Visualizing regression using caterpillar plots

Matthew Castelo

June, 2021

Caterpillar plots can be an effective way to present regression parameters or estimates from multiple sensitivity analyses. This tutorial will demonstrate how to create these plots using **ggplot2** when regression is performed within R, or in another statistical software.

We will need to install the following packages.

```
install.packages("tidyverse")
```

As usual, we will load our packages and set the working directory.

```
library(tidyverse)
library(ggsci)
library(survival)
library(broom)

#The working directory saves the following path when loading files, saving plots, etc
setwd("C:/Users/matth/Documents/R/R code and education/Tutorials/")
```

We will be working with a simulated dataset representing 80 generic cancer patients. The dataset can be loaded directly from GitHub so all the subsequent code will run on your computer.

```
urlRemote  <- "https://raw.githubusercontent.com/"
pathGithub <- "mcas-surg/Tutorials/main/Datasets/"
fileName   <- "generic_cancer.csv"

cancer <- read_csv(paste0(urlRemote, pathGithub, fileName))
```

Suppose we were interested in performing a multivariable Cox proportional hazards model, including 5 covariates: sex, patient age, tumour grade, tumour size, and treating hospital. The **survival** package requires our vital status variable to be binary. We will do this and save our new dataset as *cancer_surv*.

```
cancer_surv <- cancer %>%
  mutate(vital_status = ifelse(vital_status == "Died",
                               1, 0))
```

Now we can create the model. On the left side of the formula we pass variables indicating follow-up time and vital status. On the right side we pass our covariates of interest.

```
cox_model <- coxph(Surv(follow_up, vital_status) ~ male + age + high_grade +
                     tumour_size + hospital,
                   data = cancer_surv)

summary(cox_model)
```

```
## Call:
## coxph(formula = Surv(follow_up, vital_status) ~ male + age +
##     high_grade + tumour_size + hospital, data = cancer_surv)
##
##   n= 80, number of events= 50
##
##                               coef exp(coef) se(coef)      z Pr(>|z|)
## maleMale                   -0.25671   0.77359  0.29118 -0.882   0.3780
## age                         0.04105   1.04190  0.02360  1.740   0.0819 .
## high_gradeYes               0.36677   1.44307  0.38147  0.961   0.3363
## tumour_size                 0.11791   1.12515  0.15341  0.769   0.4421
## hospitalCommunity hospital  0.08044   1.08376  0.32124  0.250   0.8023
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##                            exp(coef) exp(-coef) lower .95 upper .95
## maleMale                      0.7736     1.2927    0.4372     1.369
## age                           1.0419     0.9598    0.9948     1.091
## high_gradeYes                 1.4431     0.6930    0.6832     3.048
## tumour_size                   1.1251     0.8888    0.8330     1.520
## hospitalCommunity hospital    1.0838     0.9227    0.5774     2.034
##
## Concordance= 0.629  (se = 0.044 )
## Likelihood ratio test= 7.99  on 5 df,   p=0.2
## Wald test            = 7.96  on 5 df,   p=0.2
## Score (logrank) test = 8.2  on 5 df,   p=0.1
```

Ideally, we wish to present the hazard ratios and confidence intervals graphically. These values are located within the model object *cox_model*. The **broom** package conveniently converts model objects into datasets than we can manipulate directly.

In this case, we need to exponentiate the estimates, and we would like 95% confidence intervals.

```
tidy_model <- tidy(cox_model,
                   exponentiate = TRUE,
                   conf.int = TRUE)

tidy_model
```

```
## # A tibble: 5 x 7
##   term                   estimate std.error statistic p.value conf.low conf.high
##   <chr>                     <dbl>     <dbl>     <dbl>   <dbl>    <dbl>     <dbl>
## 1 maleMale                  0.774    0.291     -0.882  0.378     0.437      1.37
## 2 age                       1.04     0.0236     1.74   0.0819    0.995      1.09
## 3 high_gradeYes             1.44     0.381      0.961  0.336     0.683      3.05
## 4 tumour_size               1.13     0.153      0.769  0.442     0.833      1.52
## 5 hospitalCommunity hos~    1.08     0.321      0.250  0.802     0.577      2.03
```

Let's clean up our small dataset. We will rename the values in the "term" column, and only keep our hazard ratios and 95% confidence intervals.

```
tidy_model$term <- c("Male sex",
                     "Older age (1 year)",
```

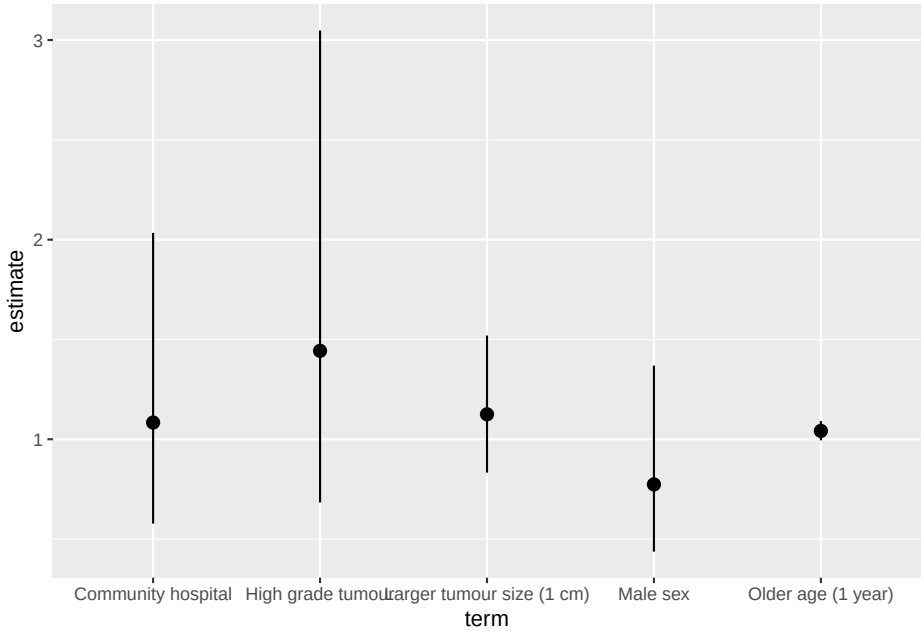
```
                      "High grade tumour",
                      "Larger tumour size (1 cm)",
                      "Community hospital")

tidy_model <- tidy_model %>%
  select(term, estimate, conf.low, conf.high)

tidy_model
```

```
## # A tibble: 5 x 4
##   term                        estimate conf.low conf.high
##   <chr>                          <dbl>    <dbl>     <dbl>
## 1 Male sex                       0.774    0.437      1.37
## 2 Older age (1 year)             1.04     0.995      1.09
## 3 High grade tumour              1.44     0.683      3.05
## 4 Larger tumour size (1 cm)      1.13     0.833      1.52
## 5 Community hospital             1.08     0.577      2.03
```

Now we can begin creating our plot. We will mainly use `geom_pointrange` and `geom_errorbar`, which together will demonstrate each hazard ratio and 95% CI.

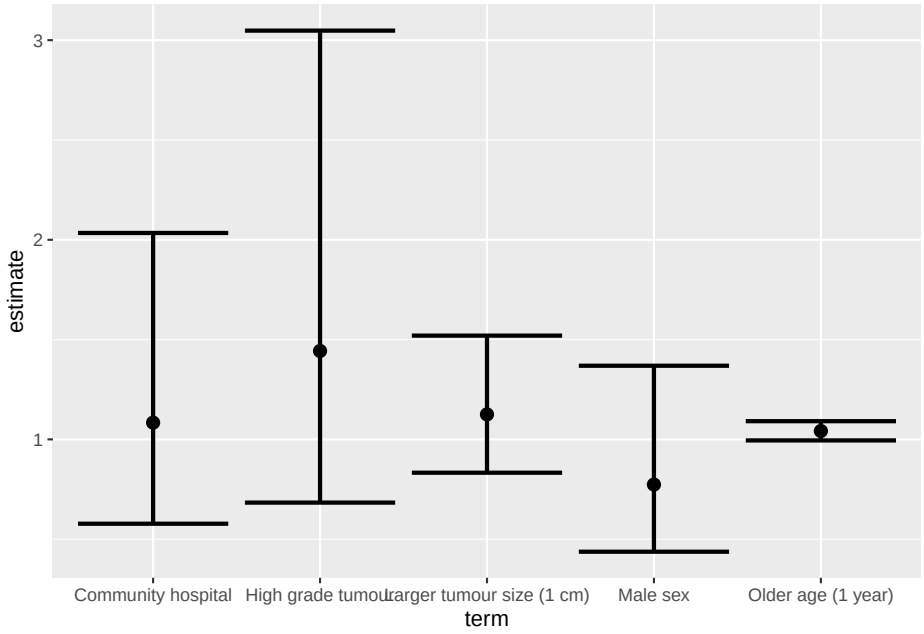
First, starting simply.

```
tidy_model %>%
  ggplot(aes(x = term,
             y = estimate, ymin = conf.low, ymax = conf.high))+
  geom_pointrange(size=0.5)
```



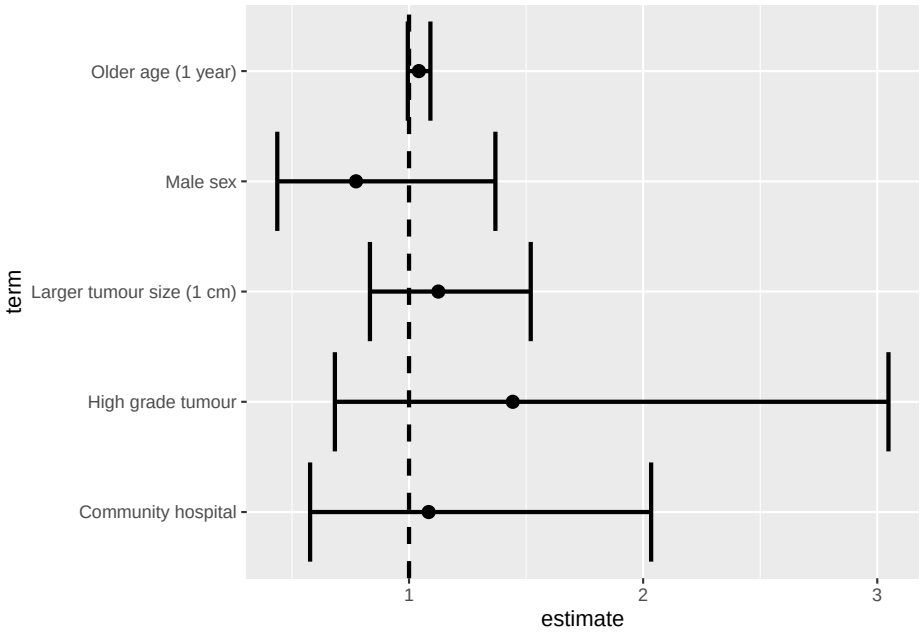
This is `geom_pointrange()` by itself. Here is how adding `geom_errorbar` changes the plot.

```
tidy_model %>%
  ggplot(aes(x = term,
             y = estimate, ymin = conf.low, ymax = conf.high))+
  geom_pointrange(size = 0.5)+
  geom_errorbar(size = 1)
```



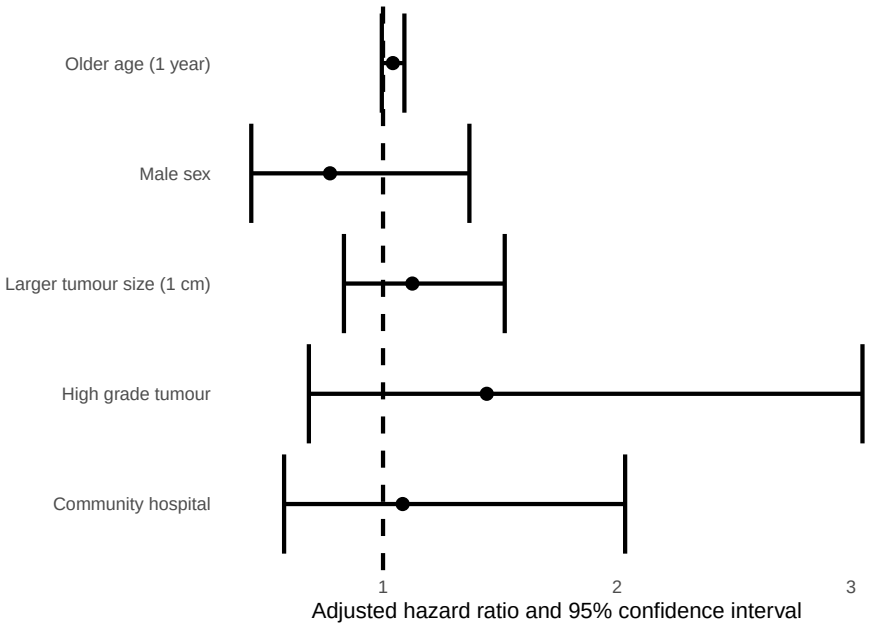
We should flip the plot vertically and add a line showing a HR = 1.

```
tidy_model %>%
  ggplot(aes(x = term,
             y = estimate, ymin = conf.low, ymax = conf.high))+
  geom_pointrange(size = 0.5)+
  geom_errorbar(size = 1)+
  geom_hline(yintercept = 1, linetype = 2, size = 1)+
  coord_flip()
```



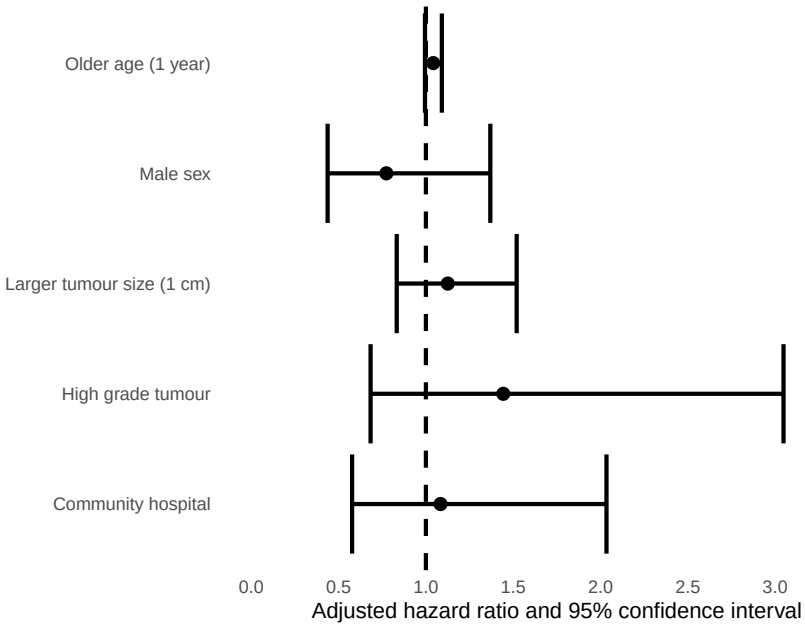
This is already looking better. Next, we can remove the y-axis label, change the x-axis label, and adjust the theme. Remember, when we flip figures the axes don't change in the code.

```
tidy_model %>%
  ggplot(aes(x = term,
             y = estimate, ymin = conf.low, ymax = conf.high))+
  geom_pointrange(size = 0.5)+
  geom_errorbar(size = 1)+
  geom_hline(yintercept = 1, linetype = 2, size = 1)+
  coord_flip()+
  labs(x = "",
       y = "Adjusted hazard ratio and 95% confidence interval")+
  theme_minimal()+
  theme(panel.grid = element_blank())
```



The figure would be improved by shifting the axis and breaks.

```
tidy_model %>%
  ggplot(aes(x = term,
             y = estimate, ymin = conf.low, ymax = conf.high))+
  geom_pointrange(size = 0.5)+
  geom_errorbar(size = 1)+
  geom_hline(yintercept = 1, linetype = 2, size = 1)+
  coord_flip()+
  labs(x = "",
       y = "Adjusted hazard ratio and 95% confidence interval")+
  theme_minimal()+
  theme(panel.grid = element_blank())+
  scale_y_continuous(limits = c(0, 3.5),
                     breaks = seq(0, 3, 0.5))
```



It is also possible to add labels to the figure indicating the hazard ratio. Text can be added dynamically as a layer using `+geom_text()`. This layer requires you to provide labels. We will create our custom labels for each covariate.

First, we need to round the hazard ratios and confidence limits to 2 decimal places, and we will save these as new variables.

```
tidy_model_labels <- tidy_model %>%
  mutate(estimate_round = round(estimate, 2),
         conf.low_round = round(conf.low, 2),
         conf.high_round = round(conf.high, 2))
```

Now we can start building our labels. This will be shown over several steps so you can follow what each piece of code is doing.

```
tidy_model_labels <- tidy_model_labels %>%
  unite("custom_label", estimate_round, conf.low_round,
        remove = TRUE,
        sep = ", 95% CI ")
```

```
## # A tibble: 5 x 1
##   custom_label
##   <chr>
## 1 0.77, 95% CI 0.44
## 2 1.04, 95% CI 0.99
## 3 1.44, 95% CI 0.68
## 4 1.13, 95% CI 0.83
## 5 1.08, 95% CI 0.58
```

The `unite()` function combines chosen columns in a character string, with a custom separator that you can specify. Here, we have combined the estimate and lower confidence limit with the string ", 95% CI"

between. This has been saved to the new variable "custom_label". The `remove = TRUE` argument will delete the original combined columns.

We will continue and add the upper confidence limit.

```
tidy_model_labels <- tidy_model_labels %>%
  unite("custom_label", custom_label, conf.high_round,
        remove = TRUE,
        sep = "-")
```

```
## # A tibble: 5 x 1
##   custom_label
##   <chr>
## 1 0.77, 95% CI 0.44-1.37
## 2 1.04, 95% CI 0.99-1.09
## 3 1.44, 95% CI 0.68-3.05
## 4 1.13, 95% CI 0.83-1.52
## 5 1.08, 95% CI 0.58-2.03
```

Finally, we can paste the string "HR" to the front of each label.

```
tidy_model_labels <- tidy_model_labels %>%
  mutate(custom_label = paste0("HR ", custom_label))
```

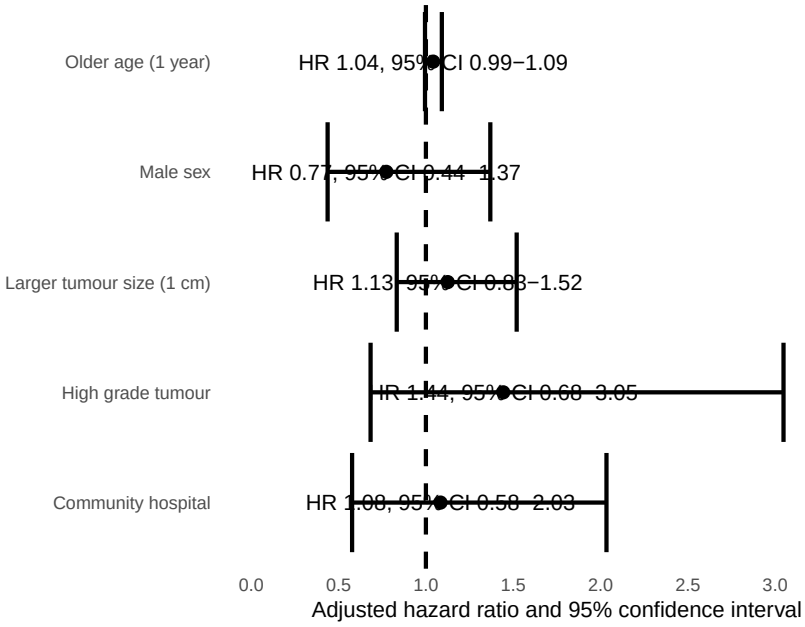
```
## # A tibble: 5 x 1
##   custom_label
##   <chr>
## 1 HR 0.77, 95% CI 0.44-1.37
## 2 HR 1.04, 95% CI 0.99-1.09
## 3 HR 1.44, 95% CI 0.68-3.05
## 4 HR 1.13, 95% CI 0.83-1.52
## 5 HR 1.08, 95% CI 0.58-2.03
```

In reality, all of these data steps could have been piped together to save time. This is what that would look like.

```
tidy_model_labels <- tidy_model %>%
  mutate(estimate_round = round(estimate, 2),
         conf.low_round = round(conf.low, 2),
         conf.high_round = round(conf.high, 2)) %>%
  unite("custom_label", estimate_round, conf.low_round,
        remove = TRUE,
        sep = ", 95% CI ") %>%
  unite("custom_label", custom_label, conf.high_round,
        remove = TRUE,
        sep = "-") %>%
  mutate(custom_label = paste0("HR ", custom_label))
```

Now, let's add these labels to our previous figure using `+geom_text()`. Note I am adding the labels aesthetic directly to `geom_text()`, but it could also have been added to the `ggplot()` statement at the top of the code.

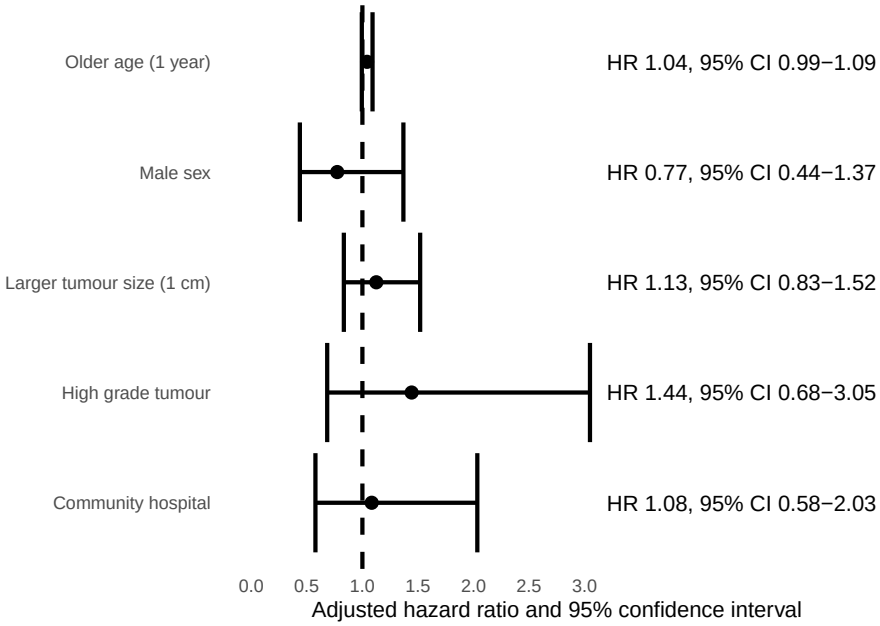
```
tidy_model_labels %>%
  ggplot(aes(x = term,
             y = estimate, ymin = conf.low, ymax = conf.high))+
  geom_pointrange(size = 0.5)+
  geom_errorbar(size = 1)+
  geom_hline(yintercept = 1, linetype = 2, size = 1)+
  coord_flip()+
  labs(x = "",
       y = "Adjusted hazard ratio and 95% confidence interval")+
  theme_minimal()+
  theme(panel.grid = element_blank())+
  scale_y_continuous(limits = c(0, 3.5),
                     breaks = seq(0, 3, 0.5))+
  geom_text(aes(label = custom_label))
```



Unfortunately, `geom_text()` inherited the x and y coordinates in the `ggplot()` statement and the result is not very attractive. We can manually set these values to shift the text. We will also adjust the x-axis limits and breaks to add more space to the right side of the plot without extending the x-axis text. Look at the following code carefully.

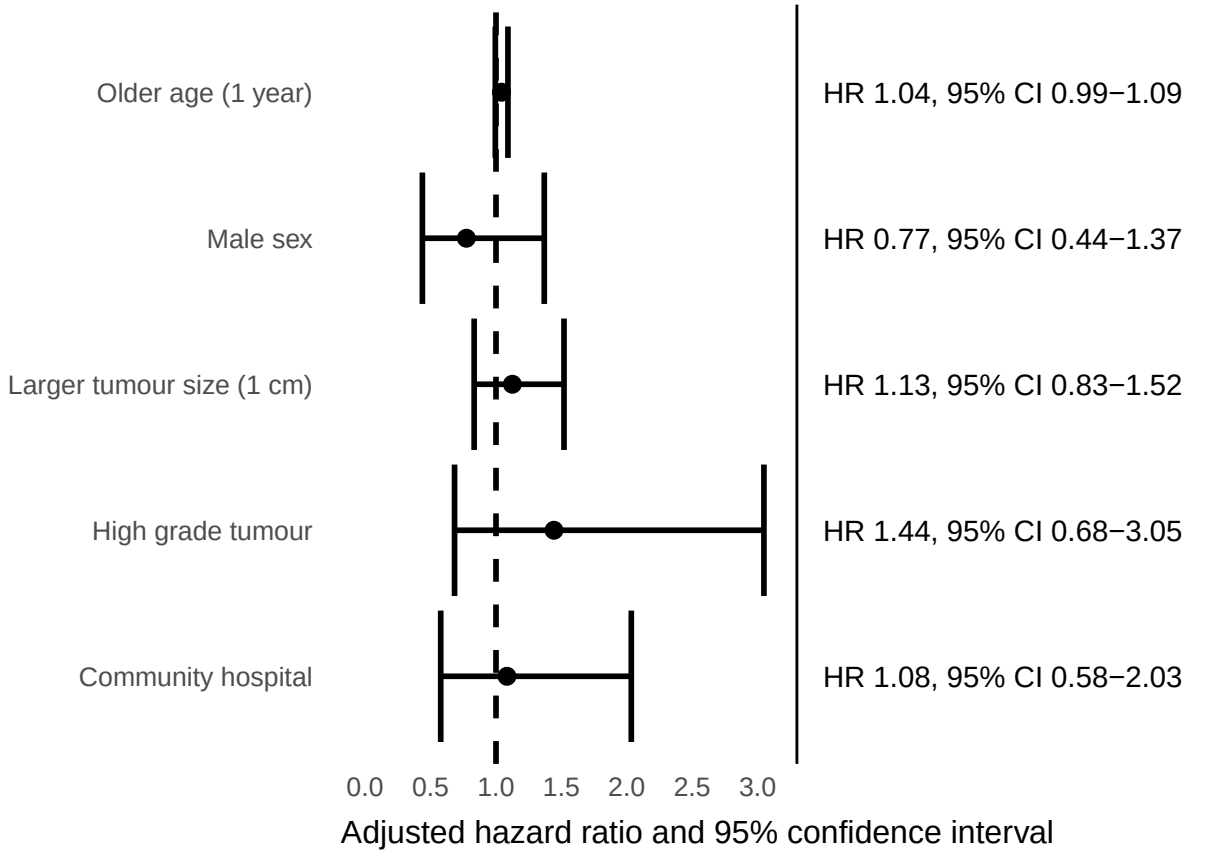
```
tidy_model_labels %>%
  ggplot(aes(x = term,
             y = estimate, ymin = conf.low, ymax = conf.high))+
  geom_pointrange(size = 0.5)+
  geom_errorbar(size = 1)+
  geom_hline(yintercept = 1, linetype = 2, size = 1)+
  coord_flip()+
  labs(x = "",
       y = "Adjusted hazard ratio and 95% confidence interval")+
  theme_minimal()+
  theme(panel.grid = element_blank())+
```

```
  scale_y_continuous(limits = c(0, 5.5),
                     breaks = seq(0, 3, 0.5))+
  geom_text(aes(label = custom_label,
                y = 3.2, hjust = 0))
```



This is looking better. For a final level of customization we can add a vertical line to give the appearance of a divide between the error bars and labels, and manually shift the x-axis title to the left so it remains centered. I will also increase the overall size of the text slightly.

```
tidy_model_labels %>%
  ggplot(aes(x = term,
             y = estimate, ymin = conf.low, ymax = conf.high))+
  geom_pointrange(size = 0.5)+
  geom_errorbar(size = 1)+
  geom_hline(yintercept = 1, linetype = 2, size = 1)+
  coord_flip()+
  labs(x = "",
       y = "Adjusted hazard ratio and 95% confidence interval")+
  theme_minimal()+
  theme(panel.grid = element_blank())+
  scale_y_continuous(limits = c(0, 6),
                     breaks = seq(0, 3, 0.5))+
  geom_text(aes(label = custom_label,
                y = 3.5, hjust = 0))+
  geom_hline(yintercept = 3.3)+
  theme(axis.title.x = element_text(hjust = 0.1, vjust = -0.5),
        text = element_text(size = 12))
```



Remember to save a publication-quality version of your plot.

```
#If you have not set the working directory, include the full file path
ggsave("my_plot.png",
       dpi = 300, dev = "png",
       height = 20, width = 25, units = "cm")
```

Although this example used hazard ratios from a model created in R, this plot can be easily replicated by manually creating an equivalent dataset from estimates given in another statistical software. Creating a .csv file with the following columns and importing this into R will provide an attractive method of presenting regression parameters.

term	estimate	conf.low	conf.high	custom_label

Future tutorials will focus on more complex plots, including using colour and facets to distinguish several different analyses for the same parameter.