Quantitative Methods for Political Economy

Seminar 2: Exploring and visualising data

Friday 23 October 2026

Wouter Leenders

wouter.leenders@kcl.ac.uk

1.1 Introduction

1.1.1 Prerequisites

We can create plots with base R. In this seminar, we will use `ggplot2` for plotting and `readr` for importing data.

```
# Install once, if needed:
# install.packages("ggplot2")
# install.packages("readr")

# Load each session:
library(ggplot2)
library(readr)
```

To install a missing package, remove # from its installation line and run it once. The `library()` lines need to be run each session.

Getting the BES data

We will use data from the **British Election Study**, collected after the 2019 general election.

We can import the CSV directly from this website:

```
bes <- read_csv("https://wouterleenders.eu/7SSPN107/bes_2019.csv")
```

This reads the data and stores it in a data frame called `bes`.

1.2 First steps

1.2.1 The BES data frame

32,177 respondents from England, Scotland and Wales. Each row represents one respondent.

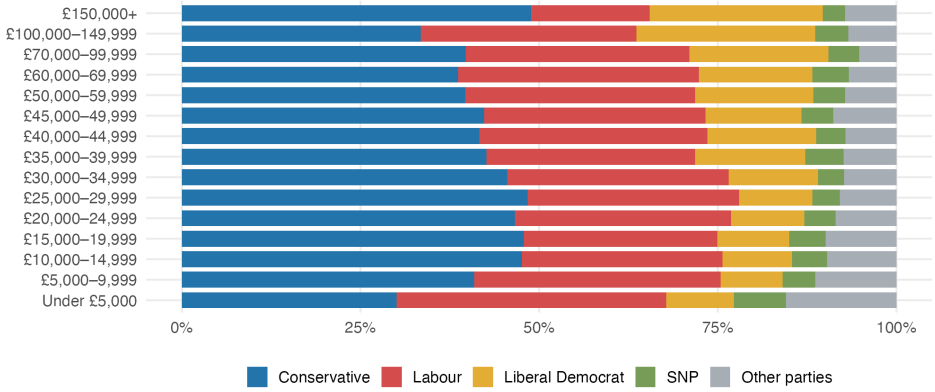
Variable	Meaning
`age`	Age in years
`vote`	Party voted for in the 2019 election
`income_band`	Annual household income before tax, in bands
`like_labour`	Labour rating: 0 = strongly dislike, 10 = strongly like
`like_conservative`	Conservative rating on the same scale

We can inspect the data by any of the following commands:

```
bes
str(bes)
print(bes,n=20)
View(bes)
```

1.2.2 Ultimate goal

How does party choice vary across household income groups?



1.2.3 Creating a ggplot: data

A plot consists of different layers. The first layer is the data that we want to plot.

We want to include respondents for whom we observe both income and vote choice:

```
plot_data <- subset(bes,!is.na(income_band) & !is.na(vote))
```

`is.na()` identifies missing values; `!` means "not" and `&` means "and".

Then, we want to tell `ggplot()` which data to use:

```
ggplot(data = plot_data)
```

The result is a blank plot. We have supplied the data, but have not specified which variables to display or how to draw them.

1.2.3 Creating a ggplot: mappings

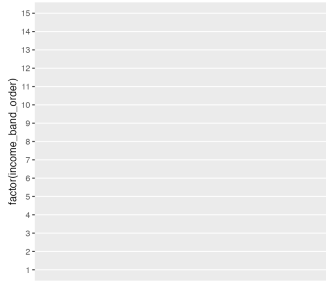
Next, we need to tell ggplot which parts of the dataset to plot.

```
ggplot(data = plot_data,
  mapping = aes(y = factor(income_band_order),
                fill = vote))
```

`aes()` specifies which variables to plot.
`y` places income groups on the vertical axis.
`factor()`, treats `income_band_order` as a categorical variable.

`fill` controls the colour inside shapes. With `fill = vote`, each party will have a different fill colour.

The plot now contains a y-axis, but still no data points. For that, we need to define geometrical objects.

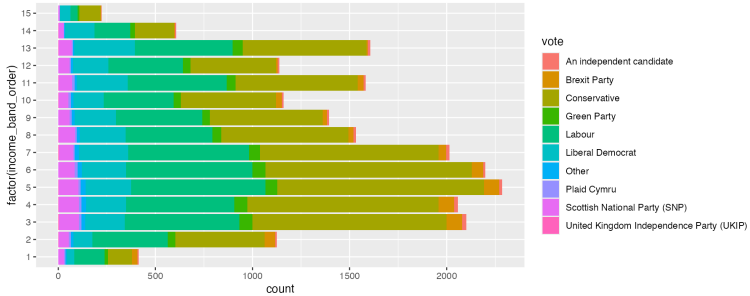


1.2.4 Adding aesthetics and layers

A **geom** is a geometrical object that ggplot uses to plot data, such as a line, a point, a bar, etc.

To add layers to a plot, we simply use +.

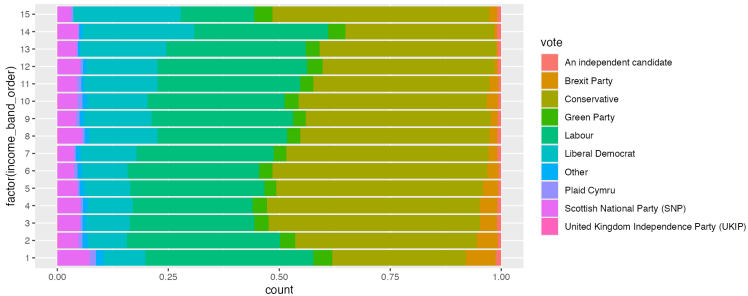
```
ggplot(data = plot_data,
  mapping = aes(y = factor(income_band_order),
                fill = vote)) +
  geom_bar()
```



1.2.4 Counts to proportions

Use `position = "fill"` to show the vote share of each party within each income band:

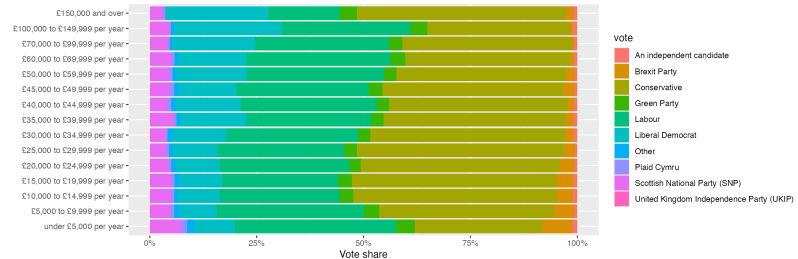
```
ggplot(data = plot_data,
  mapping = aes(y = factor(income_band_order),
                fill = vote)) +
  geom_bar(position = "fill")
```



1.2.4 Income labels and percentages

`setNames()` matches income labels to band numbers; the scales format the axes.

```
income_labels <- setNames(plot_data$income_band,
                          plot_data$income_band_order)
ggplot(plot_data,
  aes(y = factor(income_band_order), fill = vote)) +
  geom_bar(position = "fill") +
  scale_y_discrete(name = NULL, labels = income_labels) +
  scale_x_continuous(name = "Vote share", labels = scales::label_percent()
    )
```



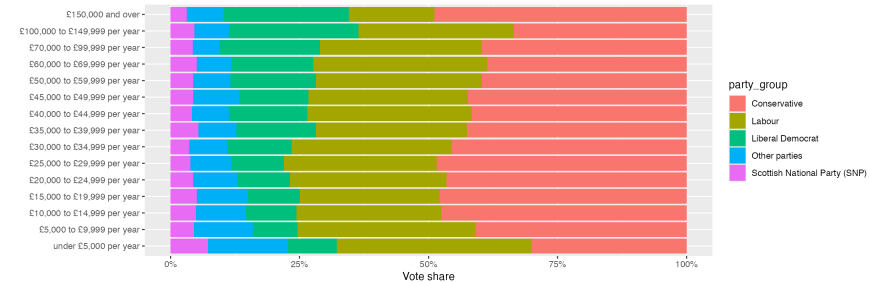
1.2.4 Grouping other parties

Let's focus on the four largest parties and group the others together:

```
main_parties <- c("Conservative", "Labour", "Liberal Democrat",
                  "Scottish National Party (SNP)")
plot_data$party_group <- ifelse(
  plot_data$vote %in% main_parties, plot_data$vote, "Other parties")
```

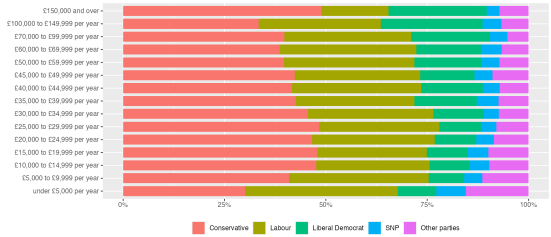
`%in%` checks whether a value matches any value in a set; `ifelse()` keeps those party names and assigns "Other parties" to the rest.

Rerun the previous plot, replacing `fill = vote` with `fill = party_group`.



1.2.4 Legend and party order

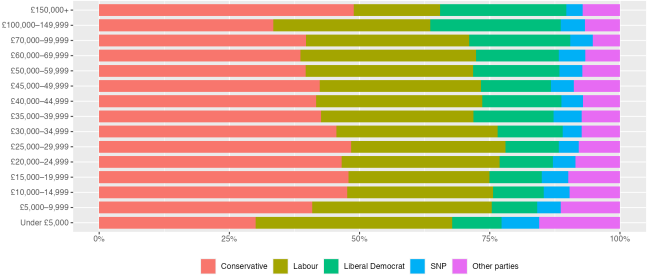
```
party_order <- c("Conservative", "Labour", "Liberal Democrat", "SNP", "
    Other parties")
plot_data$party_group <- factor(plot_data$party_group,
  levels = c(main_parties, "Other parties"), labels = party_order)
party_plot <- ggplot(plot_data,
  aes(y = factor(income_band_order), fill = party_group)) +
  geom_bar(position = position_fill(reverse = TRUE), width = 0.78) +
  scale_y_discrete(name = NULL, labels = income_labels) +
  scale_x_continuous(name = NULL, labels = scales::label_percent()) +
  labs(fill = NULL) + theme(legend.position = "bottom")
```



1.2.4 Shortening the income labels

`sub()` replaces text in each label. An empty string removes that text.

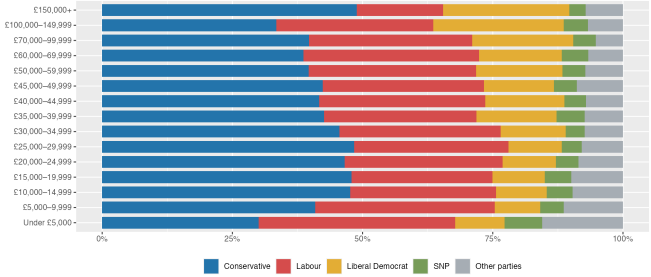
```
income_labels <- sub(" per year", "", income_labels)
income_labels <- sub(" to £", "", income_labels)
income_labels <- sub("under", "Under", income_labels)
income_labels <- sub(" and over", "+", income_labels)
party_plot <- party_plot +
  scale_y_discrete(name = NULL, labels = income_labels)
party_plot
```



1.2.4 Choosing the party colours

Match each party name to a colour, then apply it with `scale_fill_manual()`.

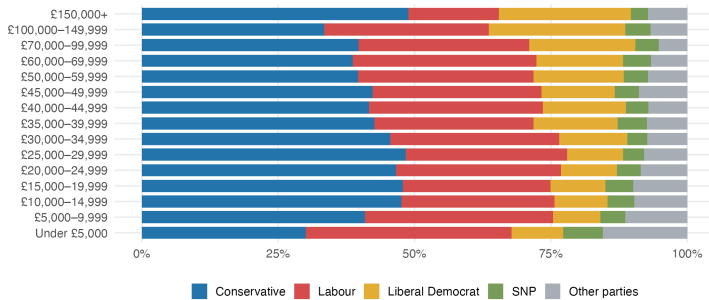
```
party_colours <- c(
  "Conservative" = "#2374AB", "Labour" = "#D54C4C",
  "Liberal Democrat" = "#E3AD35", "SNP" = "#779C58",
  "Other parties" = "#A6ADB4")
party_plot <- party_plot +
  scale_fill_manual(values = party_colours)
party_plot
```



1.2.4 Applying a theme

Use a minimal background, keep the legend below and remove minor grid lines.

```
income_plot <- party_plot +
  theme_minimal(base_size = 15) +
  theme(legend.position = "bottom",
        panel.grid.minor = element_blank())
income_plot
```



1.2.5 Exercises

1. Compare the count and percentage charts. What can you learn from the count chart that disappears when every bar totals 100%?

2. Compare the charts before and after grouping “Other parties”. Do the Conservative and Labour shares change? What would happen if we removed the other parties’ voters instead?

3. Move the legend above the finished plot using `theme(legend.position = "top")`. Add a title using `labs(title = "Party choice by income")`.

4. Describe how the Conservative share varies with income. Is the pattern consistently increasing or decreasing? Give one reason why this chart does not show that income causes party choice.

1.3 ggplot2 calls

1.3 ggplot2 calls

These two calls produce the same plot:

```
ggplot(data = plot_data,
  mapping = aes(y = factor(income_band_order),
                fill = party_group)) +
  geom_bar(position = "fill")
```

We can omit the argument names `data =` and `mapping =`:

```
ggplot(plot_data,
  aes(y = factor(income_band_order), fill = party_group)) +
  geom_bar(position = "fill")
```

R uses the argument positions: data first, mapping second. Keep + at the end of a line when adding another layer.

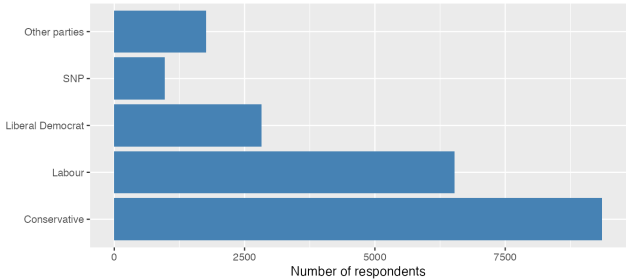
1.4 Visualizing distributions

1.4.1 A categorical variable

How do we visualise the distribution of observations across different values of a categorical variable?

For example, how many respondents voted for each party group?

```
ggplot(plot_data, aes(y = party_group)) +
  geom_bar(fill = "steelblue") +
  labs(x = "Number of respondents", y = NULL)
```

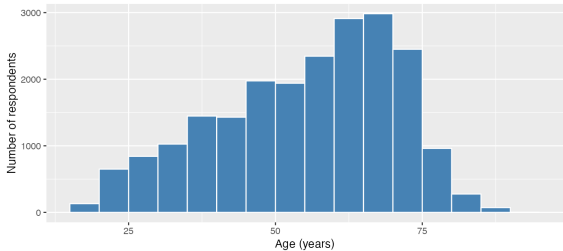


1.4.2 A numeric variable

How are respondents distributed across ages?

```
age_data <- subset(plot_data, !is.na(age))

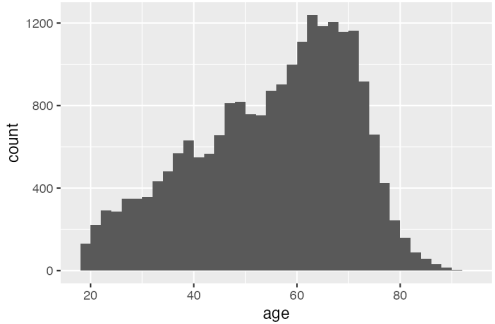
ggplot(age_data, aes(x = age)) +
  geom_histogram(binwidth = 5, boundary = 0,
                 fill = "steelblue", colour = "white") +
  labs(x = "Age (years)", y = "Number of respondents")
```



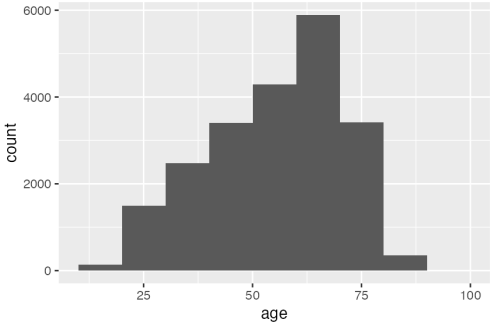
1.4.2 Choosing a bin width

The same ages can look different when we change the interval width.

```
ggplot(age_data, aes(x = age)) +
  geom_histogram(binwidth = 2,
    boundary = 0)
```



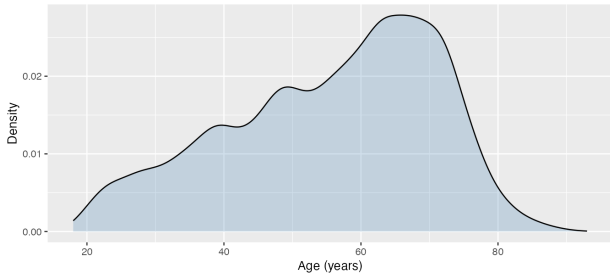

```
ggplot(age_data, aes(x = age)) +
  geom_histogram(binwidth = 10,
    boundary = 0)
```



1.4.2 A density curve

A density curve gives a smoothed view of the age distribution.

```
ggplot(age_data, aes(x = age)) +
  geom_density(fill = "steelblue", alpha = 0.25) +
  labs(x = "Age (years)", y = "Density")
```

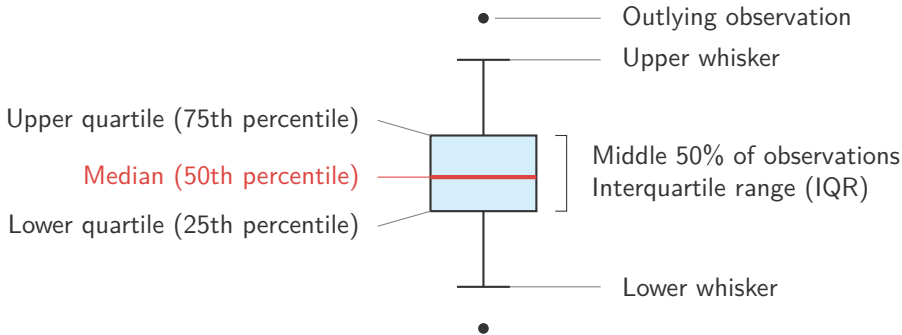


The total area under the curve is 1. Height is density.

1.5 Visualizing relationships

1.5.1 How to read a boxplot

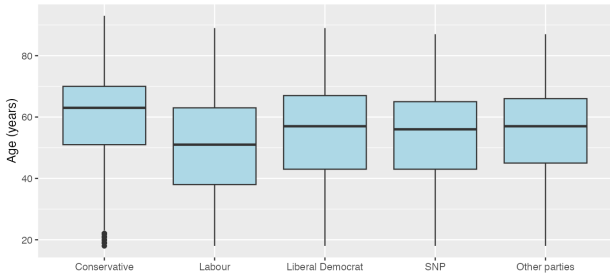
A boxplot summarises a distribution.



1.5.1 A numerical and a categorical variable

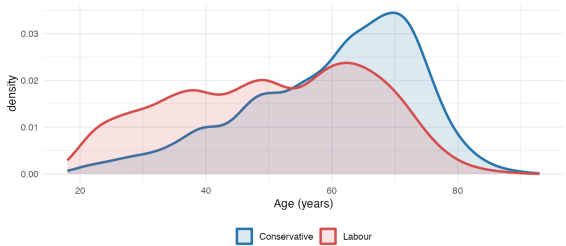
How does the age distribution differ between parties' voters?

```
ggplot(age_data, aes(x = party_group, y = age)) +
  geom_boxplot(fill = "lightblue") +
  labs(x = NULL, y = "Age (years)")
```



1.5.1 Comparing density curves

```
two_party <- subset(age_data,
  party_group %in% c("Labour", "Conservative"))
ggplot(two_party, aes(age, col = party_group, fill = party_group)) +
  geom_density(alpha = 0.16, linewidth = 1.1) +
  scale_colour_manual(values = party_colours) +
  scale_fill_manual(values = party_colours) +
  labs(x = "Age (years)", colour = NULL, fill = NULL) +
  theme_minimal() + theme(legend.position = "bottom")
```



1.5.1 Exercises: density curves

1. In the density curves by party, try `alpha = 0.1`, `alpha = 0.5` and `alpha = 1`. How does the transparency change?

2. What is the difference between `colour` (or `col`) and `fill`? In the density plot, which part of the plot does each control?

1.5.2 Two categorical variables: exercise

Our party-choice-by-income plot already compares two categorical variables.

1. Adapt that plot to show party choice by `region`, using `plot_data`. Update the axis labels to match.

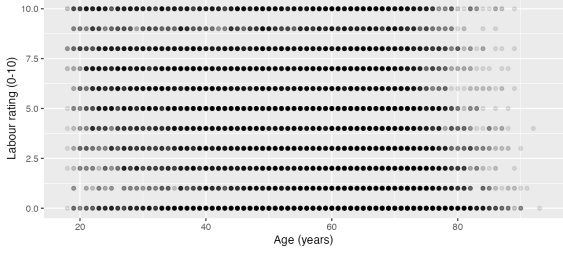
2. What does each percentage represent: the share of a region's respondents who voted for a party, or the share of a party's voters who live in a region? Explain which group totals 100%.

1.5.3 Two numerical variables

How does the Labour rating vary with age?

```
rating_data <- subset(plot_data,
  !is.na(age) & !is.na(like_labour))

ggplot(rating_data, aes(x = age, y = like_labour)) +
  geom_point(alpha = 0.1) +
  labs(x = "Age (years)", y = "Labour rating (0-10)")
```

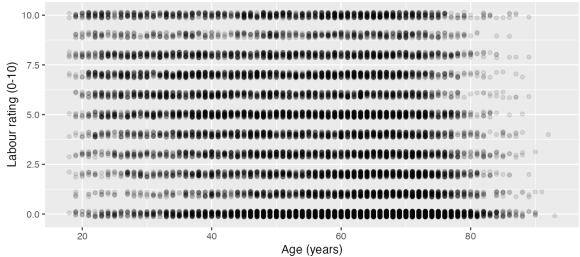


Ratings have only 11 possible values, so points overlap. `alpha` sets transparency: 0 is invisible, 1 is opaque.

1.5.3 Revealing overlapping points

Jitter adds random noise to points so that points don't overlap as much.

```
set.seed(107)
ggplot(rating_data, aes(x = age, y = like_labour)) +
  geom_jitter(width = 0, height = 0.15, alpha = 0.1) +
  labs(x = "Age (years)", y = "Labour rating (0-10)")
```

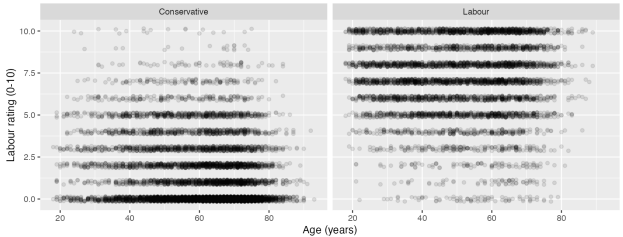


`set.seed()` makes the analysis reproducible.

1.5.4 Three or more variables

```
party_ratings <- subset(rating_data,
  party_group %in% c("Labour", "Conservative"))

set.seed(107)
ggplot(party_ratings, aes(x = age, y = like_labour)) +
  geom_jitter(width = 0, height = 0.15, alpha = 0.1) +
  facet_wrap(~ party_group) +
  labs(x = "Age (years)", y = "Labour rating (0-10)")
```



1.5.5 Exercises

1. Compare age boxplots across party groups. Describe a difference in medians and a difference in spread. Do the groups overlap?

2. Change the regional chart to `geom_bar()` without `position = "fill"`. How does the question answered by the chart change?

3. Plot `like_conservative` against age, excluding missing values for these two variables. Facet by Labour and Conservative vote.

1.6 Saving your plots

1.6 Saving a named plot

Often we want to save our plots. It is helpful to first save the plot to an object within R.

```
age_plot <- ggplot(age_data, aes(x = age)) +
  geom_histogram(binwidth = 5, boundary = 0,
                 fill = "steelblue", colour = "white") +
  labs(x = "Age (years)", y = "Number of respondents",
       title = "Age distribution in the BES teaching sample")
age_plot
```

Then we can save the object as an image file to the current working directory:

```
ggsave("bes-age.png", plot = age_plot,
       width = 8, height = 4, units = "in", dpi = 300)
getwd()
```

`getwd()` tells you where to find the file.

1.6.1 Exercises

1. Save `income_plot` as `bes-income.png`, using `ggsave()`. Use `getwd()` to find the file and open it.

2. Save another version with a different width and height, using a new filename. How do the dimensions affect the appearance of the plot, labels and legend?

3. Save the same plot as a PDF by changing the filename extension to `.pdf`. Open the file to check the result.