

# ggplot and the Grammar of Graphics

Melbourne Statistical Consulting Platform

University of Melbourne

April 2024

"a statistical graphic is a mapping from data  
to aesthetic attributes (colour, size, shape)  
of geometric objects (points ●, lines \, bars ■)"

— Hadley Wickham, *ggplot2: Elegant Graphics for Data Analysis* (2nd ed., 2016)

Image by Will Chase ([Twitter](#)).

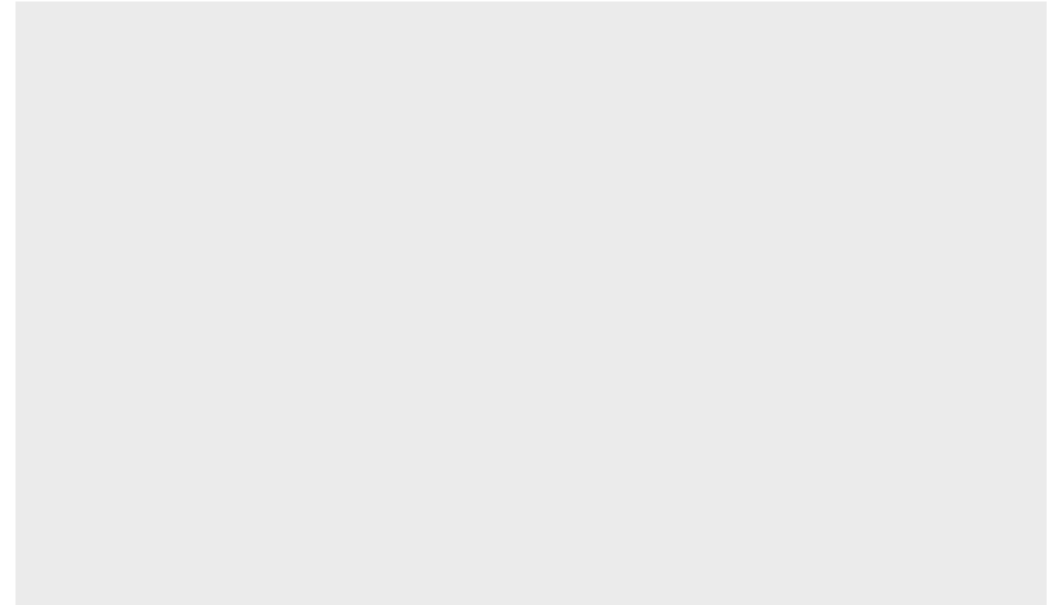
Glamour of Graphics presentation



# Structure of a plot in ggplot

- Data

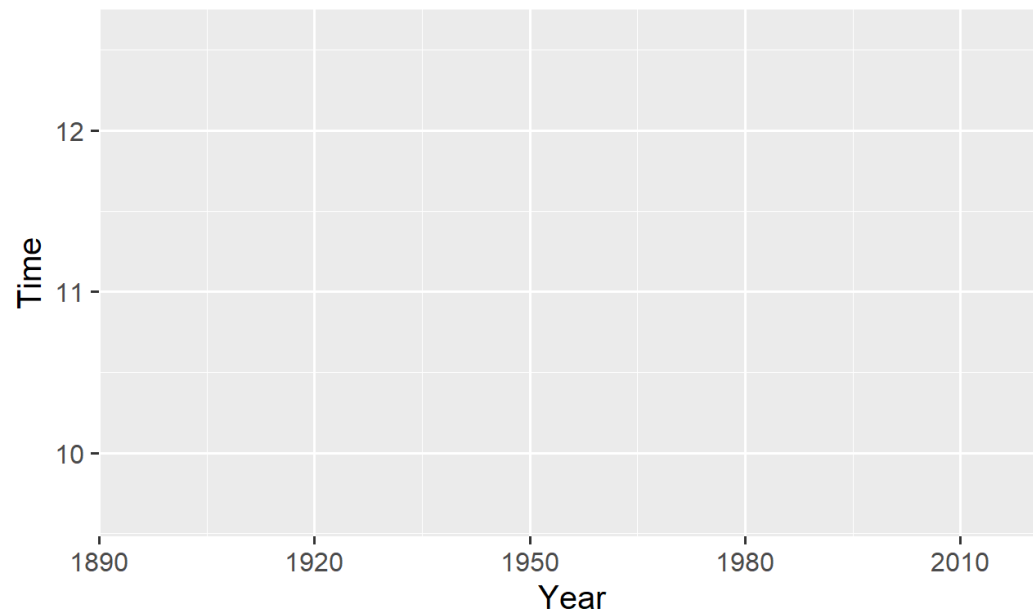
```
ggplot(olympic_100m_data)
```



# Structure of a plot in ggplot

- Data
- Mapping from columns to aesthetic attributes

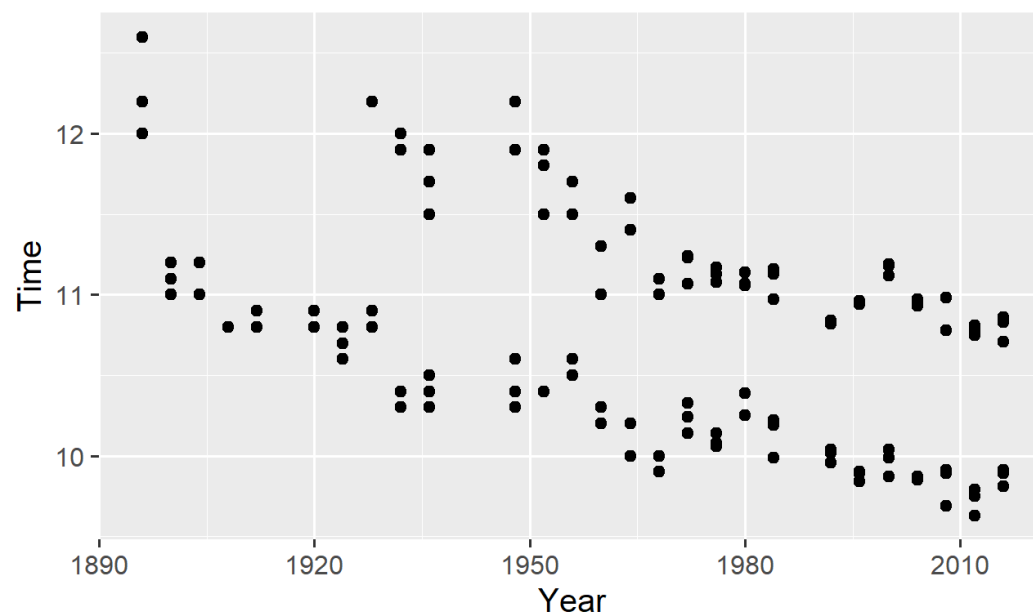
```
ggplot(olympic_100m_data,  
       aes(x = Year, y = Time))
```



# Structure of a plot in ggplot

- Data
- Mapping from columns to aesthetic attributes
- Layers (geometric objects)

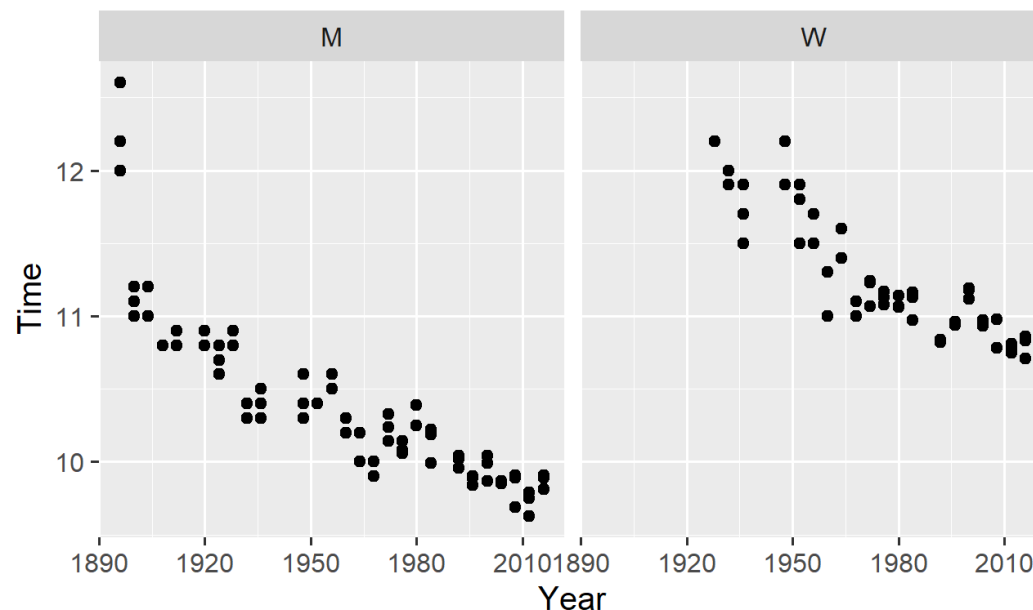
```
ggplot(olympic_100m_data,  
      aes(x = Year, y = Time)) +  
  geom_point()
```



# Structure of a plot in ggplot

- Data
- Mapping from columns to aesthetic attributes
- Layers (geometric objects)
- Facets

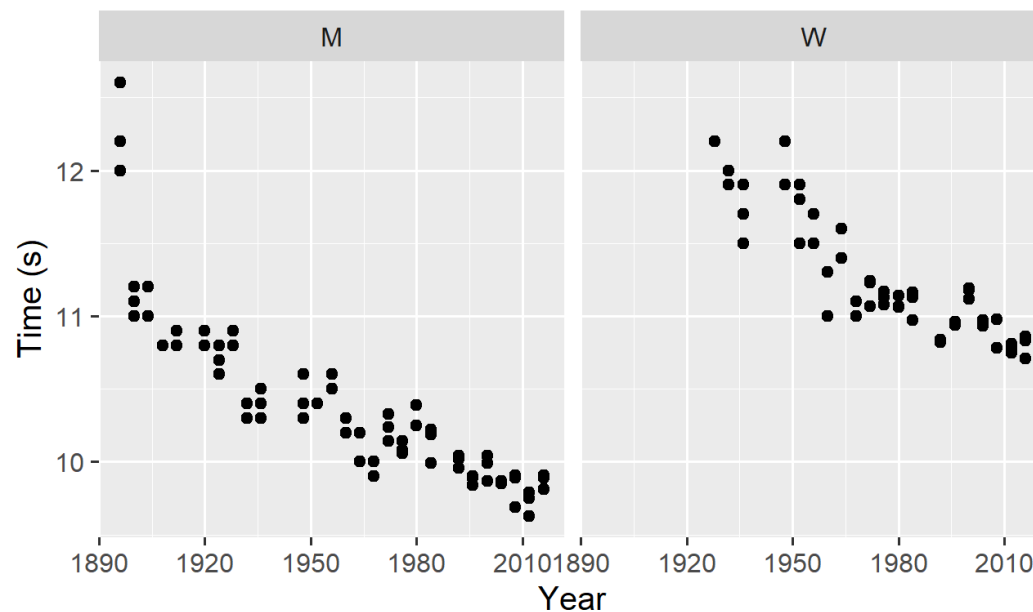
```
ggplot(olympic_100m_data,  
      aes(x = Year, y = Time)) +  
  geom_point() +  
  facet_grid(cols = vars(Gender))
```



# Structure of a plot in ggplot

- Data
- Mapping from columns to aesthetic attributes
- Layers (geometric objects)
- Facets
- Scales and labels

```
ggplot(olympic_100m_data,  
      aes(x = Year, y = Time)) +  
  geom_point() +  
  facet_grid(cols = vars(Gender)) +  
  labs(y = "Time (s)")
```

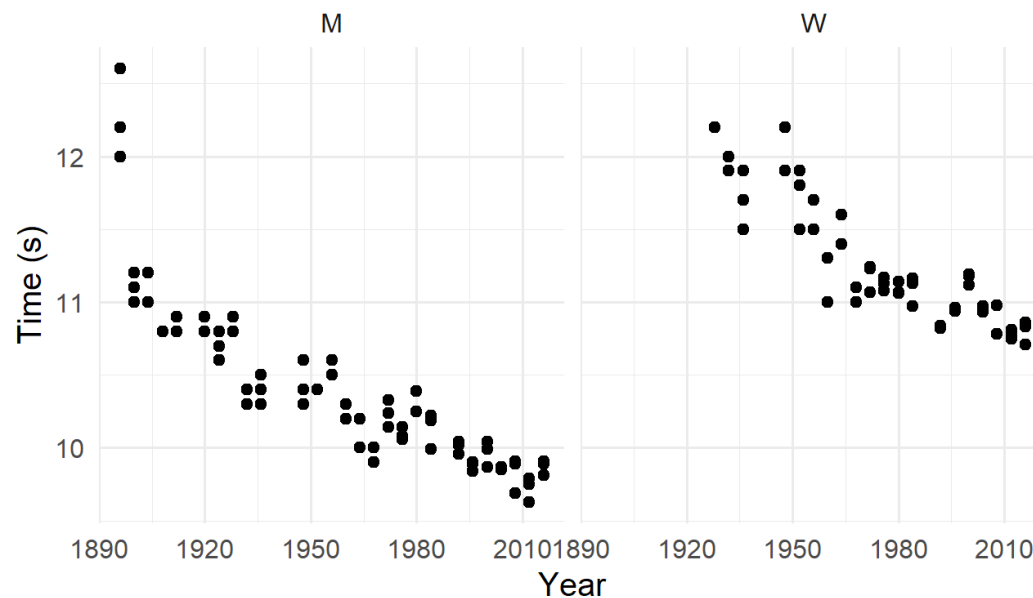




# Structure of a plot in ggplot

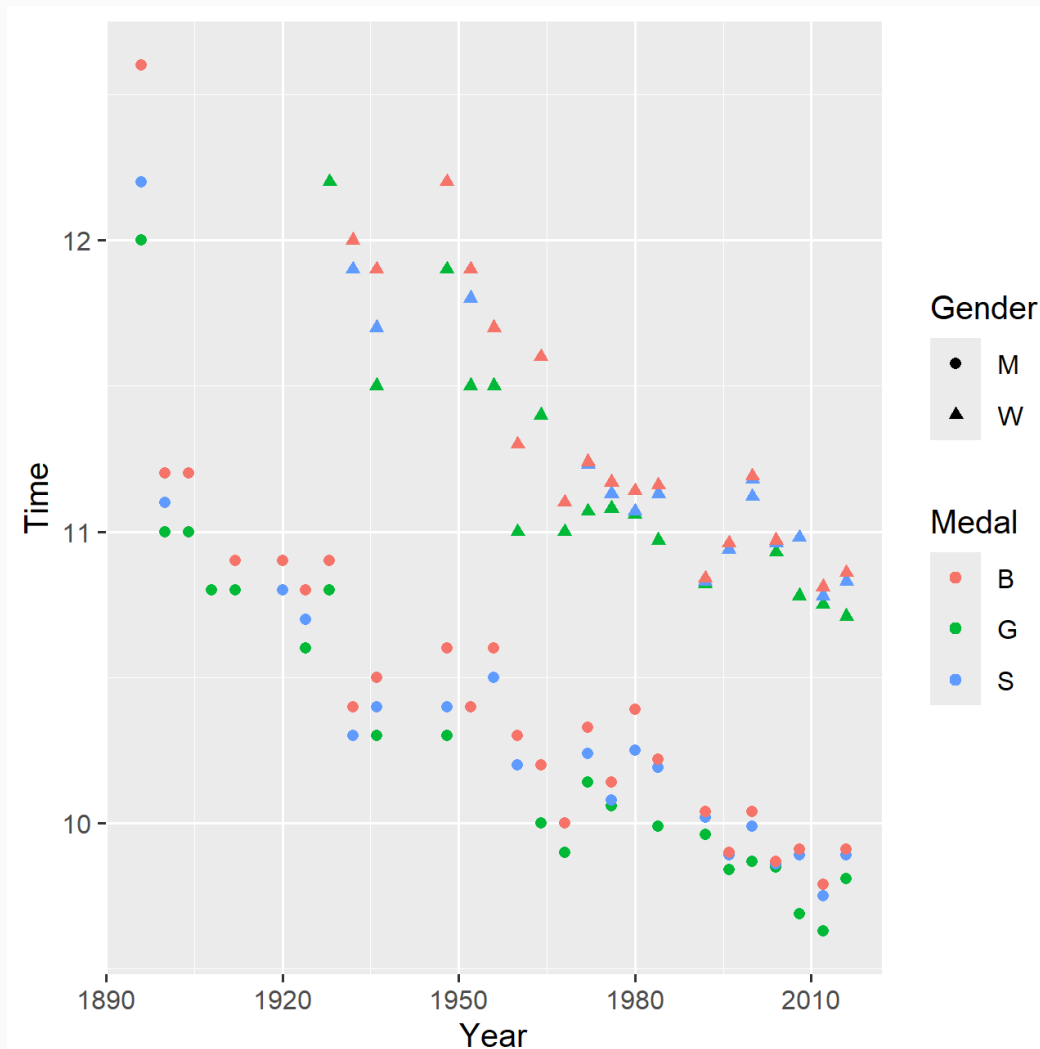
- Data
- Mapping from columns to aesthetic attributes
- Layers (geometric objects)
- Facets
- Scales and labels
- Themes

```
ggplot(olympic_100m_data,  
      aes(x = Year, y = Time)) +  
  geom_point() +  
  facet_grid(cols = vars(Gender)) +  
  labs(y = "Time (s)") +  
  theme_minimal()
```



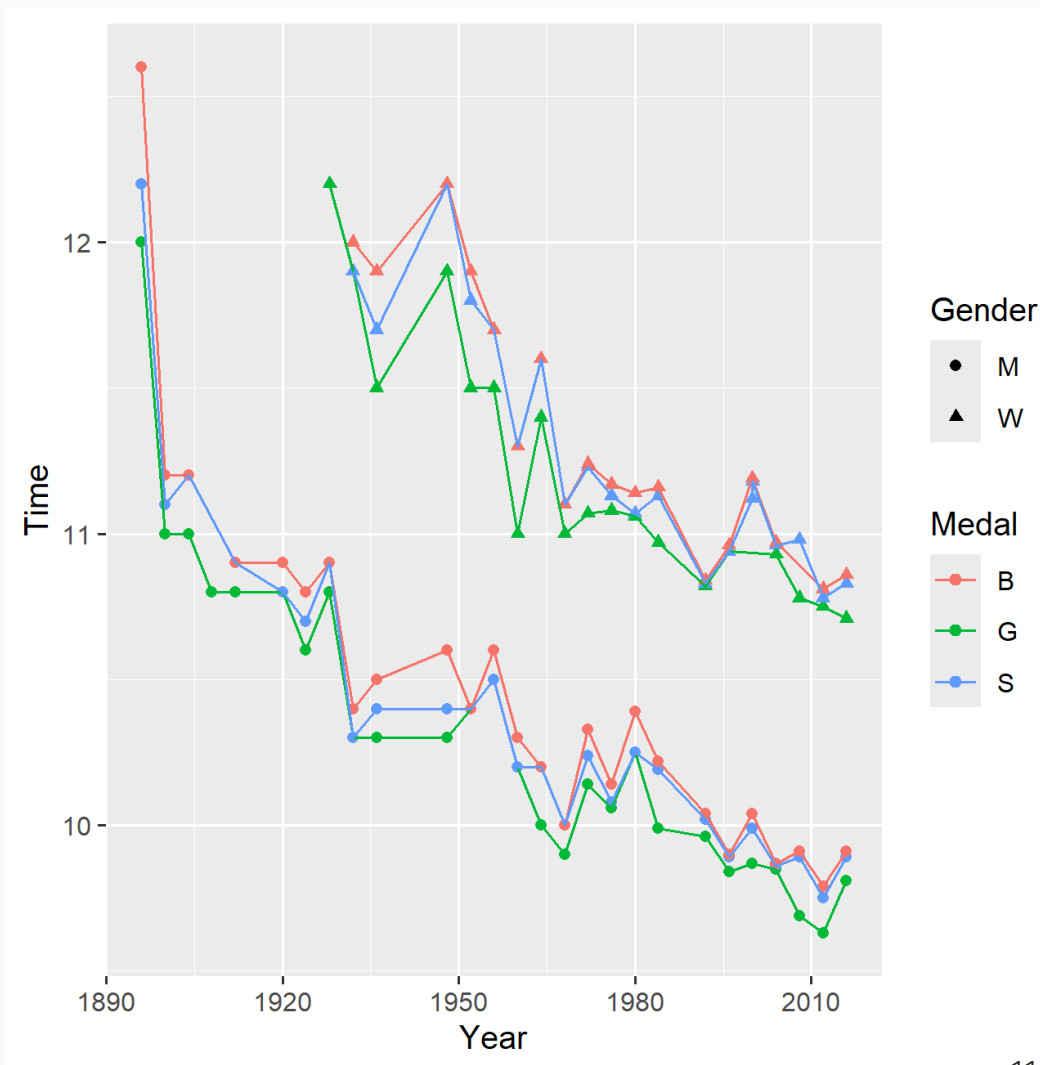
# Example: scatter plot

```
ggplot(olympic_100m_data,  
      aes(x = Year,  
          y = Time,  
          colour = Medal,  
          shape = Gender)) +  
  geom_point()
```



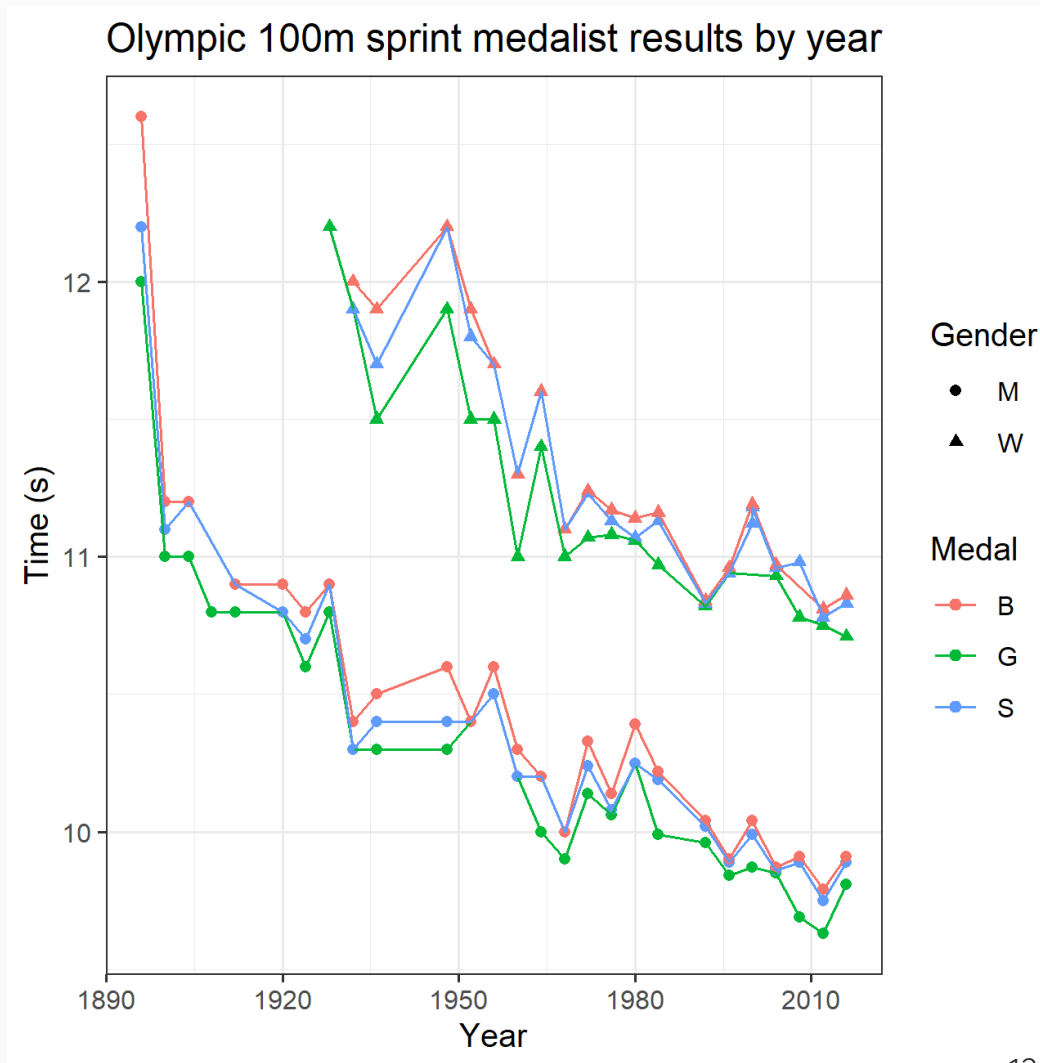
# Example: scatter plot with time series line

```
ggplot(olympic_100m_data,  
  aes(x = Year,  
      y = Time,  
      colour = Medal,  
      shape = Gender)) +  
  geom_point() +  
  geom_line()
```



# Example: scatter plot with time series line

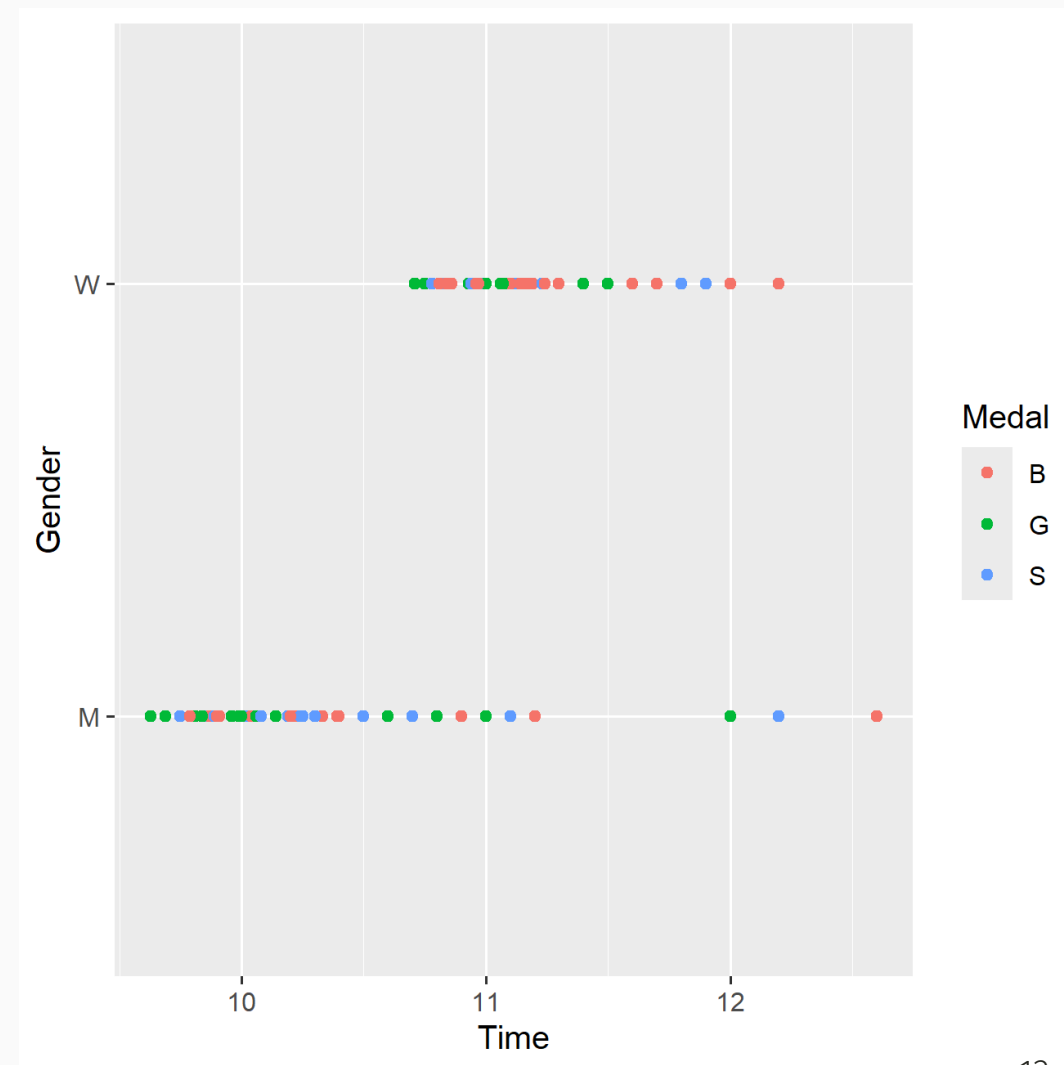
```
ggplot(olympic_100m_data,  
  aes(x = Year,  
      y = Time,  
      colour = Medal,  
      shape = Gender)) +  
  geom_point() +  
  geom_line() +  
  labs(y = "Time (s)",  
       title = "Olympic 100m sprint medalist results by year",  
       theme_bw())
```



# Example: a different scatter plot

```
ggplot(olympic_100m_data,  
      aes(x = Time,  
          y = Gender,  
          colour = Medal)) +  
  geom_point()
```

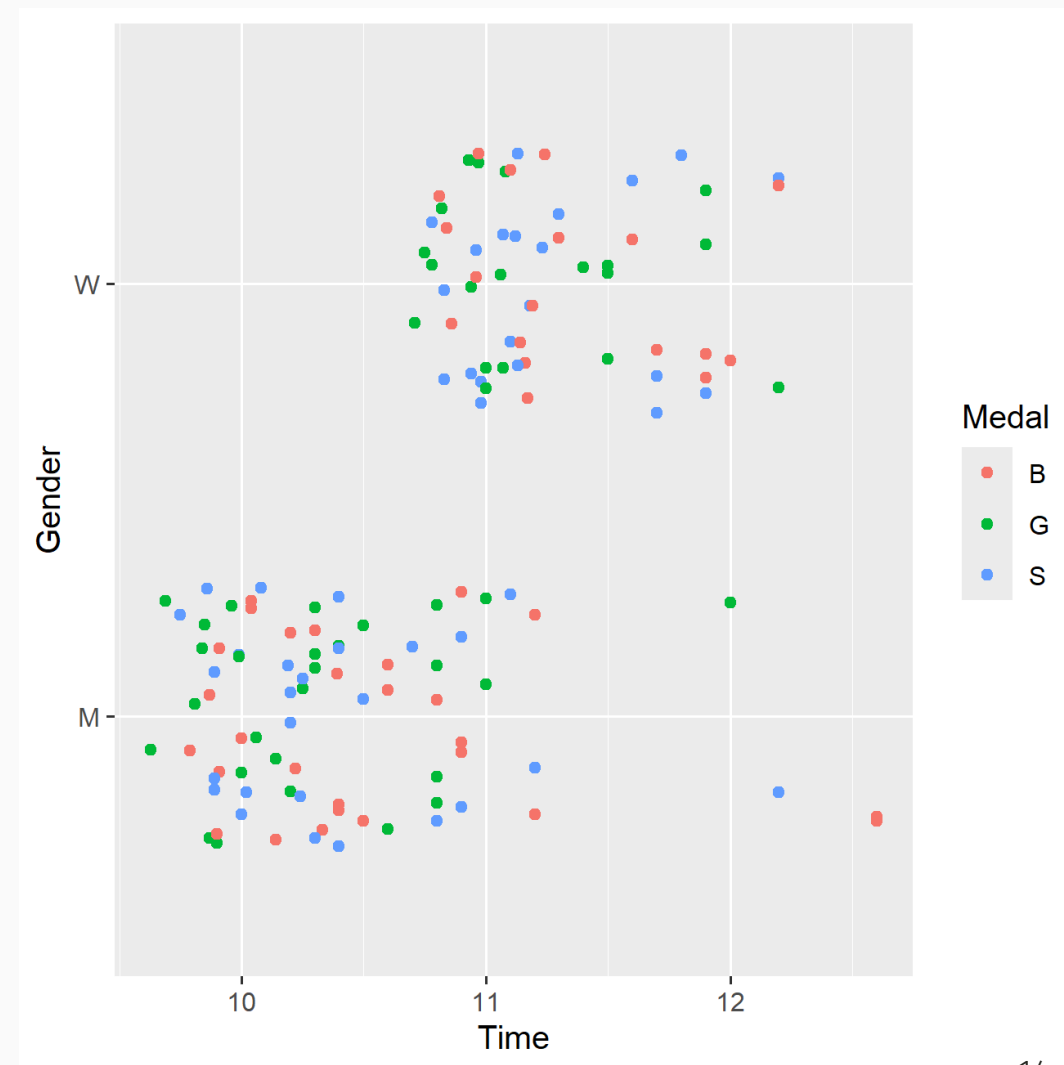
In this case we are plotting a continuous variable and a categorical variable. There are lots of points with the same y coordinates and it's hard to get an idea of what's going on.



# Example: scatter plot with jitter

```
ggplot(olympic_100m_data,  
  aes(x = Time,  
      y = Gender,  
      colour = Medal)) +  
  geom_jitter(width = 0, height = 0.3)
```

Adding jitter, or a small amount of random movement, makes the distribution of the data more apparent.

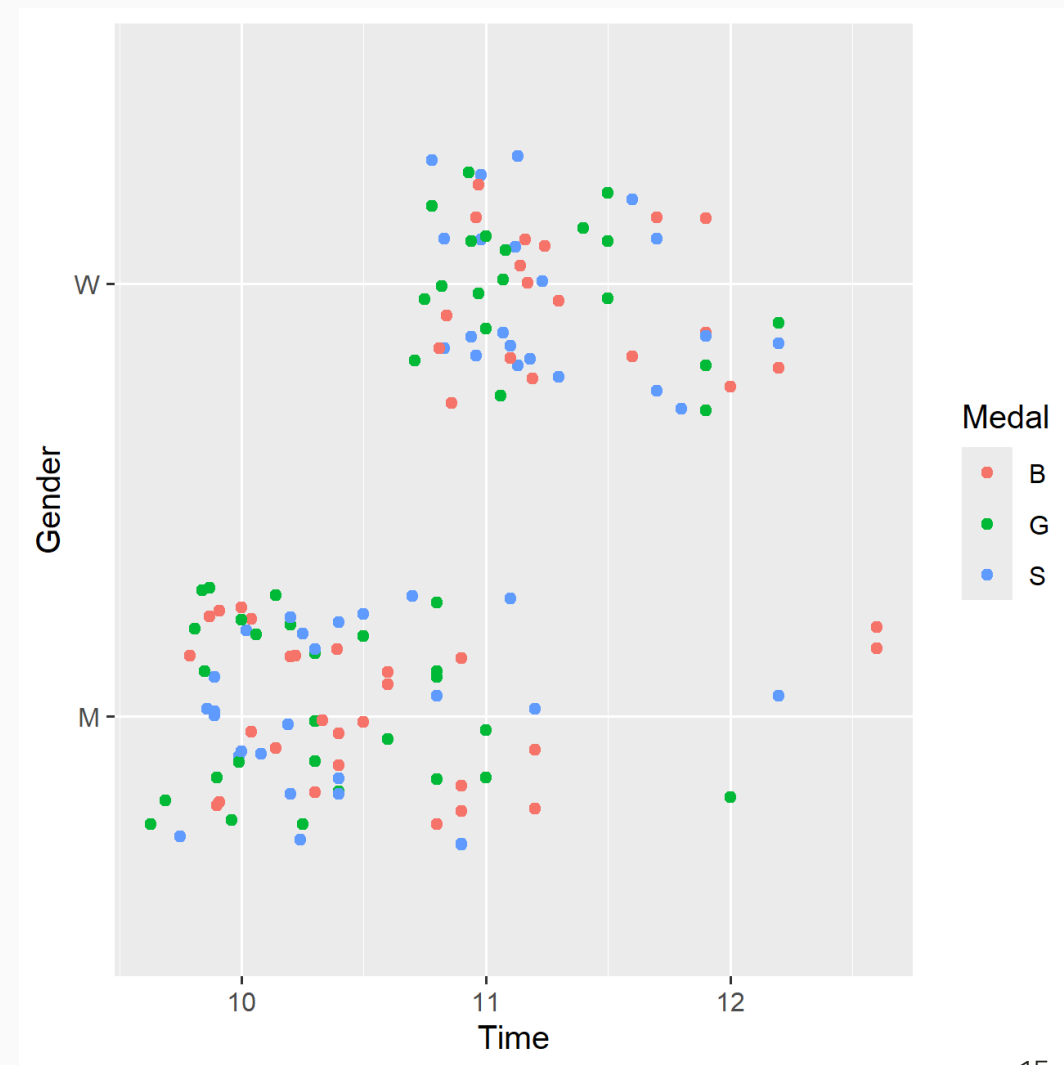


# Example: scatter plot with jitter

```
ggplot(olympic_100m_data,  
  aes(x = Time,  
      y = Gender,  
      colour = Medal)) +  
  geom_jitter(width = 0, height = 0.3)
```

Adding jitter, or a small amount of random movement, makes the distribution of the data more apparent.

Running the plot again produces slightly different output.



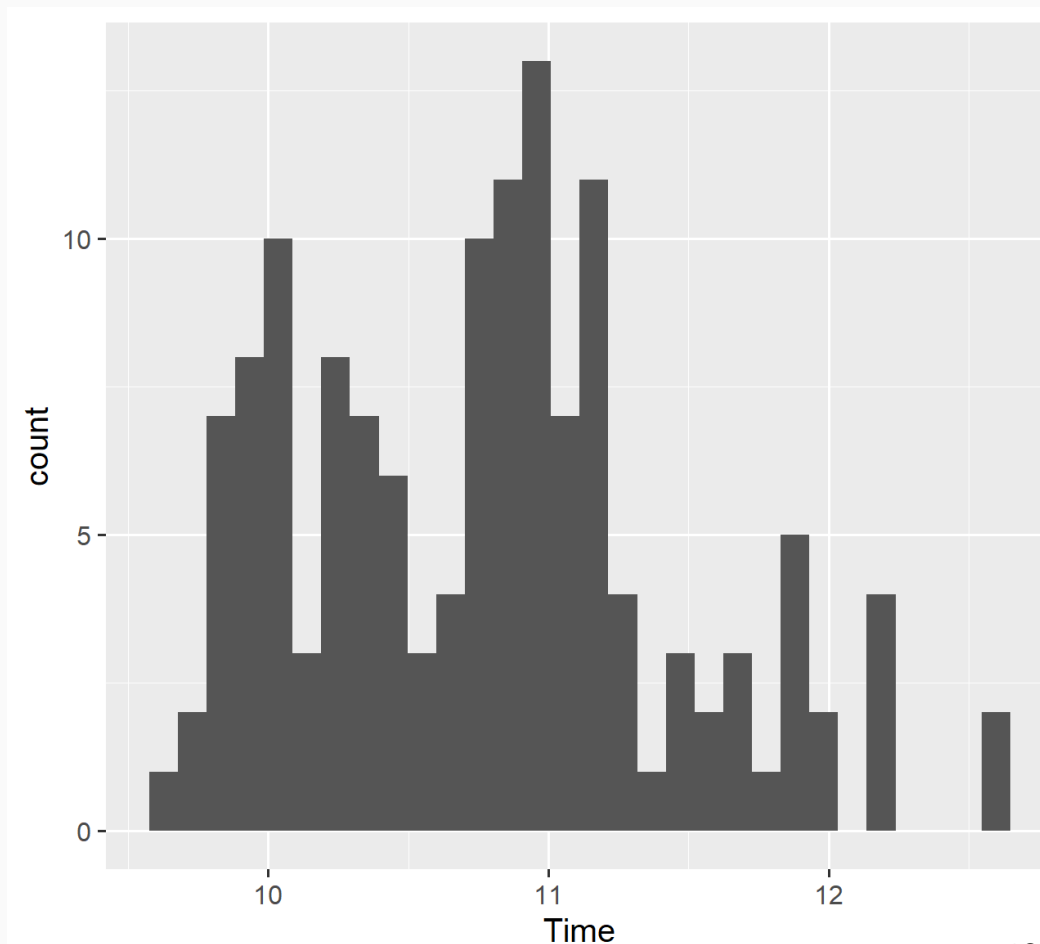
# Example: histogram

```
ggplot(olympic_100m_data,  
      aes(x = Time)) +  
  geom_histogram()
```

Unlike before, we're not plotting the raw data. We're plotting the distribution of the data: how many observations fall within each 'bin'.

`ggplot` has a fixed default for the number of bins (30), and warns us if we don't change it.

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

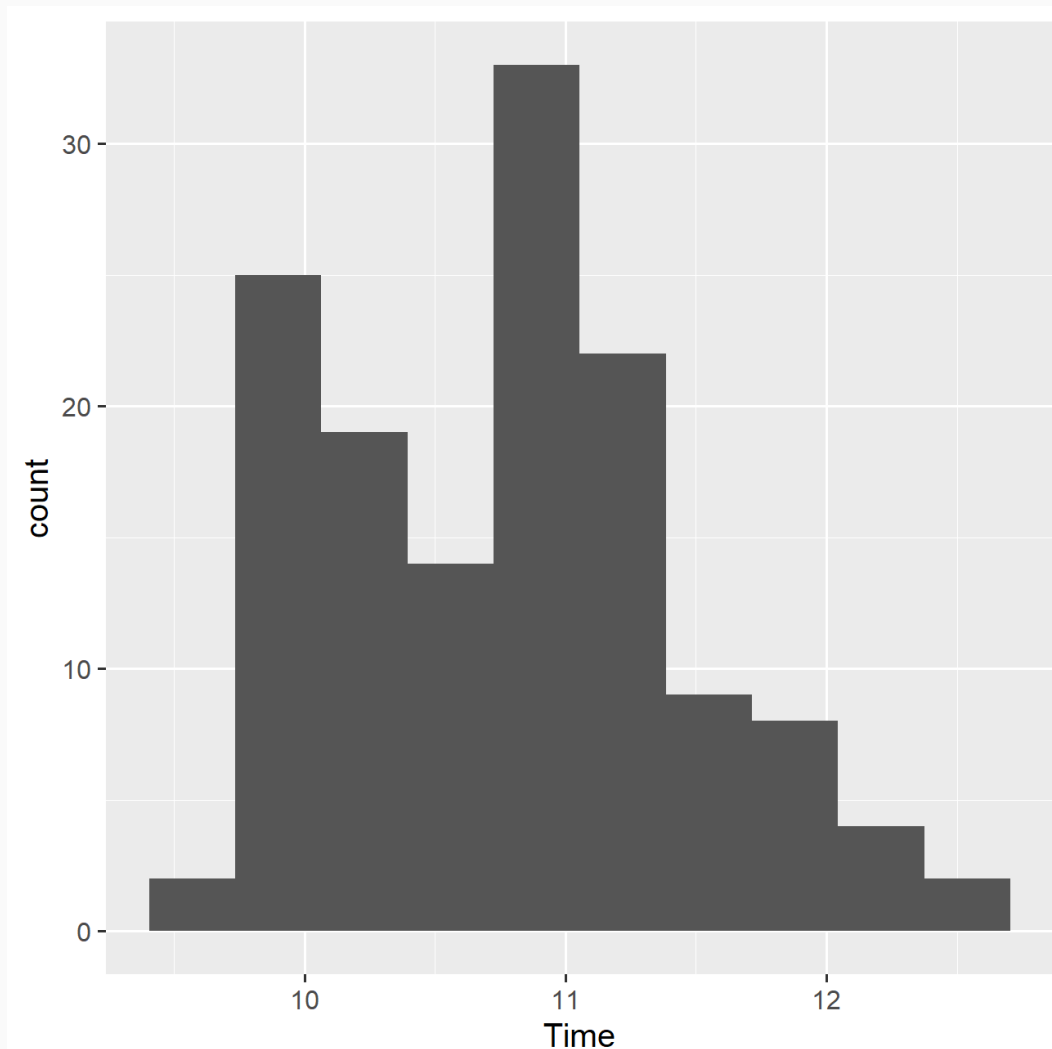




# Example: histogram

```
ggplot(olympic_100m_data,  
      aes(x = Time)) +  
      geom_histogram(bins = 10)
```

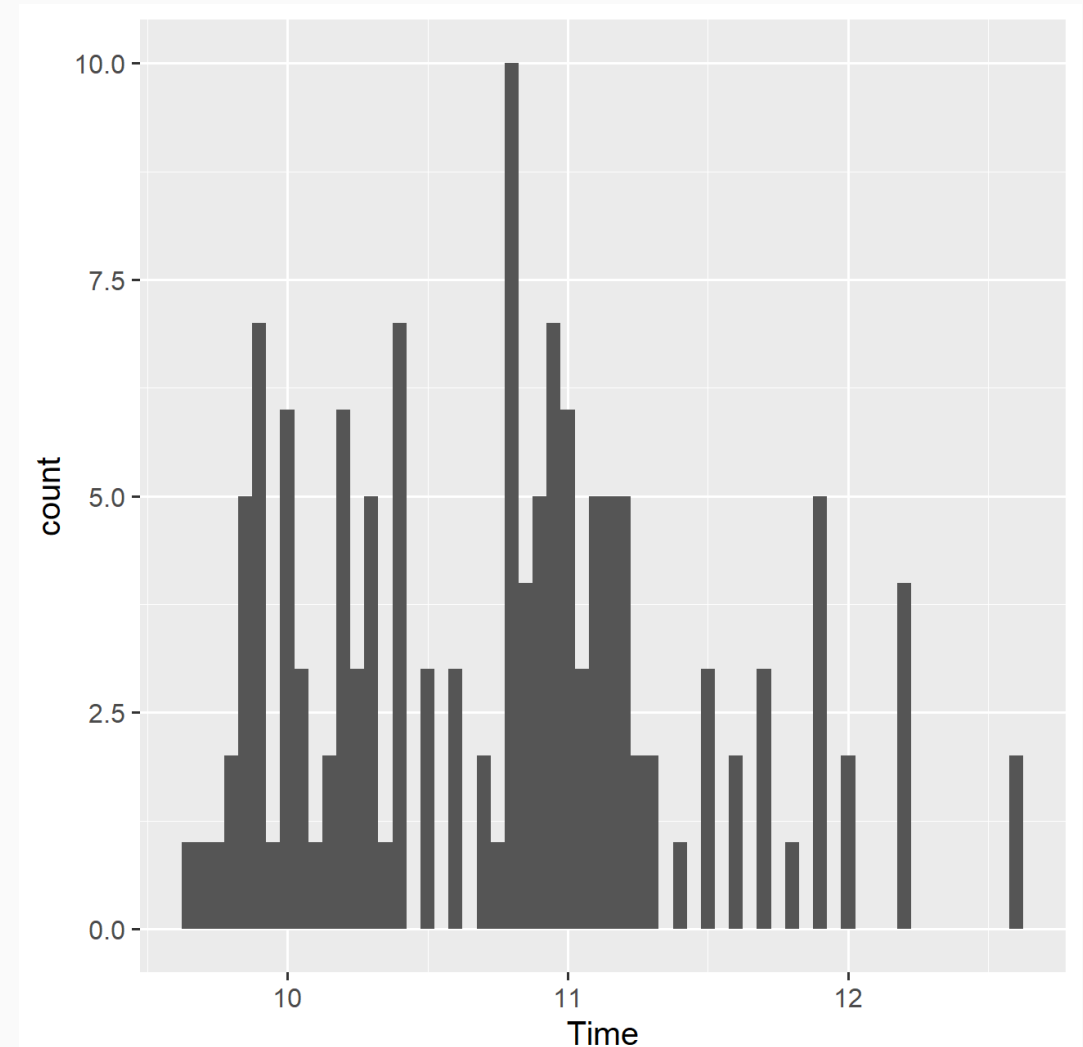
We can choose the number of bins. In this case, 10 bins was probably too few.



# Example: histogram

```
ggplot(olympic_100m_data,  
  aes(x = Time)) +  
  geom_histogram(binwidth = 0.05)
```

Or we can choose the width of the bins. In this case, a width of 0.05 seconds gives us too many bins.

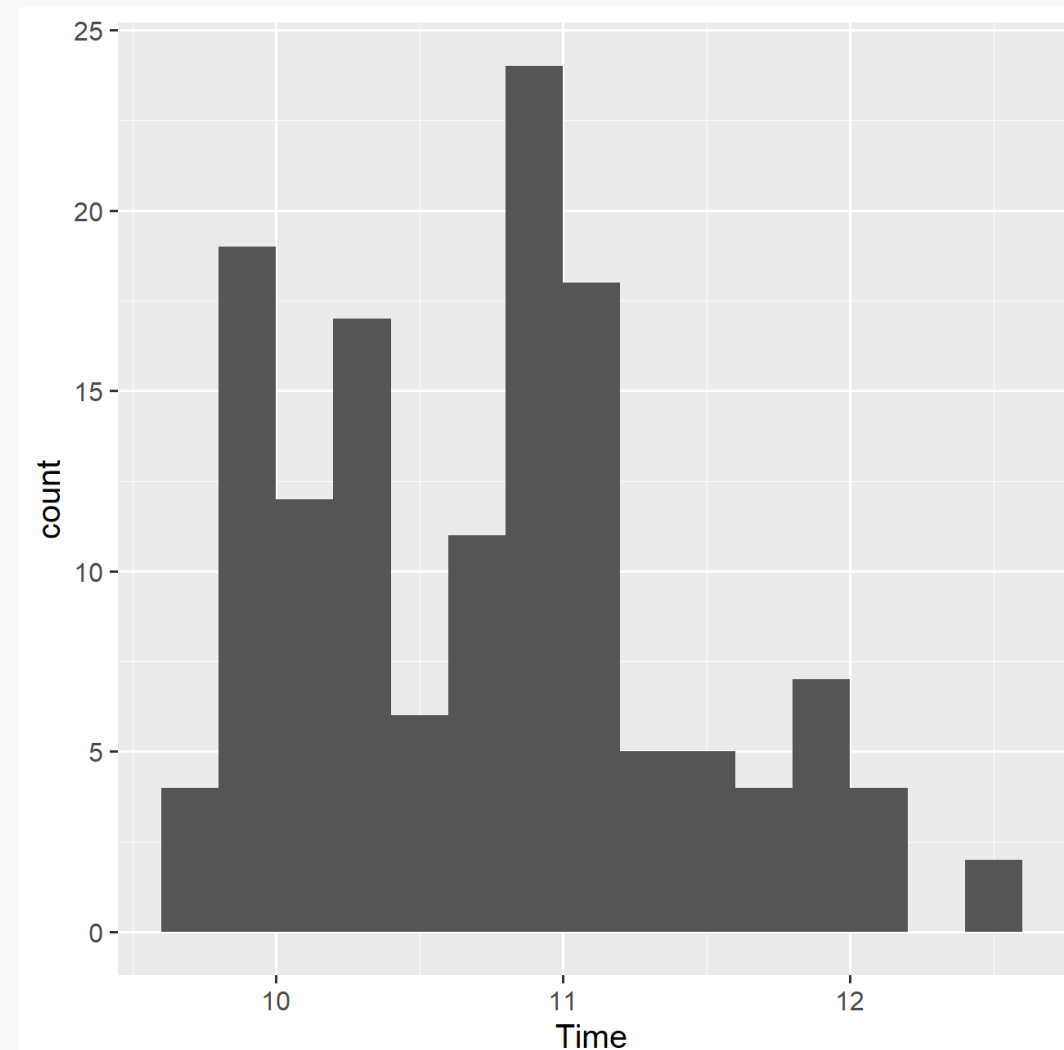


# Example: histogram

```
ggplot(olympic_100m_data,  
      aes(x = Time)) +  
  geom_histogram(binwidth = 0.2, boundary = 10.0)
```

This is a decent compromise. You can also specify a boundary to make sure the bins are aligned to meaningful values, e.g. whole numbers.

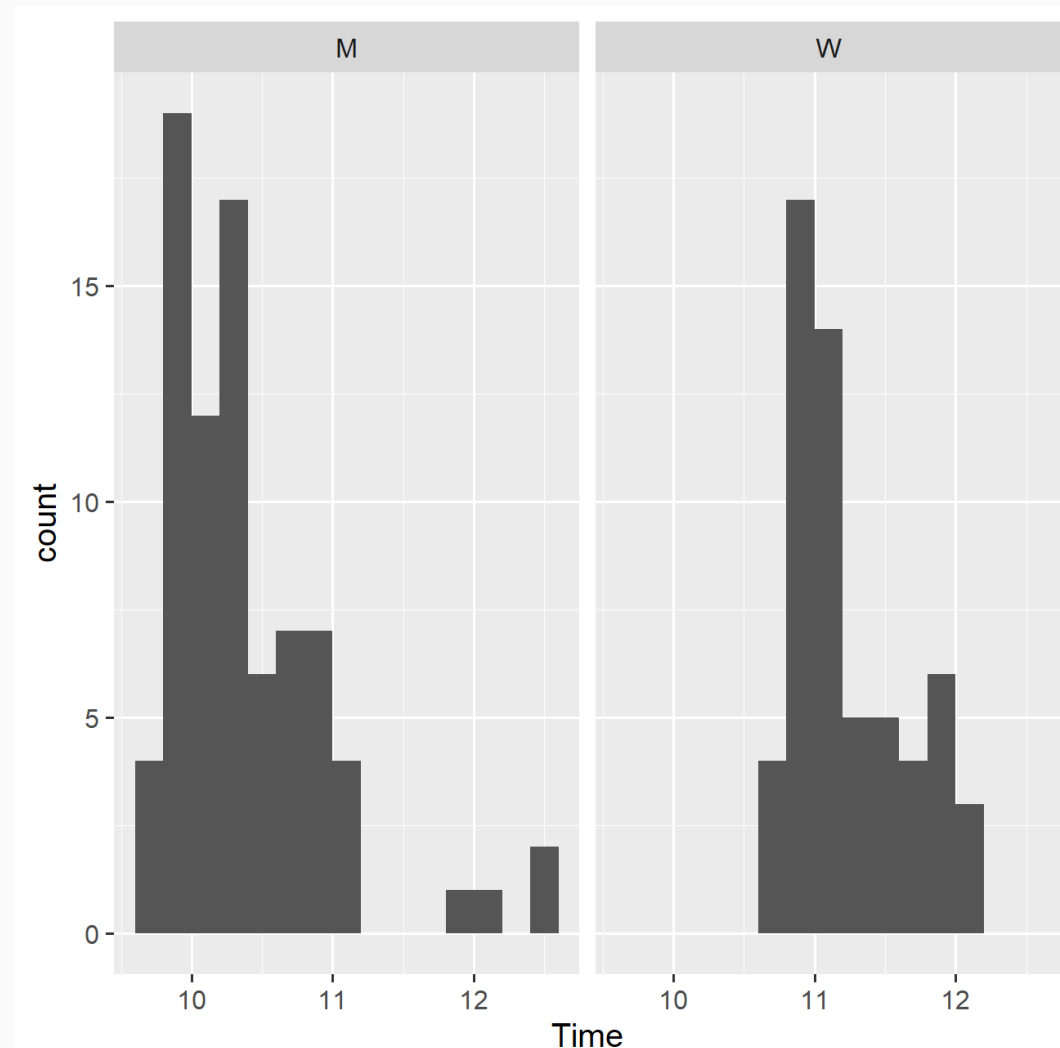
The histogram shows two distinct peaks, suggesting a bimodal distribution of the data.



# Example: histogram

```
ggplot(olympic_100m_data,  
      aes(x = Time)) +  
  geom_histogram(binwidth = 0.2, boundary = 10.0) +  
  facet_wrap(vars(Gender))
```

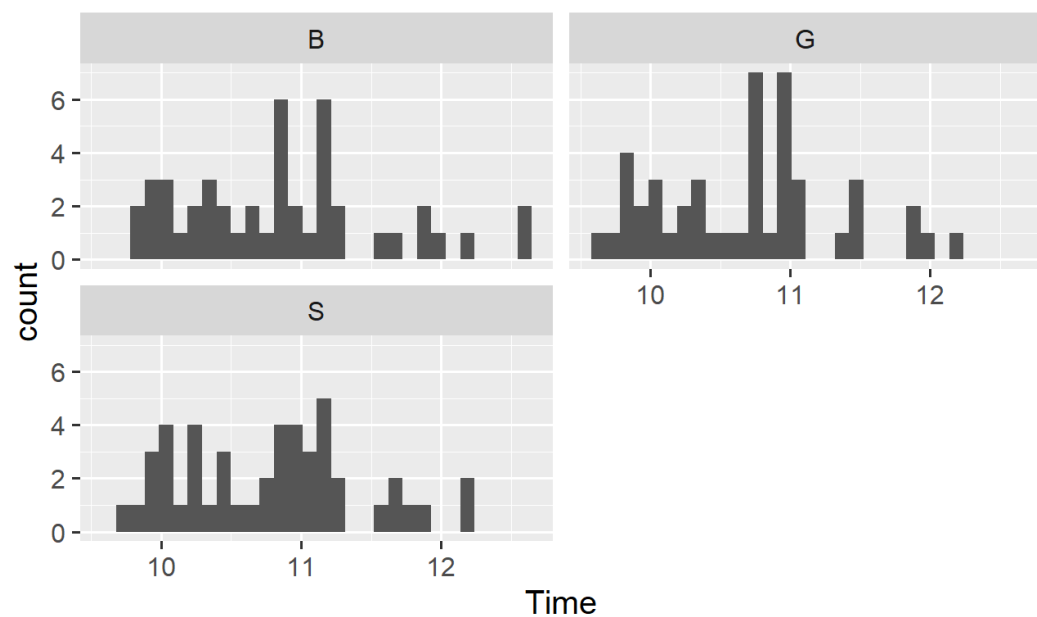
Facetting the data by gender reveals the reason for the bimodal distribution. We can also see that the slowest time and fastest time are by a man.



# Facet (panel) plots: facet\_wrap and facet\_grid

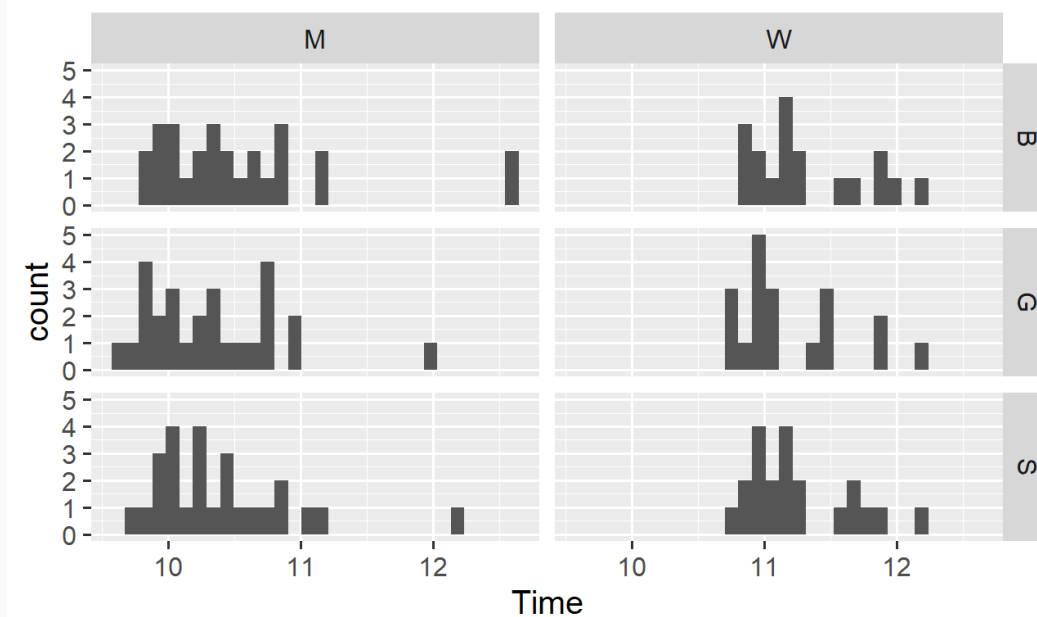
**facet\_wrap**: take one variable, split it into panels across (potentially) both rows and columns

```
ggplot(olympic_100m_data,  
  aes(x = Time)) +  
  geom_histogram() +  
  facet_wrap(vars(Medal), nrow = 2)
```



**facet\_grid**: take one variable for rows and potentially a different variable for columns

```
ggplot(olympic_100m_data,  
  aes(x = Time)) +  
  geom_histogram() +  
  facet_grid(rows = vars(Medal), cols = vars(Gender))
```

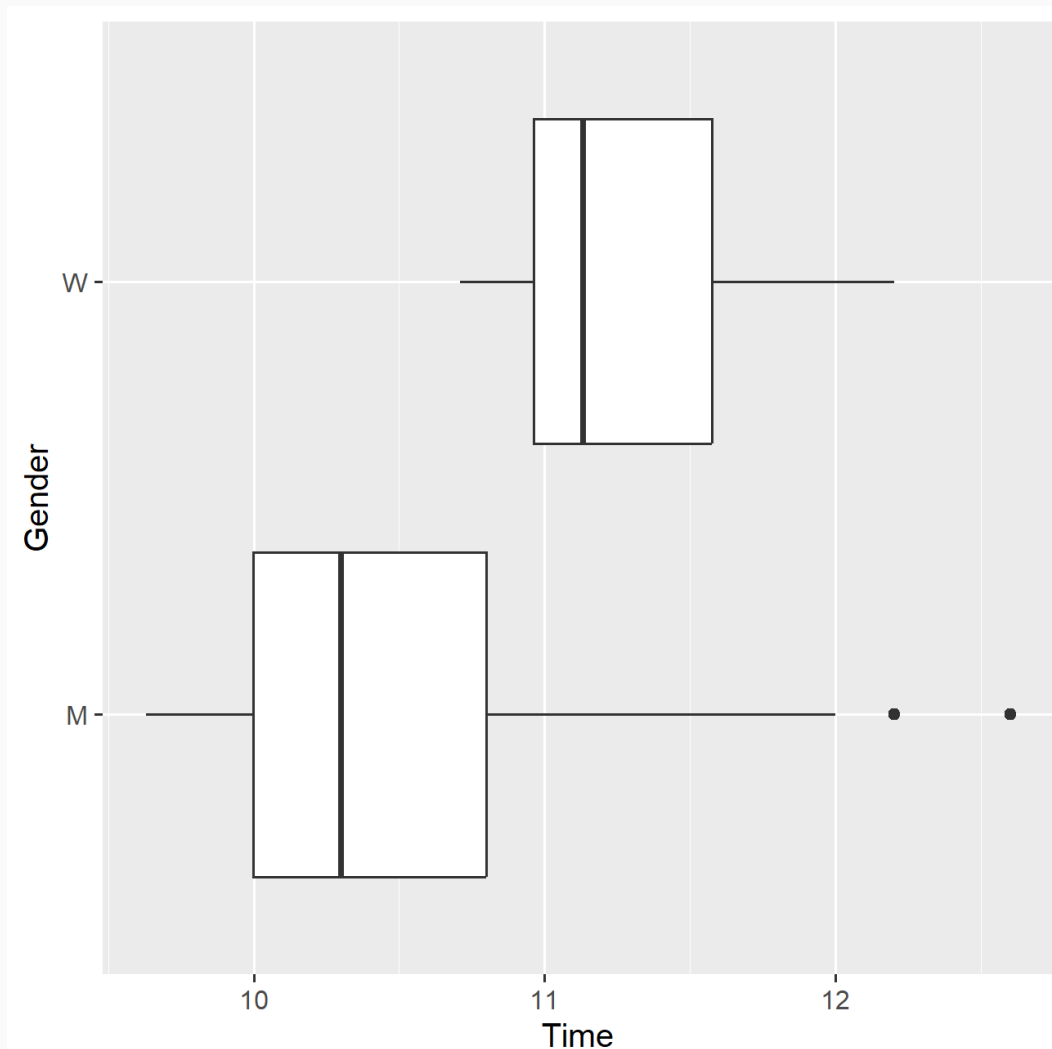


# Example: box plot

```
ggplot(olympic_100m_data,  
      aes(x = Time, y = Gender)) +  
      geom_boxplot()
```

This is another way of displaying a statistical summary of the data.

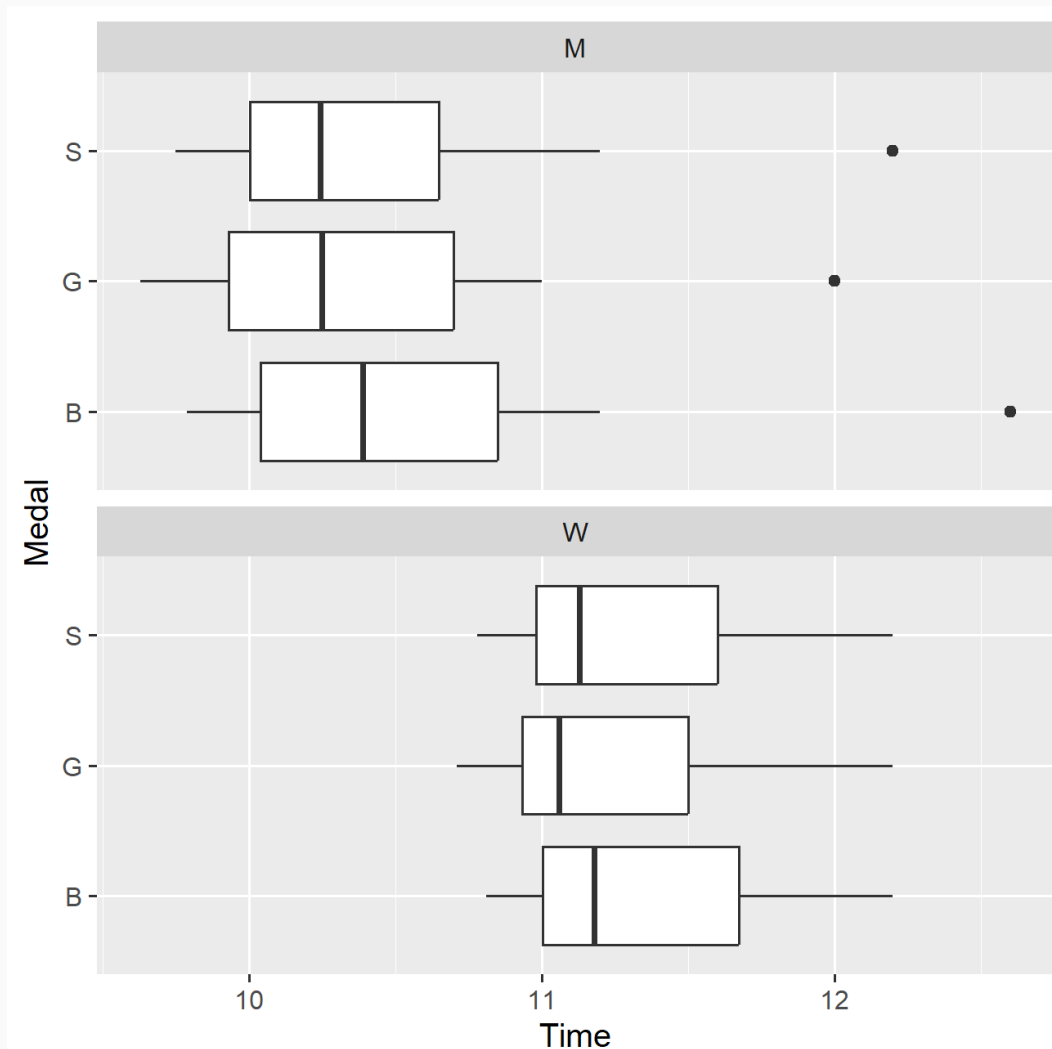
- What does the thick line in the centre represent?
- What do the two edges of the box represent?
- What do the ends of the thin lines ('whiskers') represent?
- What are the two points off to the right?



# Example: box plot

```
ggplot(olympic_100m_data,  
      aes(x = Time, y = Medal)) +  
  geom_boxplot() +  
  facet_wrap(vars(Gender), nrow = 2)
```

What's wrong with this plot?

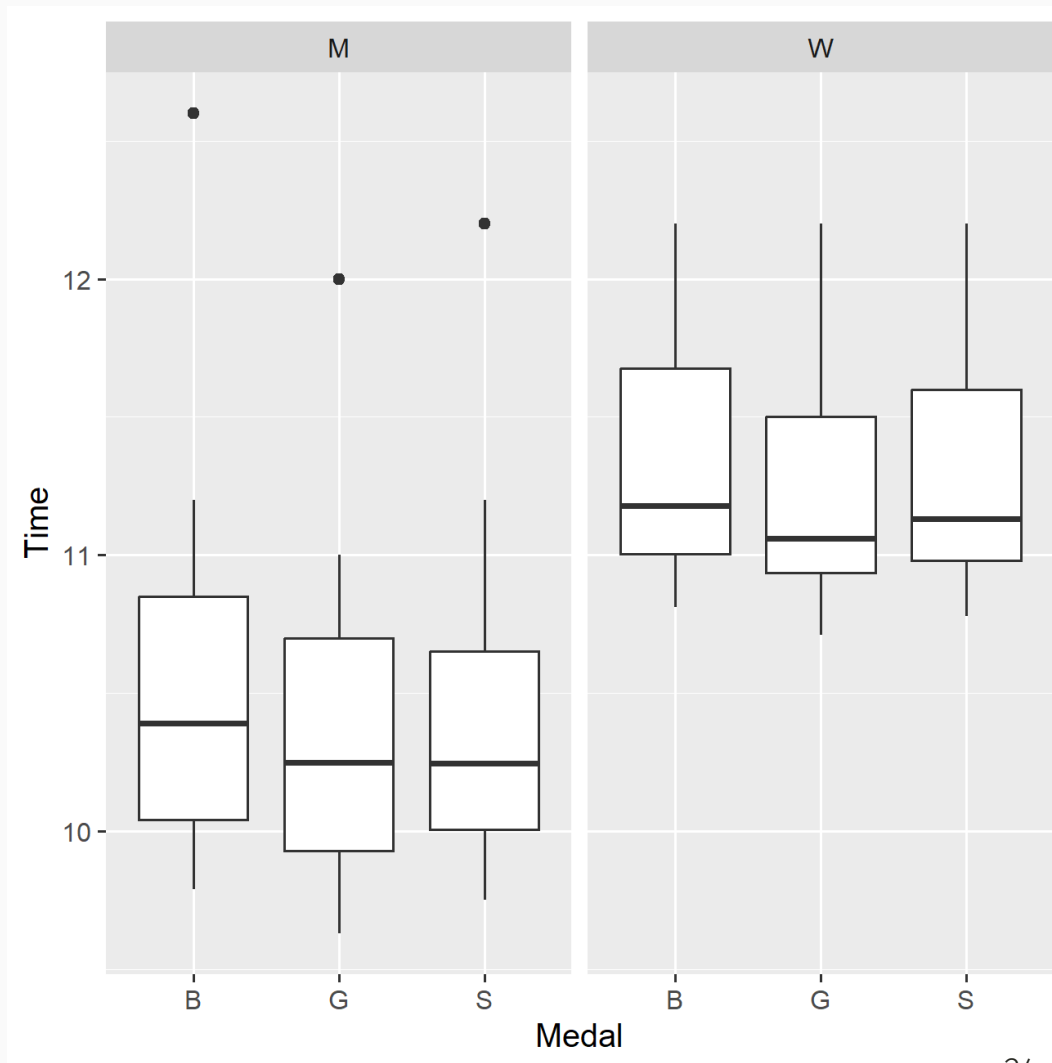


# Example: vertical box plot

```
ggplot(olympic_100m_data,  
      aes(y = Time, x = Medal)) +  
  geom_boxplot() +  
  facet_wrap(vars(Gender), ncol = 2)
```

Swap `x` and `y`; `nrow` and `ncol`.

All `ggplot` geoms work in either orientation.

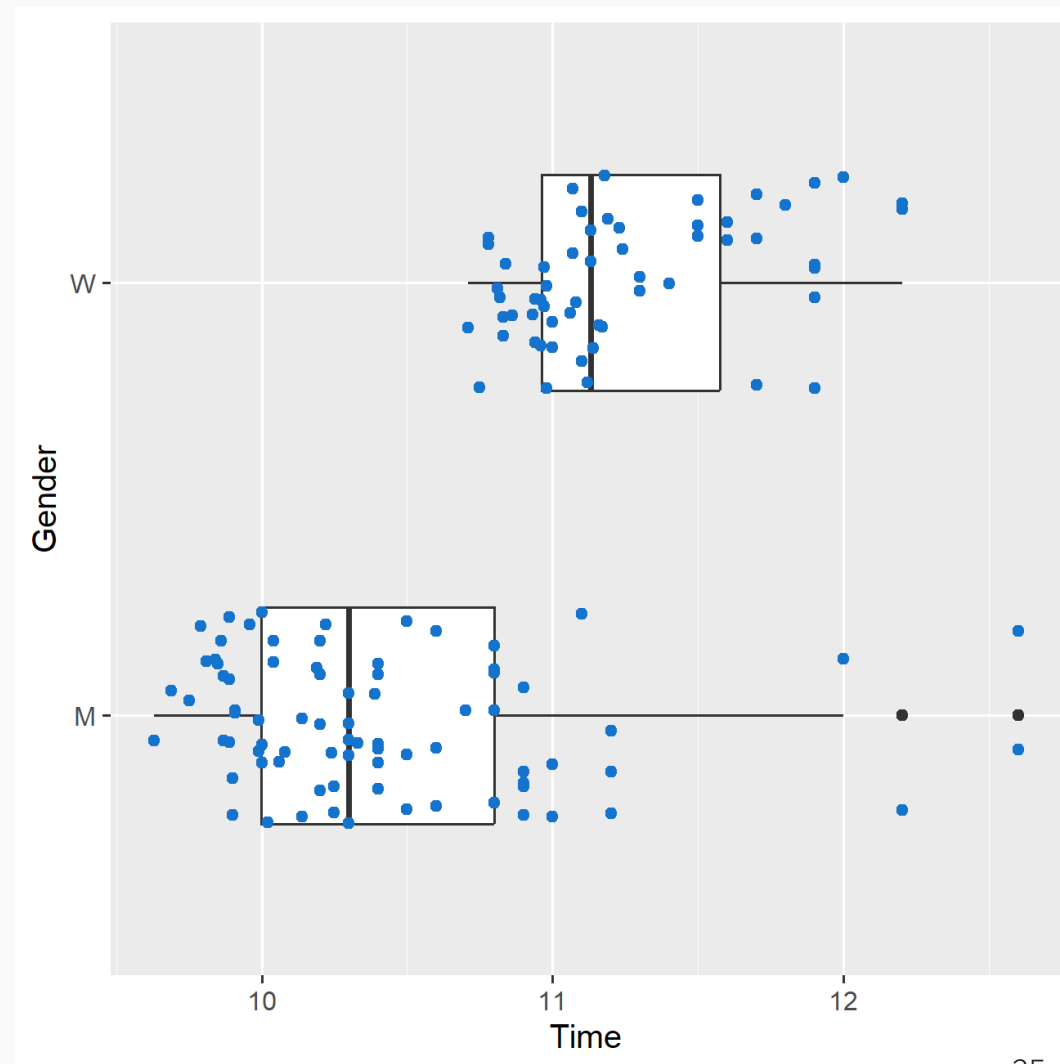




# Example: box plot with raw data display

```
ggplot(olympic_100m_data,  
      aes(x = Time, y = Gender)) +  
  geom_boxplot(width = 0.5) +  
  geom_jitter(width = 0, height = 0.25,  
             colour = "dodgerblue3")
```

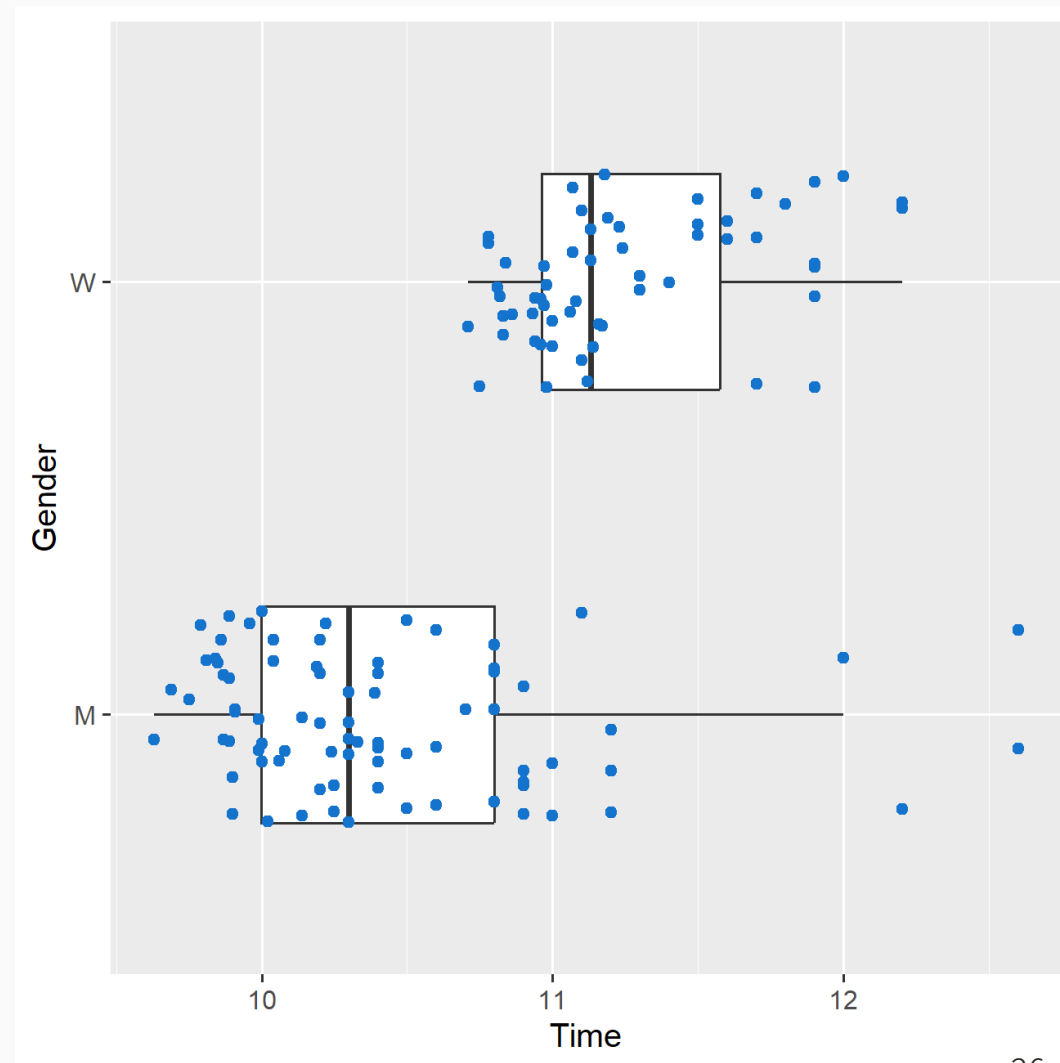
What is wrong with this plot?



# Example: box plot with raw data display

```
ggplot(olympic_100m_data,  
      aes(x = Time, y = Gender)) +  
  geom_boxplot(width = 0.5, outlier.shape = NA) +  
  geom_jitter(width = 0, height = 0.25,  
             colour = "dodgerblue3")
```

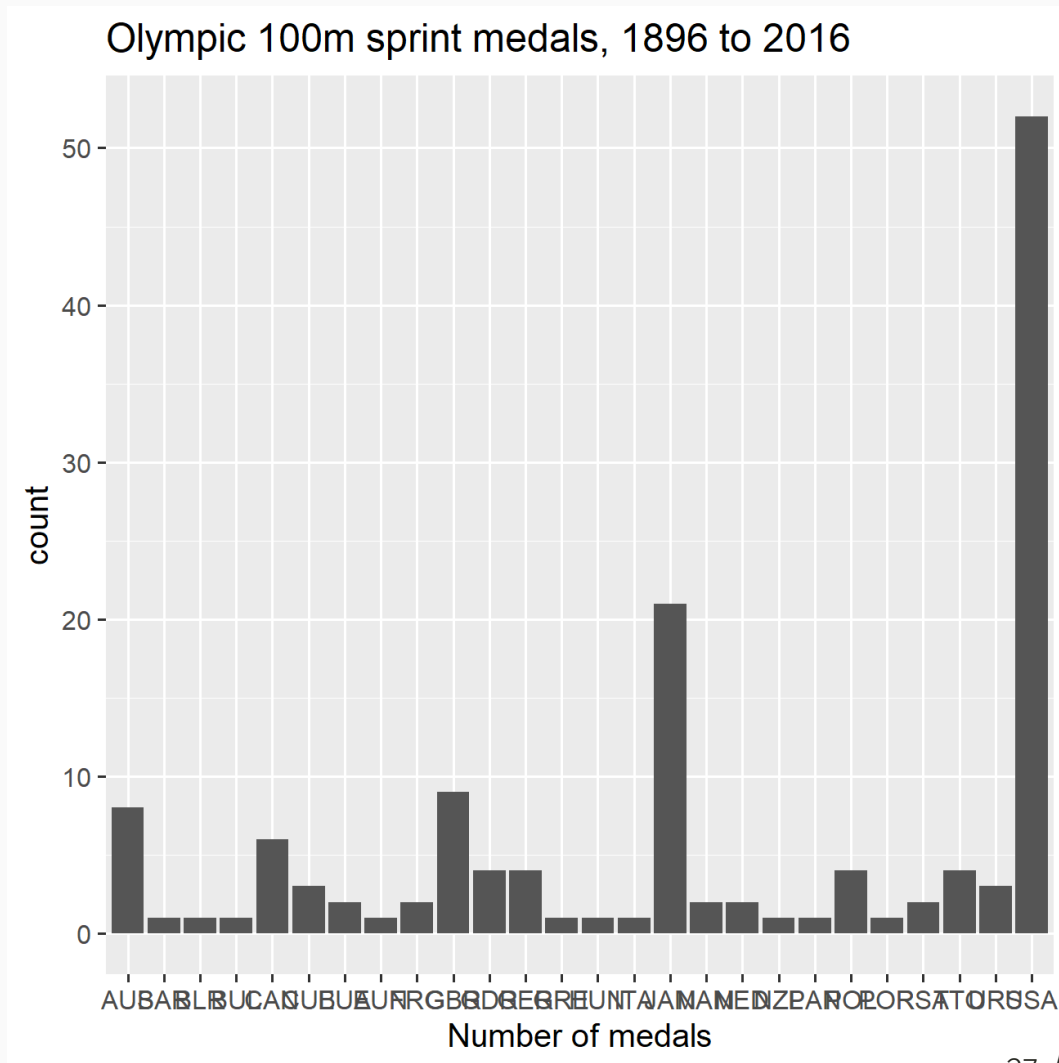
Outliers were plotted twice! We can ask `geom_boxplot` not to plot outliers.



# Example: bar chart

```
ggplot(olympic_100m_data,  
       aes(x = Nationality)) +  
  geom_bar() +  
  labs(x = "Number of medals",  
       title = "Olympic 100m sprint medals, 1896 to 2016")
```

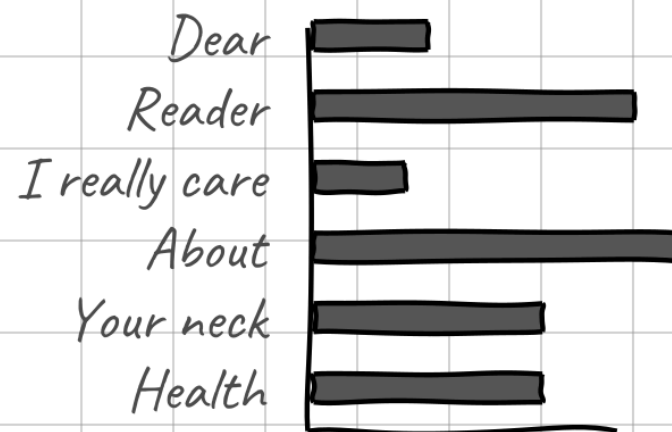
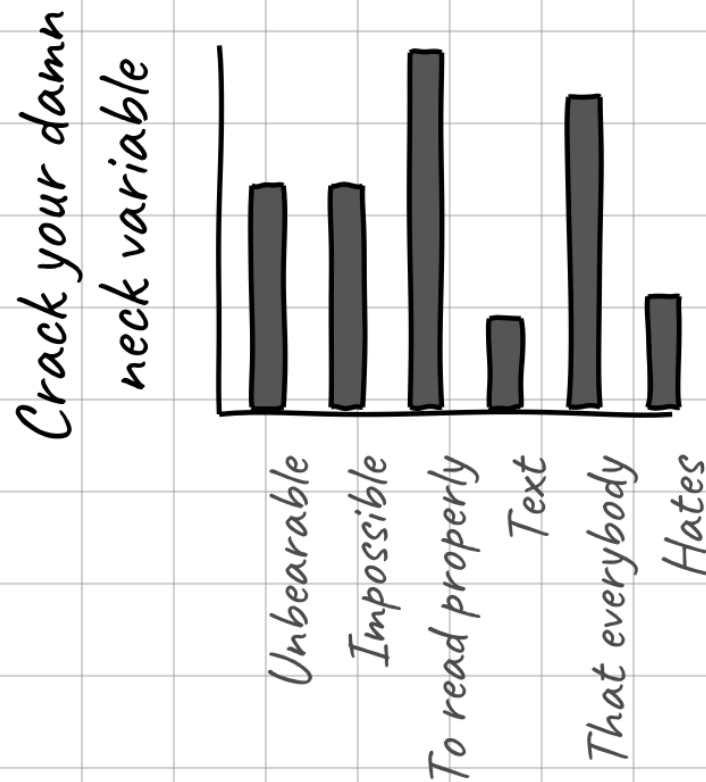
Earlier I said that I normally prefer to put categories on the y axis. Here's why.



# ROTATE THE DAMN PLOT

## THE SINGLE EASIEST AND MOST USEFUL DATAVIZ

# TRICK



Perfectly readable  
variable description

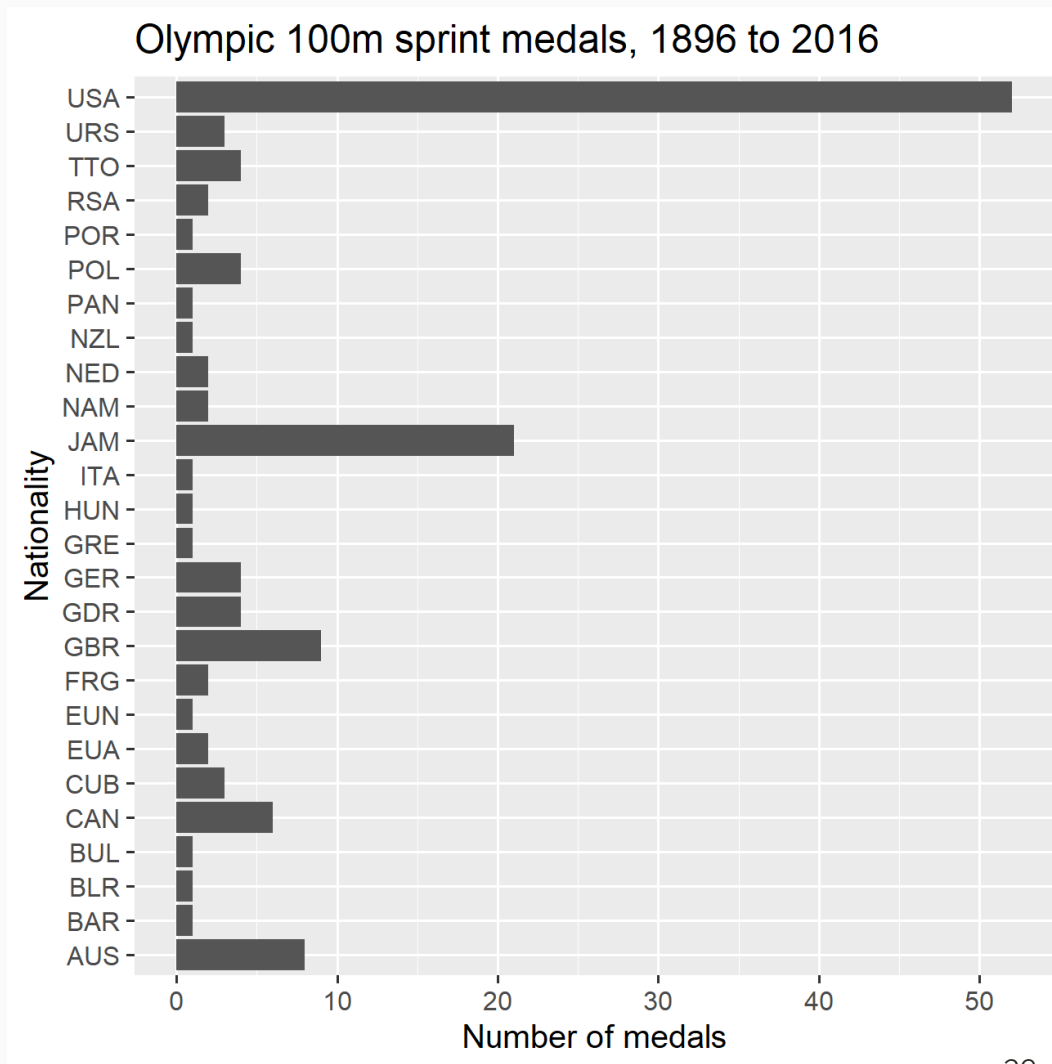
# Example: bar chart

```
ggplot(olympic_100m_data,  
      aes(y = Nationality)) +  
  geom_bar() +  
  labs(x = "Number of medals",  
       title = "Olympic 100m sprint medals, 1896 to 2016")
```

What order are the nationalities in?

We only gave ggplot a variable to plot on one axis (y).

Where are the values on the x axis coming from?

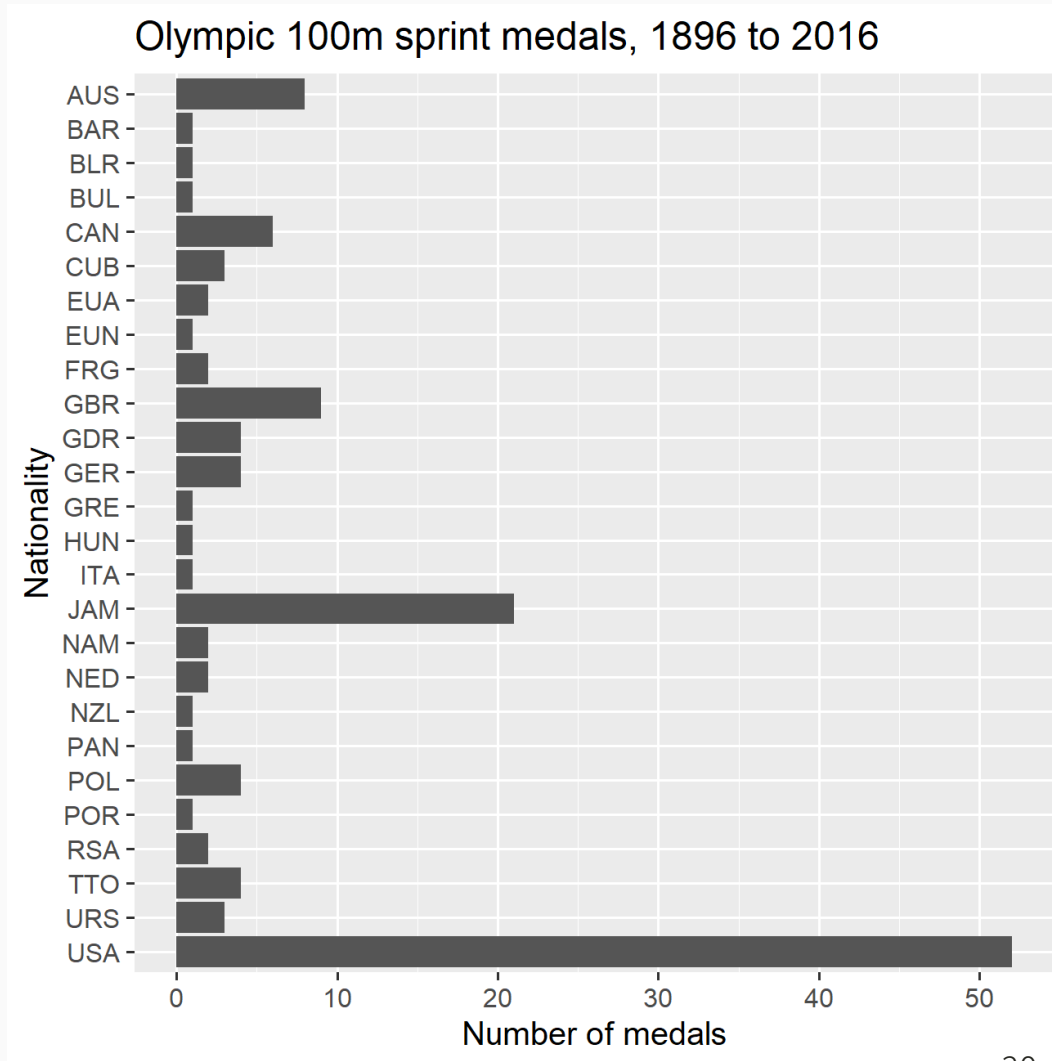


# Example: bar chart

```
ggplot(olympic_100m_data,  
      aes(y = Nationality)) +  
  geom_bar() +  
  labs(x = "Number of medals",  
       title = "Olympic 100m sprint medals, 1896 to 2016",  
       scale_y_discrete(limits = rev))
```

Is this really an improvement? What would a better order for the categories be?

Making really nice bar charts in ggplot is more work than you'd hope. More on that later.



# Commonly used aesthetic attributes

These can all be assigned to a column in the data using `aes()` or set manually for a particular `geom`.

- `x`, `y`: position
- `colour`: colour of line or symbol
- `fill`: fill colour of bars, boxplots, polygons
- `size`: symbol size or line width
- `shape`: symbol shape
- `linetype`: solid, dotted or dashed
- `alpha`: degree of transparency (where 0 is invisible, 1 is opaque)
- `group`: which observations to join with a line
- `width`, `height`: size of bars, errorbars, boxplots; or amount of jitter added
- `xmin`, `xmax`, `ymin`, `ymax`: range of error bars
- `xend`, `yend`: limits of line segments

# Commonly used geoms

## Simple shapes for each observation

- `geom_point()`: symbols
- `geom_line()`: lines
- `geom_col()`: columns (bars)

## Less common variations

- `geom_jitter()`: symbols with random shift to position
- `geom_errorbar()`, `geom_linerange()`, `geom_pointrange()`, `geom_crossbar()`: error bars of various shapes
- `geom_step()`: stair-step plots
- `geom_path()`, `geom_polygon()`: unfilled and filled polygons
- `geom_ribbon()`, `geom_area()`: ribbon plots
- `geom_tile()`, `geom_bin2d()`: heat maps

## Summarising data

- `geom_bar()`: bars for number of rows in a category
- `geom_histogram()`: histograms for continuous data
- `geom_boxplot()`: box plots
- `geom_dotplot()`: dot plots
- `geom_smooth()`: line or curve of best fit
- `stat_summary()`: plot almost any summary statistics (including means and confidence intervals)

## Useful for adding annotations

- `geom_vline()`, `geom_hline()`, `geom_abline()`, `geom_segment()`: reference lines
- `geom_text()`, `geom_label()`: text labels



# Commonly used scales

## Location

- `scale_x_continuous()`: numerical axes
- `scale_x_log10()`: log-scaled numerical axes
- `scale_x_discrete()`: categorical axes
- `scale_x_date()`, `scale_x_time()`, `scale_x_datetime()`: axes representing dates and/or times

(plus `scale_y_*` for all of the above)

## Colour

- `scale_colour_brewer()`: nice palettes for discrete variables
- `scale_colour_distiller()`: nice palettes for continuous variables
- `scale_colour_manual()`: specify colours by hand

(plus `scale_fill_*` for all of the above)

## Others

- `scale_shape_manual()`
- `scale_size_manual()`
- `scale_linetype_manual()`

These allow setting your own values for each category.

## Exercise 1.3.