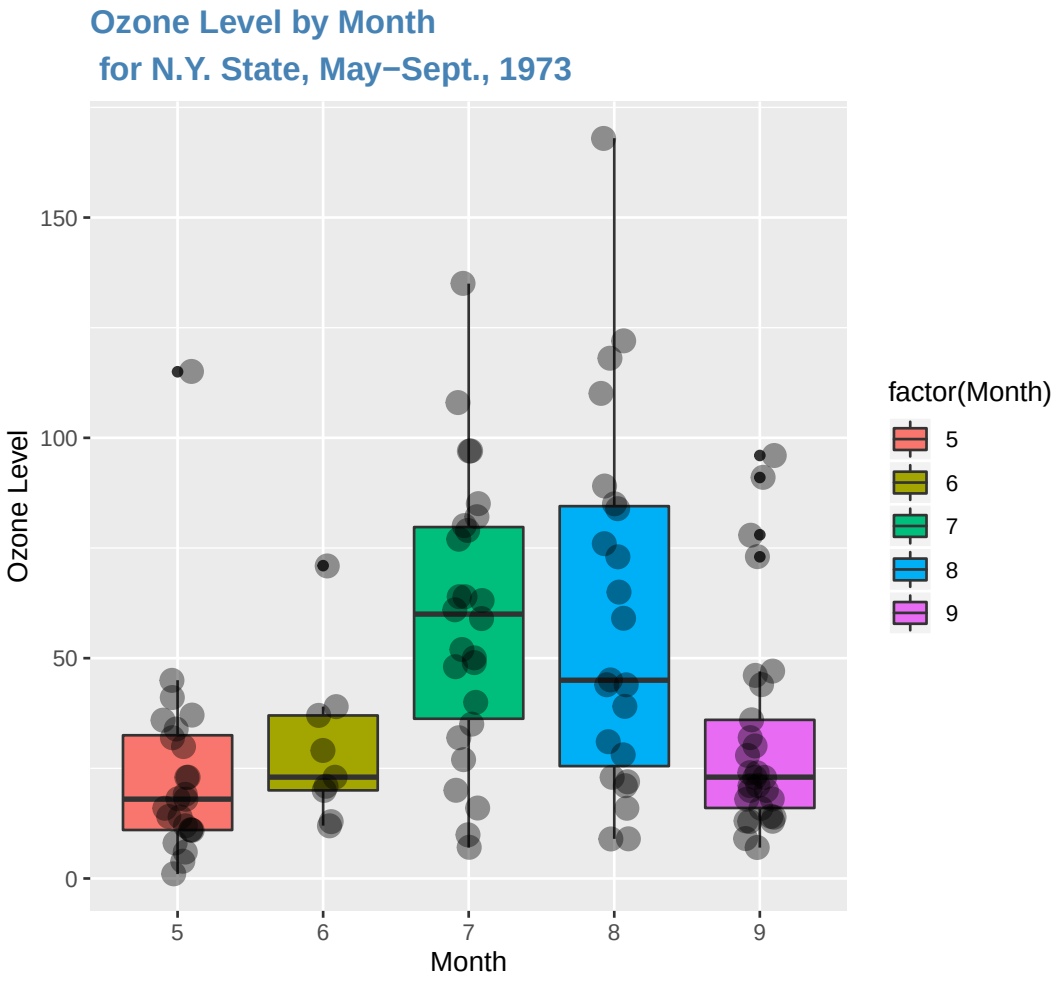


`alpha` controls transparency; `size` controls the size of the dots.

Adjust the variables for the theme

We can also make the title bold faced, change its color and adjust the height of the line between the two lines in the title. Note that we use `\n` to break the line of the title into two lines.

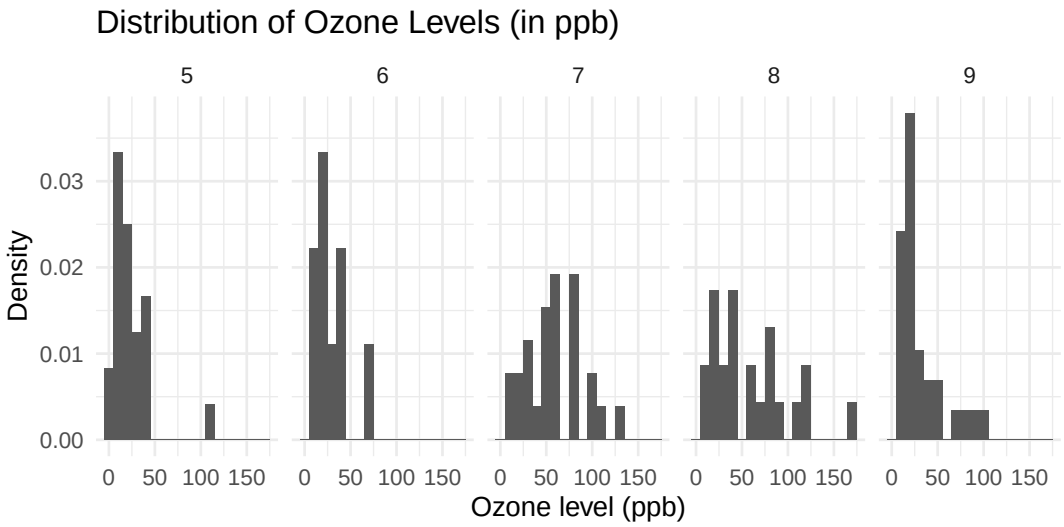
```
ggplot(data = compl.air, mapping = aes(x = factor(Month), y = Ozone)) +
    geom_boxplot(mapping = aes(fill = factor(Month))) + geom_jitter(width = 0.1,
    height = 0.1, alpha = 0.4, size = 4) + labs(title = "Ozone Level by Month \n for N.Y. State,
    x = "Month", y = "Ozone Level") + theme(plot.title = element_text(face = "bold",
    color = "steelblue", lineheight = 1.2))
```



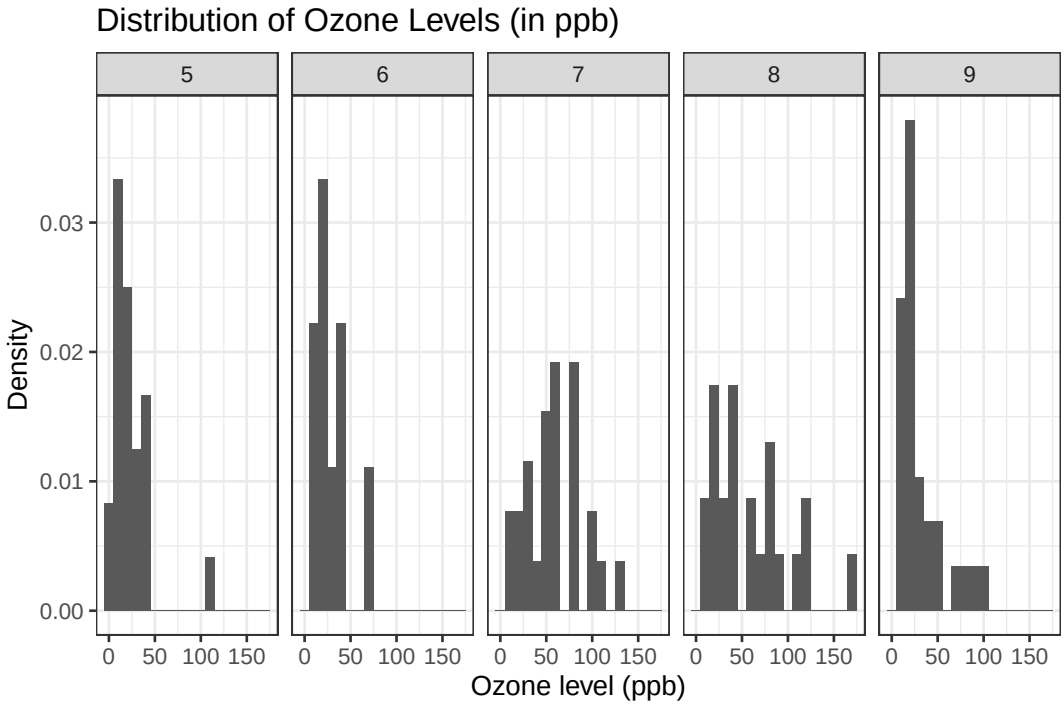
Set theme locally

You can set the theme for your graph locally by adding a layer for the theme

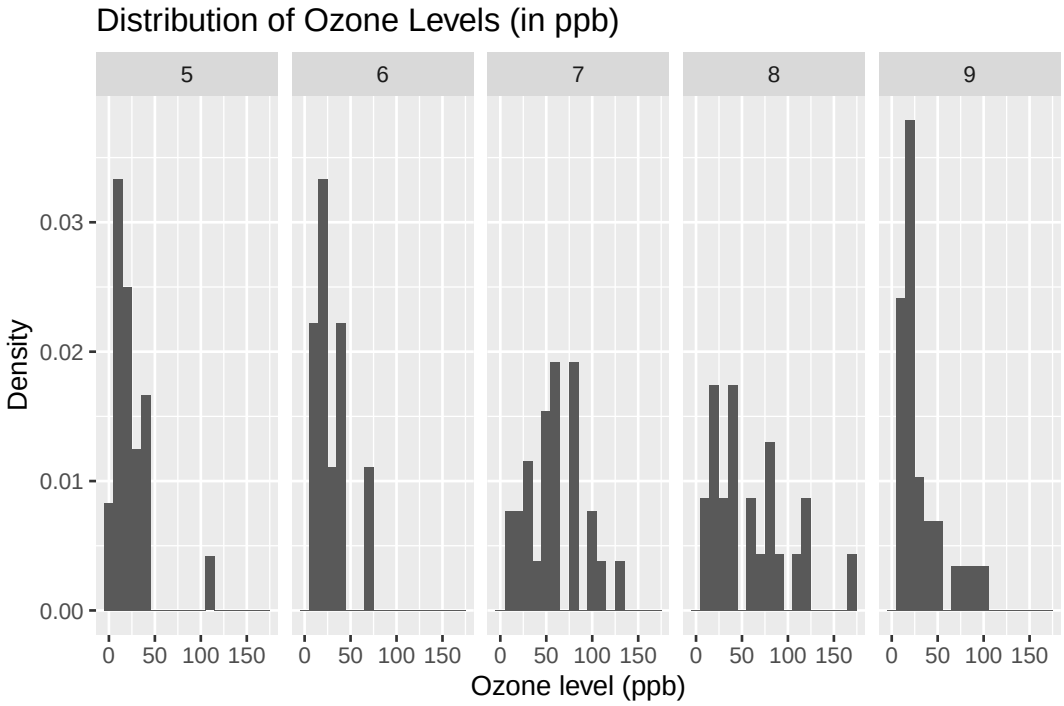
```
# use the minimal theme
g + geom_histogram(mapping = aes(x=Ozone, ..density..),
                   binwidth=10) +
  labs(x = "Ozone level (ppb)",y = "Density",
       title = "Distribution of Ozone Levels (in ppb)") +
  facet_grid(.~Month) + theme_minimal()
```




```
g + geom_histogram(mapping = aes(x=Ozone, ..density..),
                   binwidth=10) +
  labs(x = "Ozone level (ppb)",y = "Density",
       title = "Distribution of Ozone Levels (in ppb)") +
  facet_grid(.~Month) + theme_bw()
```



```
g + geom_histogram(mapping = aes(x=Ozone, ..density..),
                   binwidth=10) +
  labs(x = "Ozone level (ppb)",y = "Density",
       title = "Distribution of Ozone Levels (in ppb)") +
  facet_grid(.~Month) + theme_grey()
```

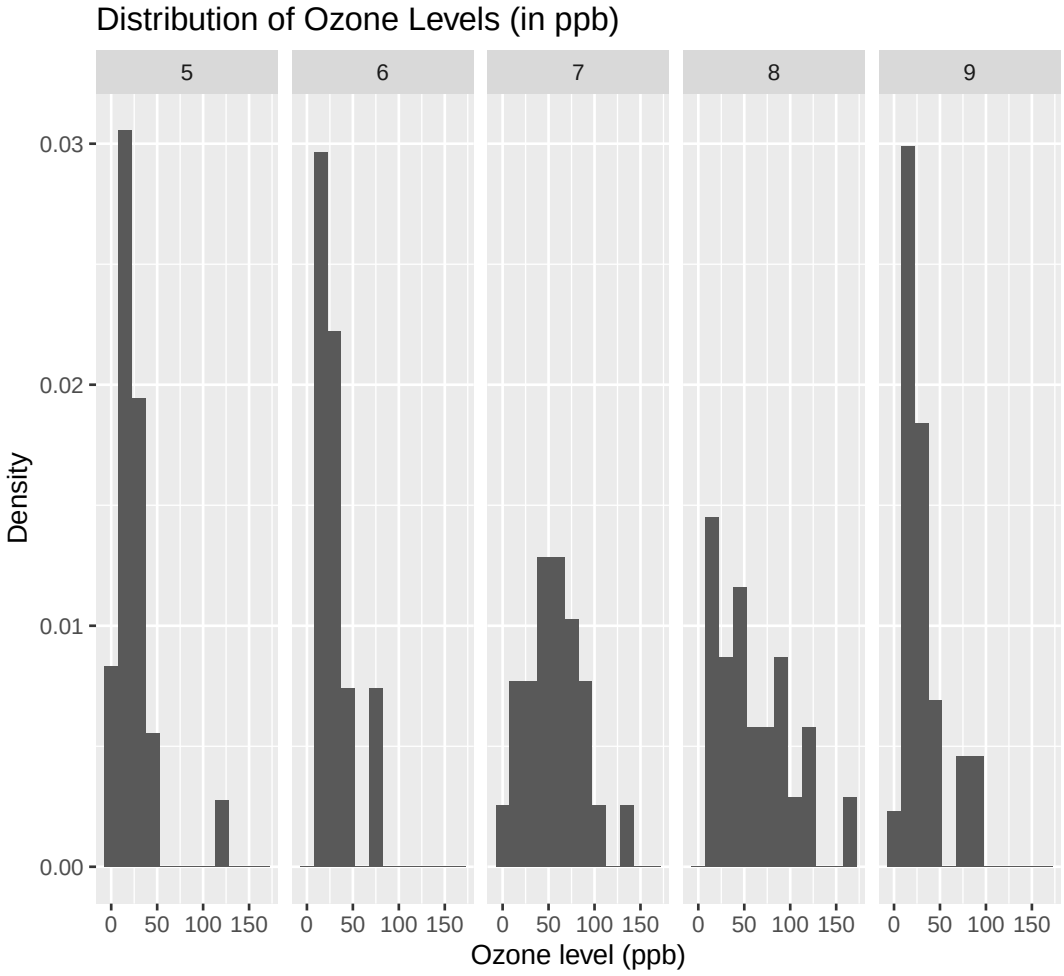


Set theme globally

You can also set the theme globally by using the `theme_set()` function

theme_set(theme_bw())
theme_set(theme_grey())
theme_set(theme_minimal())

```
theme_set(theme_grey())

g + geom_histogram(mapping = aes(x = Ozone, ..density..), binwidth = 15) +
    labs(x = "Ozone level (ppb)", y = "Density", title = "Distribution of Ozone Levels (in ppb)")
    facet_grid(. ~ Month)
```


Distribution of Ozone Levels (in ppb)
5
6
7
8
9
0.03
0.02
0.01
0.00
Density
0
50
100
150
Ozone level (ppb)