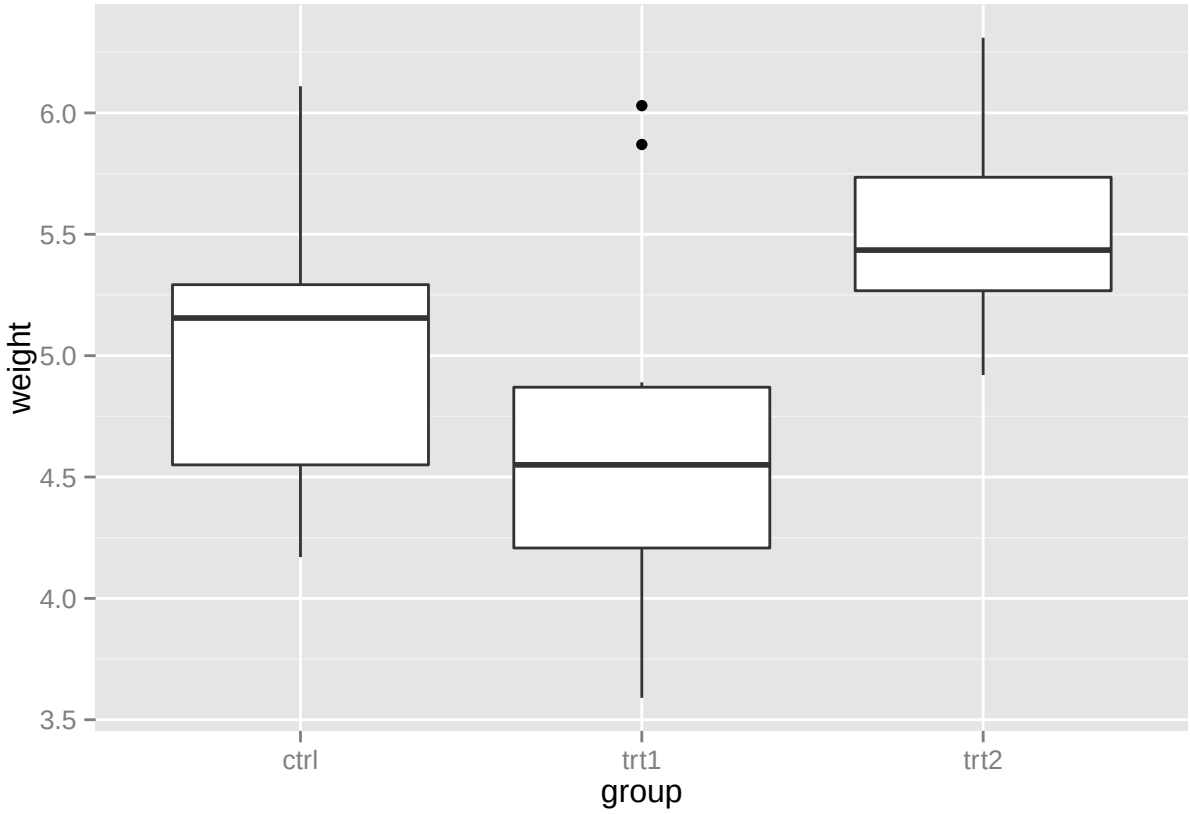
8_Axes

Gino Tesei

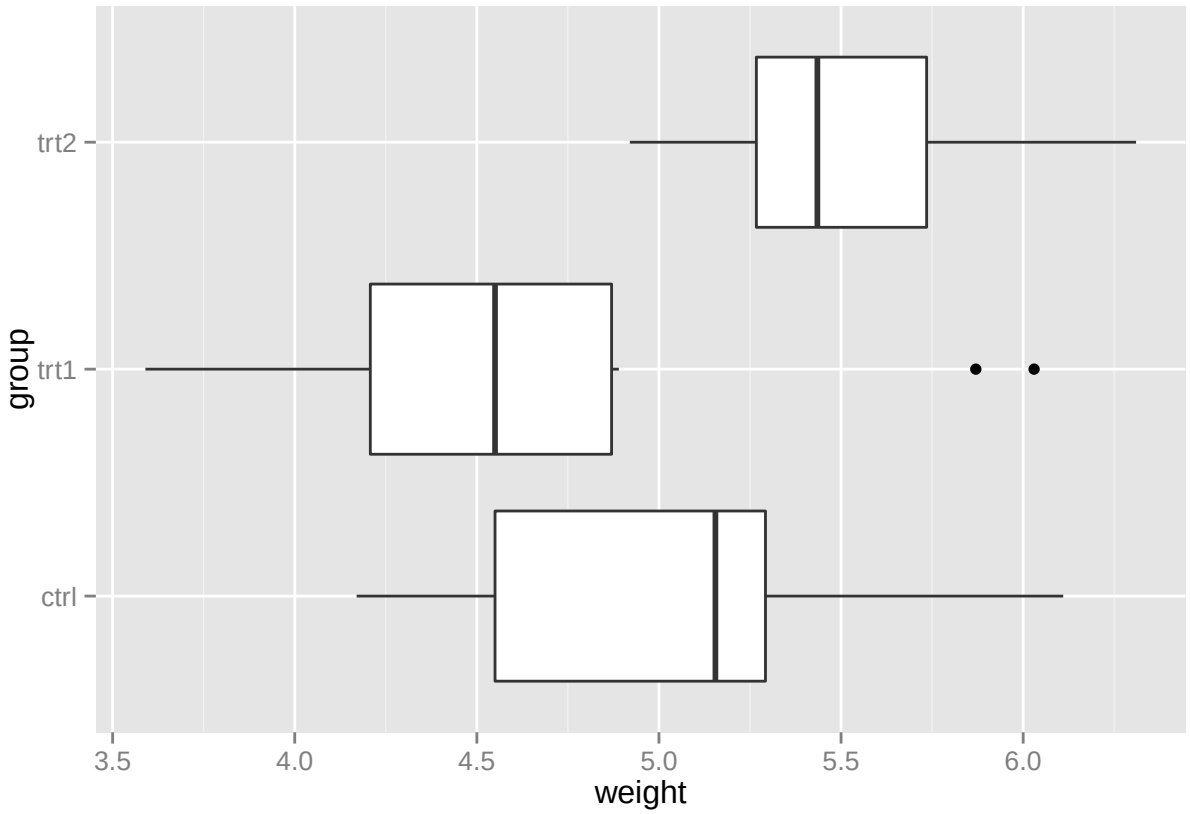
December 13, 2015

1. Swapping X- and Y-Axes

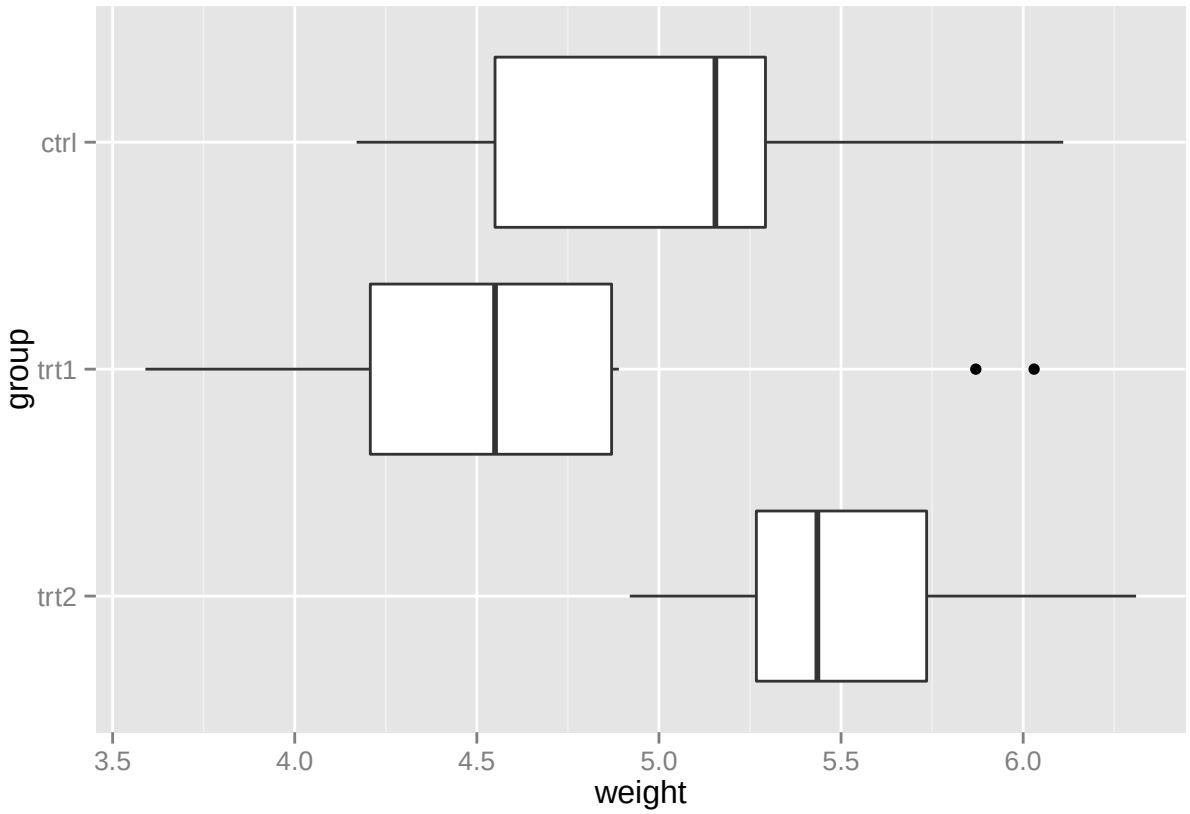
```
library(ggplot2)
library(gcookbook) # For the data set

library(plyr)

ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot()
```




```
## Use coord_flip() to flip the axes
ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot() + coord_flip()
```

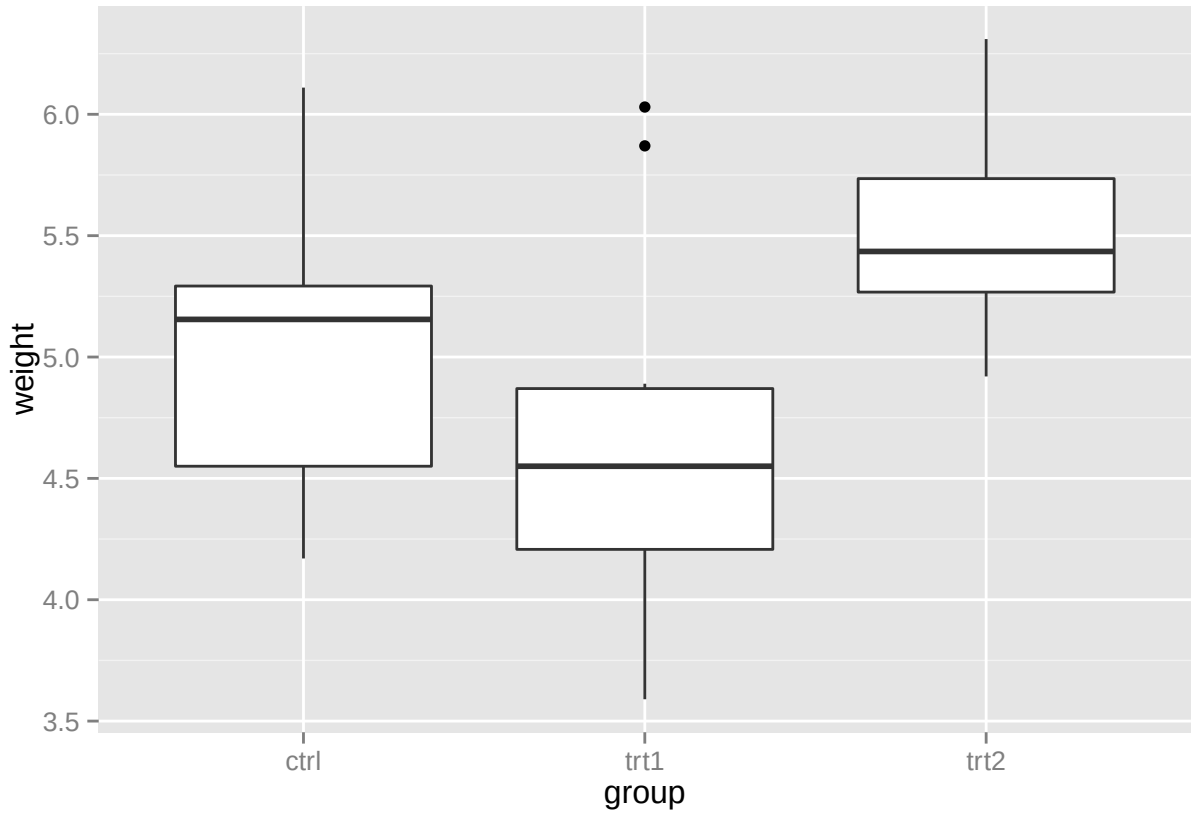



```
# If the x variable is a factor, the order can be reversed by using scale_x_discrete() with limits = re
ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot() +
  coord_flip() +
  scale_x_discrete( limits = rev( levels( PlantGrowth$group)))
```

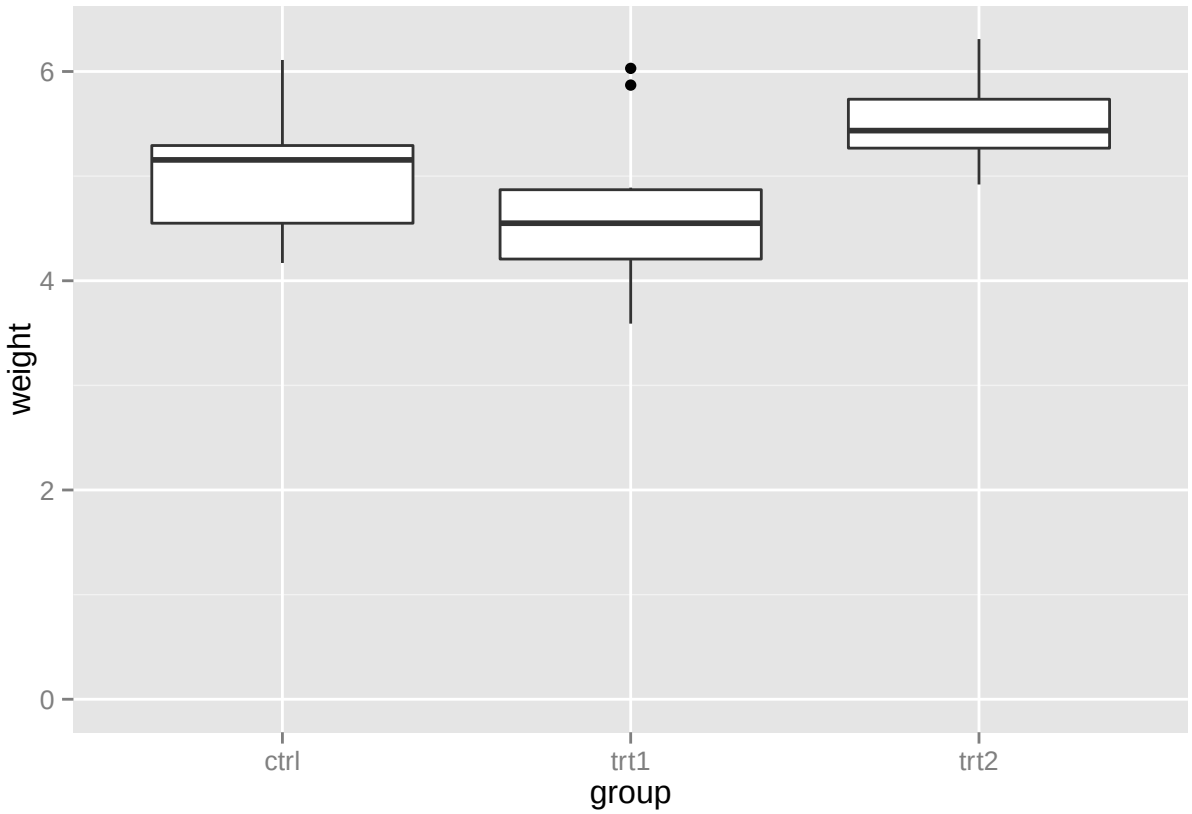


2. Setting the Range of a Continuous Axis

```
p <- ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot()

# Display the basic graph
p
```

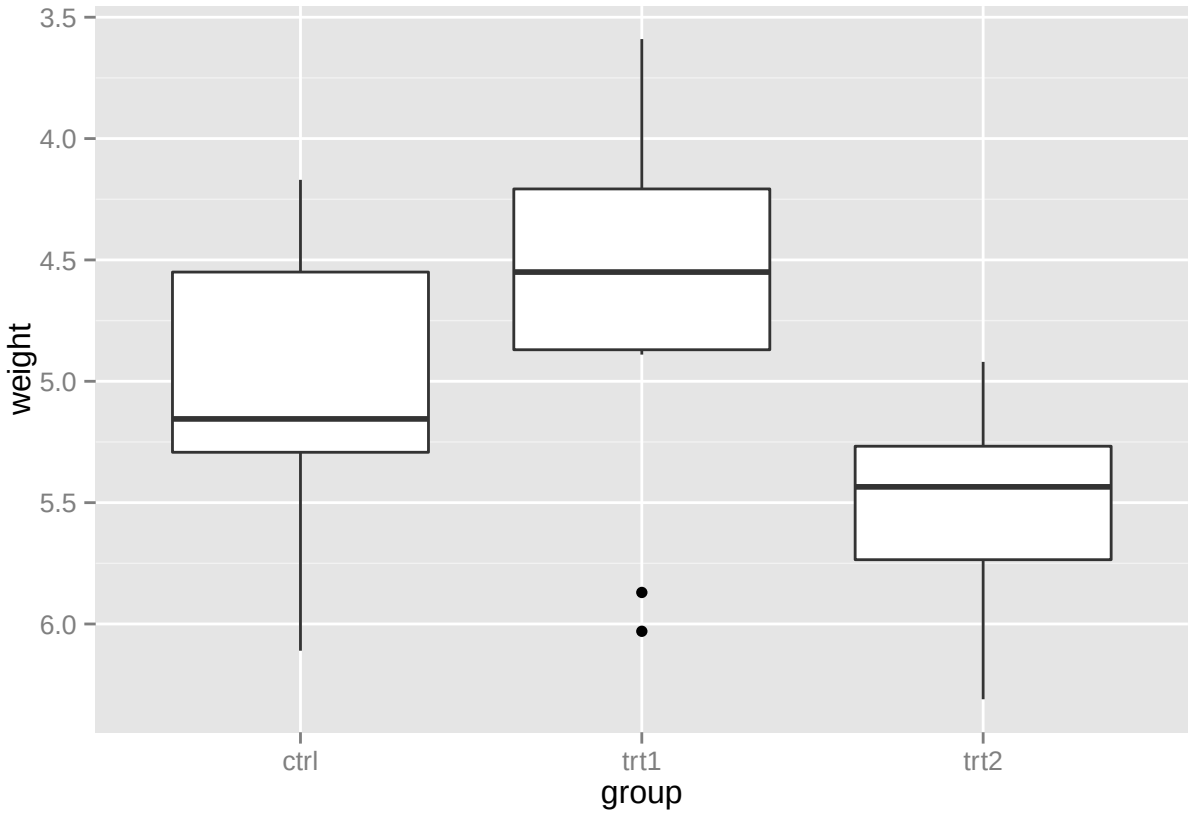



```
# Display the graph with longet y axis
p + ylim( 0, max( PlantGrowth$weight))
```

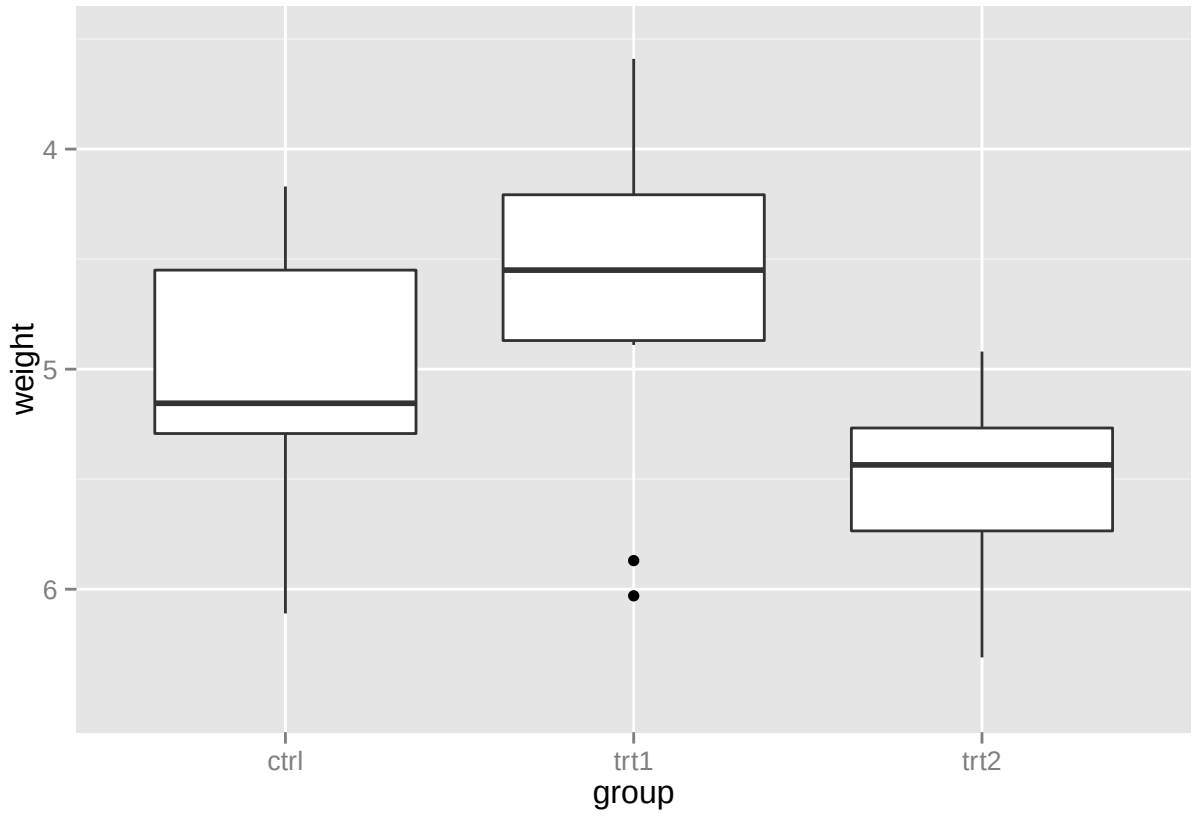


3. Reversing a Continuous Axis

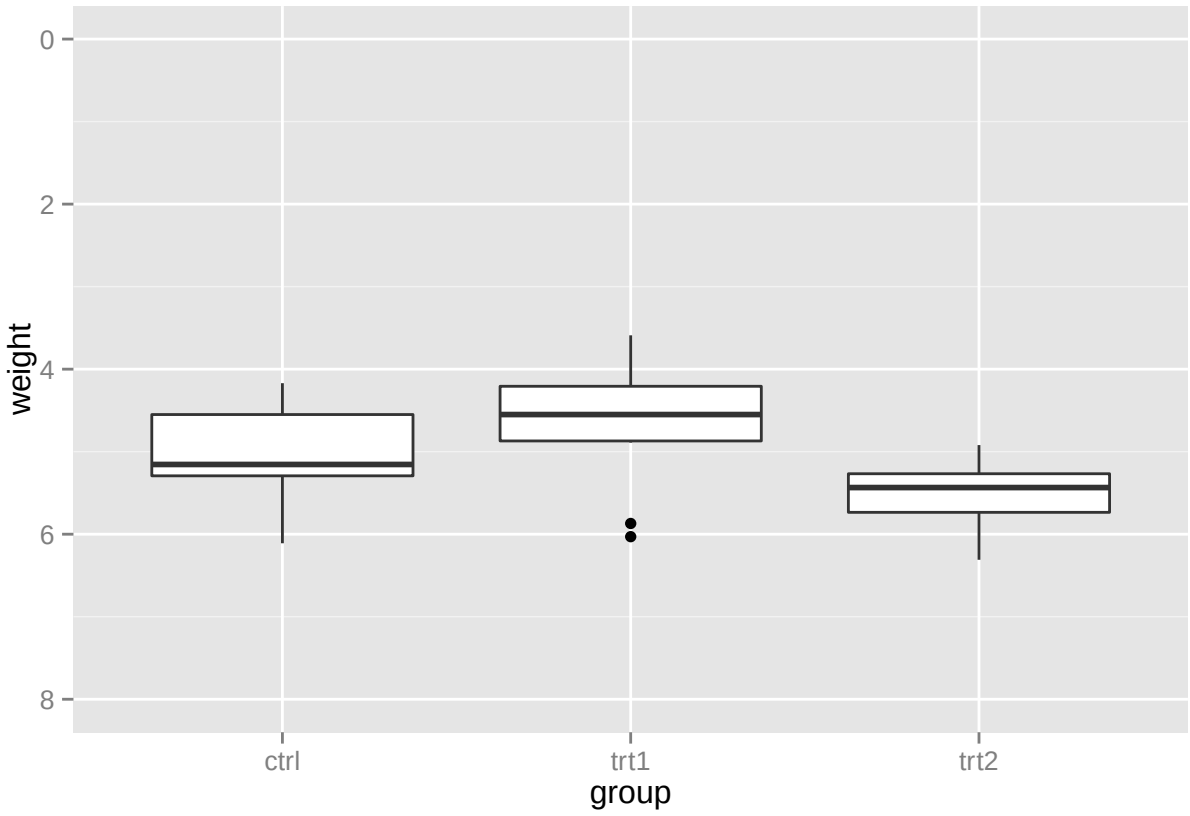
```
ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot() +
  scale_y_reverse()
```




```
# Similar effect by specifying limits in reversed order
ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot() + ylim( 6.5, 3.5)
```




```
# Like scale_y_continuous(), scale_y_reverse() does not work with ylim.
# The same is true for the x-axis properties.
# If you want to reverse an axis and set its range, you must do it within the scale_y_reverse() stateme
ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot() +
  scale_y_reverse( limits = c( 8, 0))
```

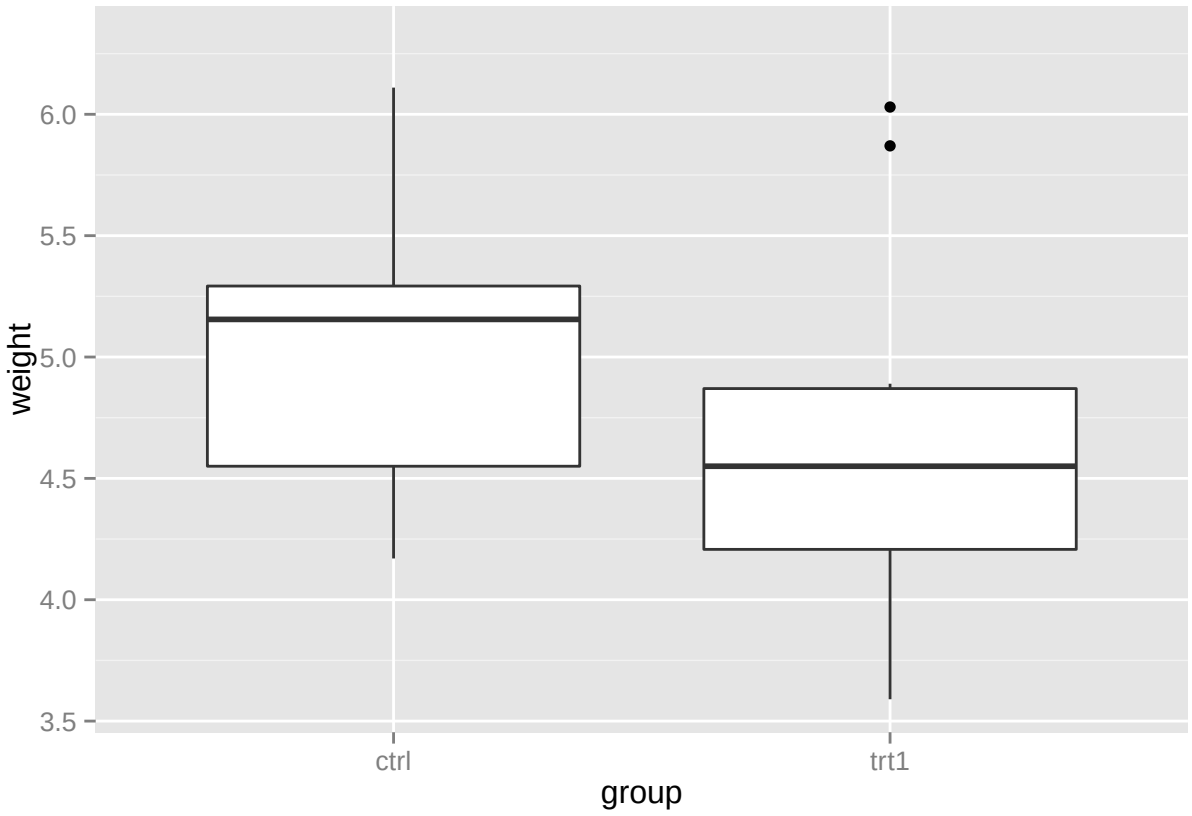


4. Changing the Order of Items on a Categorical Axis

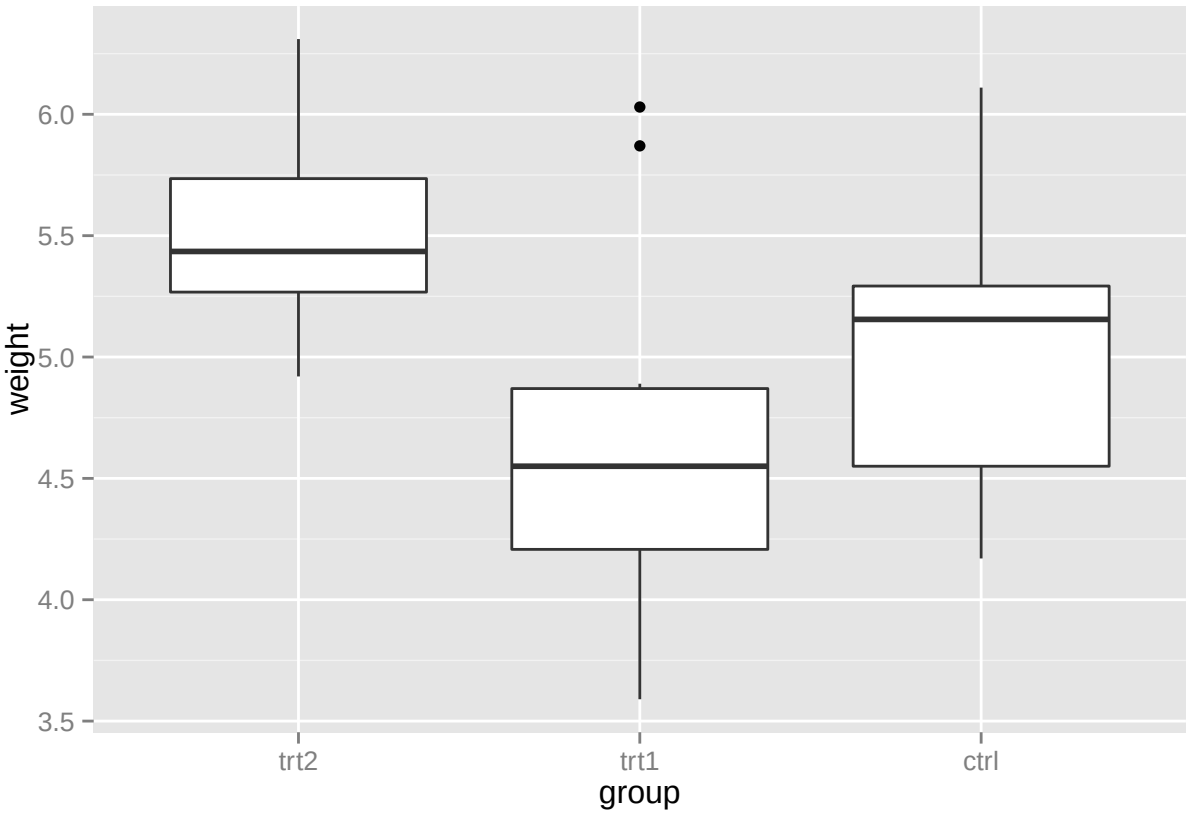
```
# To manually set the order of items on the axis, specify limits with a vector of the levels in the des
p <- ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot()

p +
  scale_x_discrete( limits = c("ctrl","trt1"))
```

```
## Warning: Removed 2 rows containing missing values (geom_segment).
```

```
## Warning: Removed 1 rows containing missing values (geom_segment).
```

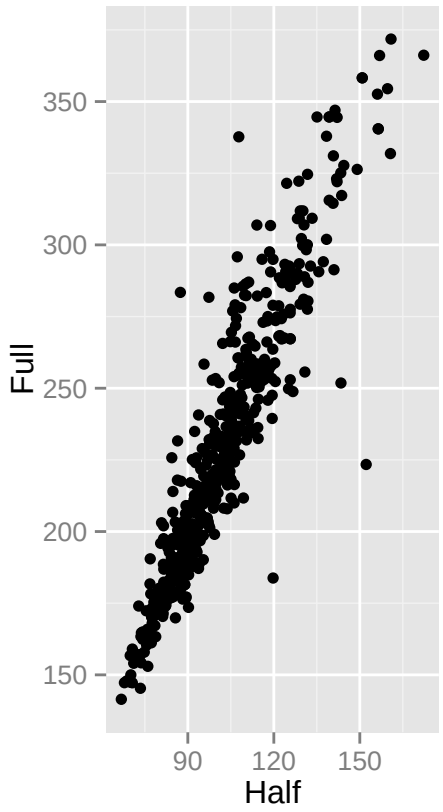



```
# To reverse the order, set limits = rev( levels(...)),
p +
  scale_x_discrete( limits = rev( levels( PlantGrowth$group)))
```

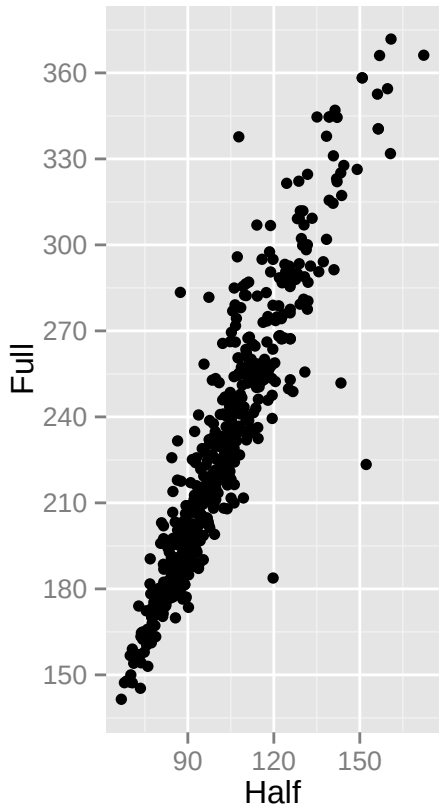


5. Setting the Scaling Ratio of the X- and Y-Axes

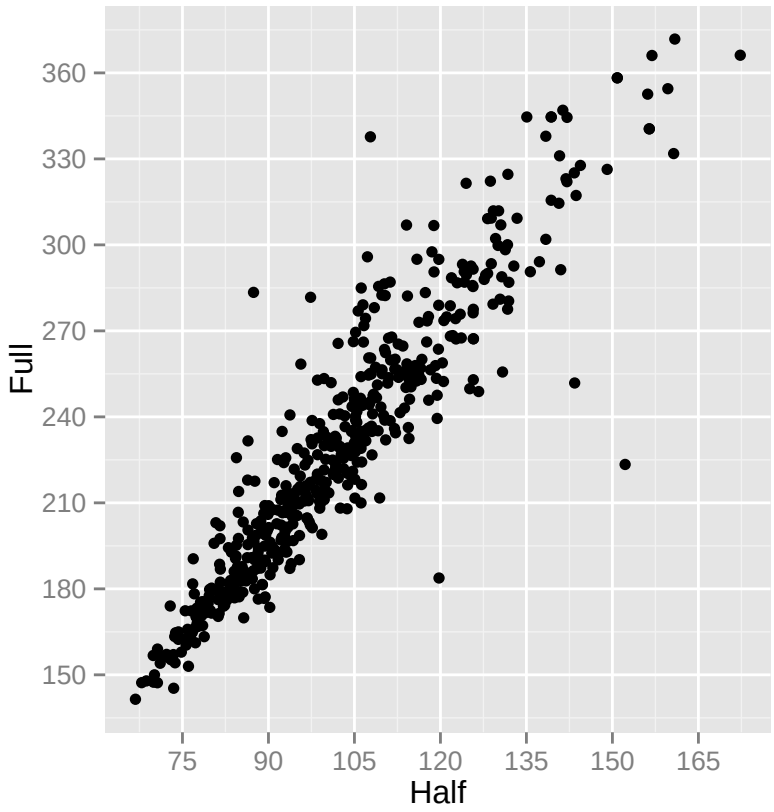
```
sp <- ggplot( marathon, aes( x = Half, y = Full)) +
  geom_point()

# Use coord_fixed(). This will result in a 1:1 scaling between the x- and y-axes
sp +
  coord_fixed()
```




```
# It's also helpful to set the tick spacing to be the same, by setting breaks in scale_y_continuous() a
sp +
  coord_fixed() +
  scale_y_continuous( breaks = seq( 0, 420, 30)) +
  scale_x_continuous( breaks = seq( 0, 420, 30))
```

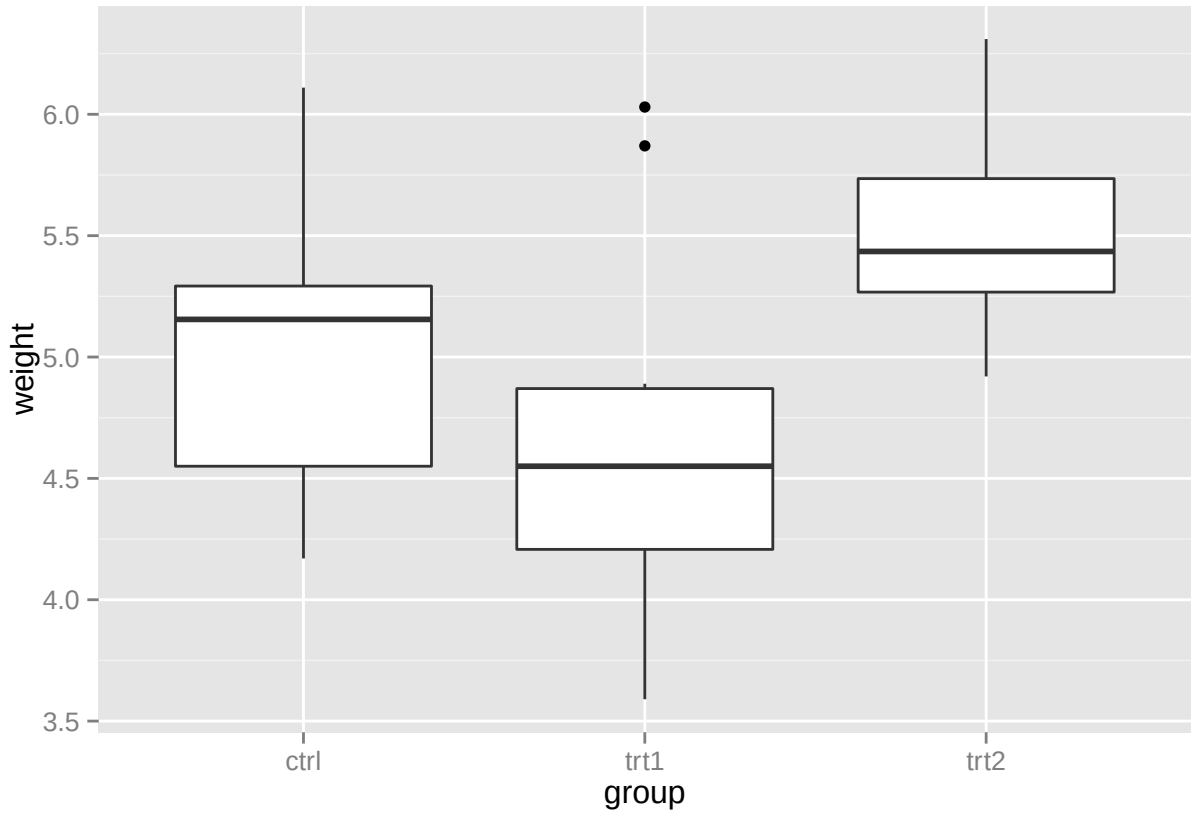



```
# If, instead of an equal ratio, you want some other fixed ratio between the axes, set the ratio parame
sp +
  coord_fixed( ratio = 1/2) +
  scale_y_continuous( breaks = seq( 0, 420, 30)) +
  scale_x_continuous( breaks = seq( 0, 420, 15))
```

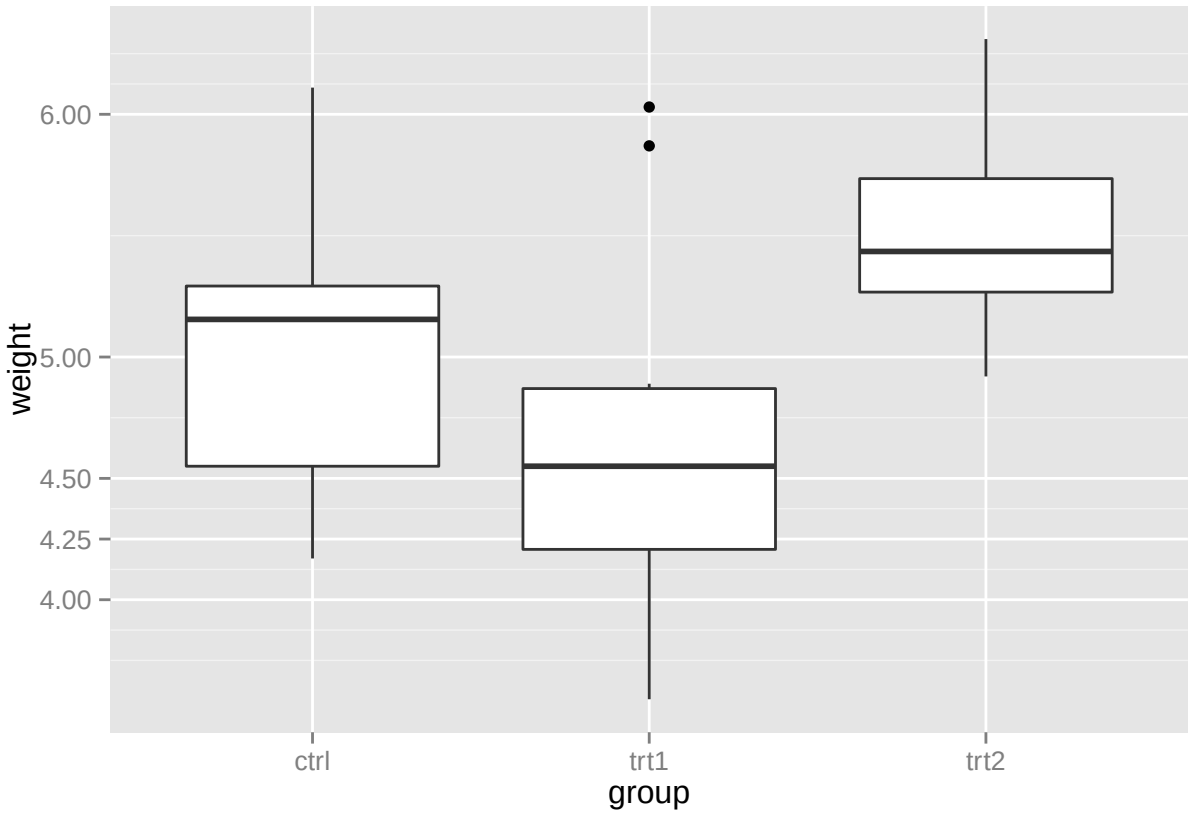


6. Setting the Positions of Tick Marks

```
# Usually ggplot() does a good job of deciding where to put the tick marks
ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot()
```




```
# but if you want to change them, set breaks in the scale
ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot() +
  scale_y_continuous( breaks = c( 4, 4.25, 4.5, 5, 6, 8))
```

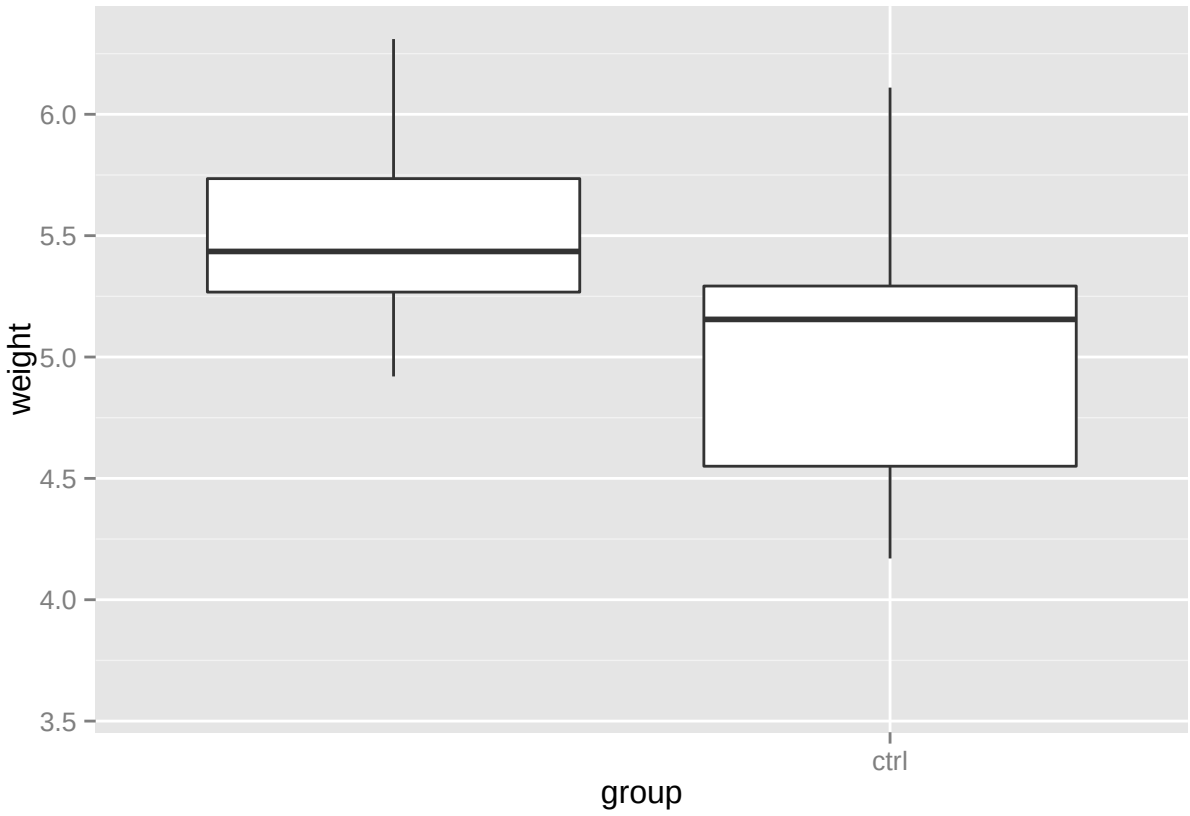



```
# Discrete axis: setting breaks will change which of the levels are labeled, but will not remove them o
ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot() +
  scale_x_discrete( limits = c("trt2", "ctrl"), breaks ="ctrl")
```

```
## Warning: Removed 2 rows containing missing values (geom_point).
```

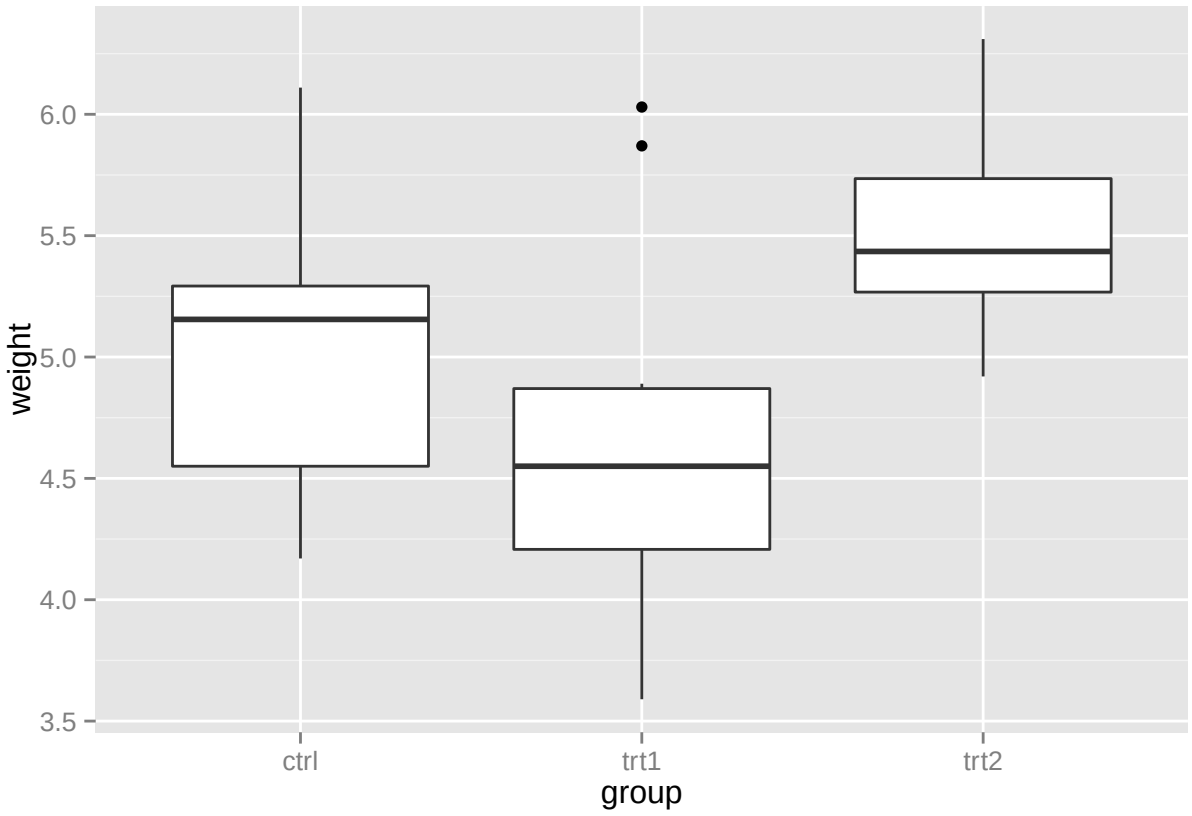
```
## Warning: Removed 2 rows containing missing values (geom_segment).
```

```
## Warning: Removed 1 rows containing missing values (geom_segment).
```

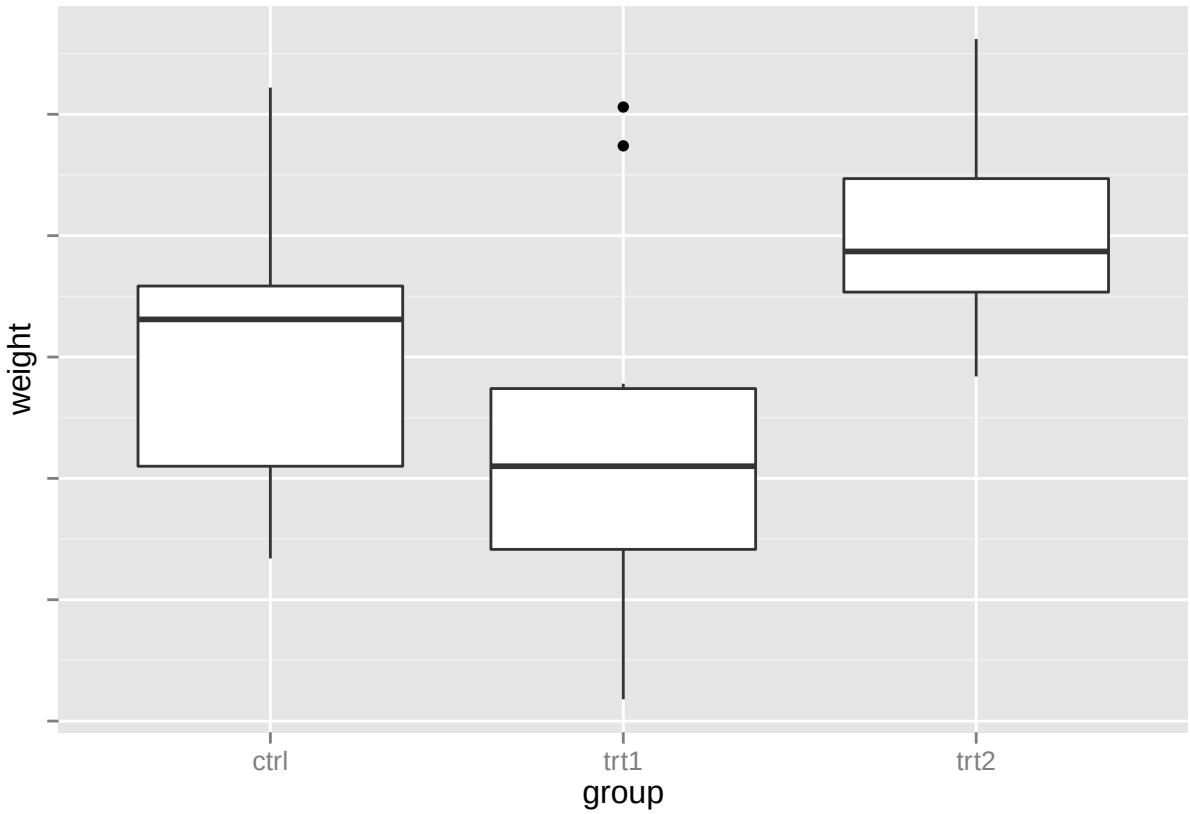


7. Removing Tick Marks and Labels

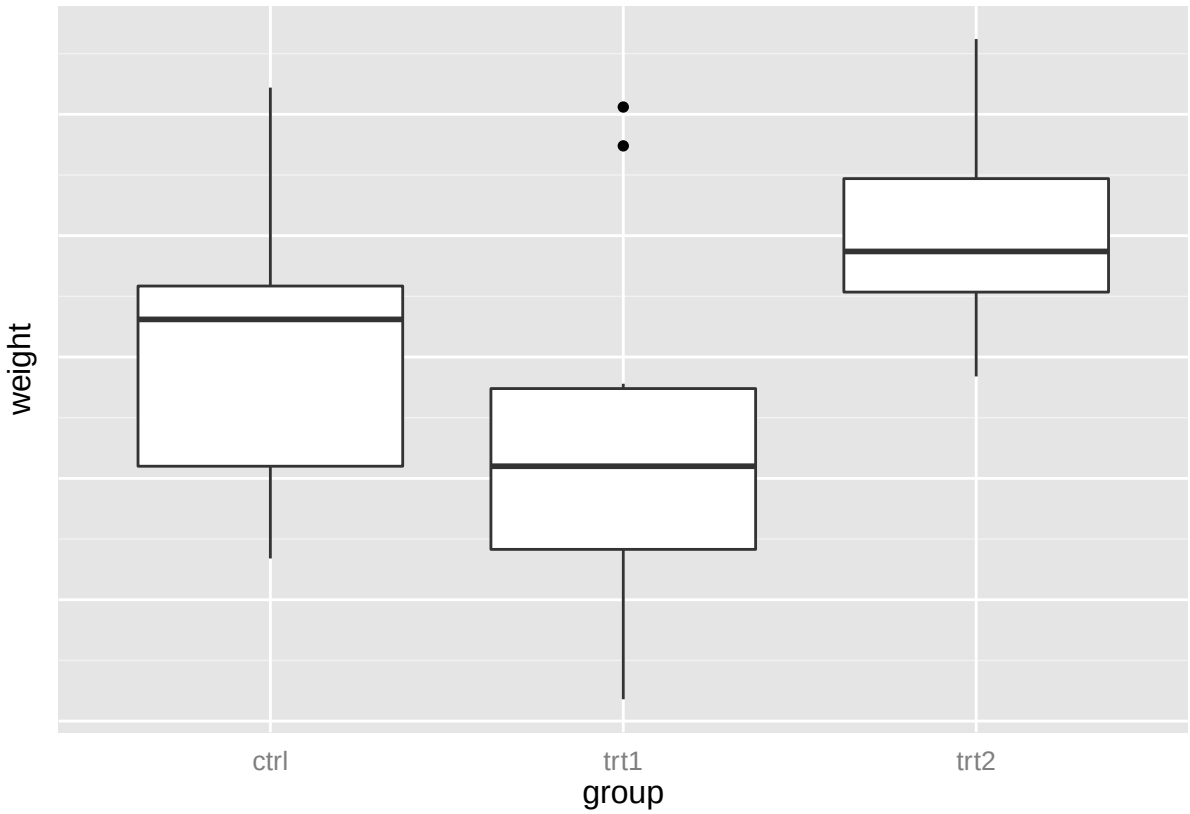
```
p <- ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot()

# default
p
```



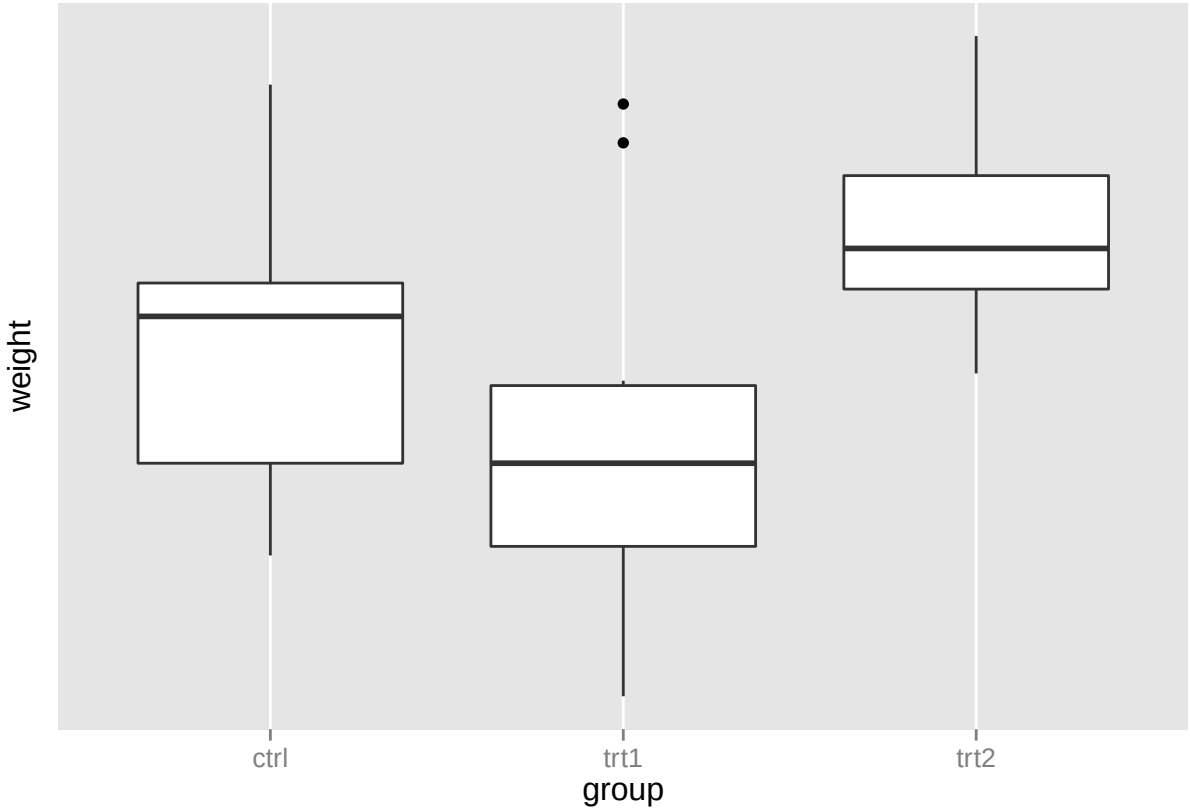

```
# To remove just the tick labels use theme( axis.text.y = element_blank())
p +
  theme( axis.text.y = element_blank())
```




```
# To remove the tick marks, use theme( axis.ticks = element_blank()).
p +
  theme( axis.ticks = element_blank(), axis.text.y = element_blank())
```

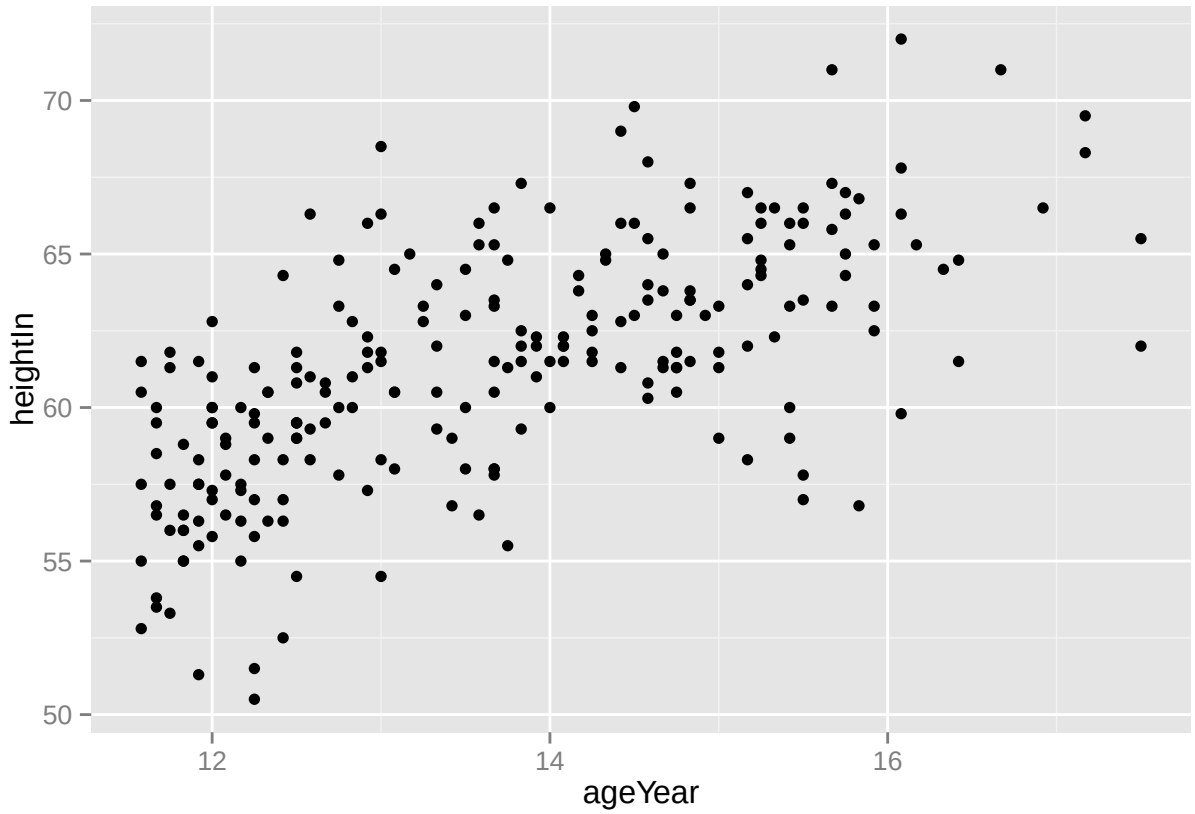



```
# To remove the tick marks, the labels, and the grid lines, set breaks to NULL
p +
  scale_y_continuous( breaks = NULL)
```

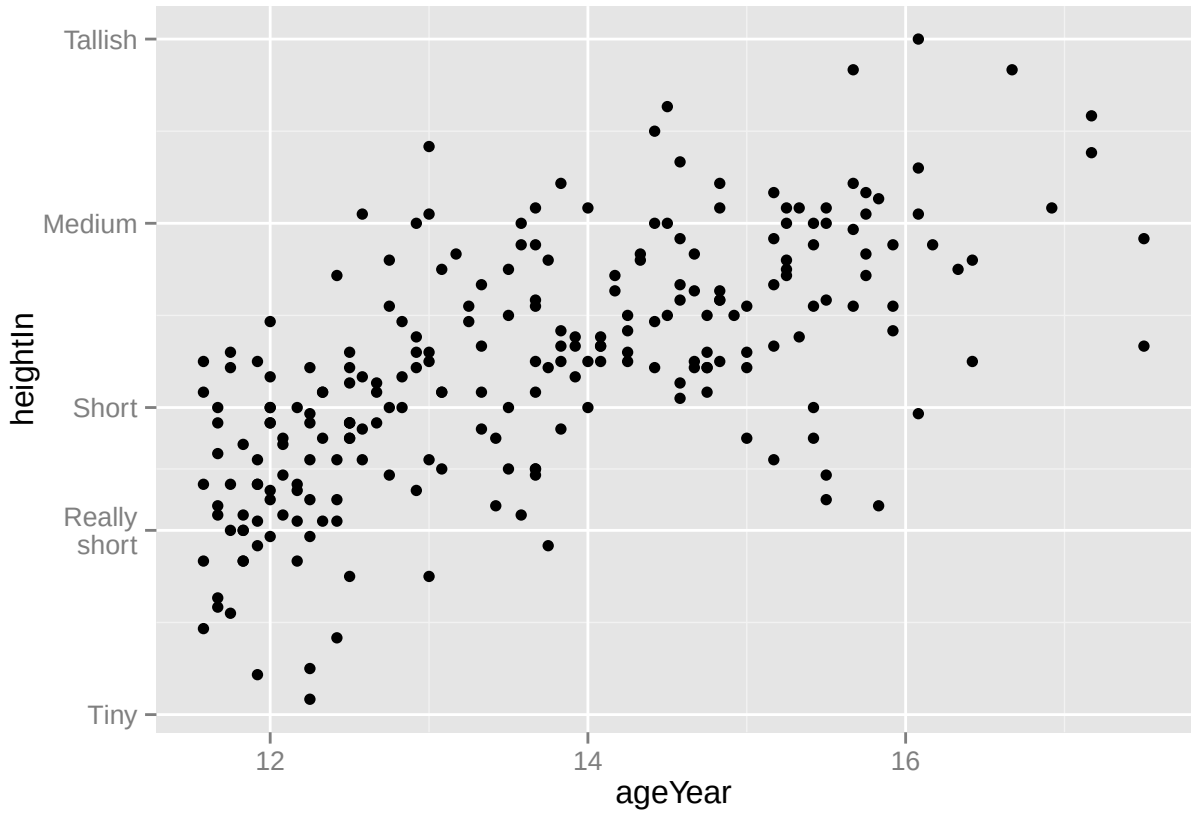


8. Changing the Text of Tick Labels

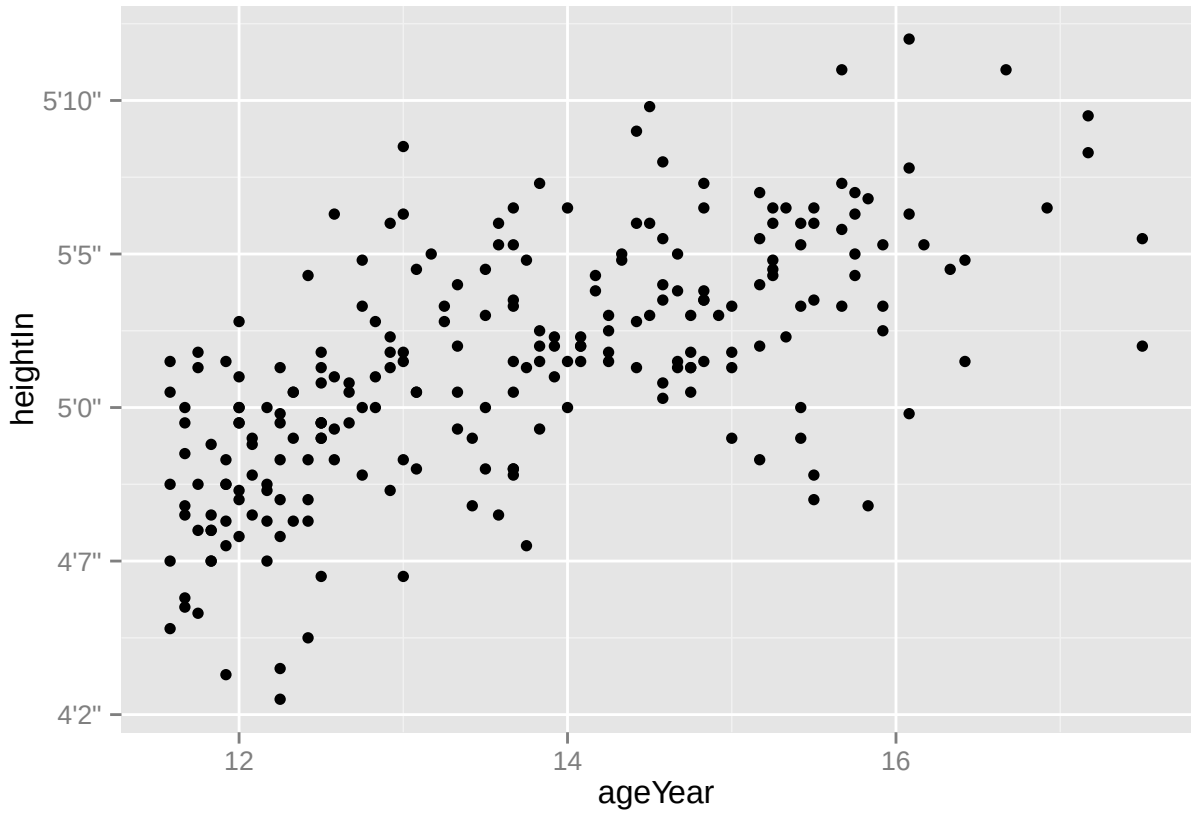
```
hwp <- ggplot( heightweight, aes( x = ageYear, y = heightIn)) +
  geom_point()

# default
hwp
```



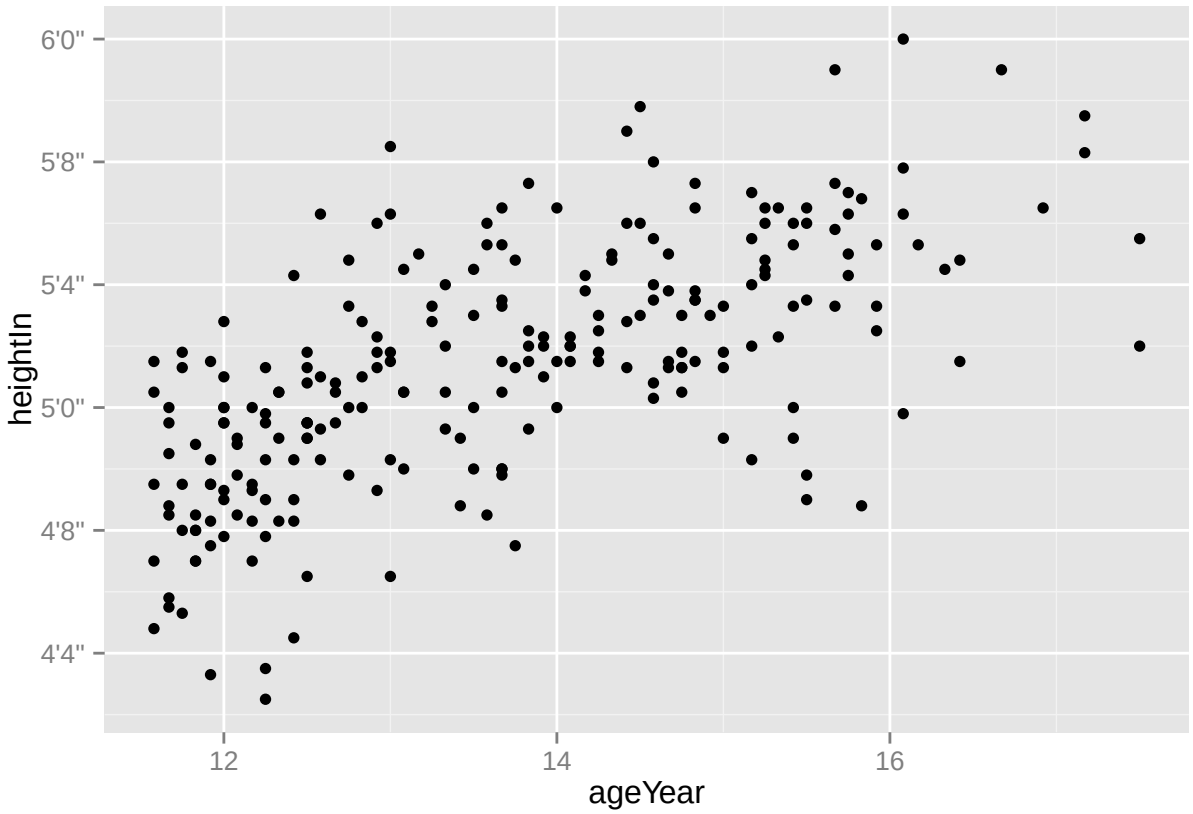

```
# To set arbitrary labels pass values to breaks and labels in the scale
hwp +
  scale_y_continuous( breaks = c( 50, 56, 60, 66, 72), labels = c("Tiny", "Really\nshort", "Short", "Me
```




```
# we can define a formatter function, which takes in a value and returns the corresponding string.
footinch_formatter <- function( x) {
  foot <- floor( x/12)
  inch <- x %% 12
  return( paste( foot, "'", inch, "\"", sep =""))
  }

# we can pass our function to the scale, using the labels parameter
hwp +
  scale_y_continuous( labels = footinch_formatter)
```




```
# We can instead have ggplot() set tick marks every four inches, by specifying breaks
hwp +
  scale_y_continuous( breaks = seq( 48, 72, 4), labels = footinch_formatter)
```

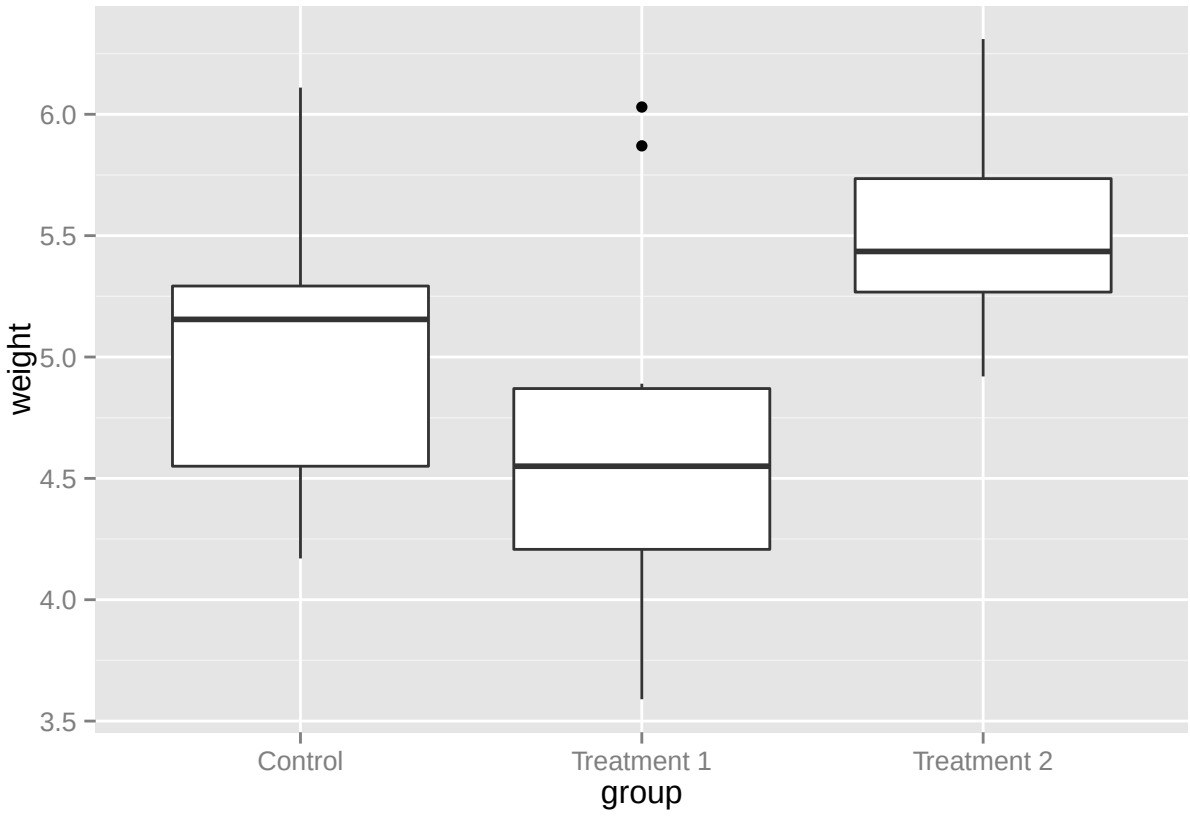



```
# Another common task is to convert time measurements to HH:MM:SS format, or something similar.
timeHMS_formatter <- function( x) {
  h <- floor( x/ 60)
  m <- floor( x %% 60)
  s <- round( 60*( x %% 1))

  # Round to nearest second
  lab <- sprintf("%02d:%02d:%02d", h, m, s)

  # Format the strings as HH:MM:SS
  lab <- gsub("^00:", "", lab)

  # Remove leading 00: if present
  lab <- gsub("^0", "", lab)

  # Remove leading 0 if present

  return(lab)
  }

timeHMS_formatter( c(.33, 50, 51.25, 59.32, 60, 60.1, 130.23))
```

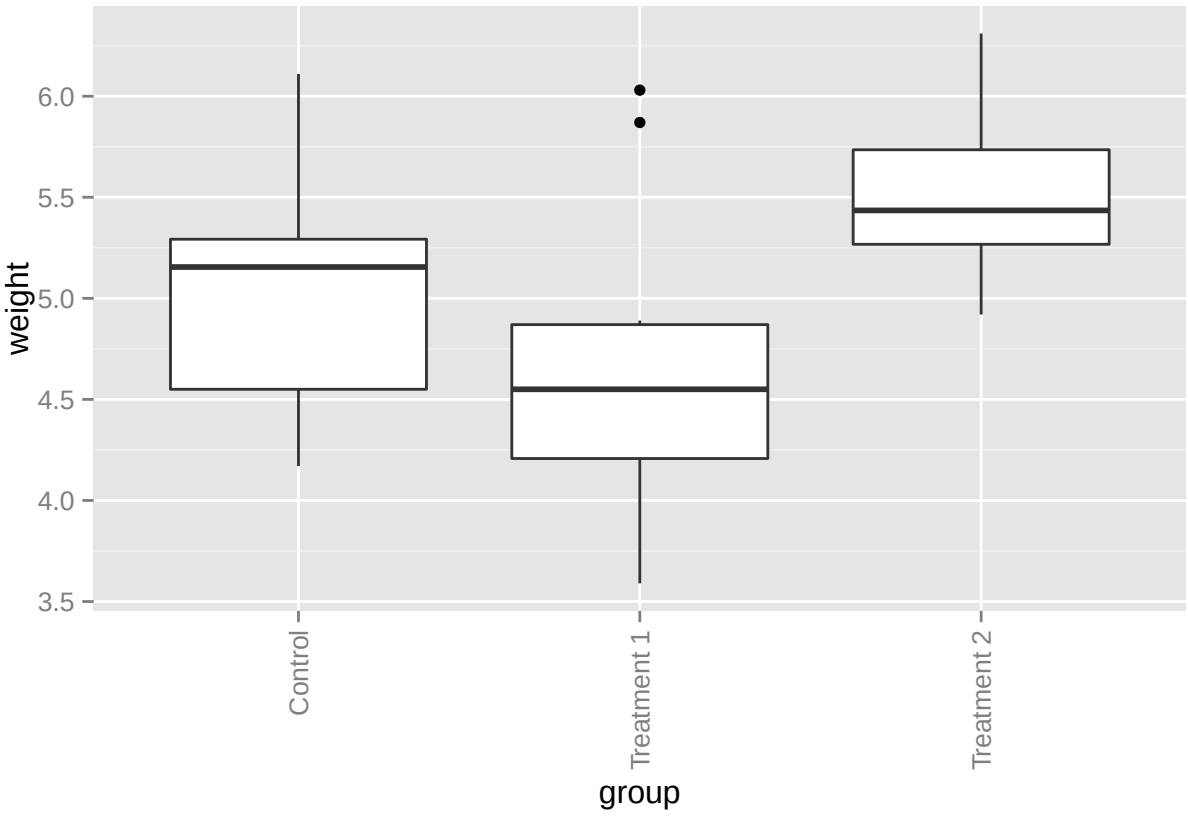
```
## [1] "0:20"    "50:00"   "51:15"   "59:19"   "1:00:00" "1:00:06" "2:10:14"
```

9. Changing the Appearance of Tick Labels

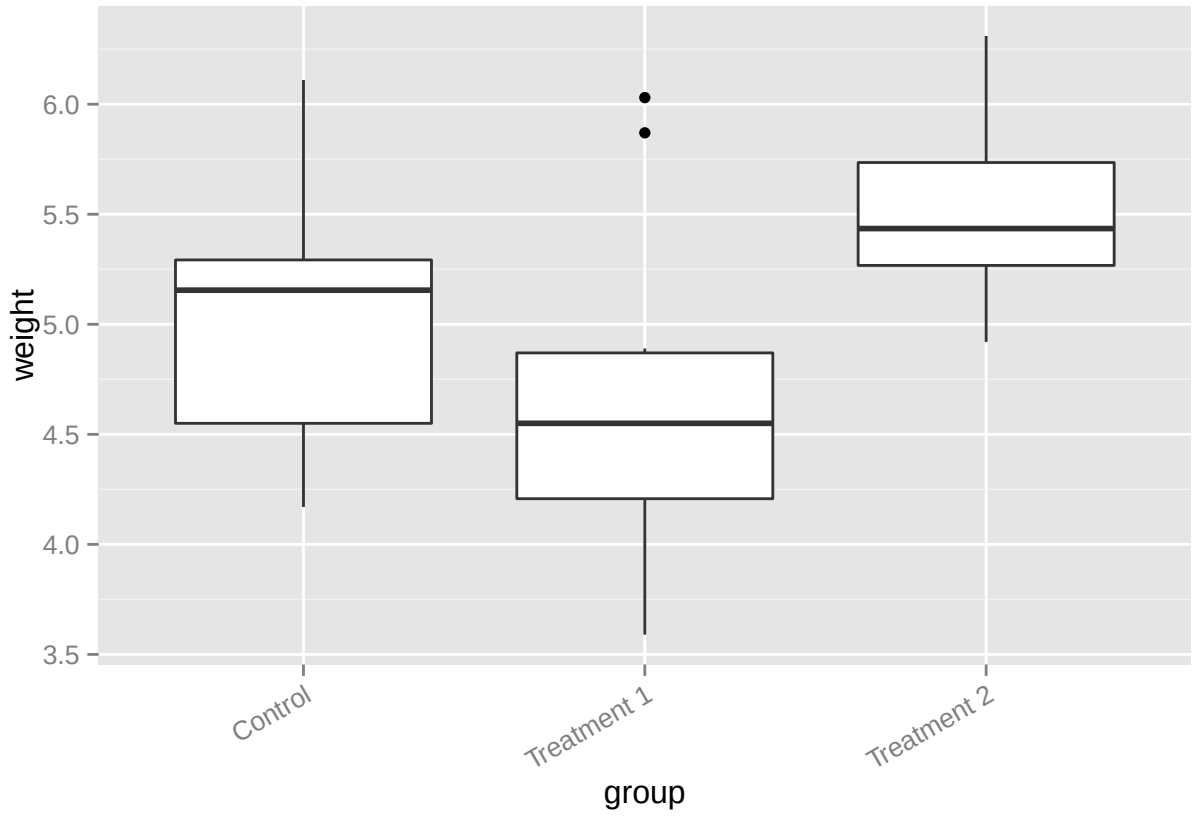
```
# manually set the labels to be long - long enough that they overlap:
bp <- ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot() + scale_x_discrete( breaks = c("ctrl", "trt1", "trt2"),
                                     labels = c("Control", "Treatment 1", "Treatment 2"))
bp
```



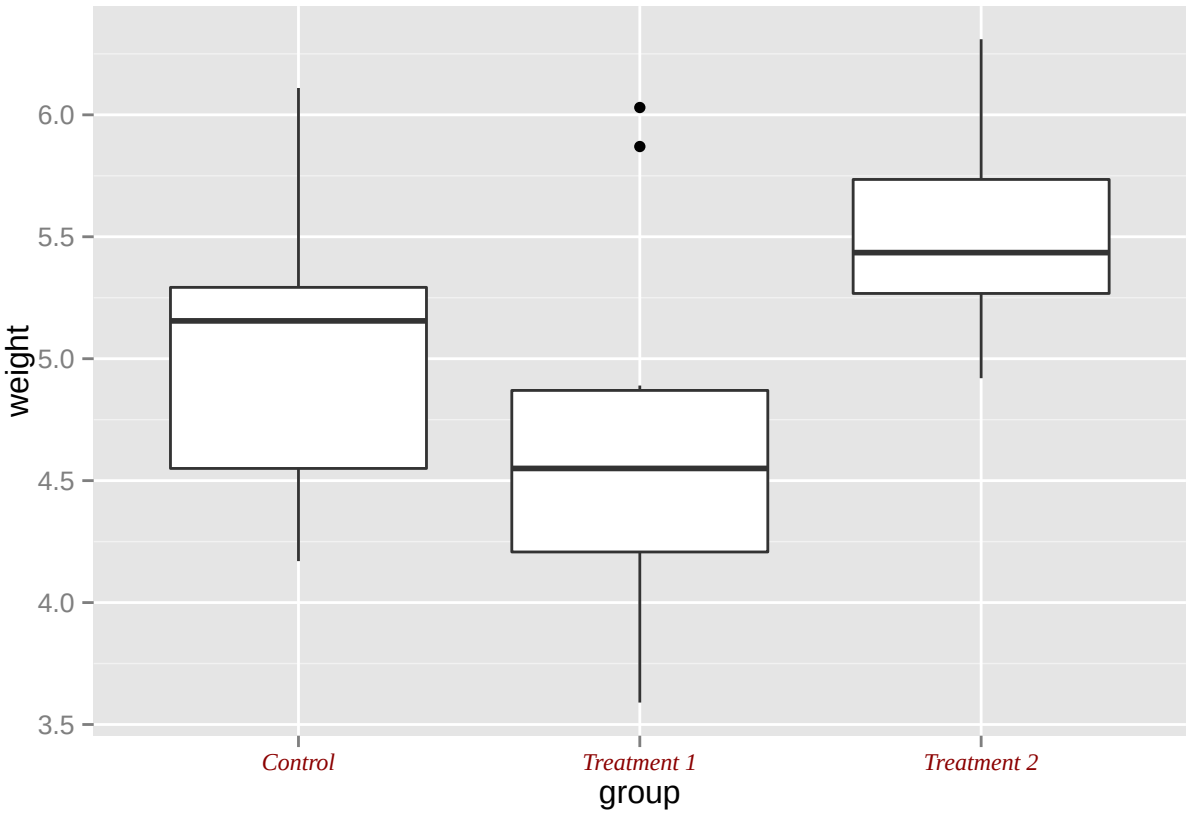

```
# To rotate the text 90 degrees counterclockwise
bp +
  theme( axis.text.x = element_text( angle = 90, hjust = 1, vjust =.5))
```




```
# Rotating the text 30 degrees
bp +
  theme( axis.text.x = element_text( angle = 30, hjust = 1, vjust = 1))
```

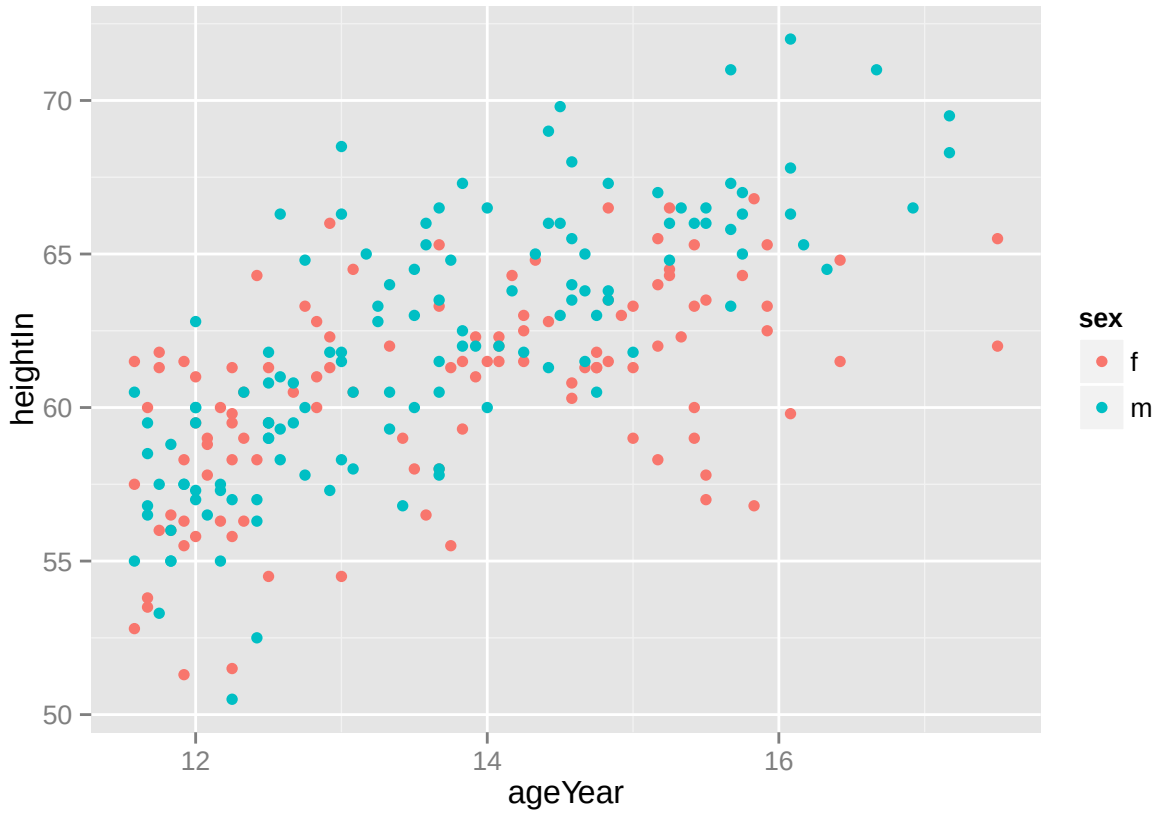



```
# Besides rotation, other text properties, such as size, style (bold/ italic/ normal), and the font fam
bp +
  theme( axis.text.x = element_text( family ="Times", face ="italic", colour ="darkred", size = rel( 0.
```

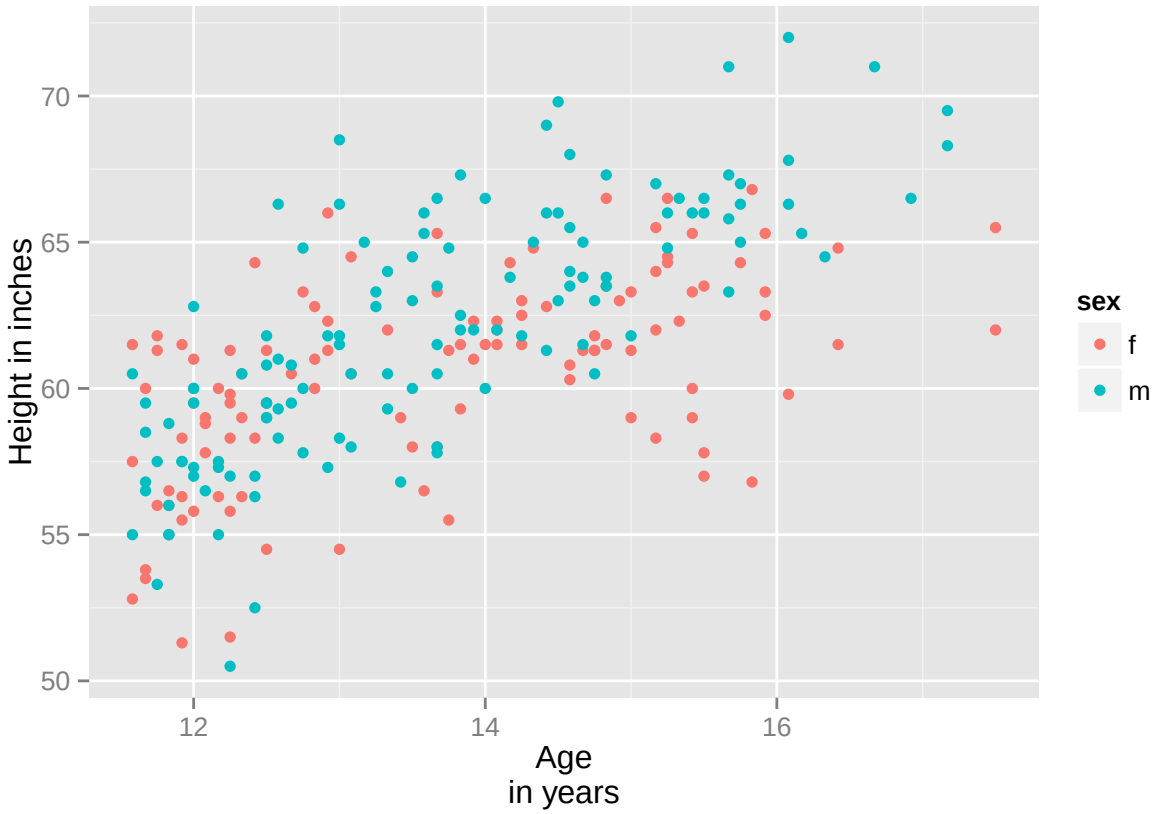


10. Changing the Text of Axis Labels

```
hwp <- ggplot( heightweight, aes( x = ageYear, y = heightIn, colour = sex)) +
  geom_point()

# With default axis labels
hwp
```

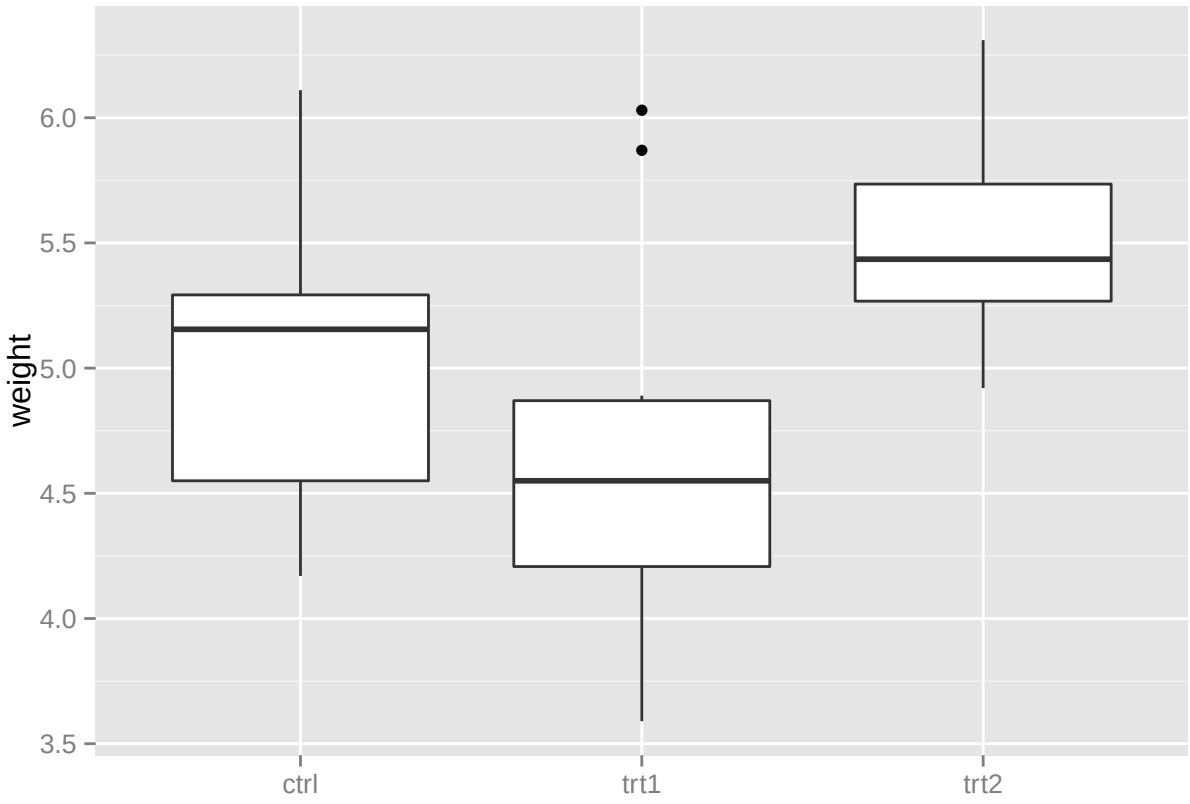



```
# Set the axis labels
hwp +
  xlab("Age\nin years") +
  ylab("Height in inches")
```



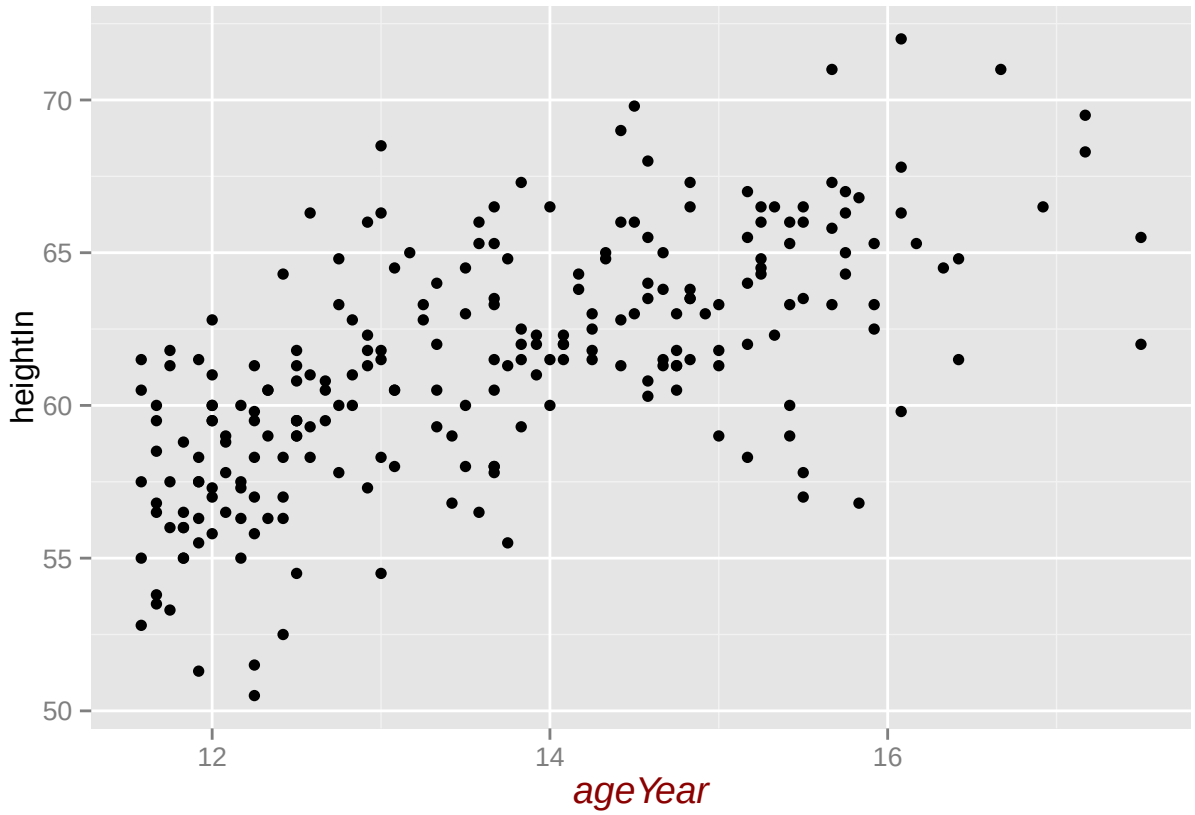
11. Removing Axis Labels

```
p <- ggplot( PlantGrowth, aes( x = group, y = weight)) +
  geom_boxplot()

# We'll hide the x-axis
p +
  theme( axis.title.x = element_blank())
```

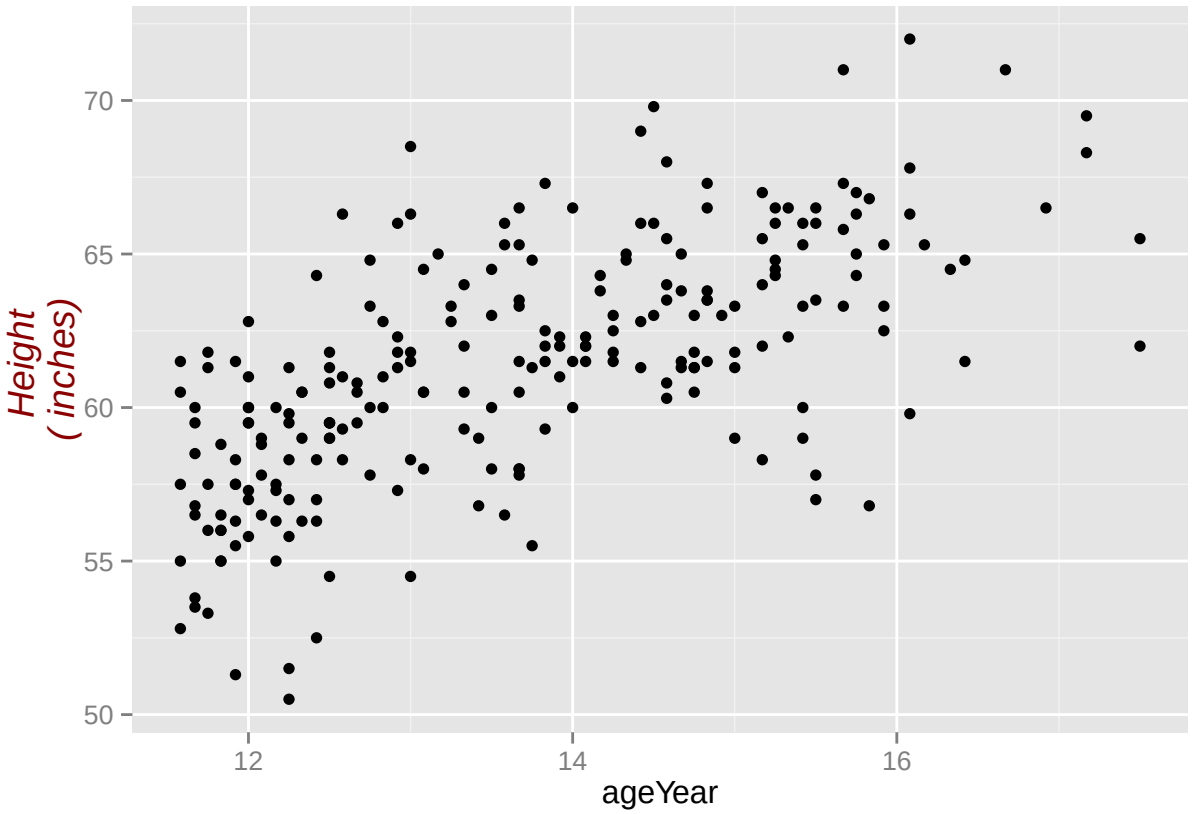


12. Changing the Appearance of Axis Labels

```
# To change the appearance of the x-axis use axis.title.x
hwp <- ggplot( heightweight, aes( x = ageYear, y = heightIn)) +
  geom_point()

hwp +
  theme( axis.title.x = element_text( face ="italic", colour ="darkred", size = 14))
```

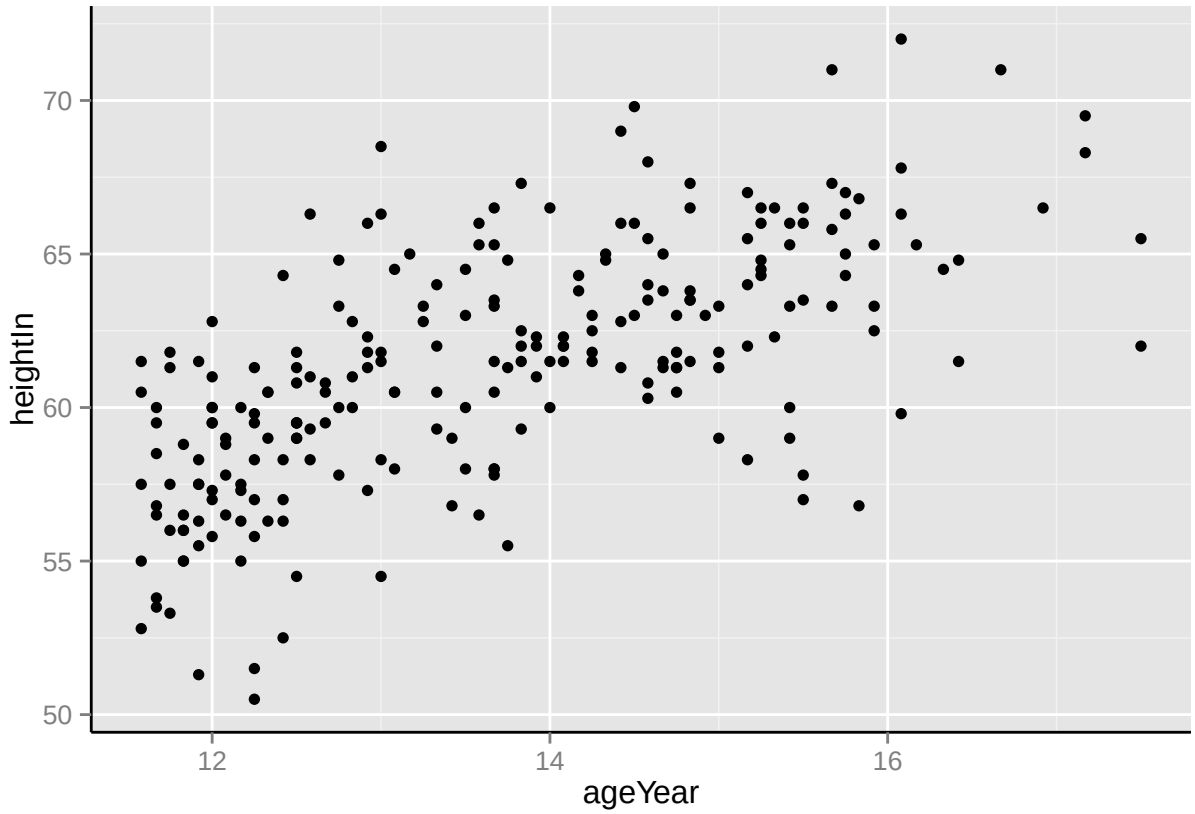



```
# y axis
hwp +
  ylab(" Height\n( inches)") +
  theme( axis.title.y = element_text( angle = 90, face ="italic", colour ="darkred", size = 14))
```

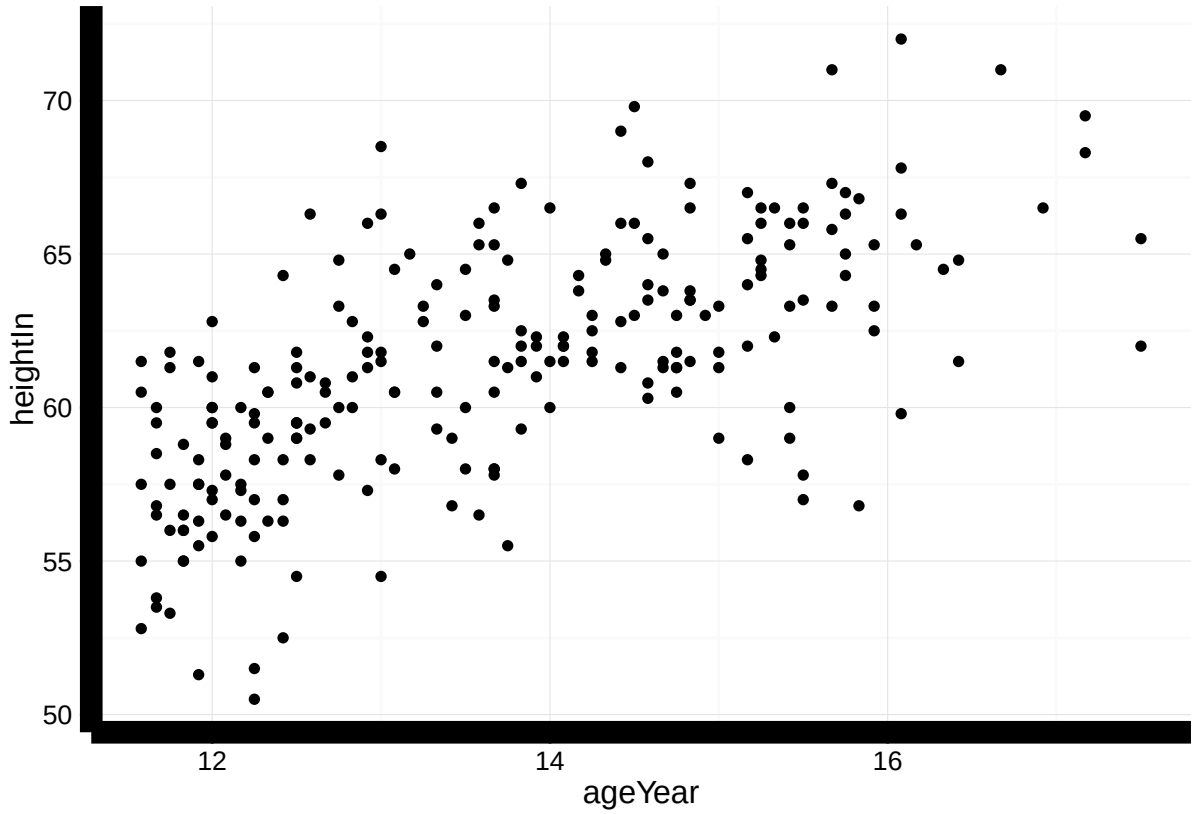


13. Showing Lines Along the Axes

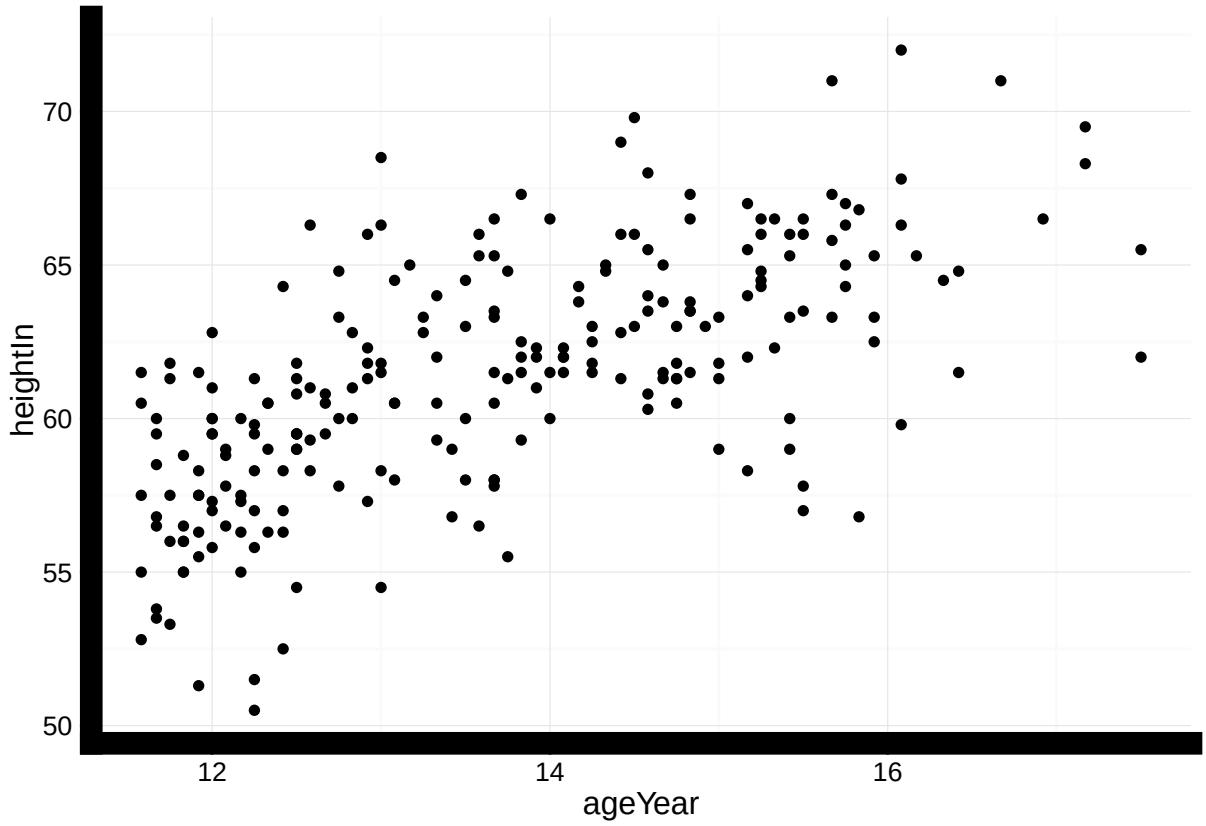
```
# You want to display lines along the x- and y-axes, but not on the other sides of the graph.
p <- ggplot( heightweight, aes( x = ageYear, y = heightIn)) +
  geom_point()

p + theme( axis.line = element_line( colour ="black"))
```




```
# With thick lines, only half overlaps
p +
  theme_bw() +
  theme( panel.border = element_blank(), axis.line = element_line( colour ="black", size = 4))
```

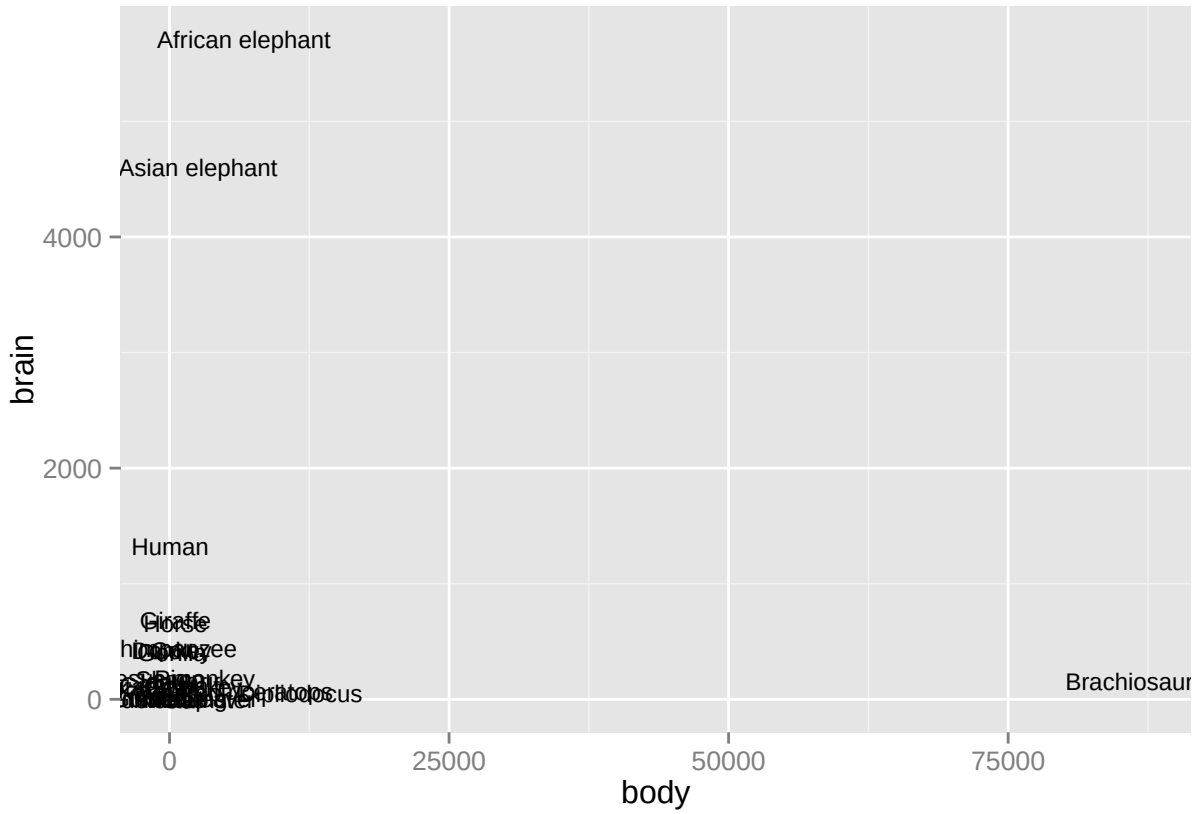



```
# Full overlap
p +
  theme_bw() +
  theme( panel.border = element_blank(), axis.line = element_line( colour ="black", size = 4, lineend =
```

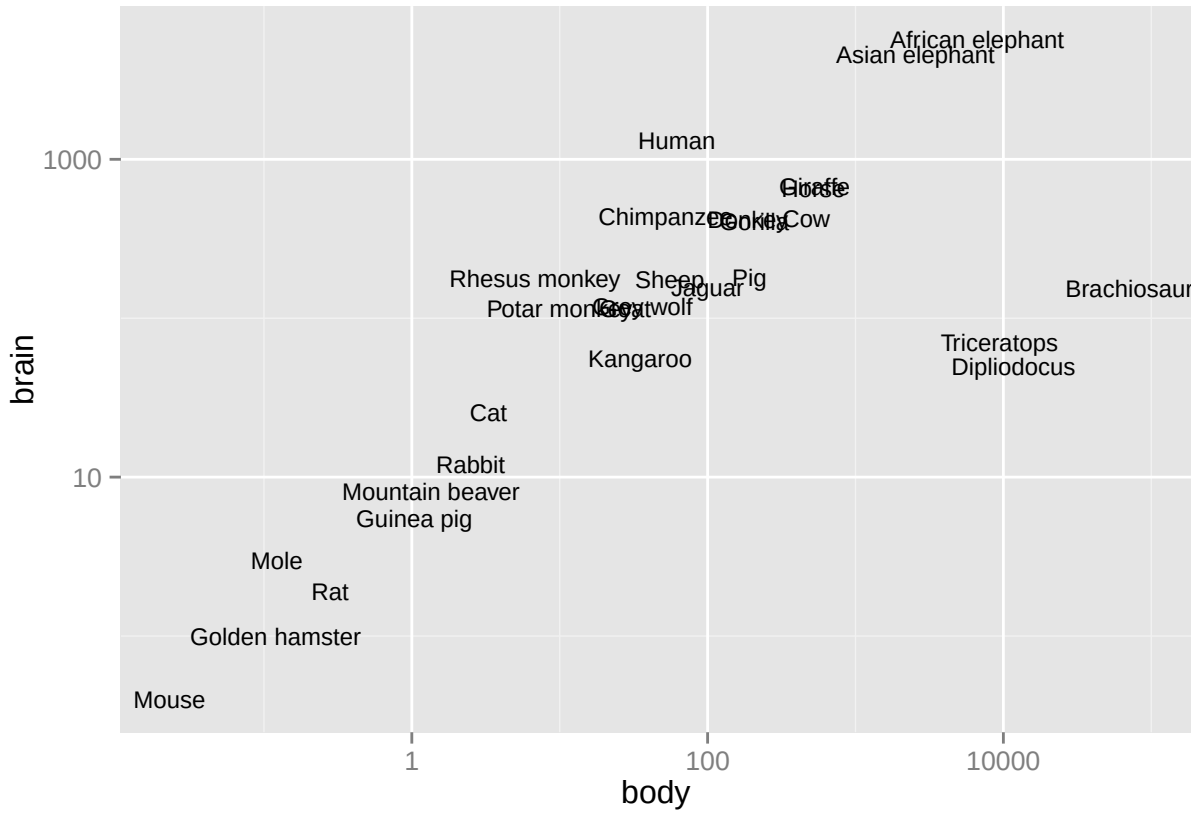


14. Using a Logarithmic Axis

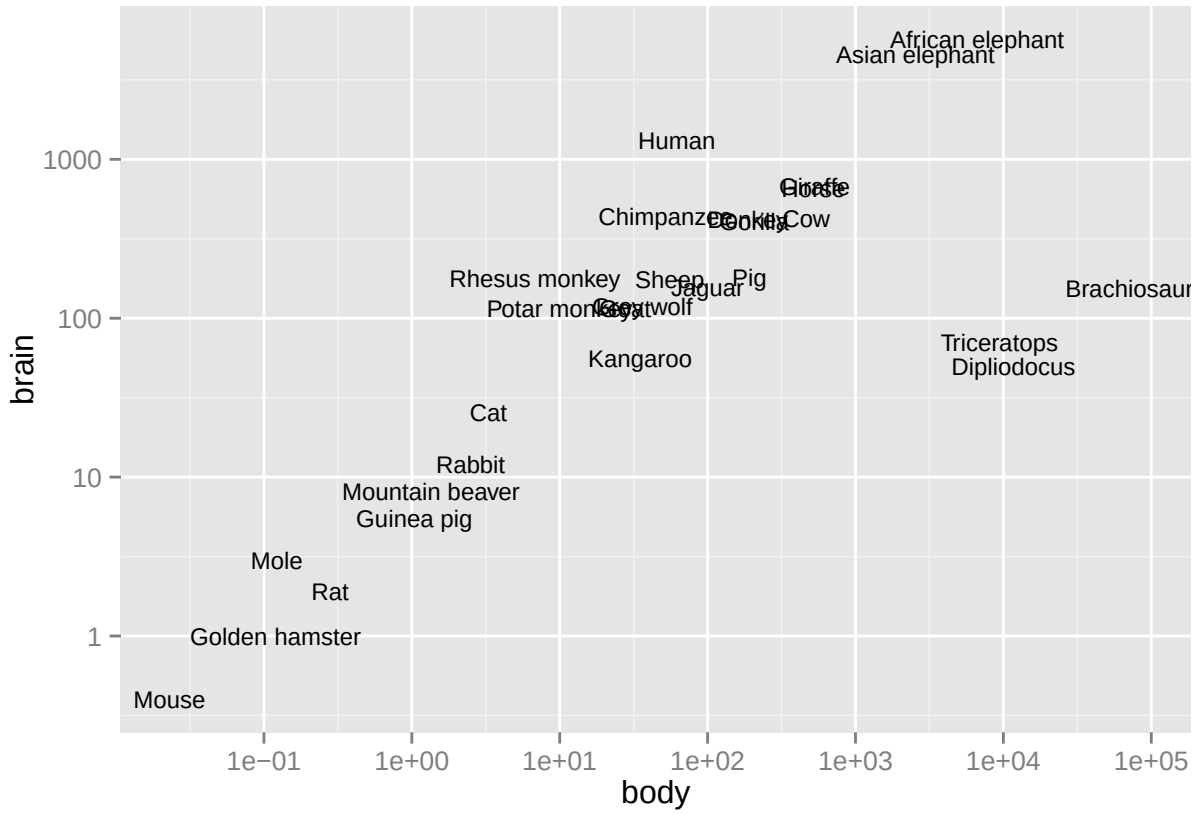
```
library( MASS)

# The base plot
p <- ggplot( Animals, aes( x = body, y = brain, label = rownames( Animals))) +
  geom_text( size = 3)

p
```



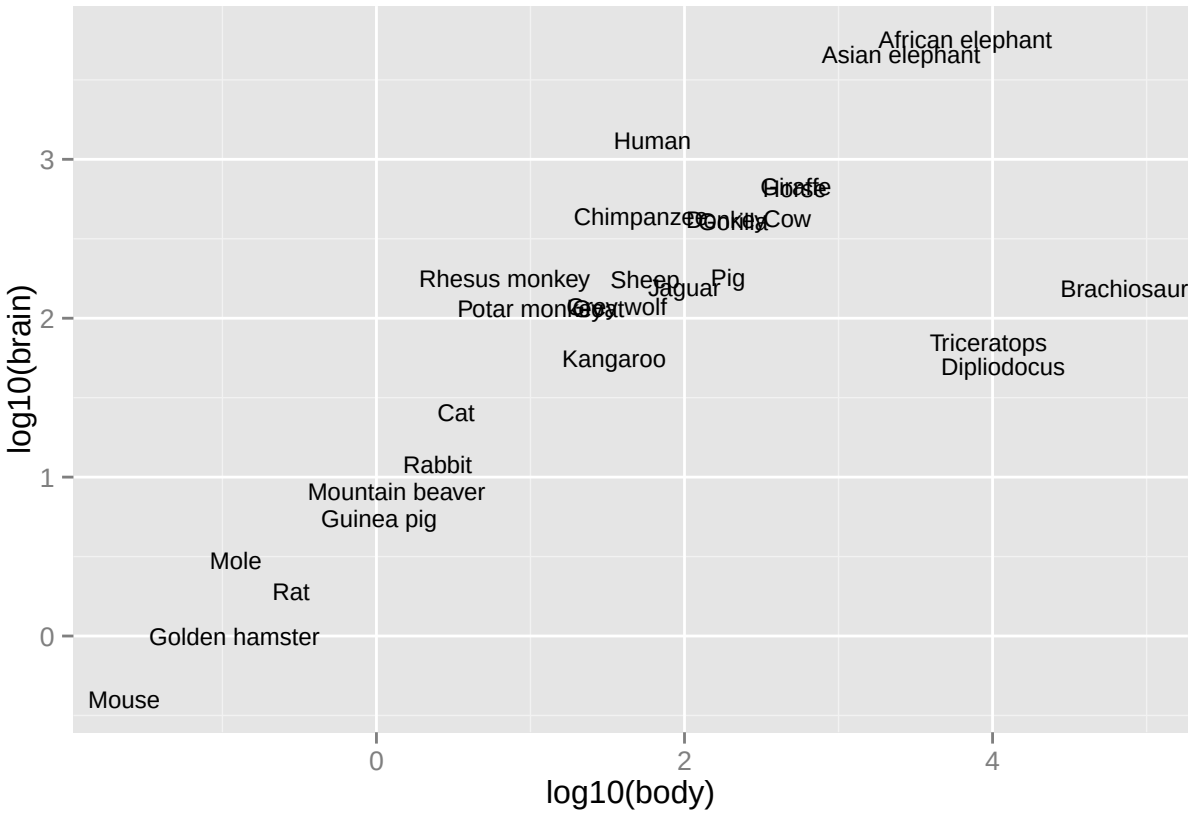

```
# With logarithmic x and y scales
p +
  scale_x_log10() +
  scale_y_log10()
```




```
# breaks
p +
  scale_x_log10( breaks = 10^(-1:5)) + scale_y_log10( breaks = 10^(0:3))
```

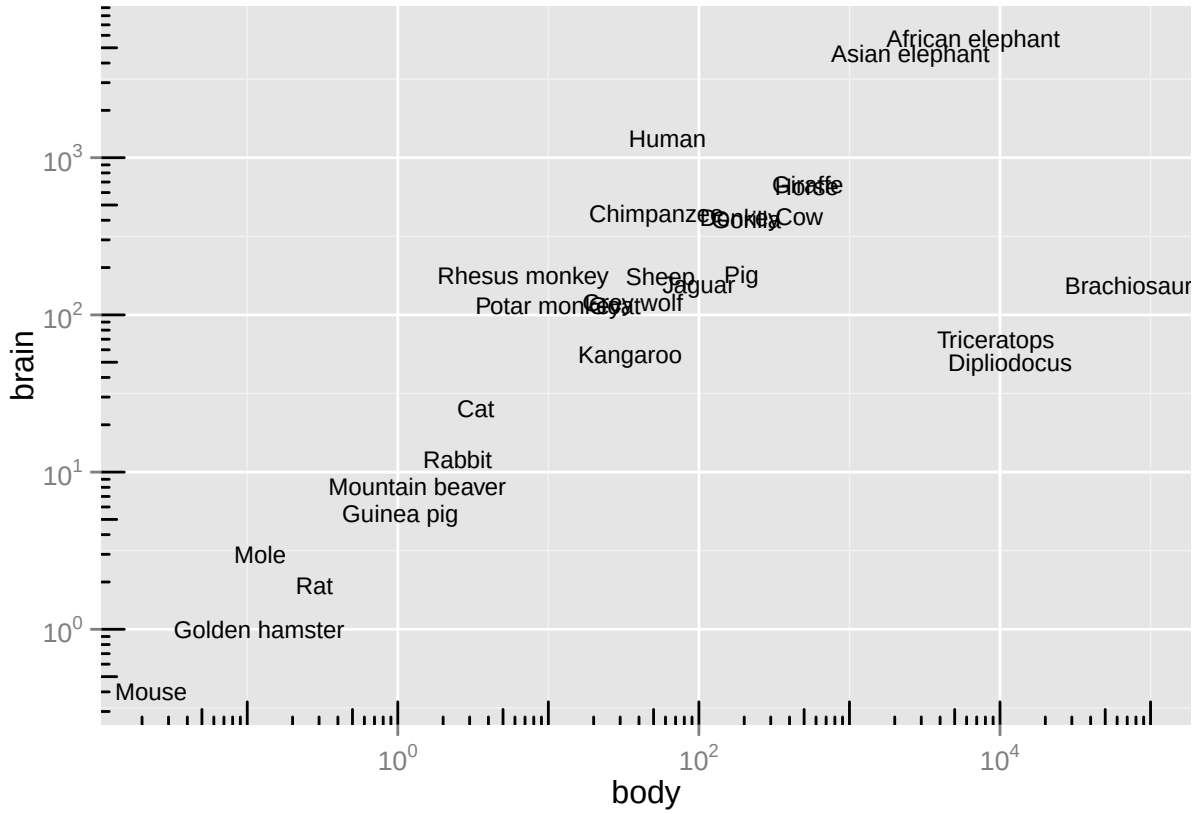



```
# Another way to use log axes is to transform the data before mapping it to the x and y coordinates
ggplot( Animals, aes( x = log10( body), y = log10( brain), label = rownames( Animals))) +
  geom_text( size = 3)
```

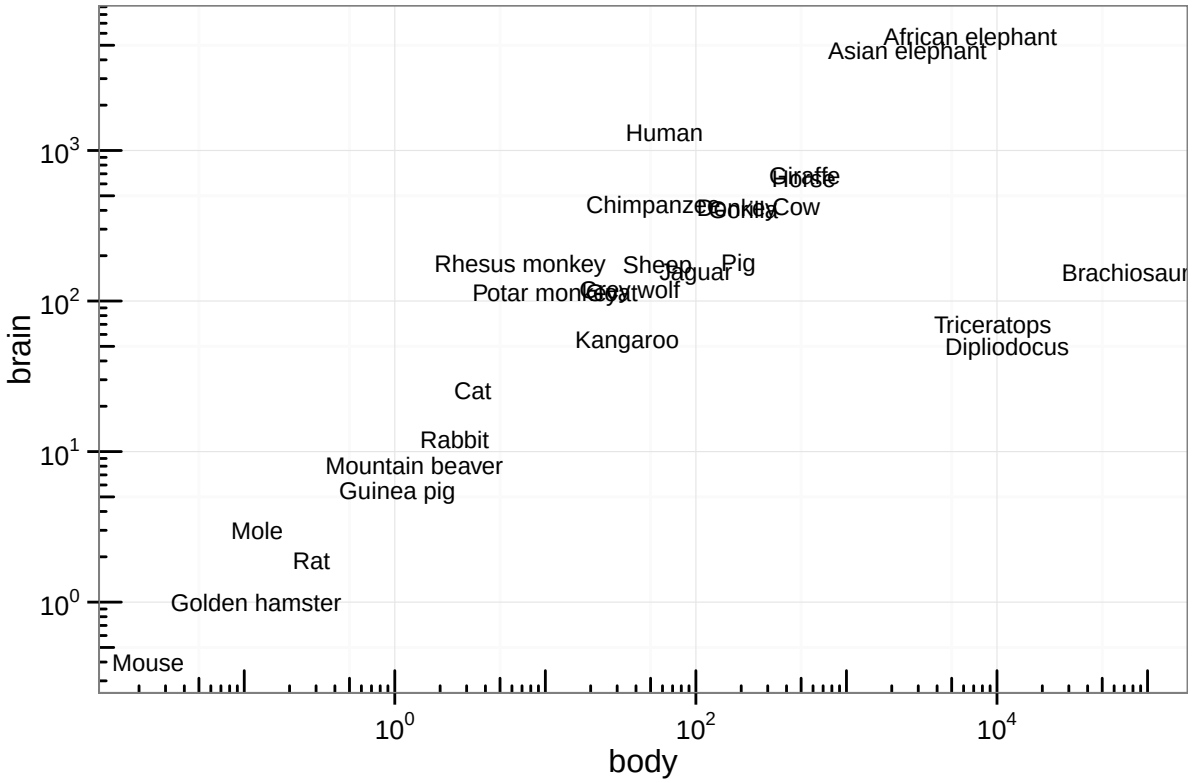


15. Adding Ticks for a Logarithmic Axis

```
# Use annotation_logticks()
library( MASS) # For the data set
library( scales) # For the trans and format functions

ggplot( Animals, aes( x = body, y = brain, label = rownames( Animals))) +
  geom_text( size = 3) + annotation_logticks() +
  scale_x_log10( breaks = trans_breaks("log10", function(x) 10^x), labels = trans_format("log10", math_
  scale_y_log10( breaks = trans_breaks("log10", function(x) 10^x), labels = trans_format("log10", math_
```

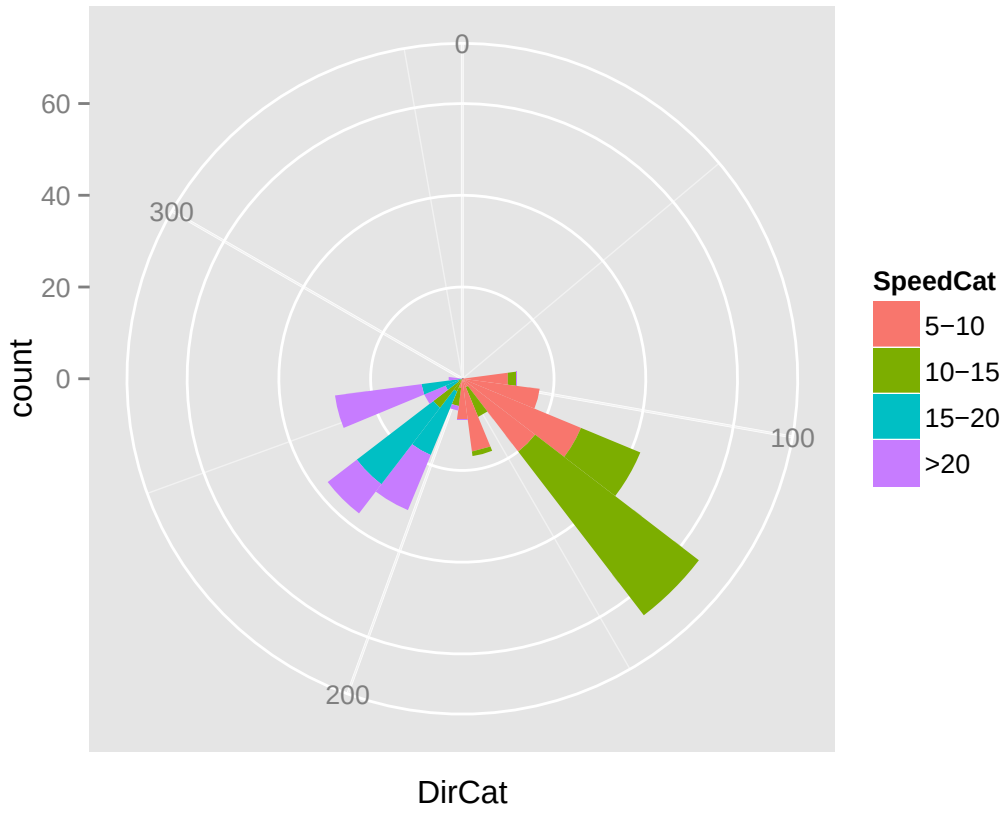



```
# To get the colors of the tick marks and the grid lines to match up a bit better, you can use theme_bw
ggplot( Animals, aes( x = body, y = brain, label = rownames( Animals))) +
  geom_text( size = 3) +
  annotation_logticks() +
  scale_x_log10( breaks = trans_breaks("log10", function(x) 10^x),
                 labels = trans_format("log10", math_format(10^.x)),
                 minor_breaks = log10( 5) + -2: 5) +
  scale_y_log10( breaks = trans_breaks("log10", function(x) 10^x),
                 labels = trans_format("log10", math_format(10^.x)),
                 minor_breaks = log10(5) + -1:3) +
  coord_fixed() +
  theme_bw()
```

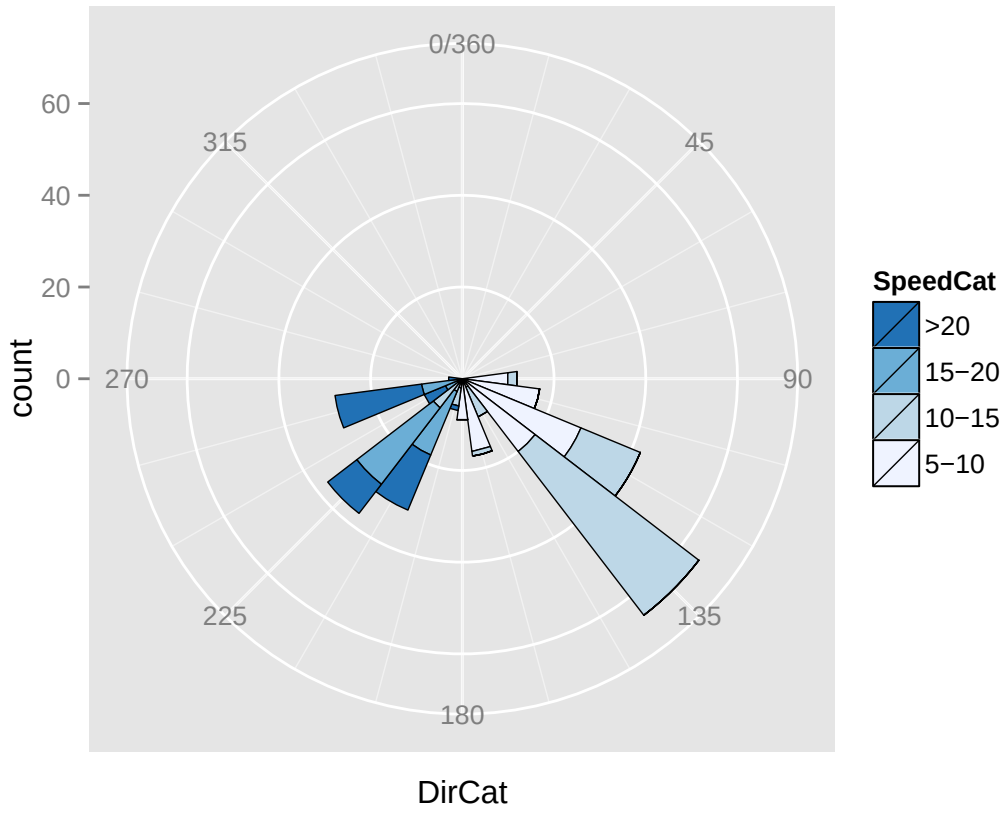


15. Making a Circular Graph

```
ggplot( wind, aes( x = DirCat, fill = SpeedCat)) +
  geom_histogram( binwidth = 15, origin = -7.5) +
  coord_polar() +
  scale_x_continuous( limits = c( 0,360))
```



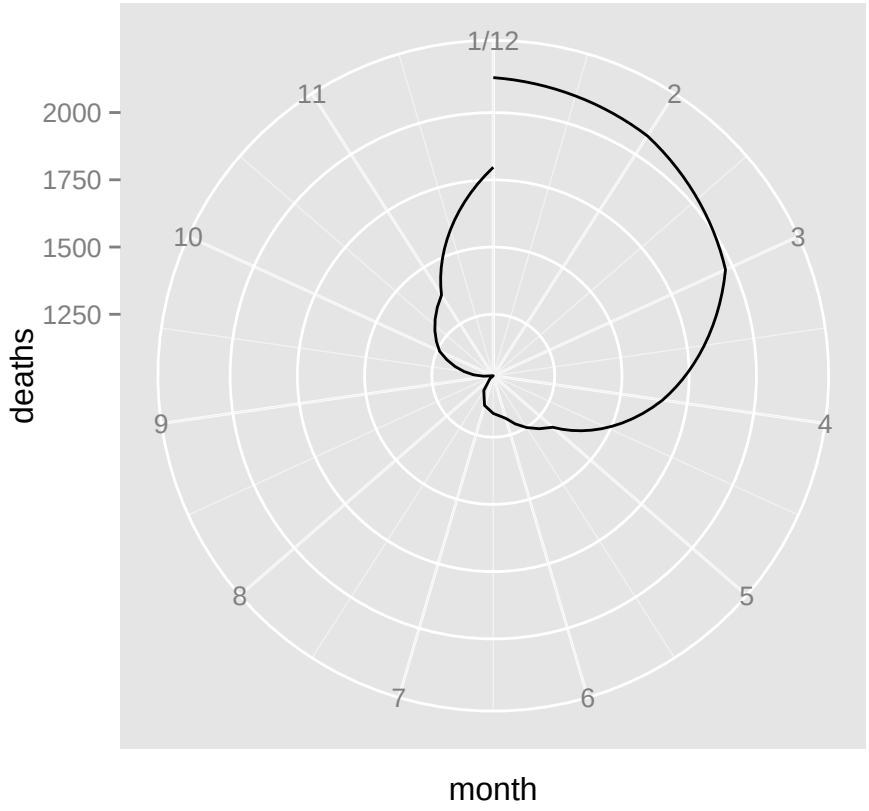

```
# we can make the plot a little prettier by reversing the legend, using a different palette, adding an
ggplot( wind, aes( x = DirCat, fill = SpeedCat)) +
  geom_histogram( binwidth = 15, origin =-7.5, colour ="black", size =.25) +
  guides( fill = guide_legend( reverse = TRUE)) +
  coord_polar() +
  scale_x_continuous( limits = c( 0,360), breaks = seq( 0, 360, by = 45), minor_breaks = seq( 0, 360, by
  scale_fill_brewer()
```



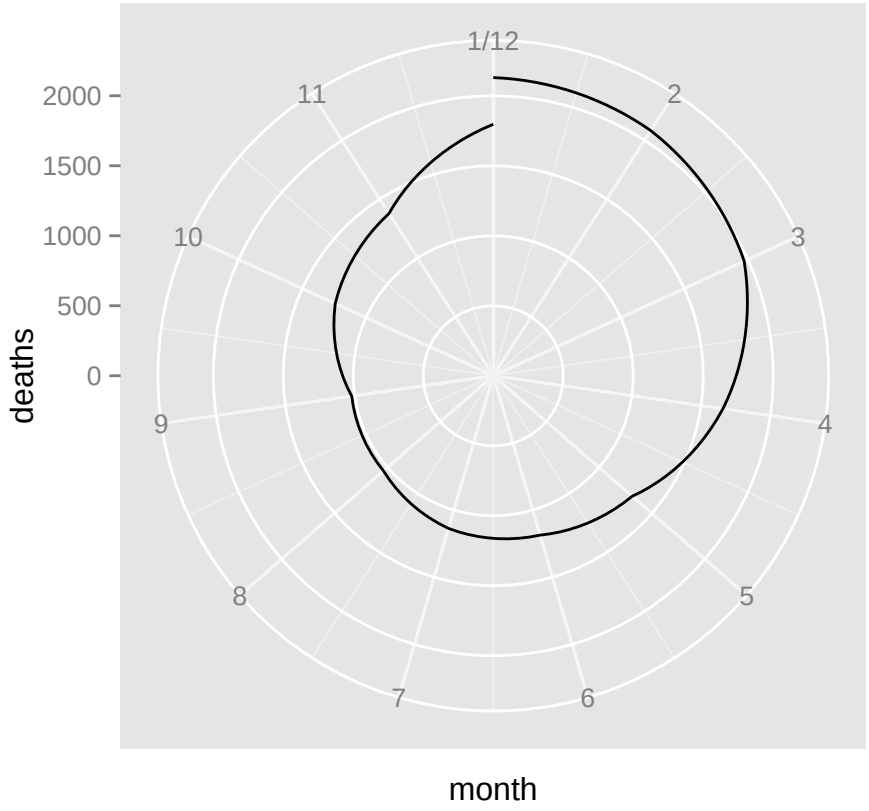

```
# Put mdeaths time series data into a data frame
md <- data.frame( deaths = as.numeric( mdeaths),
                  month = as.numeric( cycle( mdeaths)))

# Calculate average number of deaths in each month
library( plyr) # For the ddply() function
md <- ddply( md, "month", summarise, deaths = mean( deaths))
str(md)
```

```
## 'data.frame':    12 obs. of  2 variables:
##  $ month : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ deaths: num  2130 2081 1970 1657 1314 ...
```

```
# Make the base plot
p <- ggplot( md, aes( x = month, y = deaths)) +
  geom_line() + scale_x_continuous( breaks = 1:12)

# With coord_polar
p +
  coord_polar()
```



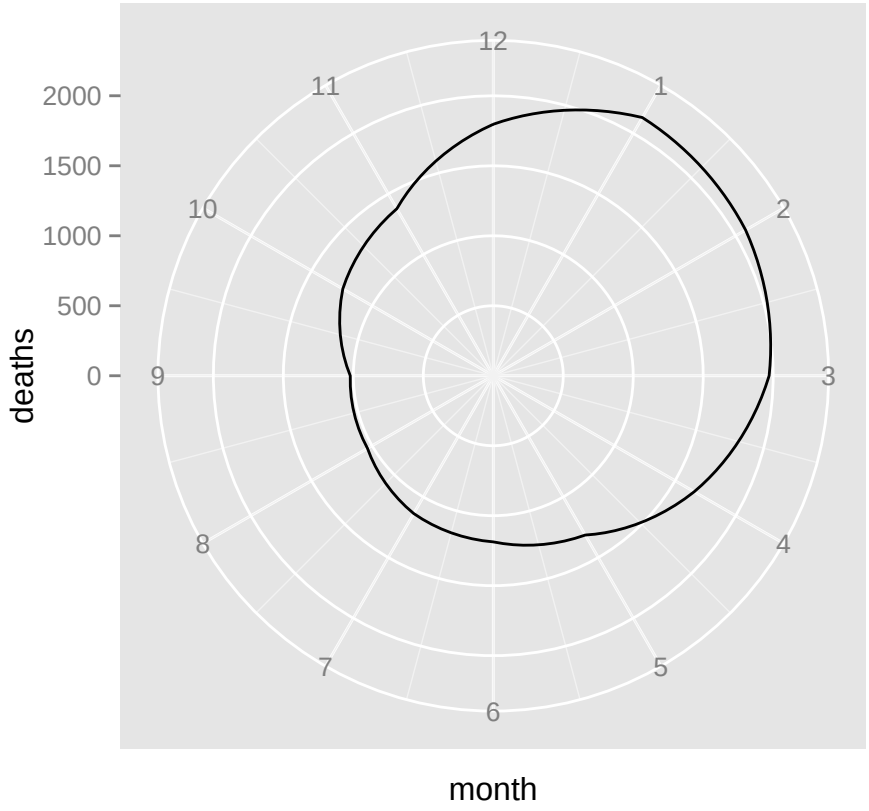

```
# With coord_polar and y (r) limits going to zero
p +
  coord_polar() + ylim( 0, max( md$deaths))
```




```
# also xlim
p +
  coord_polar() +
  ylim( 0, max( md$deaths)) +
  xlim( 0, 12)
```

```
## Scale for 'x' is already present. Adding another scale for 'x', which will replace the existing scal
```


```
# Connect the lines by adding a value for 0 that is the same as 12
mdx <- md[ md$month == 12, ]
mdx $ month <- 0
mdnew <- rbind( mdx, md)

# Make the same plot as before, but with the new data, by using %+%
p %+% mdnew +
  coord_polar() +
  ylim( 0, max( md$deaths))
```



15. Using Dates on an Axis

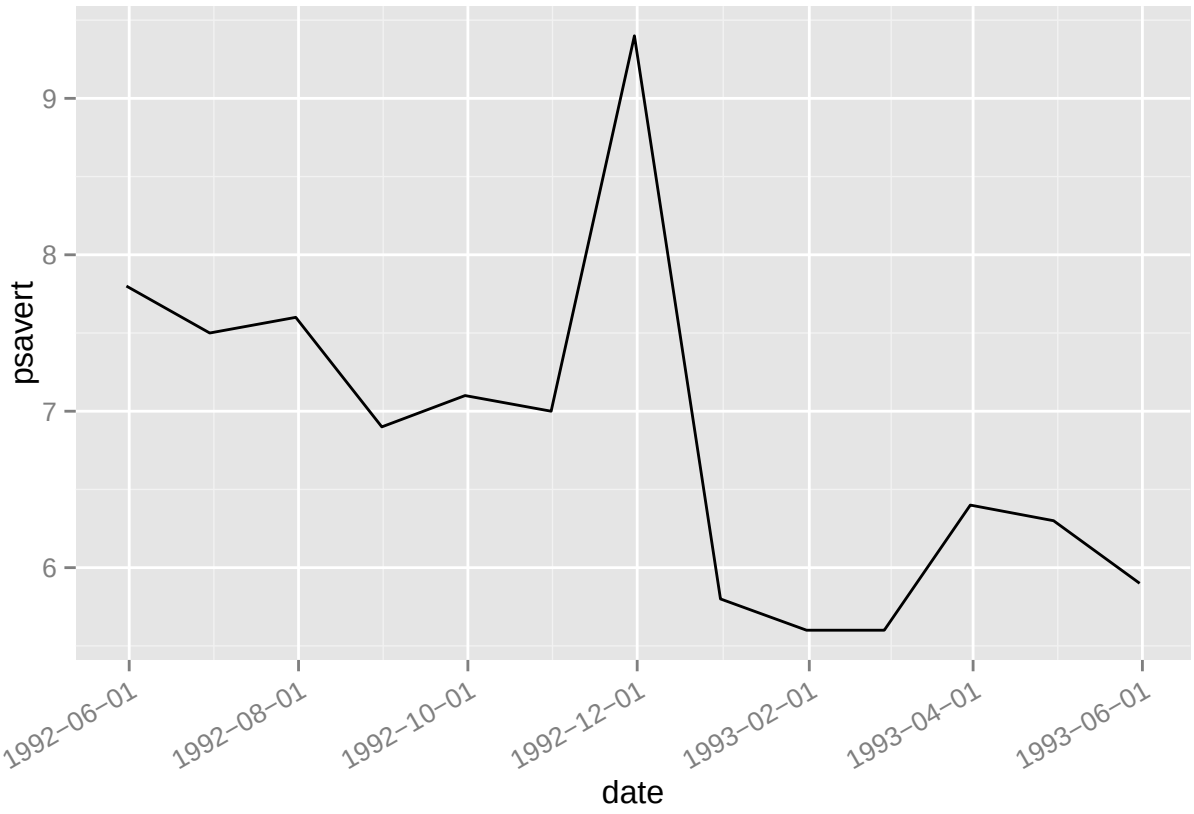
```
# Look at the structure
str( economics)
```

```
## 'data.frame':    478 obs. of  6 variables:
##  $ date    : Date, format: "1967-06-30" "1967-07-31" ...
##  $ pce     : num  508 511 517 513 518 ...
##  $ pop     : int  198712 198911 199113 199311 199498 199657 199808 199920 200056 200208 ...
##  $ psavert : num  9.8 9.8 9 9.8 9.7 9.4 9 9.5 8.9 9.6 ...
##  $ uempmed : num  4.5 4.7 4.6 4.9 4.7 4.8 5.1 4.5 4.1 4.6 ...
##  $ unemploy: int  2944 2945 2958 3143 3066 3018 2878 3001 2877 2709 ...
```

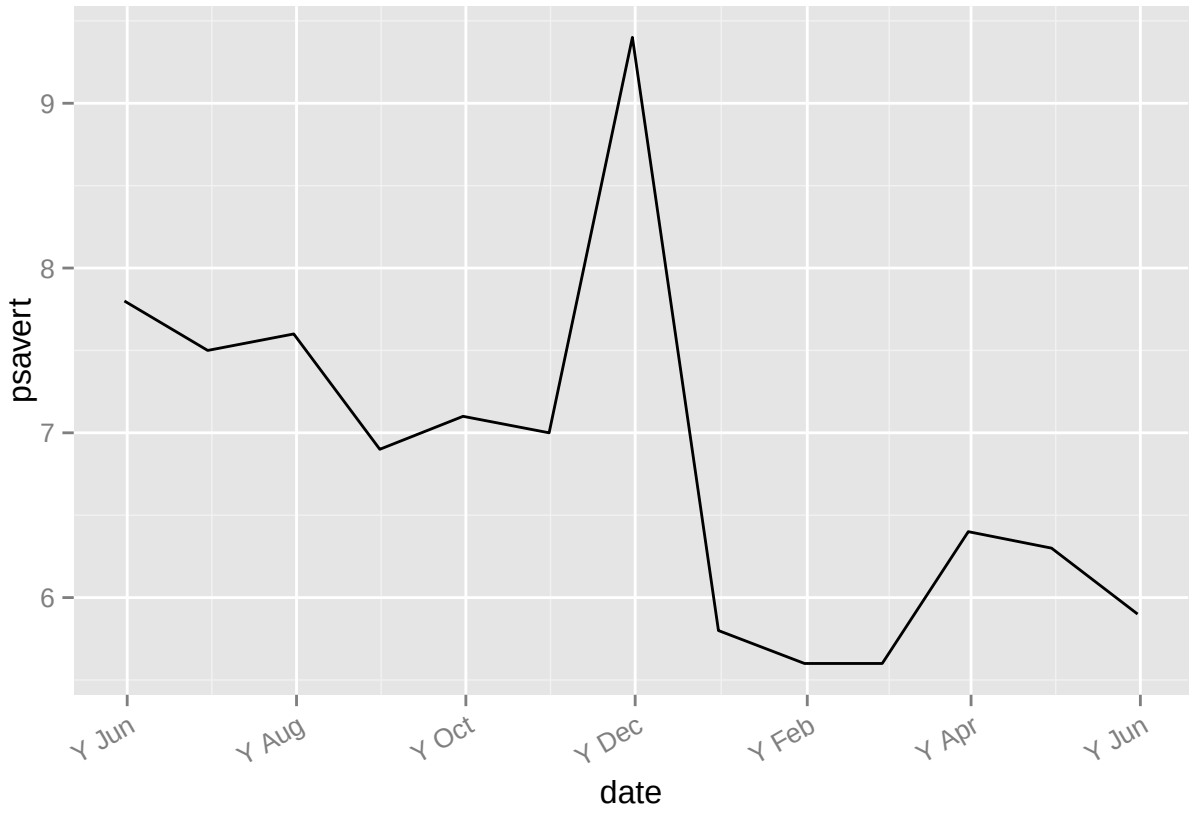
```
ggplot( economics, aes( x = date, y = psavert)) +
  geom_line()
```


```
# Take a subset of economics
econ <- subset( economics, date >= as.Date("1992-05-01") & date < as.Date("1993-06-01"))

# Base plot - without specifying breaks
p <- ggplot( econ, aes( x = date, y = psavert)) +
  geom_line()

p
```


```
# Specify breaks as a Date vector
datebreaks <- seq( as.Date("1992-06-01"), as.Date("1993-06-01"), by ="2 month")

# Use breaks, and rotate text labels
p +
  scale_x_date( breaks = datebreaks) +
  theme( axis.text.x = element_text( angle = 30, hjust = 1))
```

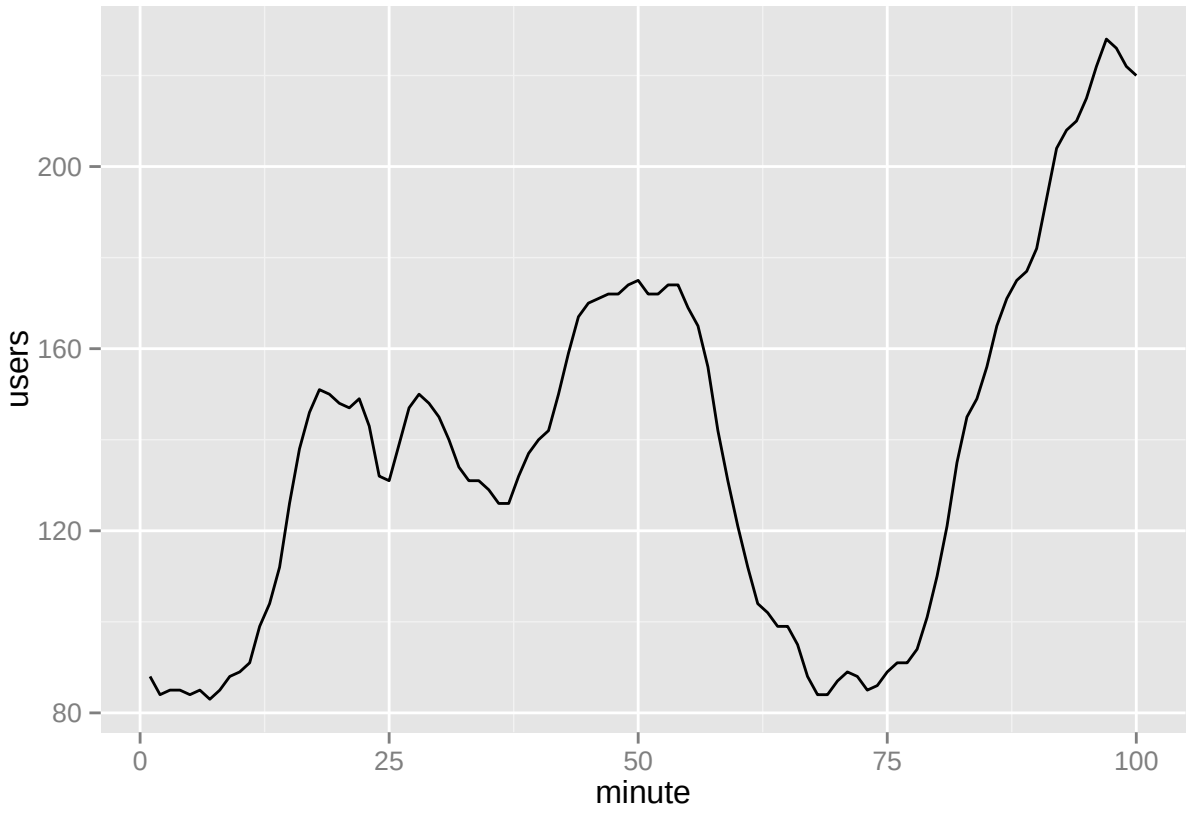



```
# Line graph with date format specified
library( scales)
p +
  scale_x_date( breaks = datebreaks, labels = date_format("% Y %b")) +
  theme( axis.text.x = element_text( angle = 30, hjust = 1))
```

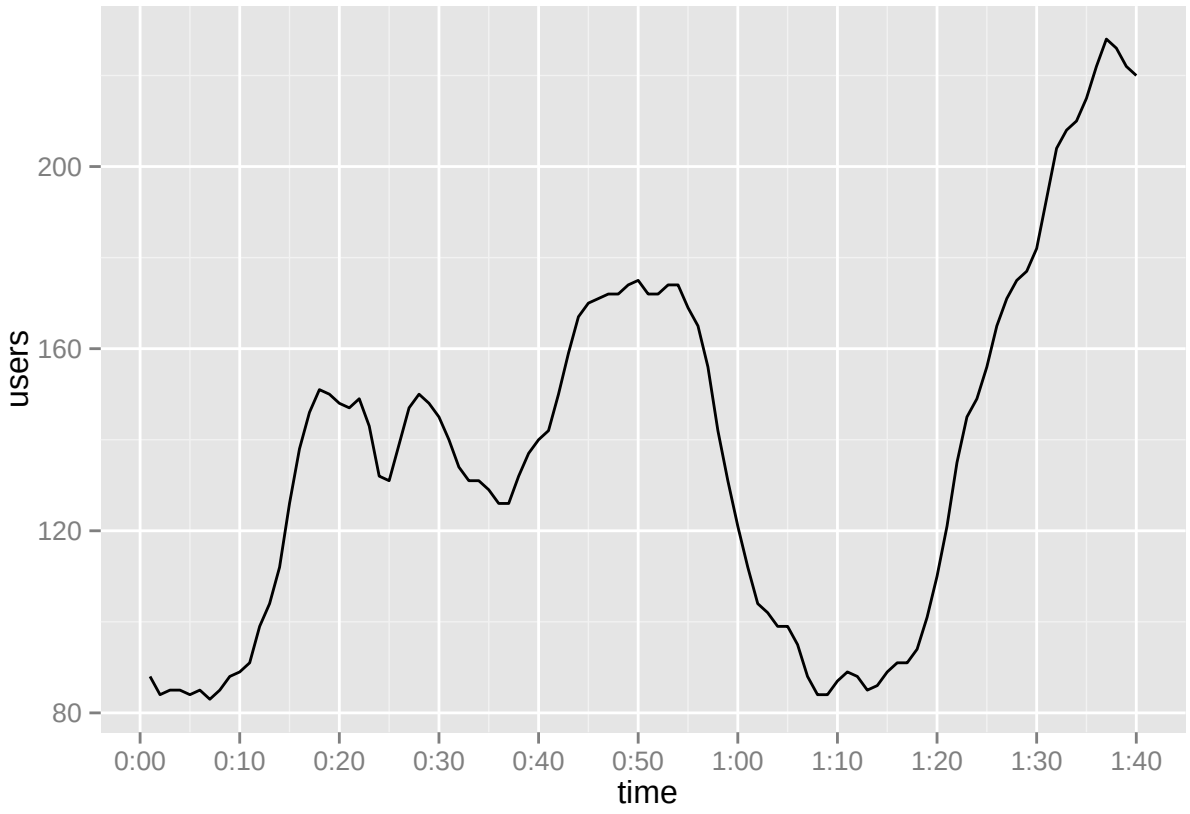


16. Using Relative Times on an Axis

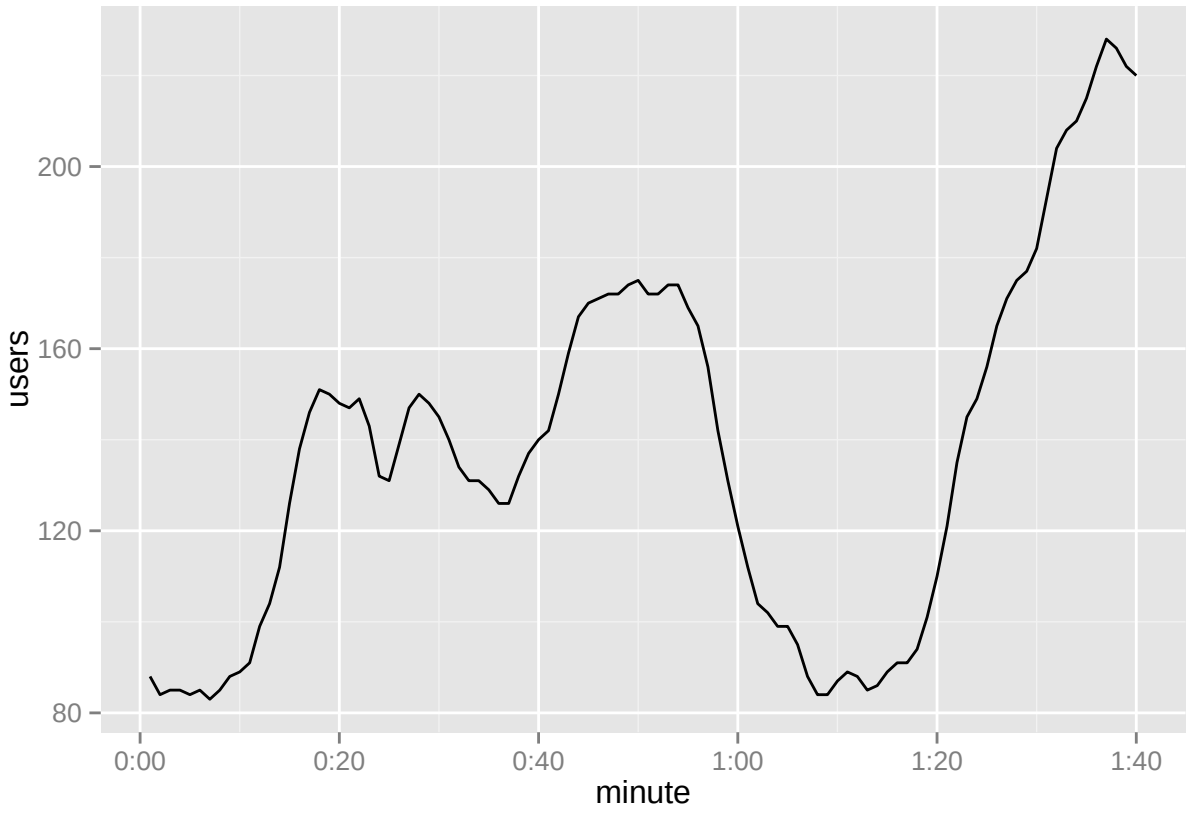
```
# Convert WWWusage time-series object to data frame
www <- data.frame( minute = as.numeric( time( WWWusage)), users = as.numeric( WWWusage))
# Define a formatter function - converts time in minutes to a string
timeHM_formatter <- function( x) {
  h <- floor( x/ 60)
  m <- floor( x %% 60)
  lab <- sprintf("%d:%02d", h, m)
  # Format the strings as HH:MM
  return( lab)
  }

# Default x axis
ggplot( www, aes( x = minute, y = users)) +
  geom_line()
```




```
# With formatted times
ggplot( www, aes( x = minute, y = users)) +
  geom_line() +
  scale_x_continuous( name =" time", breaks = seq( 0, 100, by = 10), labels = timeHM_formatter)
```




```
# To convert to HH:MM:SS format, you can use the following formatter function
timeHMS_formatter <- function(x) {
  h <- floor( x/ 3600)
  m <- floor(( x/ 60) %% 60)
  s <- round( x %% 60)

  # Round to nearest second
  lab <- sprintf("%02d:%02d:%02d", h, m, s)
  # Format the strings as HH:MM:SS
  lab <- sub("^00:", "", lab)
  # Remove leading 00: if present
  lab <- sub("^0", "", lab)
  # Remove leading 0 if present
  return( lab)
  }

ggplot( www, aes( x = minute, y = users)) +
  geom_line() +
  scale_x_continuous( breaks = c( 0, 20, 40, 60, 80, 100), labels = c("0:00", "0:20", "0:40", "1:00", "
```


users
200
160
120
80
0:00
0:20
0:40
1:00
1:20
1:40
minute