Chapter 11: Grammar of `ggplot2`

Contents

1 Introduction

1.1 What is ggplot2?

- Graphical package of R
- Works in layered fashion
- Each layer contains information about:
  - **Data**: should be data frame
  - **Mapping**: how your data corresponds to visual elements on your plot (=aesthetics).
  - **Geom**etric Objects(points, lines. . .)
  - **Stat**istics: how to summarize your data
  - **Facet**: how to break up the data

How to start?

```
install.packages("ggplot2")
library(ggplot2)
```

1.2 Documentation

ggplot2 documentation: http://ggplot2.org/

1.3 Grammar

Example *fish*
Data: *fish.xlsx* (Peck et all.)

The state of Maine conducted a field study of 115 lakes to characterize mercury levels in fish by measuring mercury (Hg) and other variables on lake characteristics.

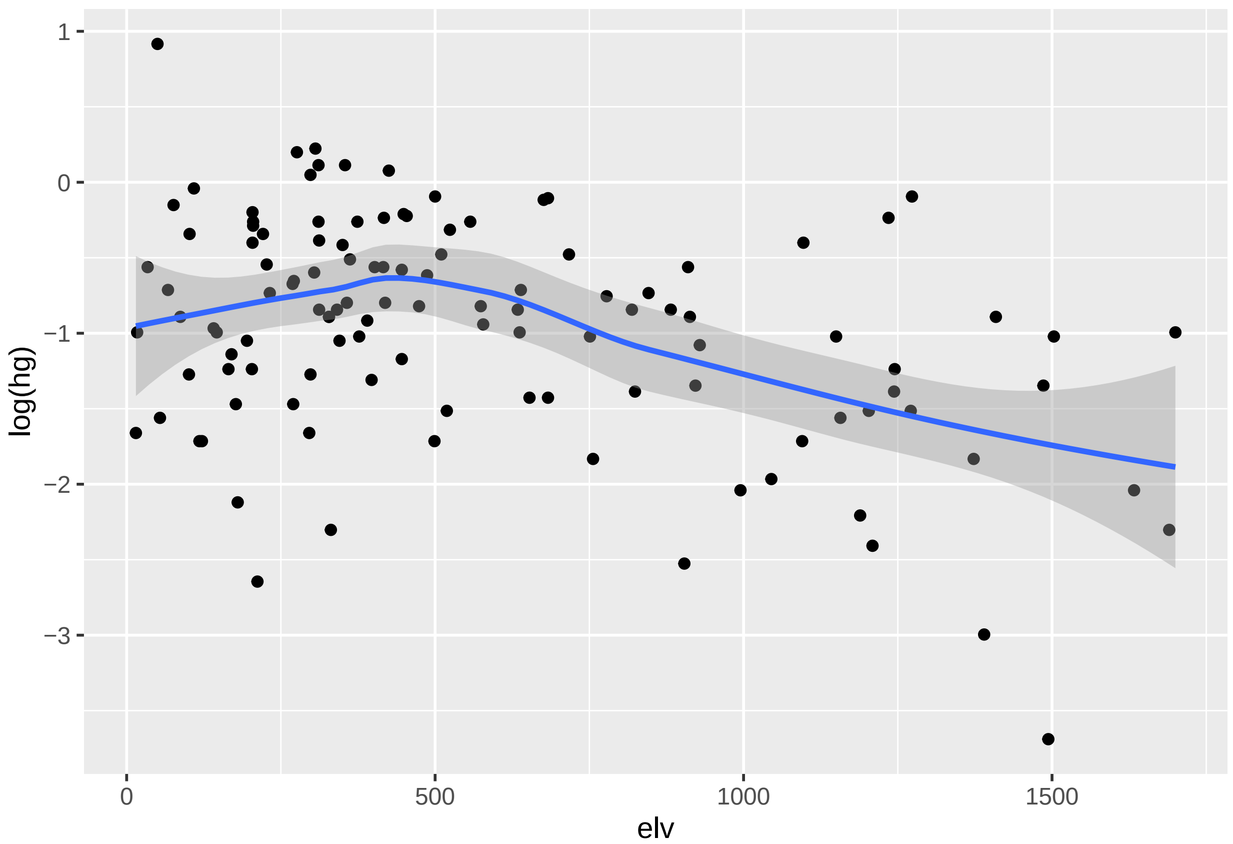
This data frame contains information on the average mercury (Hg) level of fish in lakes in order to determine whether the fishes are safe to eat. The following variables are given:

Name variable	Description
name	Name of the lake
hg	Mercury level, expressed in parts per million
number	Number of fish in the composite
elv	Elevation (feet)
sa	Surface area (acres)
z	Maximum depth (feet)
lt	Lake type: 1 = oligotrophic; 2 = eutrophic; 3 = mesotropic
st	Lake stratification indicator (1 = yes, 0 = no)
dam	Some lakes have a dam: 0 = the lake does not have a dam; 1 = the lake has a dam

```
head(fish)
```

```
## # A tibble: 6 x 15
##    name     hg number   elv    sa     z    lt    st   dam  lat1  lat2  lat3 long1
##   <chr> <dbl>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 ALLE~ 1.08       3   425    83    27     3     1     1    44    57    44    68
## 2 ALLI~ 0.025      2  1494    47    26     2     0     1    45    37    50    69
## 3 ANAS~ 0.57       5   402   568    54     2     1     0    44    25    13    70
## 4 BALC~ 0.77       5   557   704    44     2     1     0    43    37     0    70
## 5 BASK~ 0.79       5   417  6944    22     2     0     1    45    30    32    67
## 6 BAUN~ 0.75       4   205   200    29     2     1     0    43    21    46    70
## # ... with 2 more variables: long2 <dbl>, long3 <dbl>
```

```
ggplot(fish, aes(elv, log(hg))) + geom_point() + stat_smooth()
```



Layers:
Which part of this code belongs to which layer?

- `fish` → **Data**: should be data frame
- `aes(elv, log(hg))` → **Mapping**: tell R how your data corresponds to visual elements of your plot (= aesthetics). Use the function `aes()`.
- `geom_point()` → **Geom**etric Objects (point, lines…)
- `stat_smooth()` → **Stat**istics: how to summarize your data
- **Facet**: how to break up the data

2 Build a plot layer by layer by `ggplot()`

2.1 `ggplot()`

```
ggplot(data, mapping) + geom_XXX() + stat_YYY() + facet_ZZZ()
```

2.2 Data and mapping

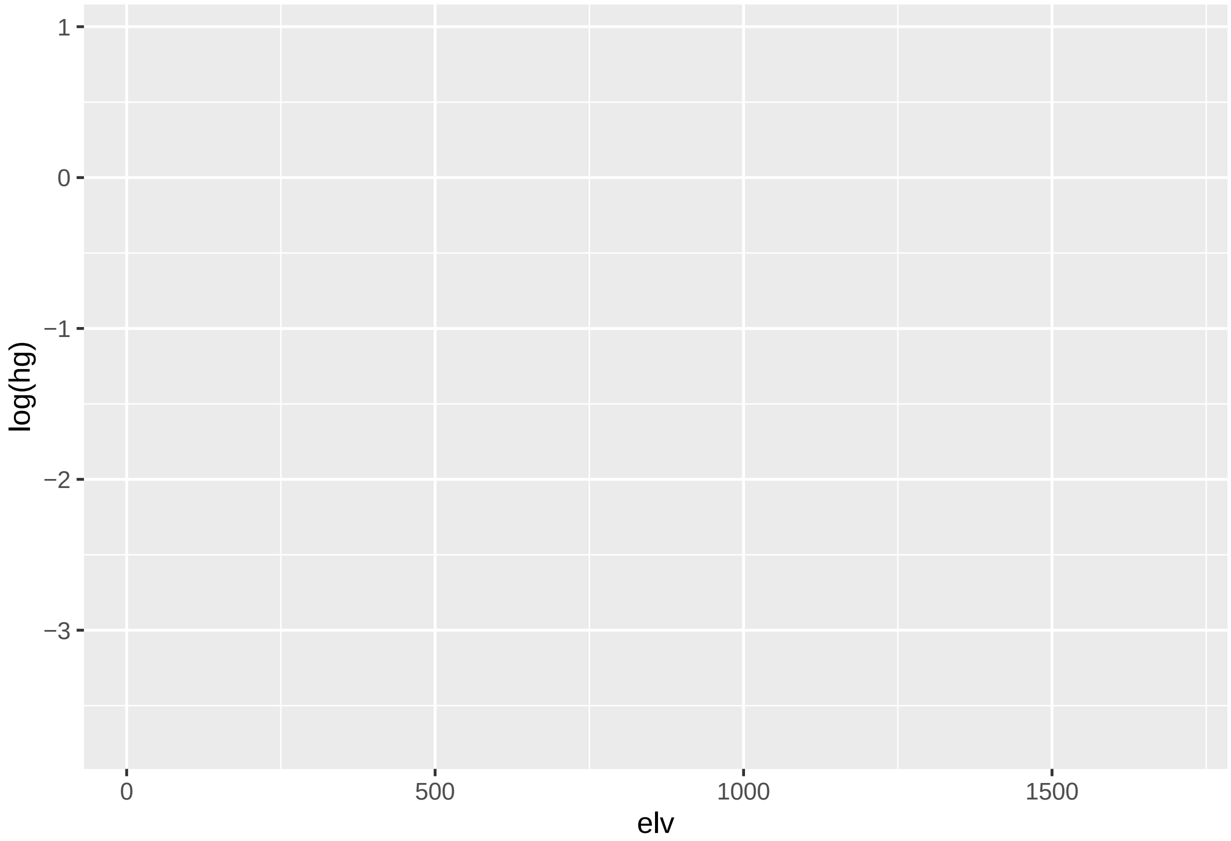
1. **Data**: must be a data frame
2. **Mapping**: The aesthetic mappings describe the way that variables in the data are mapped to the plot. Therefore we use **aes** function.

Example *fish*

For data `fish`:

1. **Data** → `fish`
2. **Mapping** → `aes(x = elv, y = log(hg), colour = factor(dam))`

```
pl1 <- ggplot(fish, aes(x = elv, y = log(hg), colour = factor(dam)))
pl1
```



Remark:

1. Here, we map the *x position* to `elv`, the *y position* to `log(hg)` and *colour* to `dam`. The first two arguments can be left without names.
2. You should never refer to variables outside the data frame (e.g. `fish$hg`)
3. This code produces an empty plot! We have to add layers.

2.3 Layer Geoms

Layers define the basic "shape" of the elements on the plot. Layers can be added to plots created by `ggplot` or by `qplot`.

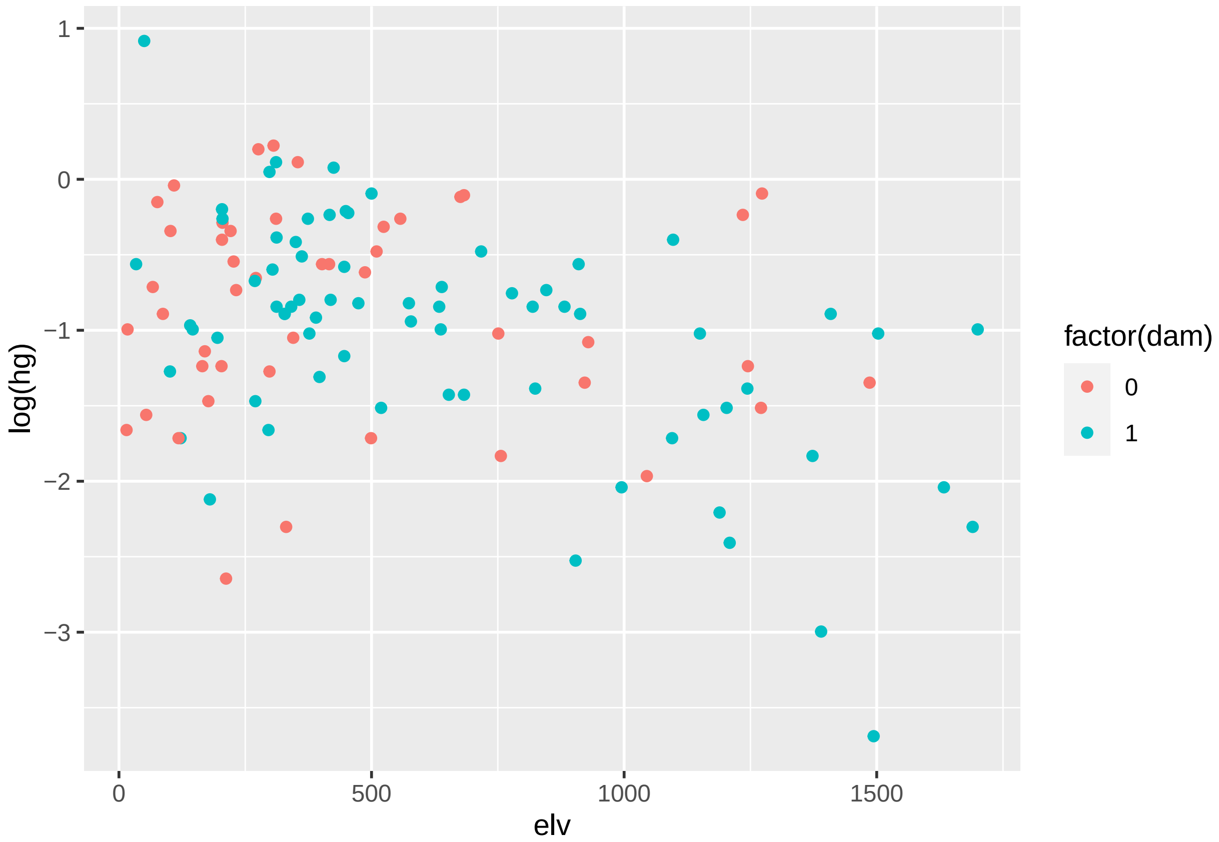
A geom defines the **layout** of a ggplot2 layer.

A selection of geoms and associated default stats:

geom	description	default stat
`geom_bar()`	Bar chart for categorical variable	`stat_bin()`
`geom_point()`	Scatterplot	`stat_identity()`
`geom_line()`	Line, connecting observations in ordered x value	`stat_identity()`
`geom_boxplot()`	Boxplot	`stat_boxplot()`
`geom_smooth()`	Fits a smoother to the data	`stat_smooth()`
`geom_histogram()`	Histogram for continuous variable	`stat_bin()`
`geom_density()`	Smooth density estimate	`stat_density()`

2.3.1 To create scatterplot: `geom_point`

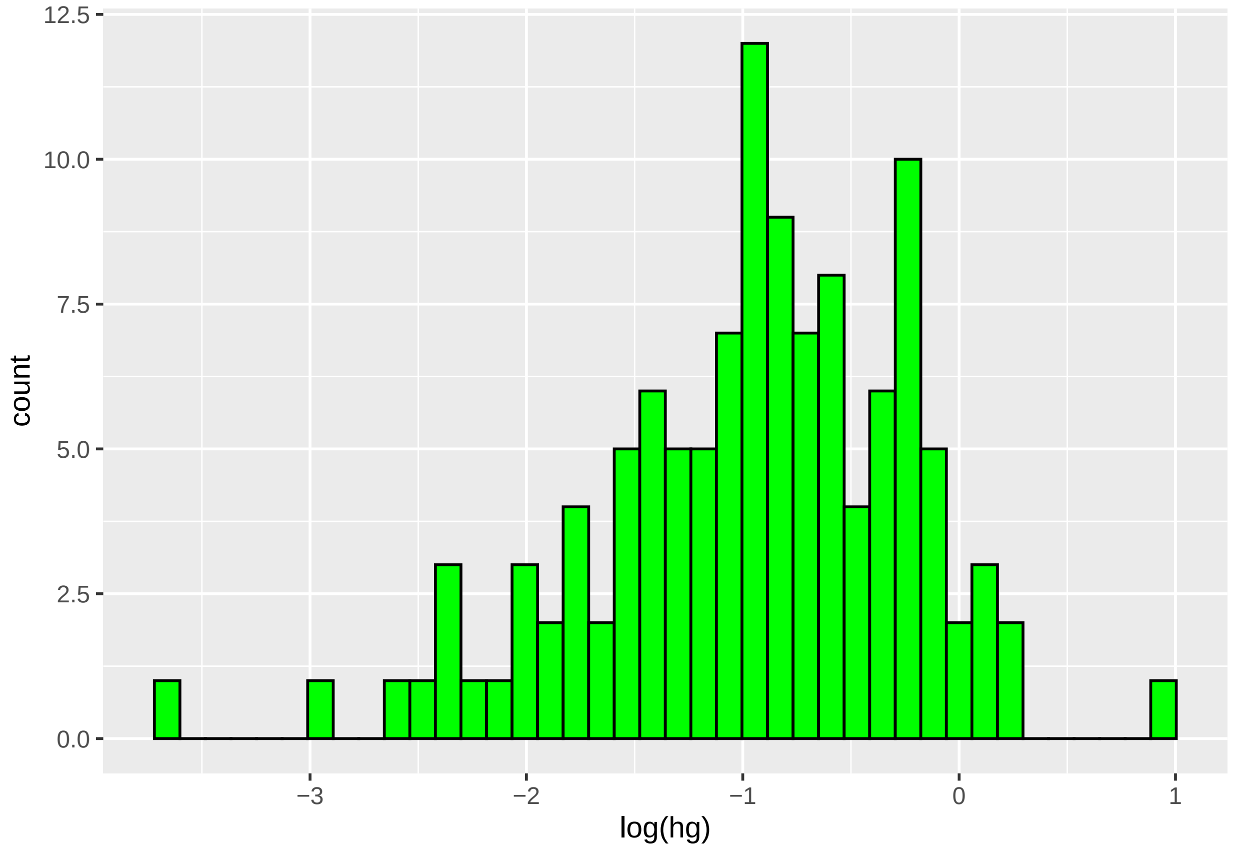
```
pl2 <- pl1 + geom_point()
pl2
```



2.3.2 To create a histogram: `geom_histogram`

To create a histogram for continuous variable `log(hg)`

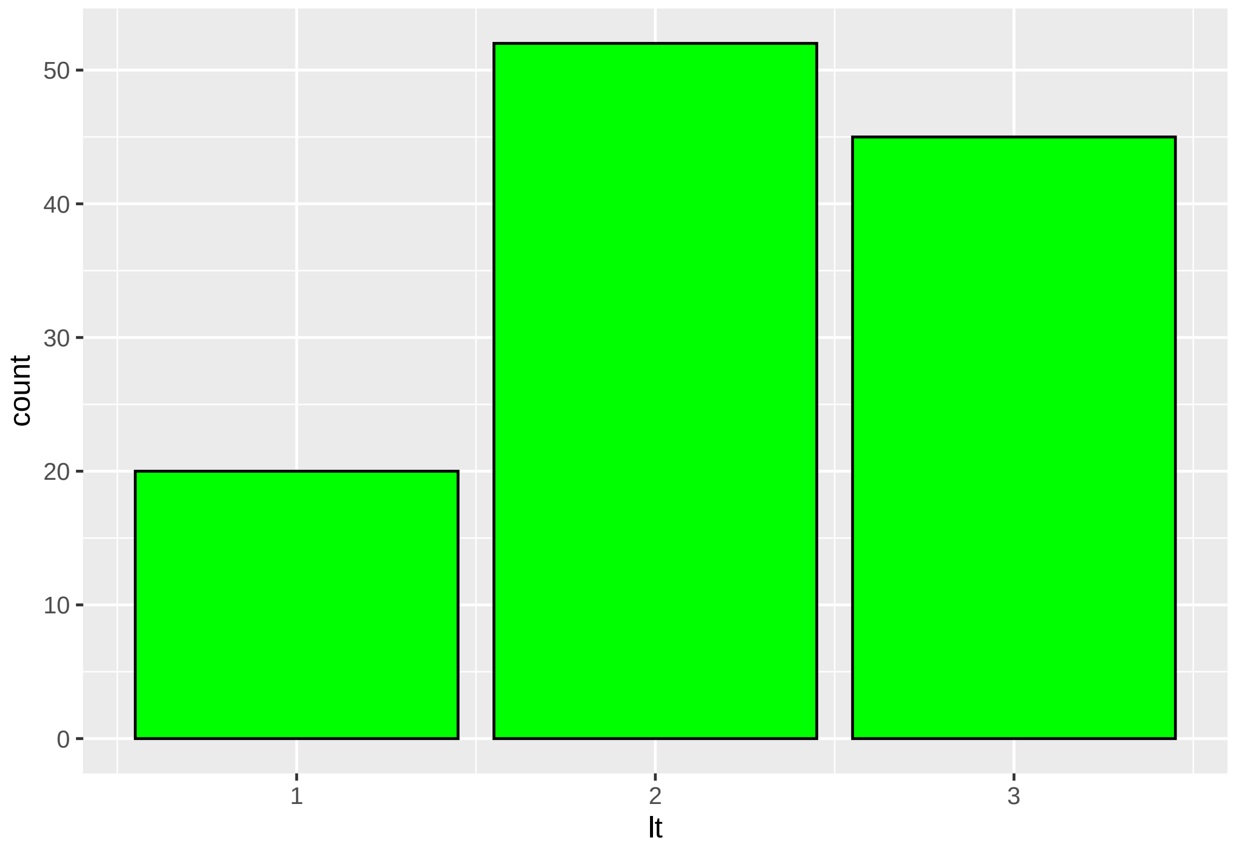
```
pl3 <- ggplot(fish, aes(log(hg))) + geom_histogram(bins = 40, colour = "black", fill = "green")
pl3
```



2.3.3 To create a bar chart for a categorical variable: `geom_bar`

To create a bar chart for categorical variable lake type (`lt`)

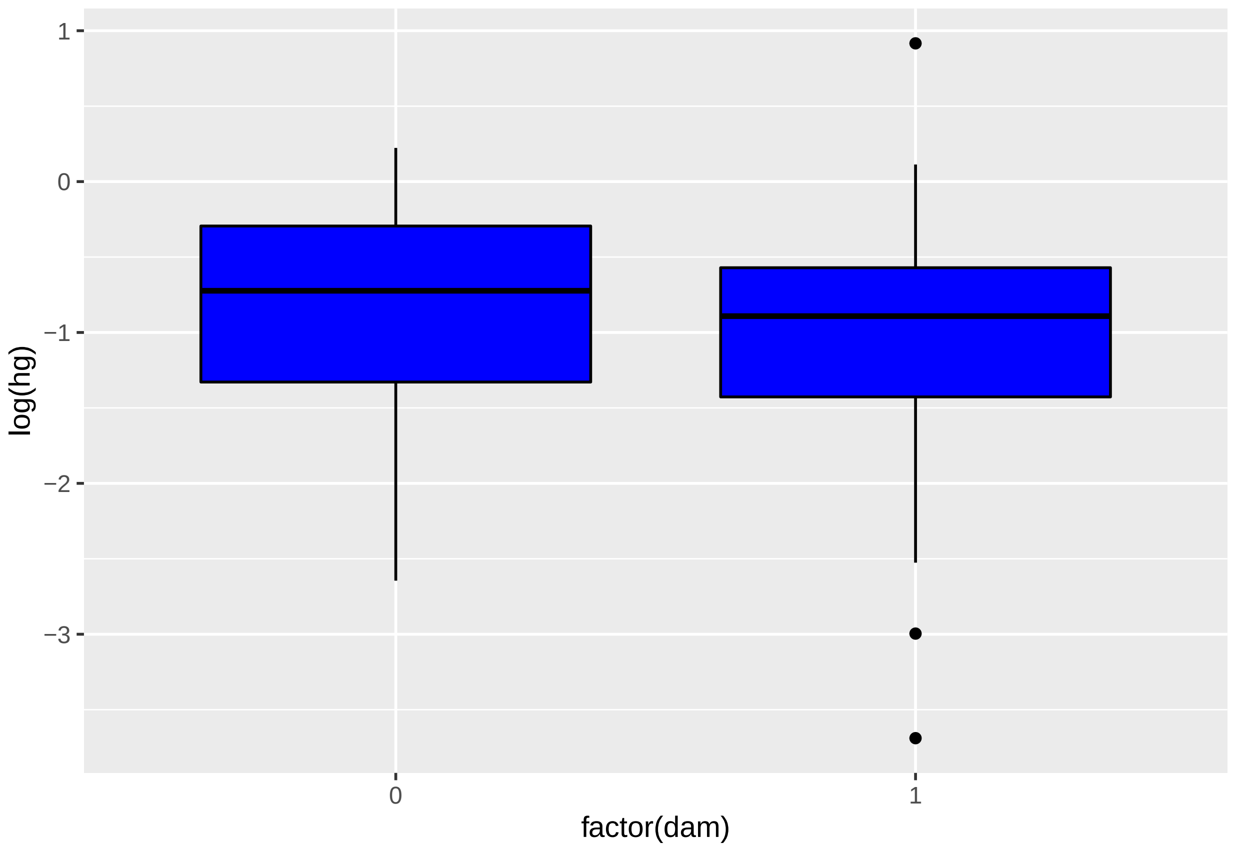
```
pl4 <- ggplot(fish, aes(lt)) + geom_bar(colour = "black", fill = "green")
pl4
```



2.3.4 To create a boxplot: geom_boxplot

To create a boxplot of log(hg) for lakes with and without dam

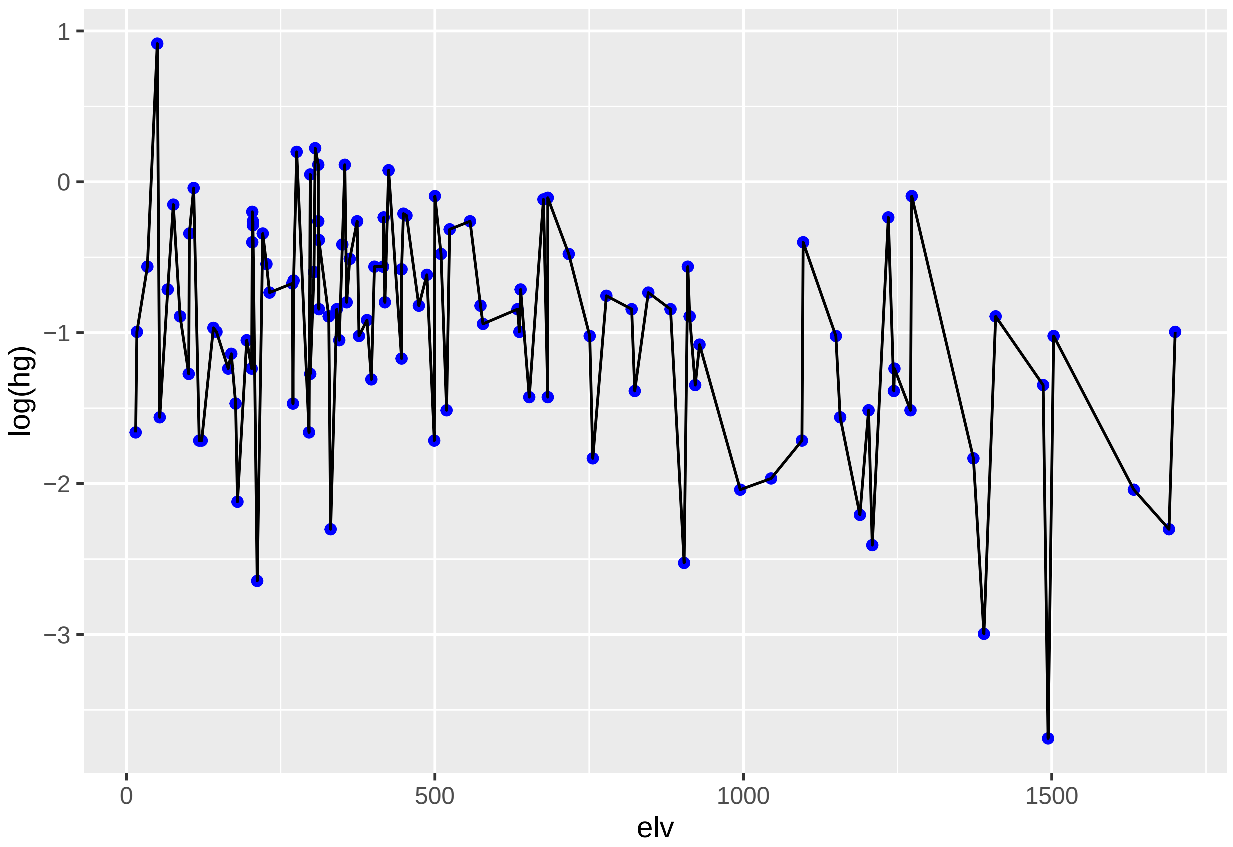
```
pl5 <- ggplot(fish, aes(x = factor(dam), y = log(hg))) +
  geom_boxplot(colour = "black", fill = "blue")
pl5
```



2.3.5 To produce a line: `geom_line`

Create a scatterplot and connect the dots by a line

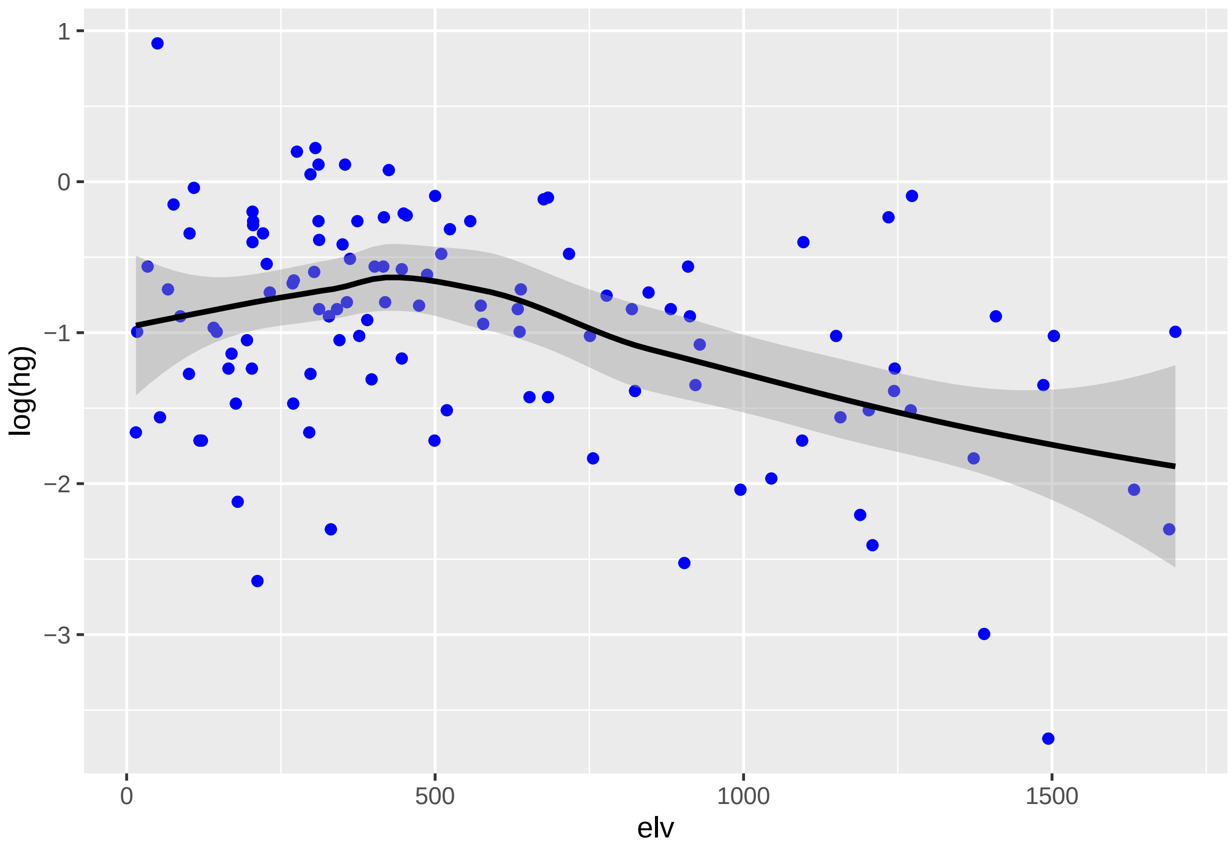
```
ggplot(fish, aes(x = elv, y = log(hg))) + geom_point(colour = "blue") +
  geom_line(colour = "black")
```



2.3.6 To produce a smooth trend line: `geom_smooth`

```
pl6 <- ggplot(fish, aes(x = elv, y = log(hg))) + geom_point(colour = "blue") +
  geom_smooth(colour = "black")
pl6
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

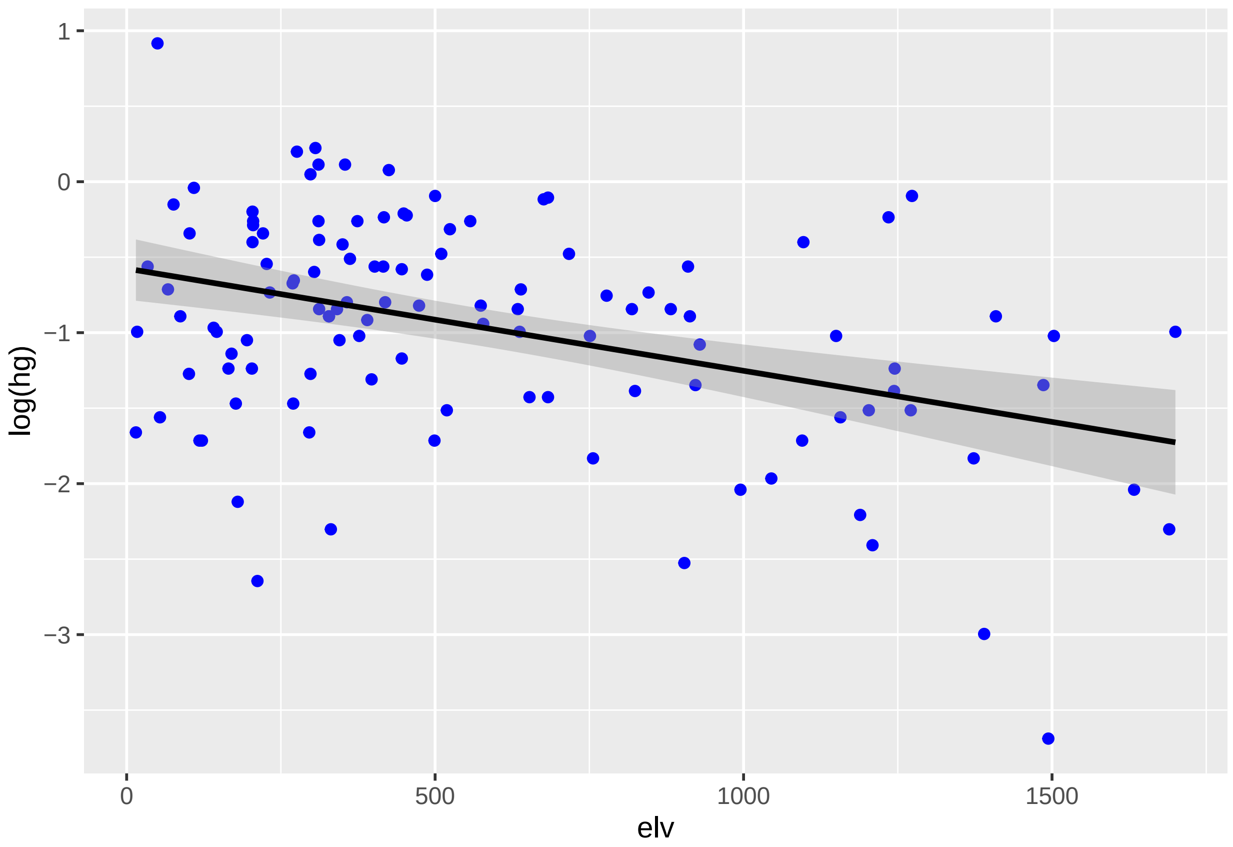


Remark:

- For small n (< 1000), a `loess` smoother is used by default.
- If you want to fit a linear model, you can change the method to `lm`.
- If you want a robust fitting line, you can use `method = rlm`. (You then first have to load the `MASS` package).

Add regression line

```
pl7 <- ggplot(fish, aes(x = elv, y = log(hg))) + geom_point(colour = "blue") +
  geom_smooth(colour = "black", method = "lm")
pl7
```



2.3.7 Remarks

Remark 1:

By default, ggplot2 gives you the 95% confidence interval.

- In case you want to change the confidence level, use e.g., `level = 0.9`.
- In case you do not want to see the 95% confidence band, use `se = FALSE`.
- In case you want to have the 95% prediction interval: you can use the `geom_ribbon` function.

In case you want to see the prediction interval instead of confidence interval

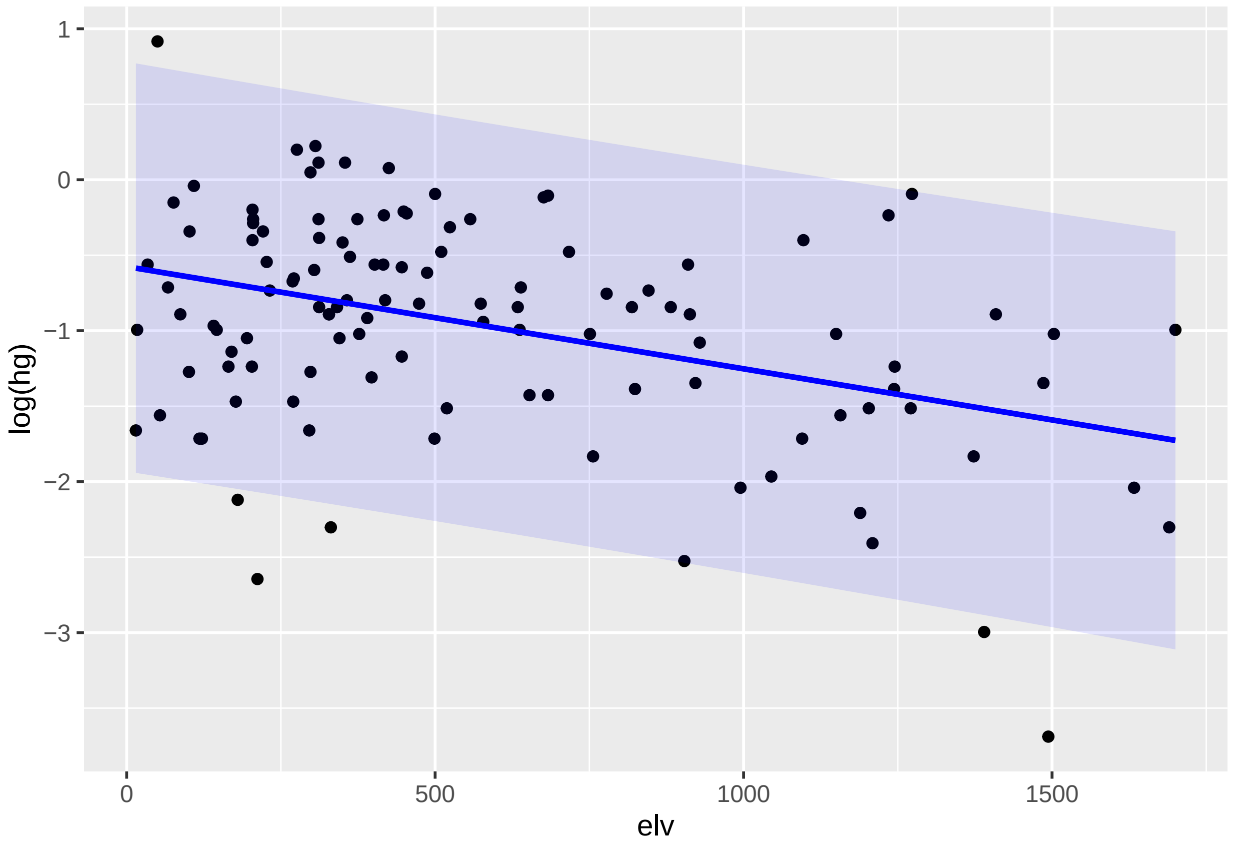
```
# Fit a linear model
m.lm <- lm(log(hg) ~ elv, data = fish)
res.pred <- predict(m.lm, interval = "predict")
head(res.pred)
```

```
##          fit       lwr        upr
## 1 -0.8632111 -2.210718  0.4842960
## 2 -1.5869632 -2.959411 -0.2145157
## 3 -0.8476393 -2.195373  0.5000948
## 4 -0.9525800 -2.299410  0.3942505
## 5 -0.8577948 -2.205377  0.4897876
## 6 -0.7142631 -2.065263  0.6367366
```

```
# cbind the predictions to fish
fish.pred <- cbind(fish, res.pred)
names(fish.pred)
```

```
##  [1] "name"   "hg"     "number" "elv"    "sa"     "z"      "lt"     "st"
##  [9] "dam"    "lat1"   "lat2"   "lat3"   "long1"  "long2"  "long3"  "fit"
## [17] "lwr"    "upr"
```

```
# Make now the plot
# remark that not all aesthetics are defined beforehand
pl7a <- ggplot(fish.pred, aes(x = elv)) +
  geom_point(aes(y = log(hg))) +
  geom_line(aes(y = fit), colour = "blue", size = 1)
pl7b <- pl7a + geom_ribbon(aes(ymin = lwr, ymax = upr), fill = "blue", alpha = 0.1)
pl7b
```



Remark 2:
Example *fish*
We use previous example as how you can **use multiple data frames** in one and the same `ggplot`.
We here use the data frame `fish` and the data frame `pred_fish`.

1. Fit a linear model and create the data frame `pred_fish`

```
m.lm <- lm(log(hg) ~ elv, data = fish)
pred_fish <- cbind(elv = fish$elv, data.frame(predict(m.lm, interval = "prediction")))
names(pred_fish)
```

```
## [1] "elv" "fit" "lwr" "upr"
```

2. Make now the scatterplot of `elv` versus `log(hg)` **based on `fish` data frame**

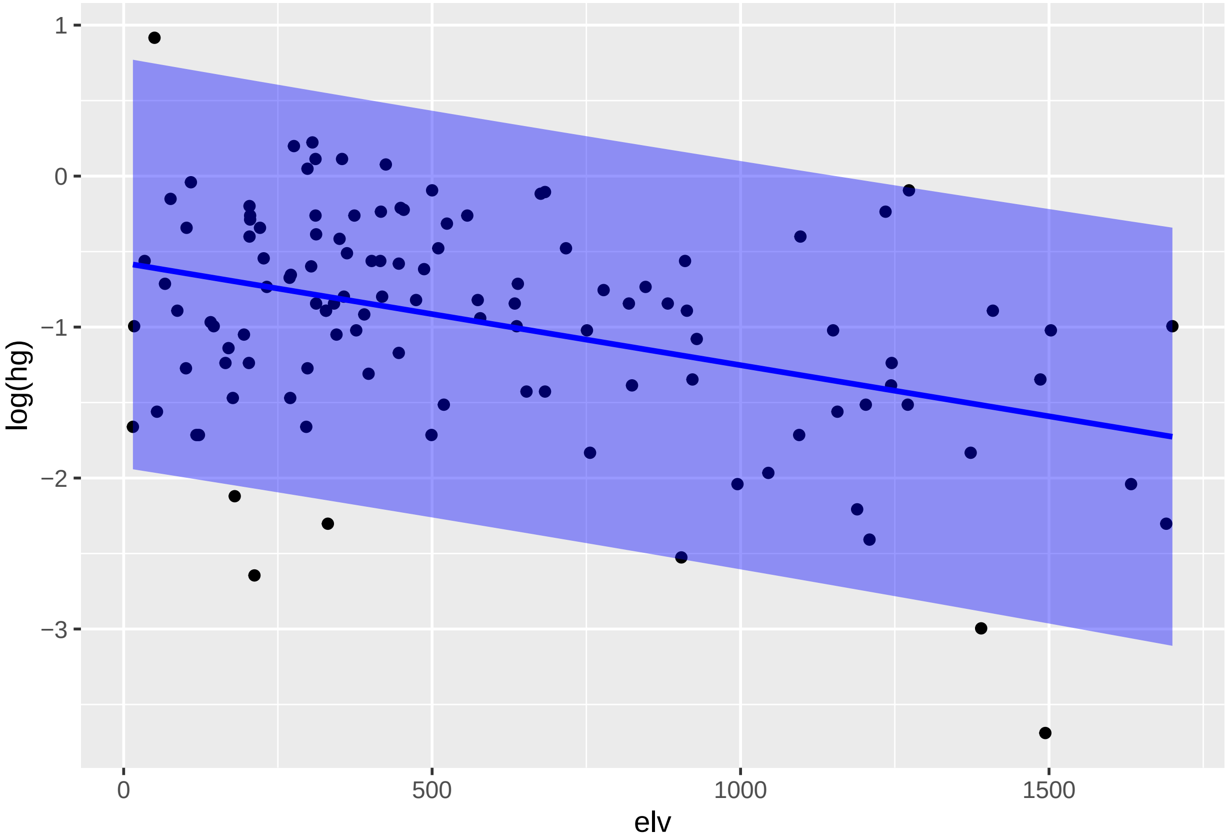
```
pl7d <- ggplot() + geom_point(data = fish, aes(x = elv, y = log(hg)))
```

3. Add now the fitted values **based on `pred_fish` data frame**

```
pl7e <- pl7d + geom_line(data = pred_fish, aes(x = elv, y = fit), colour = "blue",
                         size = 1)
```

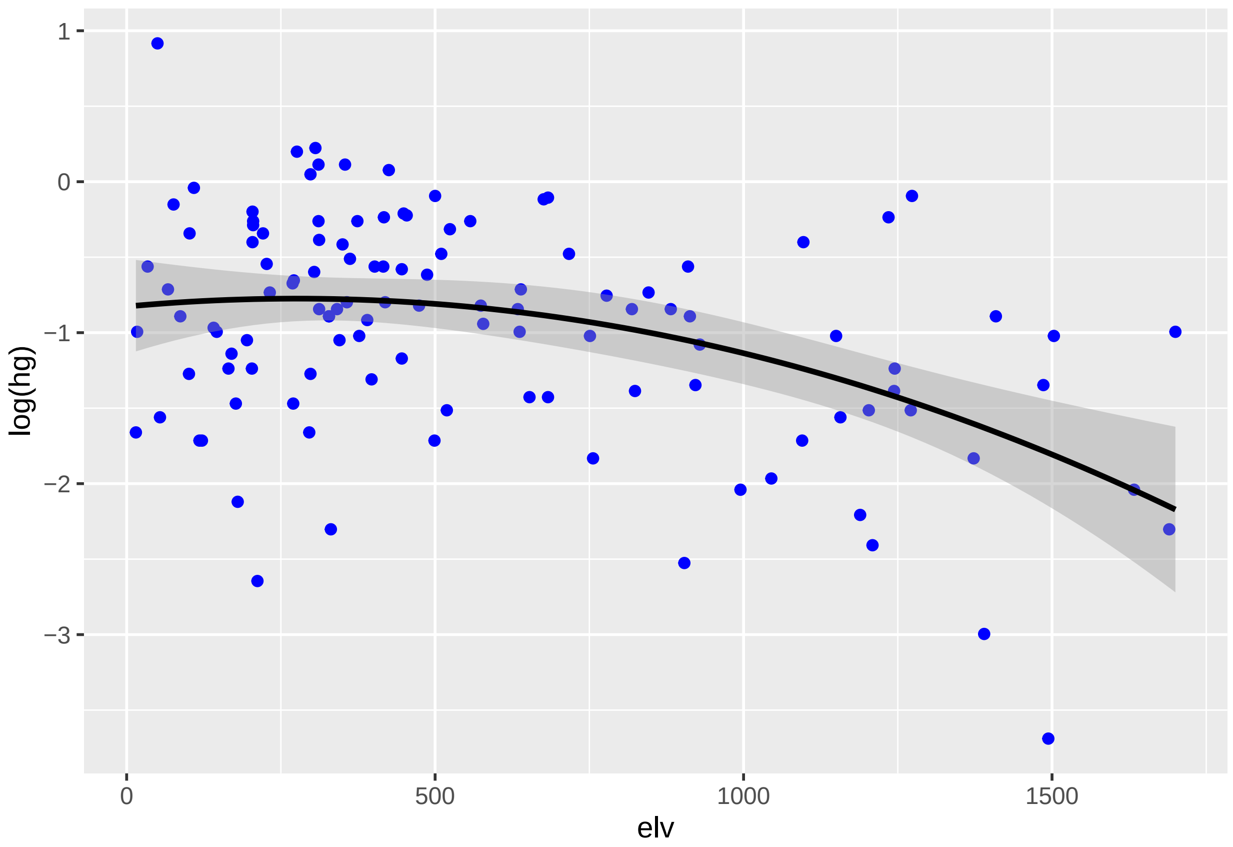
4. Add now the prediction limits **based on `pred_fish` data frame.
   Alpha expresses the density of the ribbon.

```
pl7f <- pl7e + geom_ribbon(data = pred_fish, aes(x = elv, ymin = lwr, ymax = upr),
                           fill = "blue", alpha = 0.4)
pl7f
```



Remark 3:
You can also specify the underlying model by for example `formula = y ~ x`

```
ggplot(fish, aes(x = elv, y = log(hg))) + geom_point(colour = "blue") +
  geom_smooth(colour = "black", formula = y ~ x + I(x^2), method = "lm")
```



2.4 Layer Stat

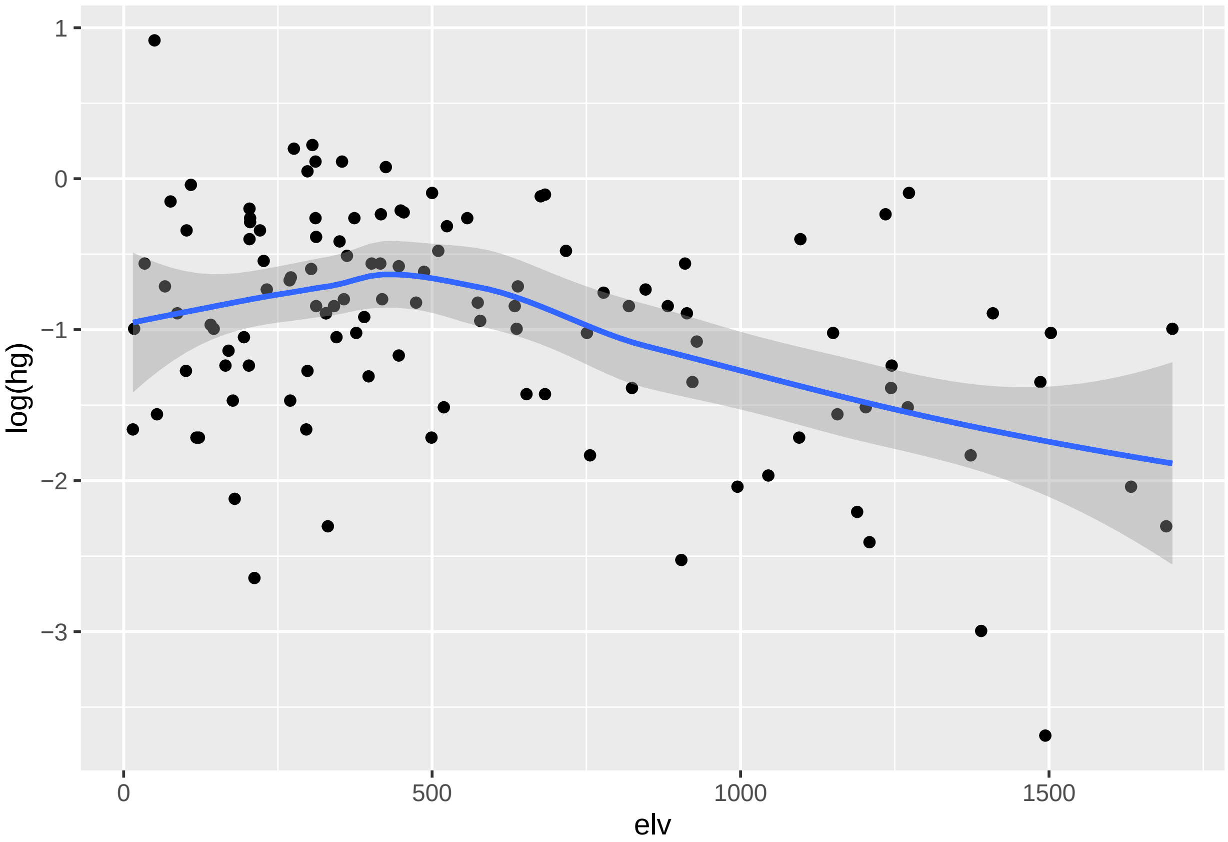
The *Stat* layer describes how the data should be summarized.

Some useful stats and default geoms:

stat	description	default geom
`stat_bin()`	Counts number of observations per bin	`geom_bar()`
`stat_smooth()`	Creates a smooth line	`geom_smooth()`
`stat_sum()`	Adds values	`geom_point()`
`stat_identity()`	No summary, plots data as it	`geom_point()`
`stat_boxplot()`	Summarizes data for boxplot	`geom_boxplot()`

2.4.1 Create a smooth line: `stat_smooth`

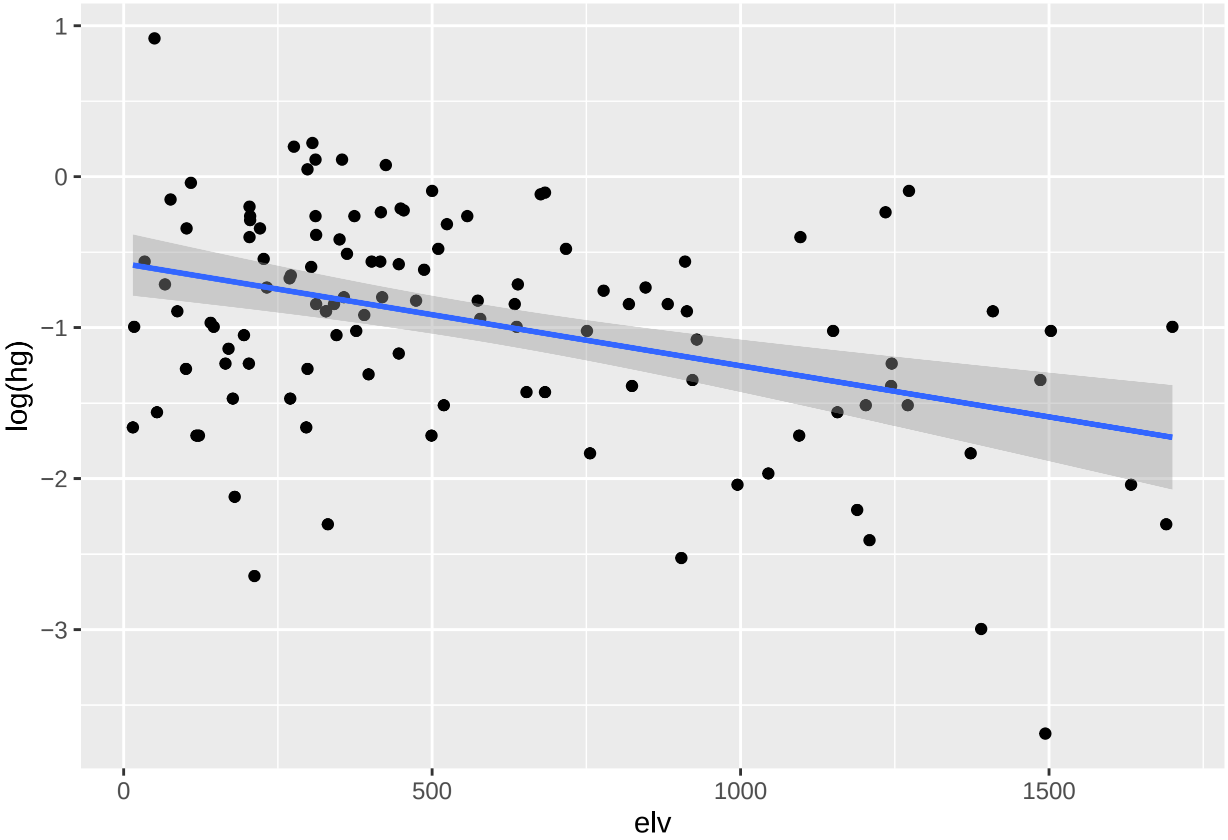
```
ggplot(fish, aes(elv, log(hg))) + geom_point() + stat_smooth()
```



- You first create a scatterplot by `geom_point()`
- You then add a smooth line with `stat_smooth()`: it uses `loess()` regression

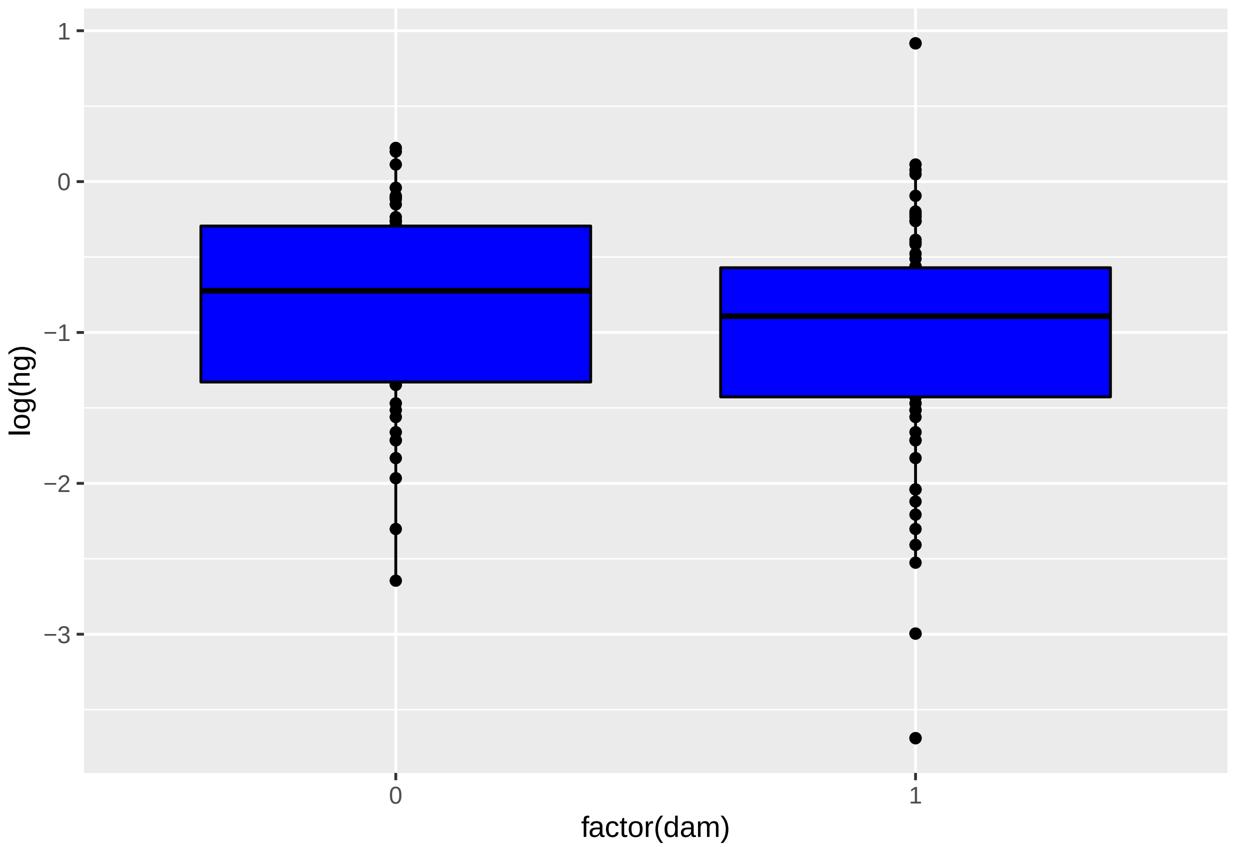
2.4.2 Create a regression line: `stat_smooth(method = 'lm')`

```
ggplot(fish, aes(elv, log(hg))) + geom_point() + stat_smooth(method = 'lm')
```



2.4.3 Add boxplot

```
ggplot(fish, aes(x = factor(dam), y = log(hg))) + geom_point(colour = "black") +
  stat_boxplot(colour = "black", fill = "blue")
```



2.5 Layer Facet

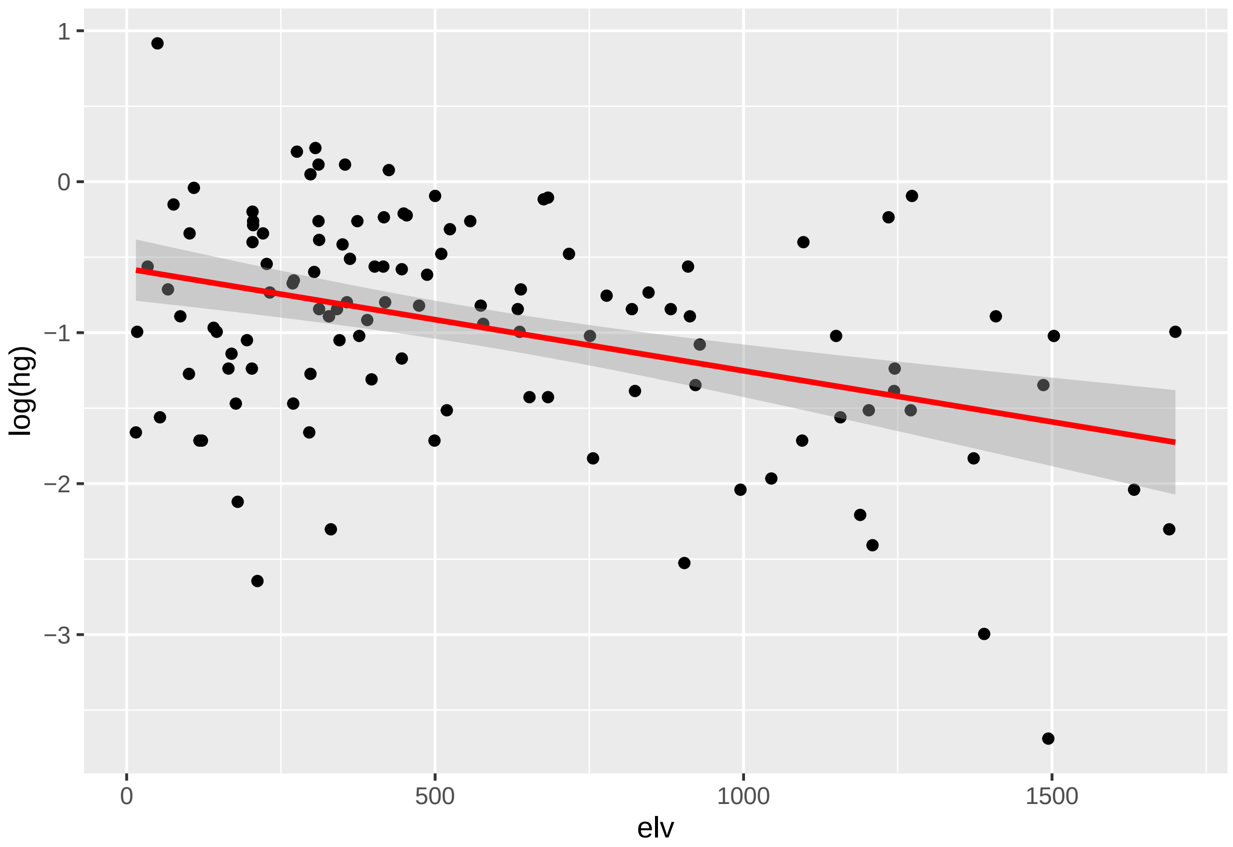
There are two types of faceting provided by ggplot2: **facet_grid** and **facet_wrap**

2.5.1 facet_grid

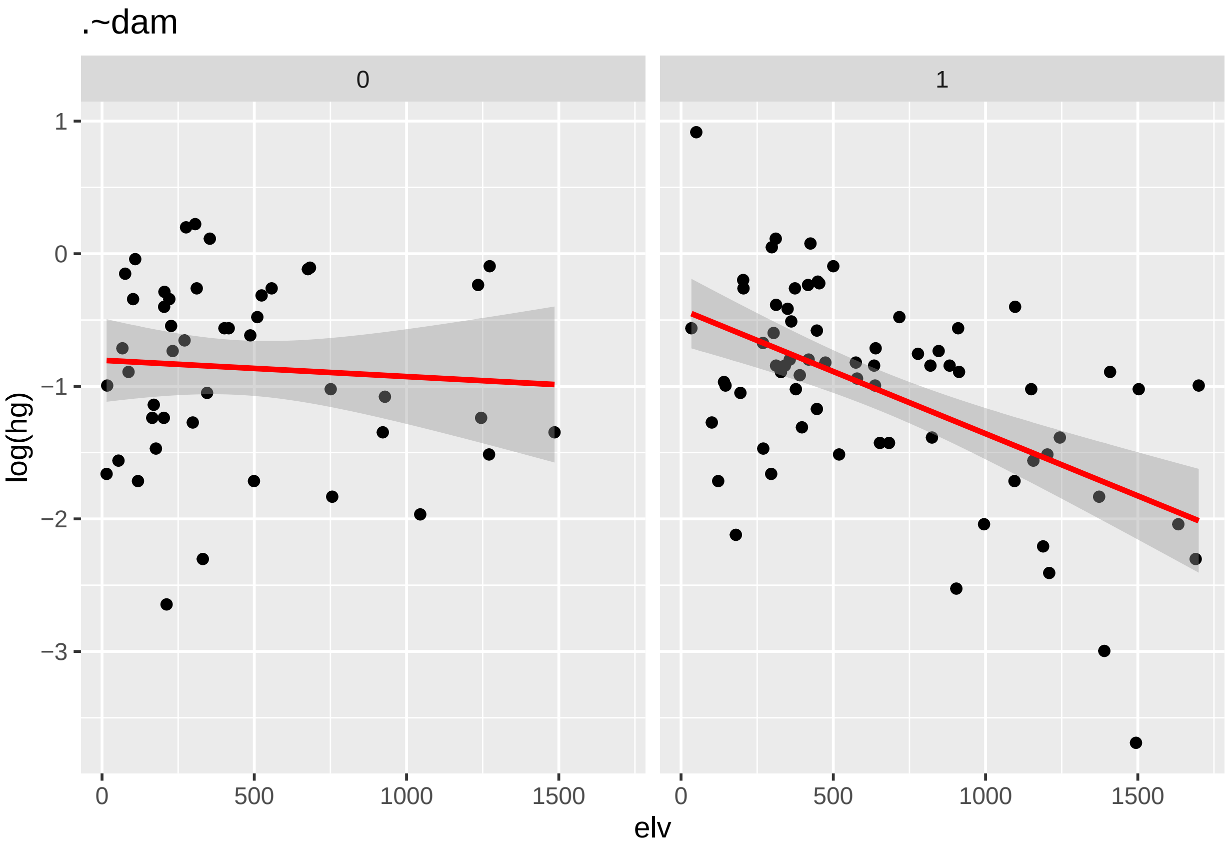
The specification of faceting variables is of the form (row ~ column)
. ~ a
a ~ .
a ~ b

Using the data set fish:

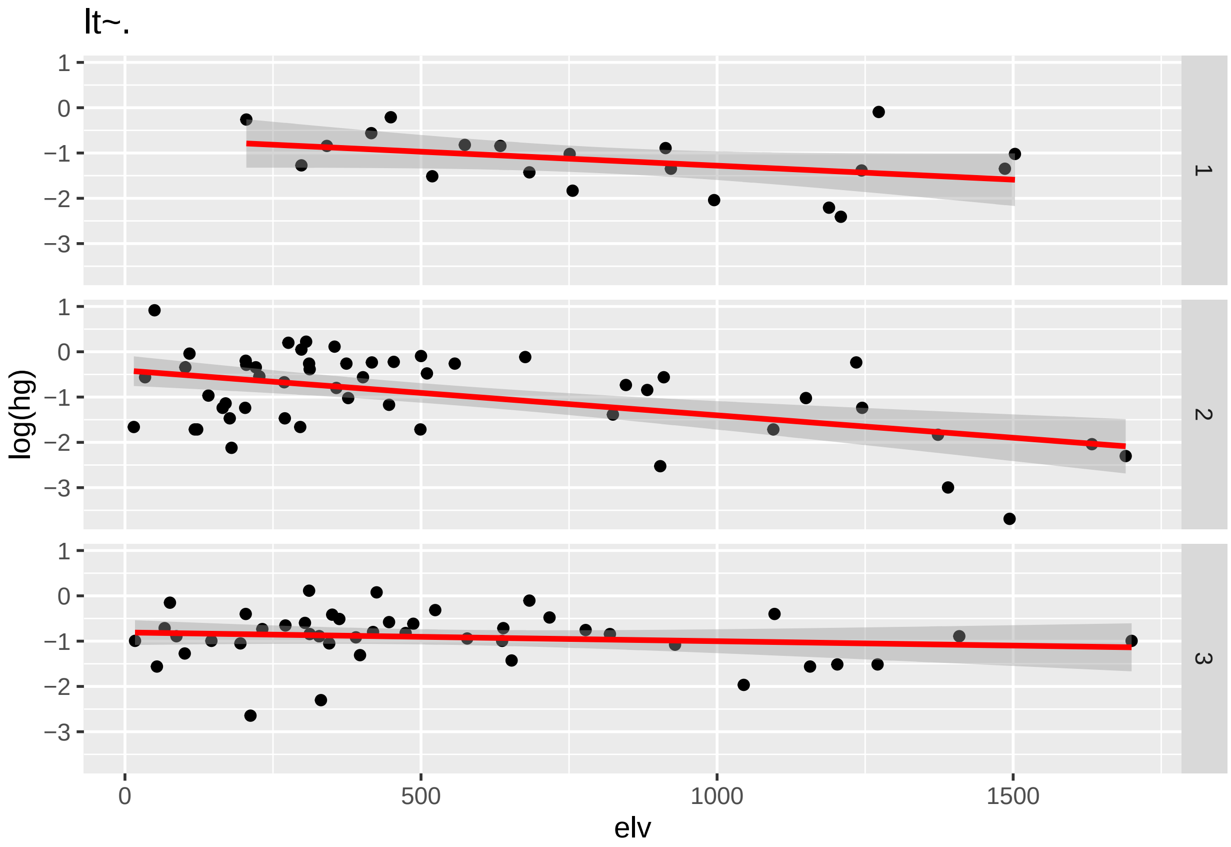
```
fac1 <- ggplot(fish, aes(elv, log(hg))) + geom_point() +
  geom_smooth(colour = "red", method = "lm")
fac1
```



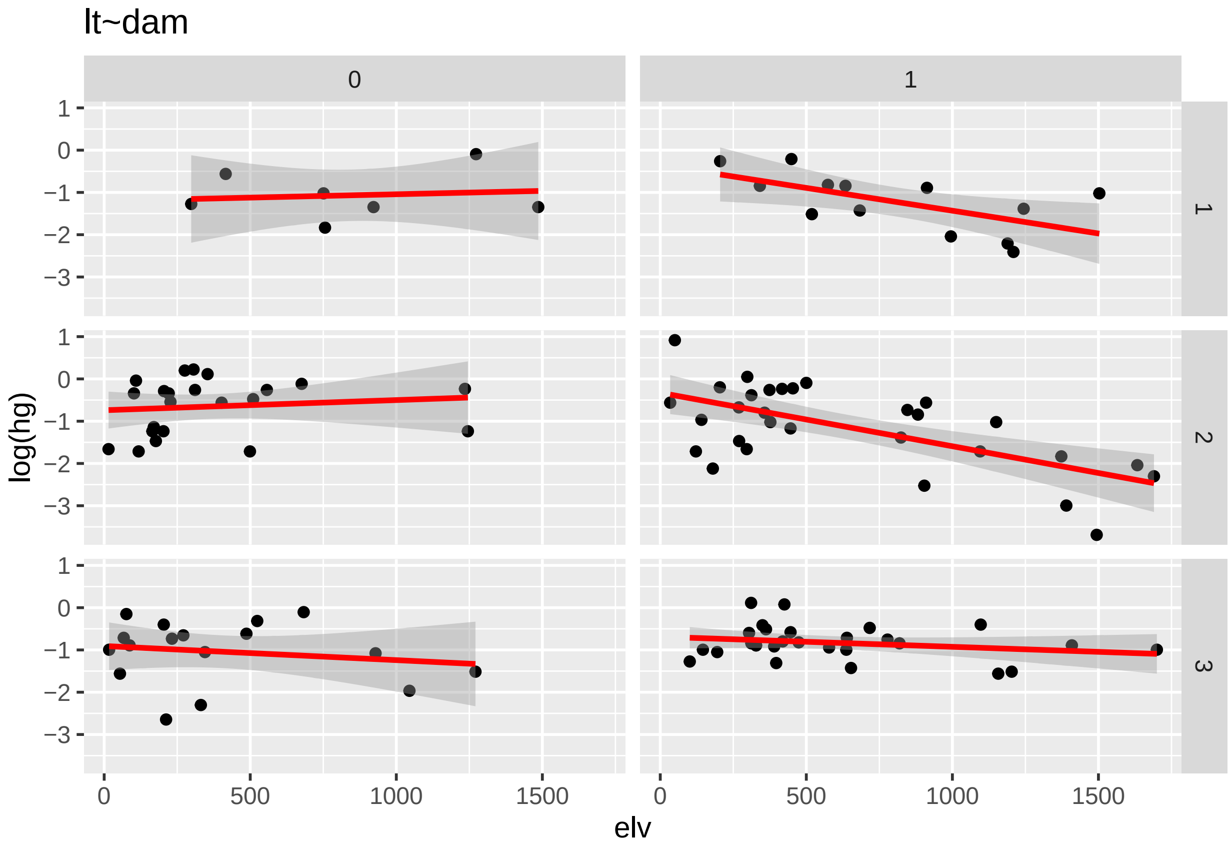
```
fac2 <- fac1 + facet_grid(. ~ dam) + labs(title = ".~dam")
fac2
```



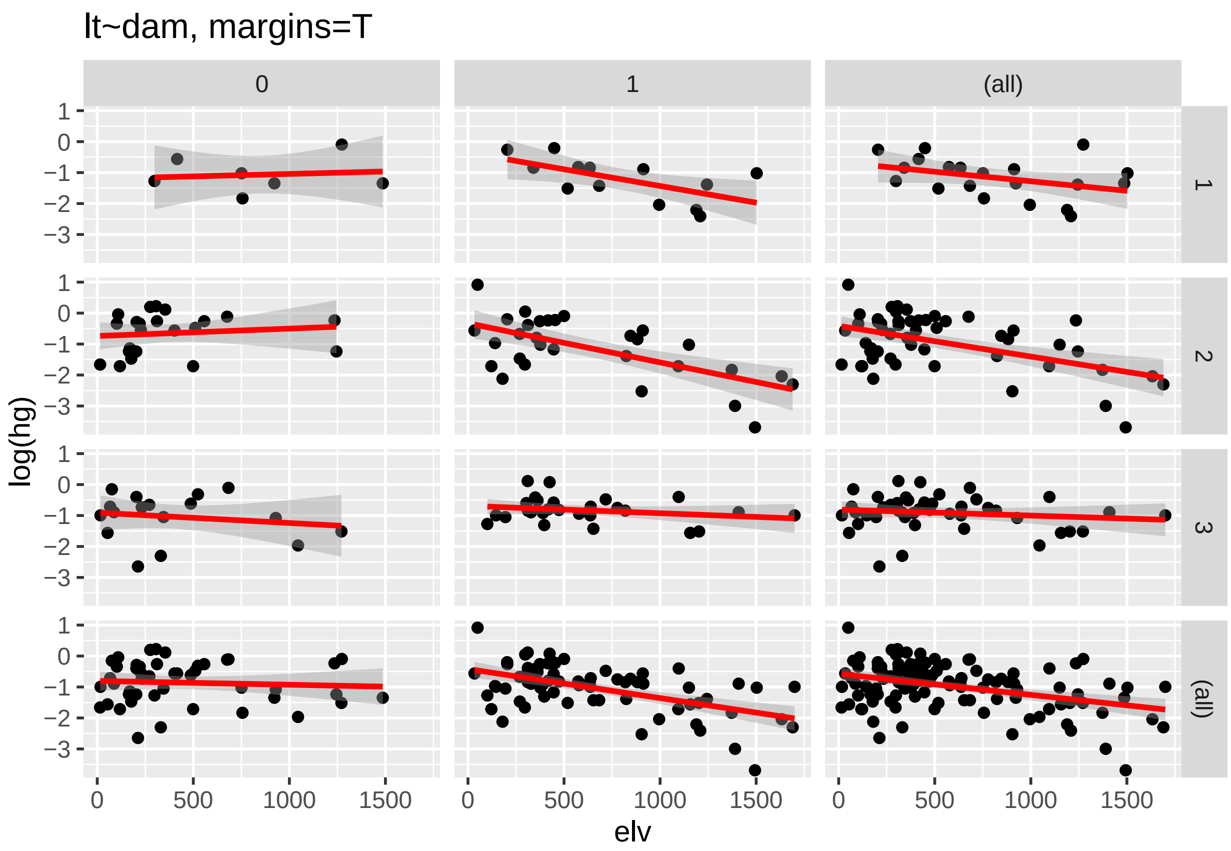
```r
fac3 <- fac1 + facet_grid(lt ~ .) + labs(title = "lt~.")
fac3
```



```
fac4 <-  fac1 + facet_grid(lt ~ dam) +  labs(title = "lt~dam")
fac4
```



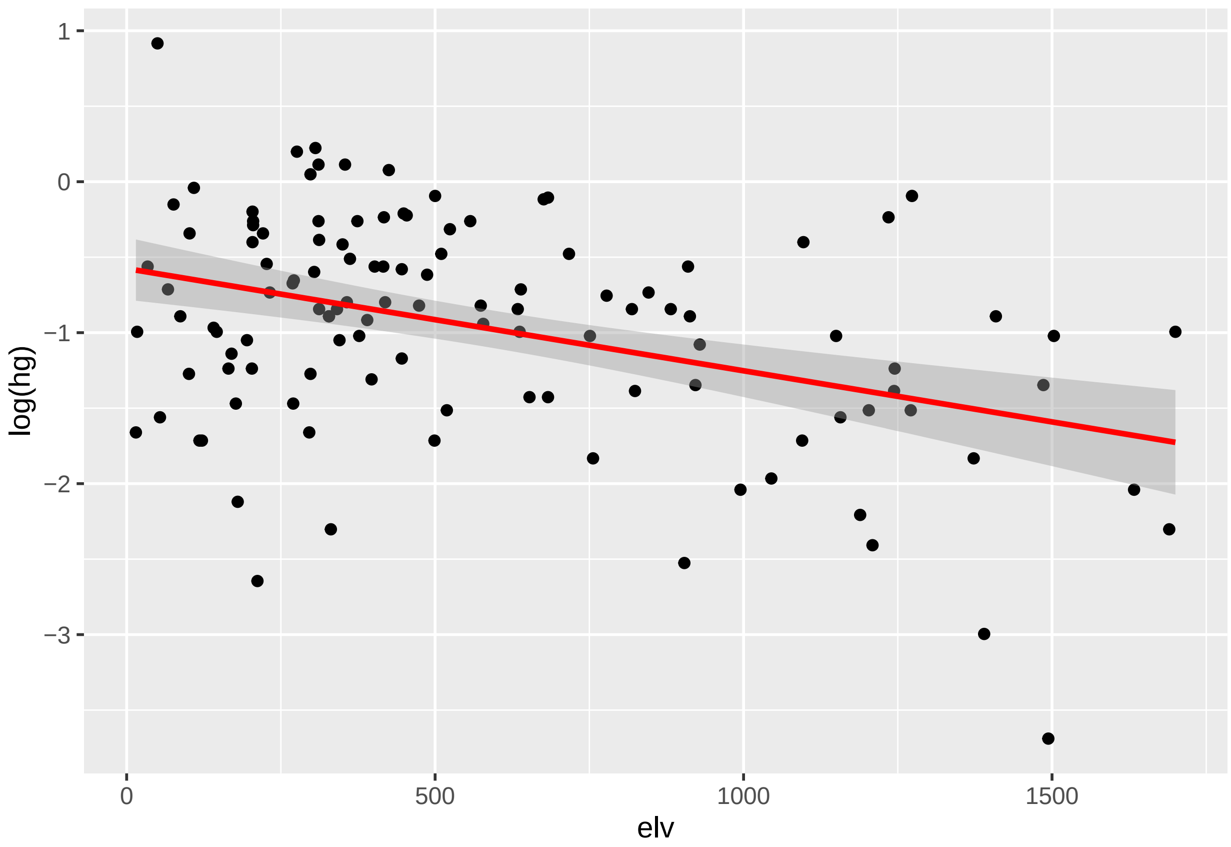
```
fac5 <-  fac1 + facet_grid(lt ~ dam, margins = T) + labs(title = "lt~dam, margins=T")
fac5
```



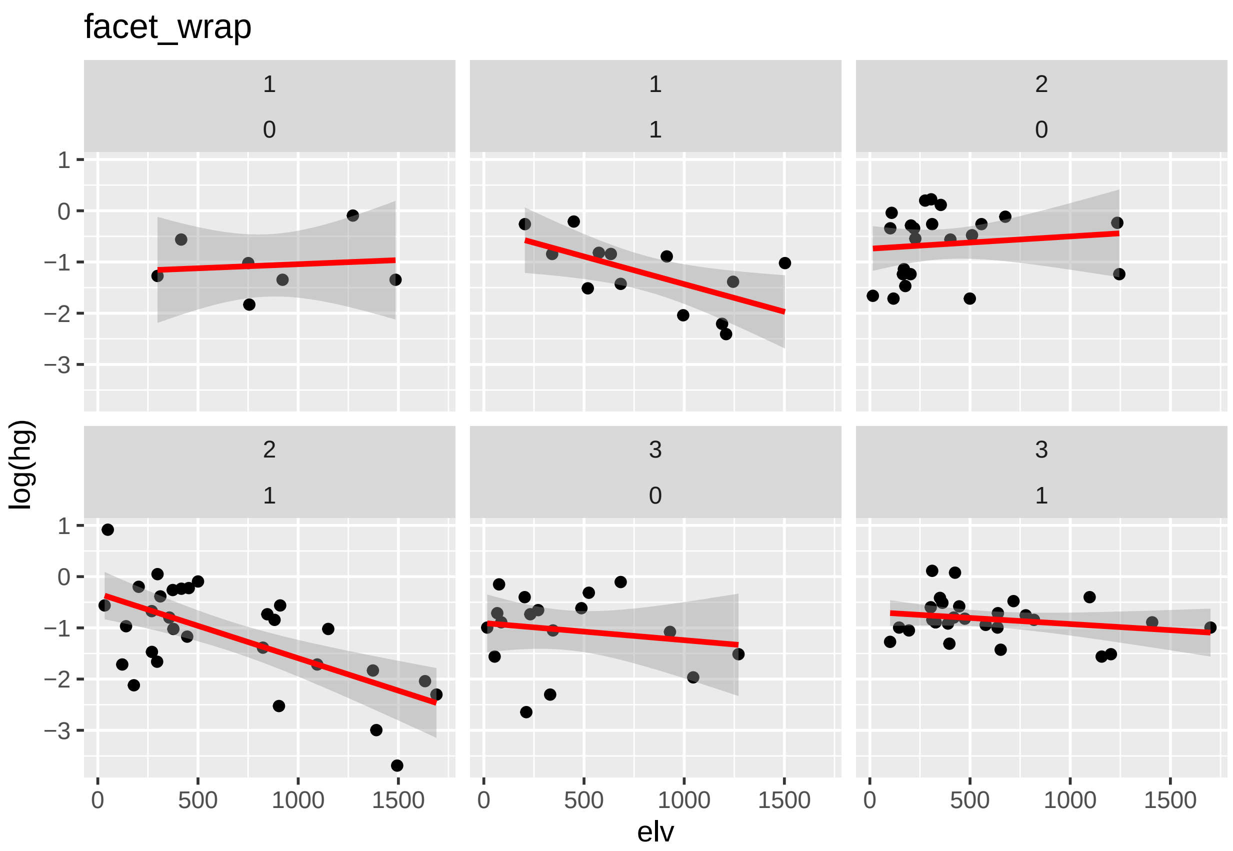
2.5.2 `facet_wrap`

The specification of faceting variables in `facet_wrap` is of the form `~ a + b`

```
fac1 <- ggplot(fish, aes(elv, log(hg))) + geom_point() +
  geom_smooth(colour = "red", method = "lm")
fac1
```



```
fac2 <- fac1 + facet_wrap(~ lt + dam) + labs(title = "facet_wrap")
fac2
```



2.5.3 Difference between faceting and grouping

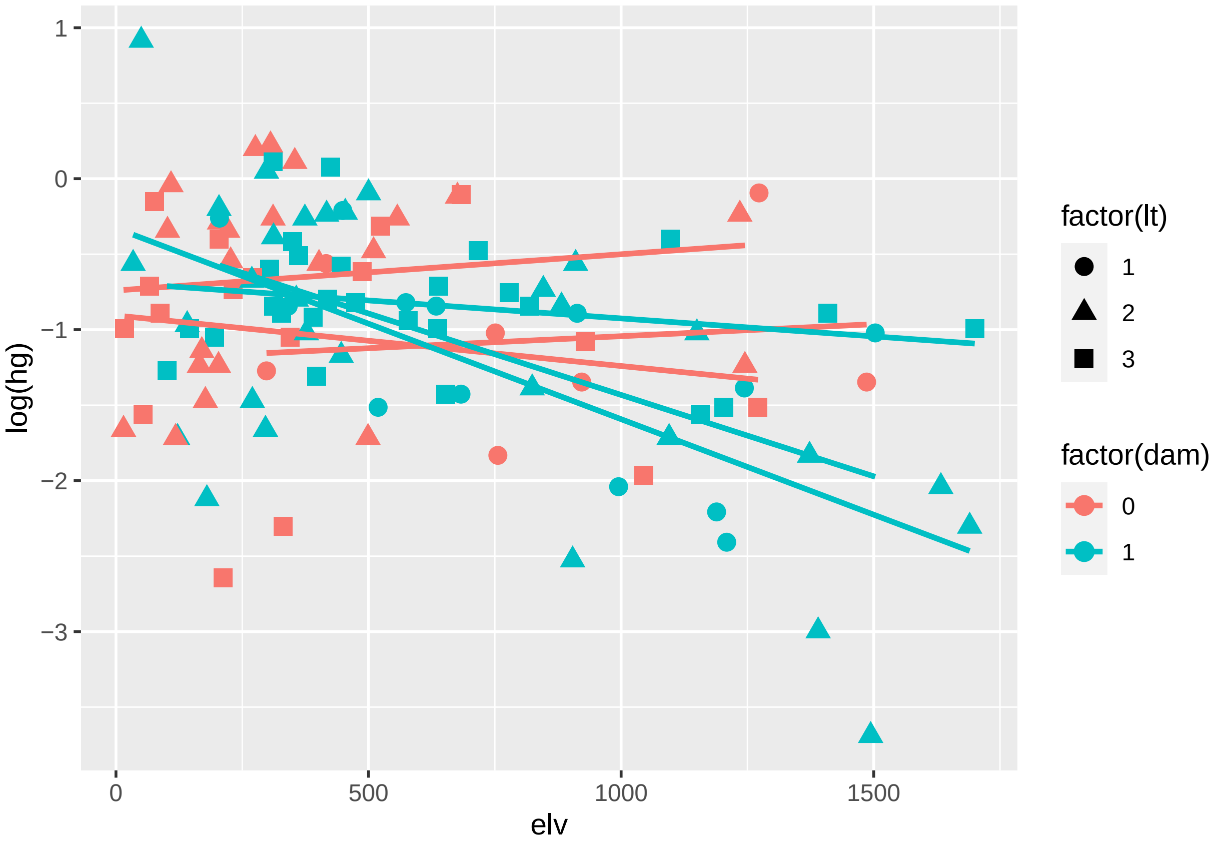
- With **faceting**: each group is quite far apart in its own panel, and there is no overlap between the groups. If there are small differences between groups, then these are harder to detect.
- When using **grouping**, the groups are close together and may overlap, but small differences are easier to detect.

With faceting, you can split in two dimensions and that is harder with grouping (using different colours and different symbols).

Example *fish*

Example of grouping:

```
ggplot(fish, aes(elv, log(hg), colour = factor(dam), shape = factor(lt))) +
  geom_point(size = 3) + geom_smooth(method = "lm", se = FALSE)
```



2.5.4 Faceting by continuous variables

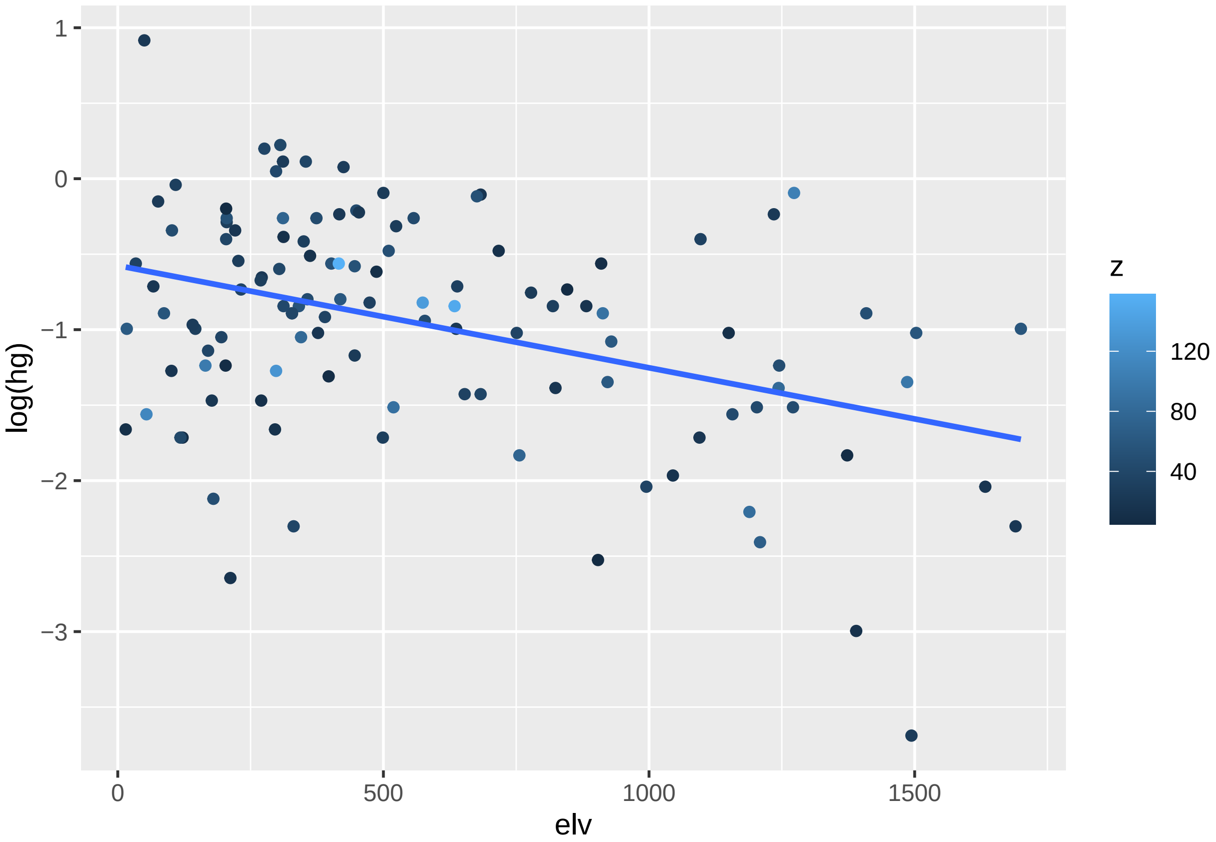
You first need to convert the continuous variables into discrete categories.

Assume that we want to see the scatterplot of `log(hg)` versus elevation **according to the value of z** (max. depth of the lake):

2.5.4.1 Method 1: Grouping by a continuous variable

Color points by value of continuous variable

```
ggplot(fish, aes(elv, log(hg), colour = z)) + geom_point() +
  geom_smooth(method = "lm", se = FALSE)
```



2.5.4.2 Method 2: Categorize your continuous variable

Step 1: Convert the continuous variable into a variable with discrete categories.

```
fish$z_cat1 <- cut_interval(fish$z, n=3)
# Or you can use
fish$z_cat2 <- cut_number(fish$z, n=3)

xtabs(~ fish$z_cat1)
```

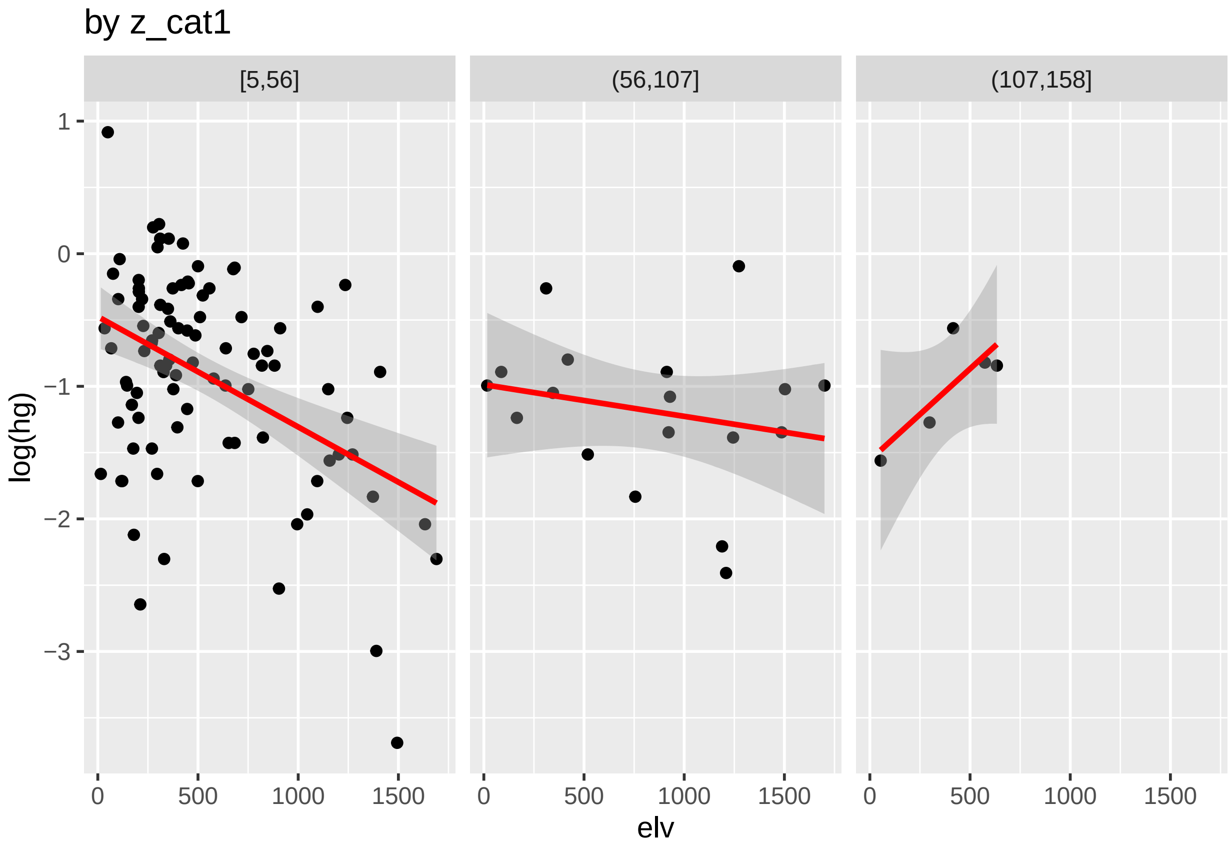
```
## fish$z_cat1
##    [5,56]  (56,107] (107,158]
##        94        18         5
```

```
xtabs(~ fish$z_cat2)
```

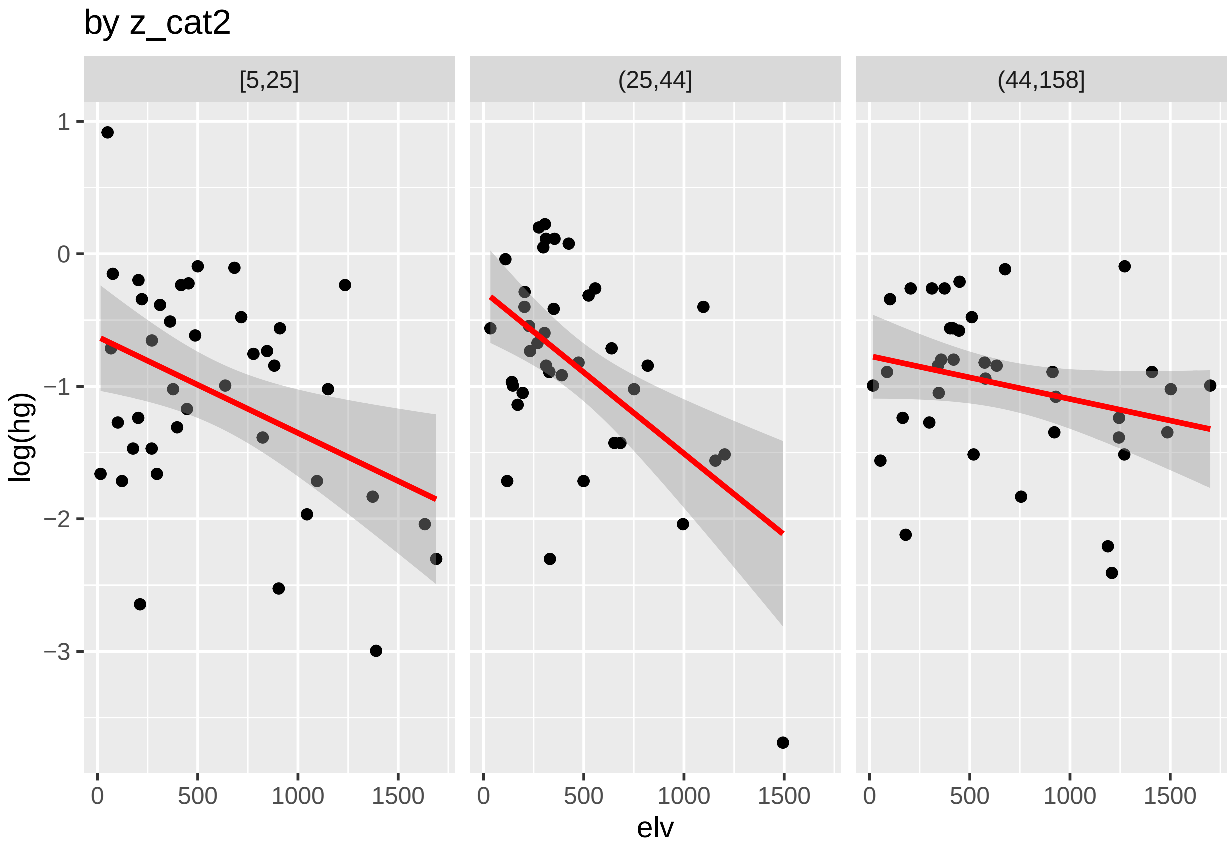
```
## fish$z_cat2
##   [5,25]  (25,44] (44,158]
##       40       39       38
```

Step 2: We now can use this variable for faceting

```
fac1 <- ggplot(fish, aes(elv, log(hg))) + geom_point() +
  geom_smooth(colour = "red", method = "lm")
fac2 <- fac1 + facet_wrap(~ z_cat1) + labs(title = "by z_cat1")
fac2
```



```
fac3 <- fac1 + facet_wrap(~ z_cat2) + labs(title = "by z_cat2")
fac3
```



3 Extras

3.1 Themes

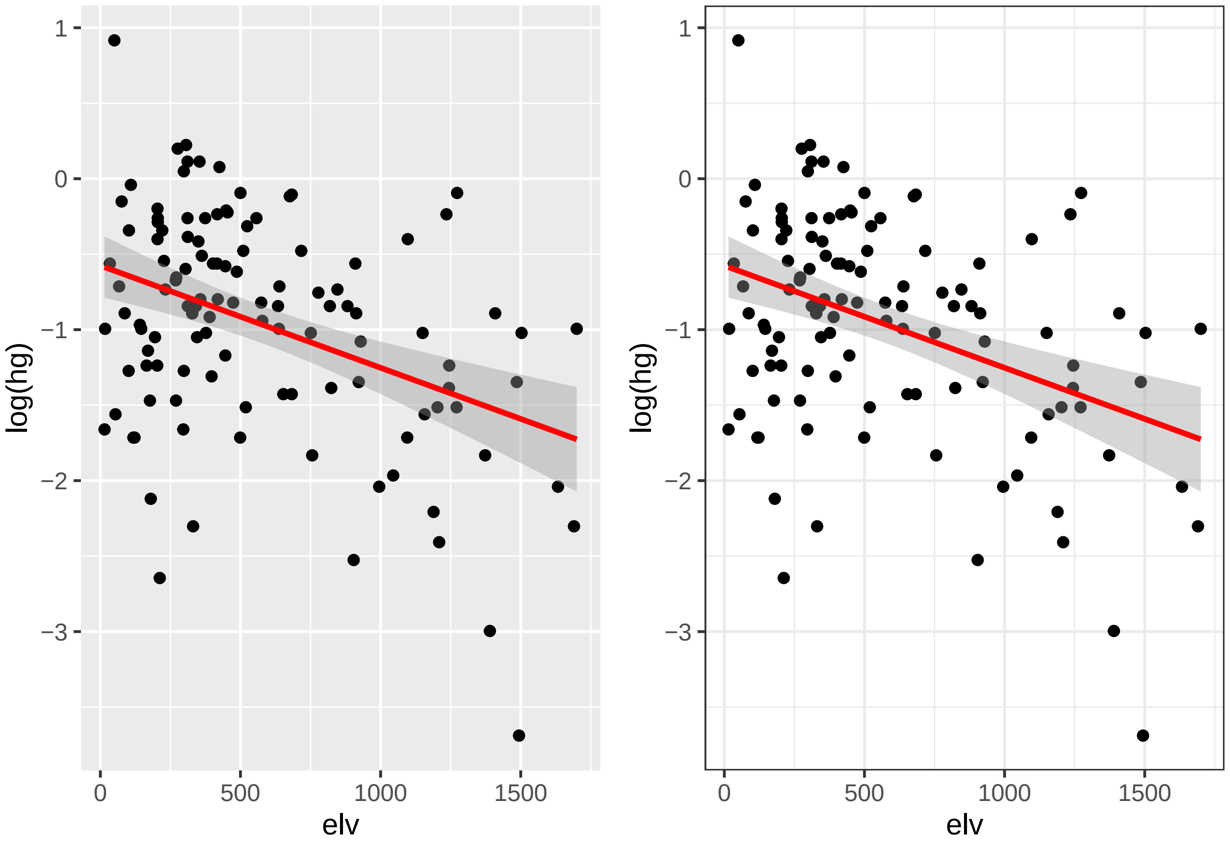
There are two built-in themes: `theme_gray()` and `theme_bw()`.

- The **default** `theme_gray()` uses a very light grey background with white grid lines.
- `theme_bw()` uses a white background with dark grey grid lines.

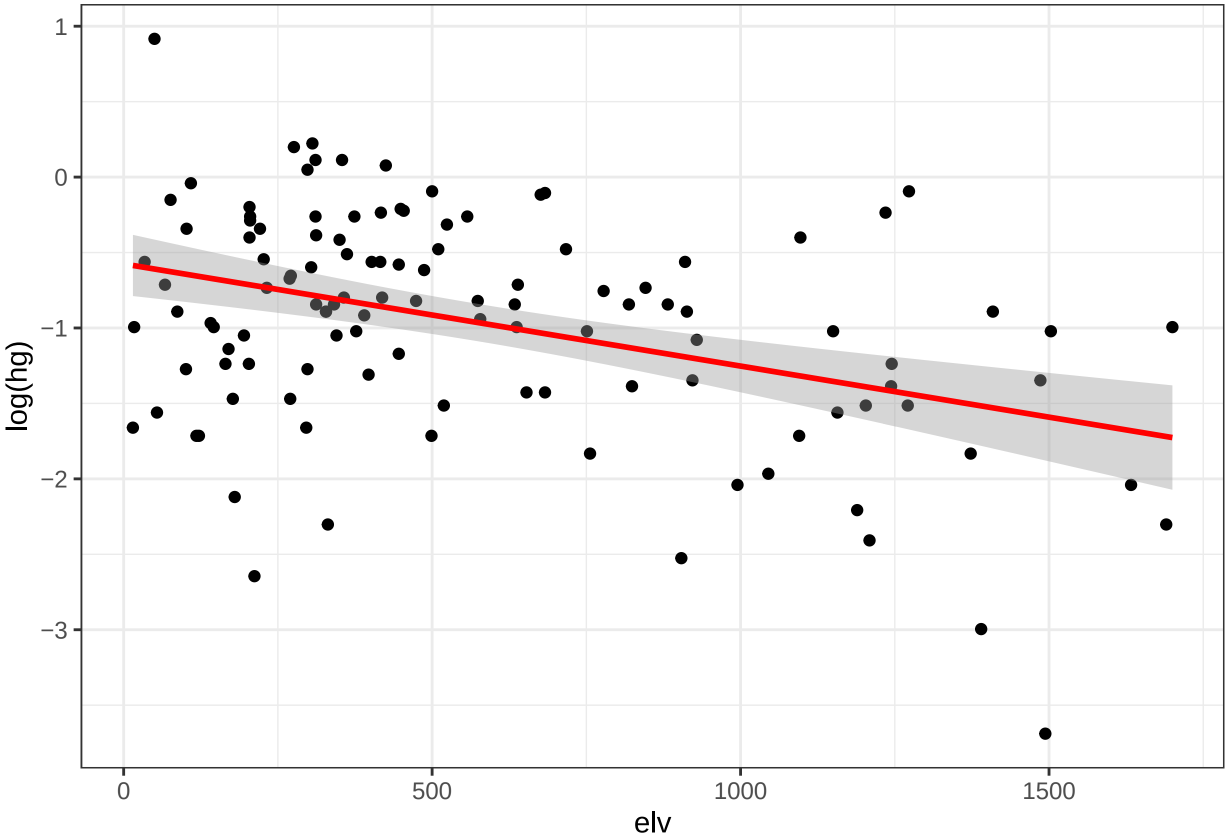
Example *fish*

Example: use of theme

```
fac1 <- ggplot(fish, aes(elv, log(hg))) + geom_point() +
  geom_smooth(colour = "red", method = "lm")
fac1
# You can override the theme for a single plot
fac2 <- fac1 + theme_bw()
fac2
```



```r
# Affecting all plots
previous_theme <- theme_set(theme_bw())
fac1
```



3.2 Multiple plots on the same page

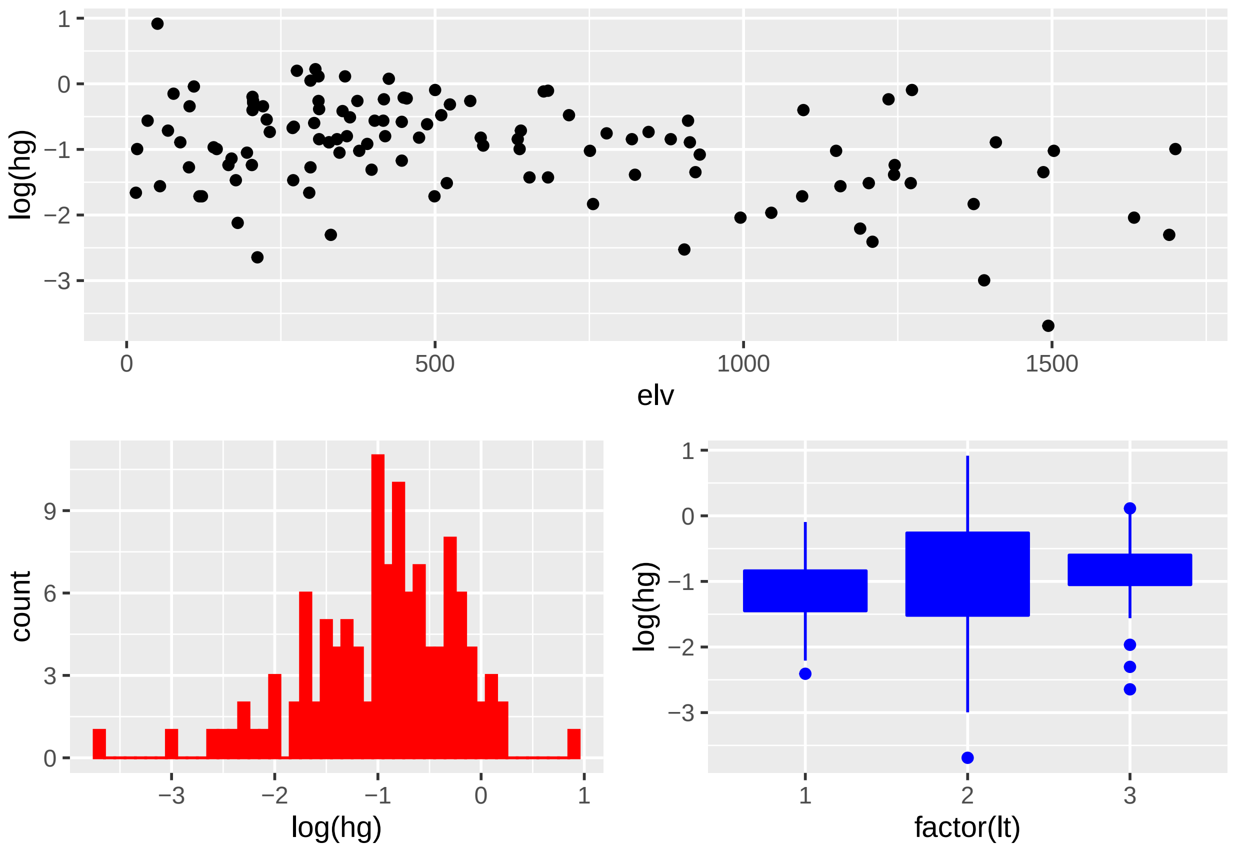
A viewport is a specified region in the entire plotting area. By customizing the viewport, you can arrange a set of plots.

First create the plots, assign them to objects and then plot the objects.

```
plot1 <- ggplot(fish, aes(elv,log(hg))) + geom_point()
plot2 <- ggplot(fish, aes(log(hg))) +
  geom_histogram(binwidth = 0.1, colour = "red", fill = "red" )
plot3 <- ggplot(fish, aes(factor(lt),log(hg))) +
  geom_boxplot(colour = "blue", fill = "blue" )
```

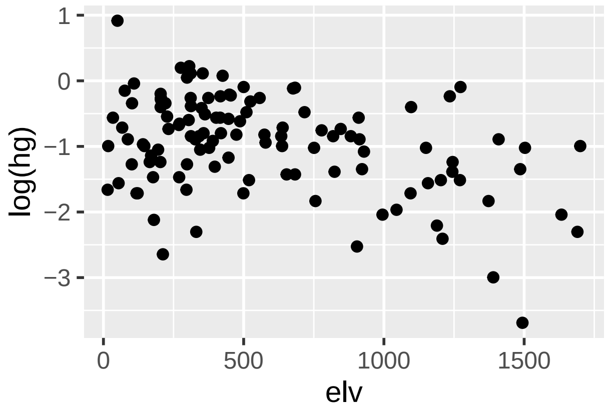
3.2.1 Use rectangular grids: use grid.layout()

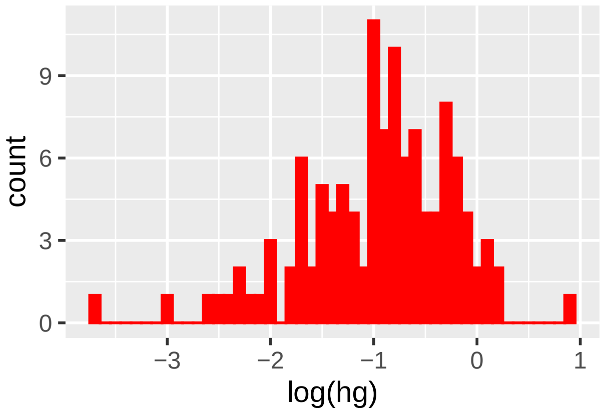
```
grid.newpage()
pushViewport(viewport(layout=grid.layout(2,2)))
vplayout <- function(x,y){
  viewport(layout.pos.row=x, layout.pos.col=y)}
print(plot1, vp=vplayout(1,1:2))
print(plot2, vp=vplayout(2,1))
print(plot3, vp=vplayout(2,2))
```

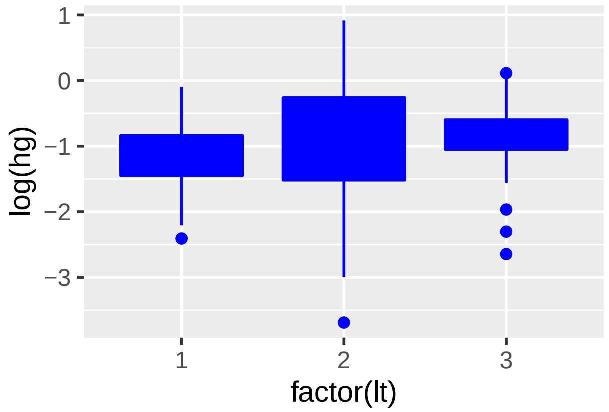


### Use rectangular grids: use `ggarrange`

```
ggarrange(plot1,plot2, plot3, ncol = 2, nrow = 2)
```







3.3 Save your output

Save your output to a *pdf* file

```
plot1 <- qplot(elv, log(hg), data = fish, geom = c("point"))
ggsave(file = "output1.pdf", plot = plot1)
```

Remark:
When you use *Latex*, it is recommended to save your work to a *.ps* file

Save your output to a *ps* file

```
ggsave(file = "output2.ps", plot = plot1)
```

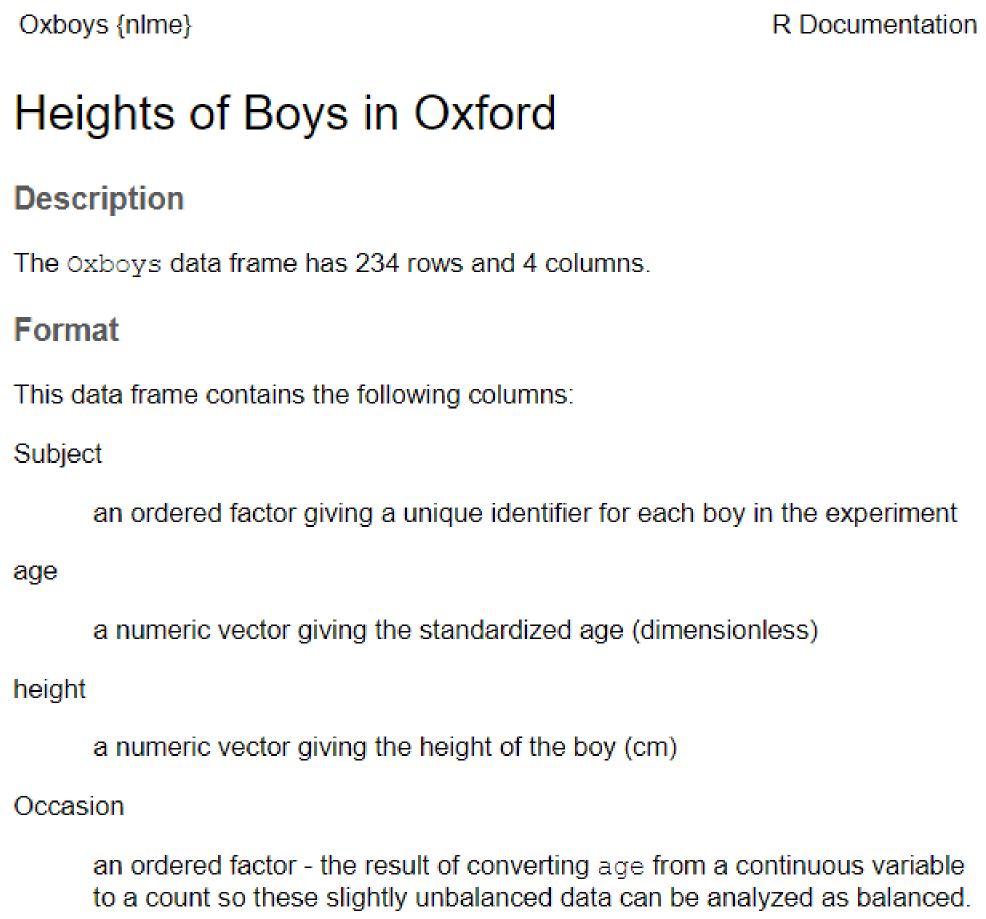
```
## Saving 6.5 x 4.5 in image
```

4 Applications

4.1 Profile plots for visualizing longitudinal data

In this section, we are using the data set `Oxboys` from the package `nlme`.

```
install.packages("nlme")
library(nlme)
?Oxboys
```



Oxboys {nlme} R Documentation

Heights of Boys in Oxford

Description

The `Oxboys` data frame has 234 rows and 4 columns.

Format

This data frame contains the following columns:

Subject

an ordered factor giving a unique identifier for each boy in the experiment

age

a numeric vector giving the standardized age (dimensionless)

height

a numeric vector giving the height of the boy (cm)

Occasion

an ordered factor - the result of converting `age` from a continuous variable to a count so these slightly unbalanced data can be analyzed as balanced.

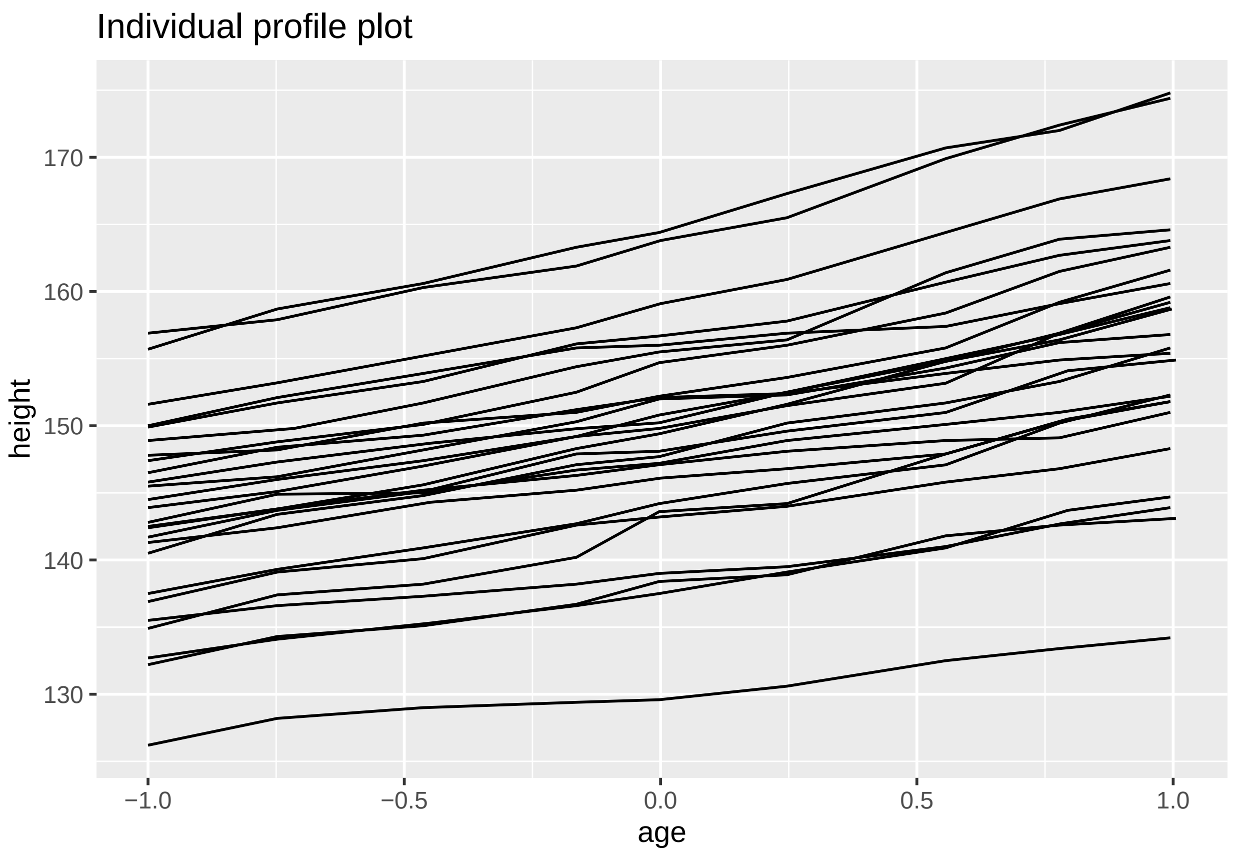
```r
head(Oxboys, n = 12)
```

```
## Grouped Data: height ~ age | Subject
##    Subject     age height Occasion
## 1        1 -1.0000  140.5        1
## 2        1 -0.7479  143.4        2
## 3        1 -0.4630  144.8        3
## 4        1 -0.1643  147.1        4
## 5        1 -0.0027  147.7        5
## 6        1  0.2466  150.2        6
## 7        1  0.5562  151.7        7
## 8        1  0.7781  153.3        8
## 9        1  0.9945  155.8        9
## 10       2 -1.0000  136.9        1
## 11       2 -0.7479  139.1        2
## 12       2 -0.4630  140.1        3
```

- Individual profile plots: specify **subject as grouping variable**.

Visualize longitudinal data

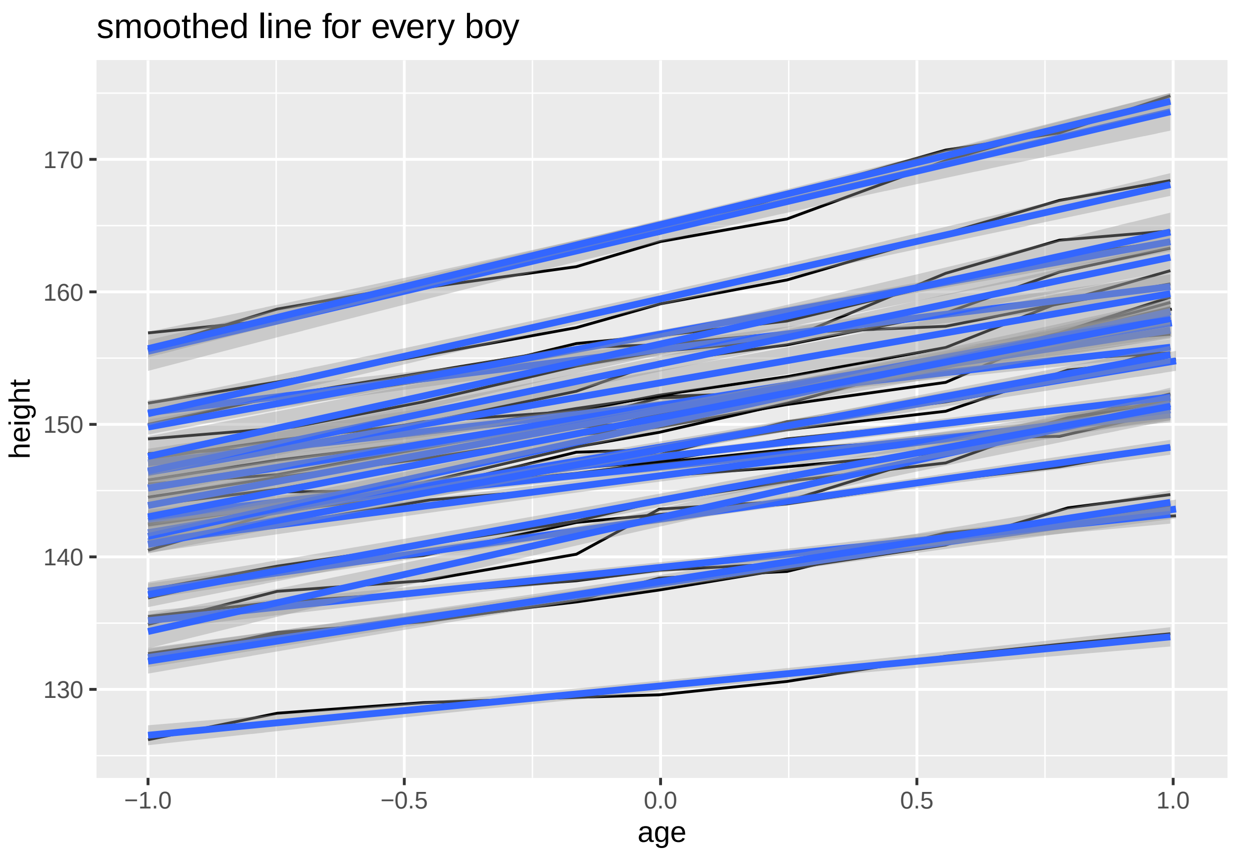
```r
pl1 <- ggplot(Oxboys, aes(age, height, group = Subject))
pl2 <- pl1 + geom_line() + labs(title = "Individual profile plot")
pl2
```



- Individual profile plot with common trend

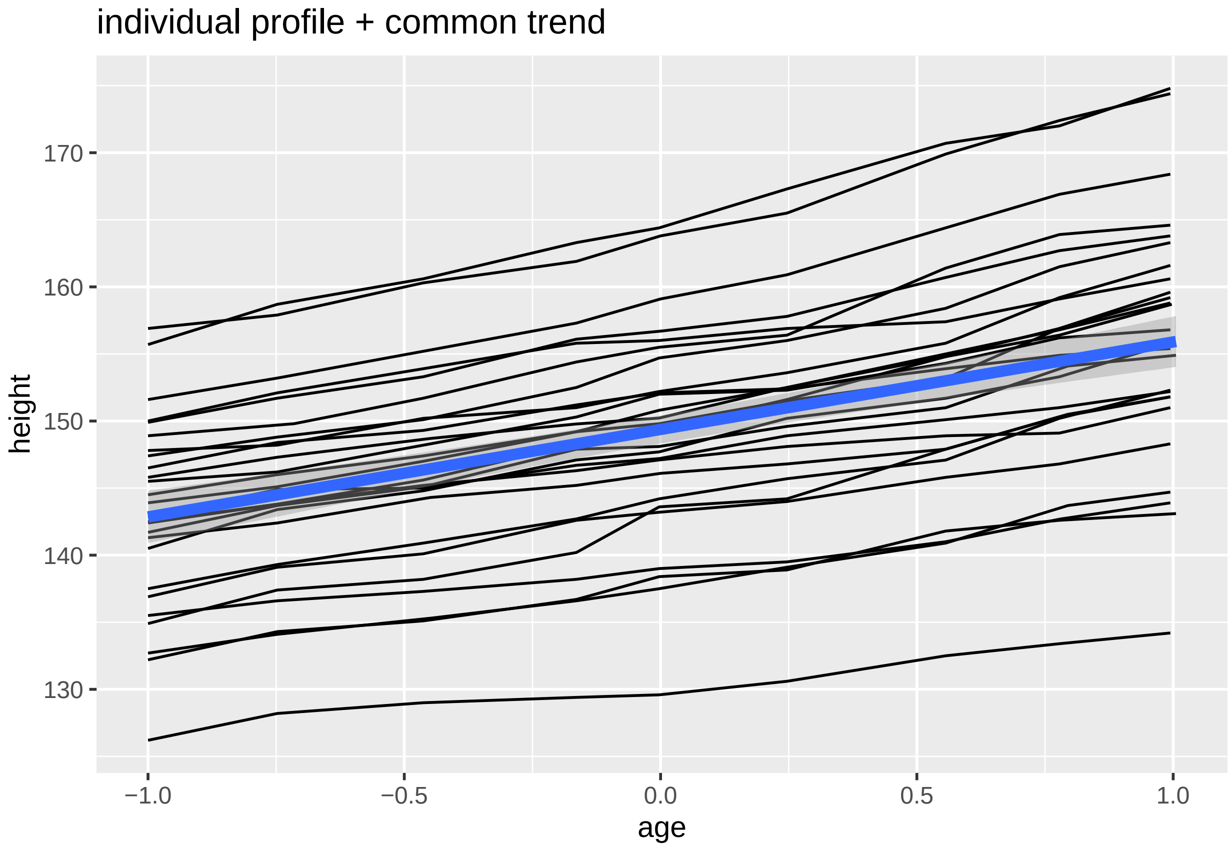
Adding a smoothed line for every boy

```
pl3 <- pl2 + geom_smooth(method = "lm", size = 1.2) +
  labs(title = "smoothed line for every boy")
pl3
```



Adding a smooth line based on ages and heights of all the boys.
Specifying `group = 1` indicates that you want a single line (and consider all data points as 1 group)

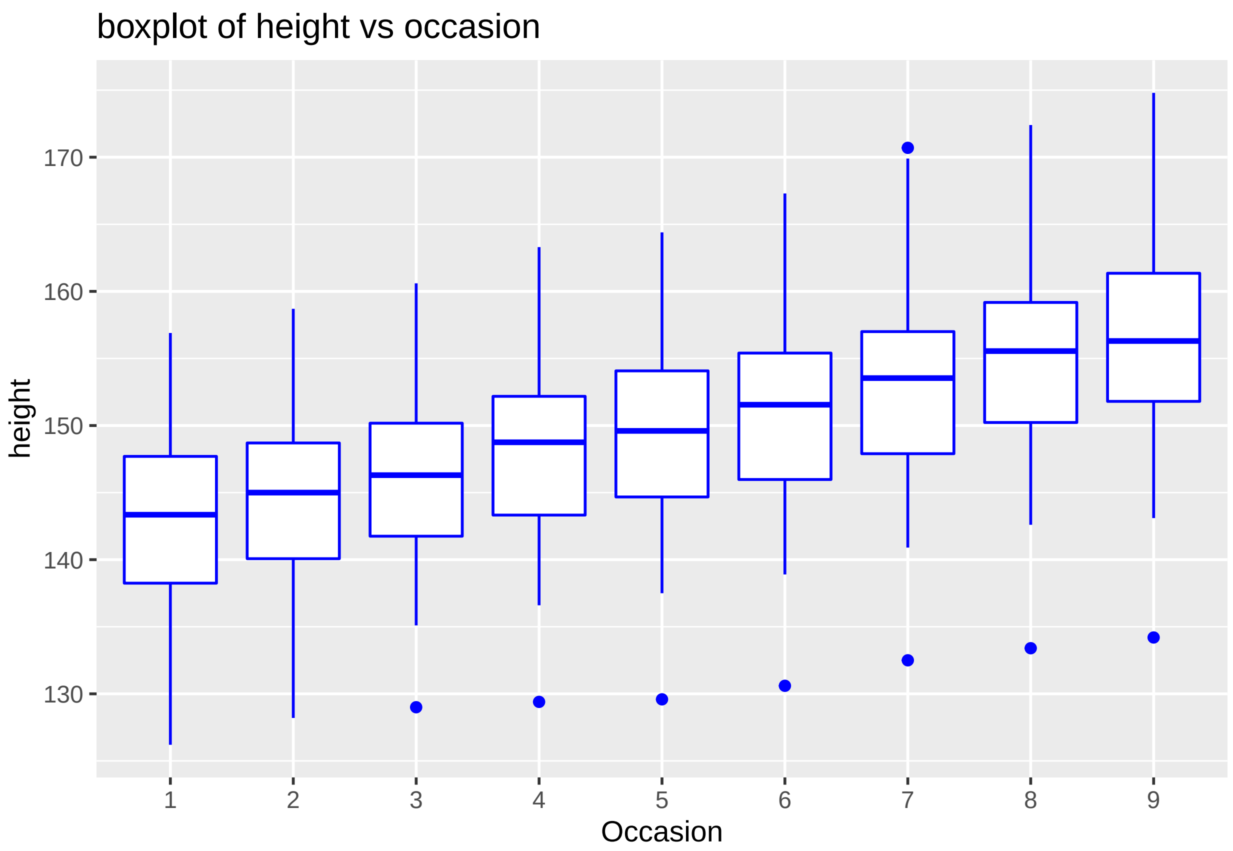
```
pl4 <- pl2 + geom_smooth(aes(group = 1), method = "lm", size = 2) +
  labs(title = "individual profile + common trend")
pl4
```



Connecting data points across groups

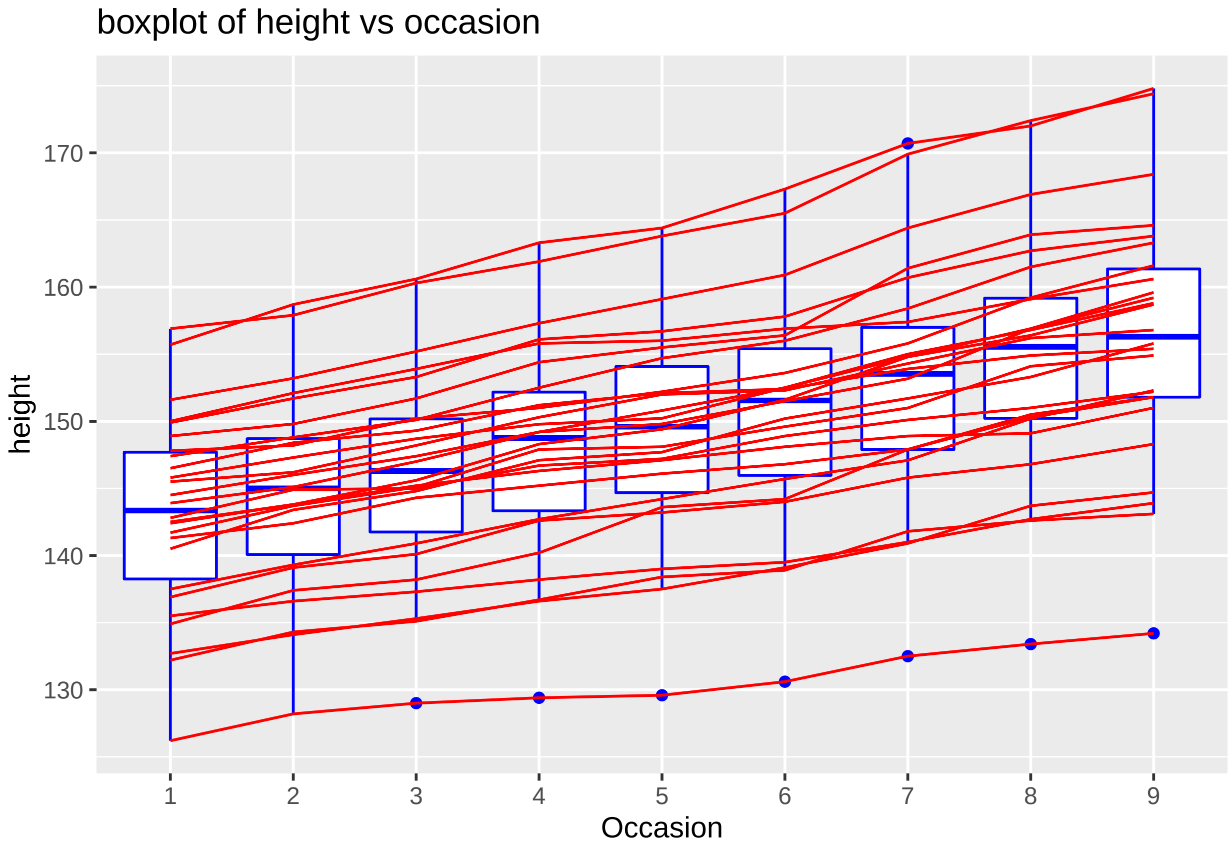
Construct a boxplot of `height` versus `Occasion`.

```
pl1 <- ggplot(Oxboys, aes(Occasion, height))
pl2 <- pl1 + geom_boxplot(colour = "blue") + labs(title = "boxplot of height vs occasion")
pl2
```



If you want to connect the values per boy over the several occasions

```
pl3 <- pl2 + geom_line(aes(group = Subject), colour = "red")
pl3
```

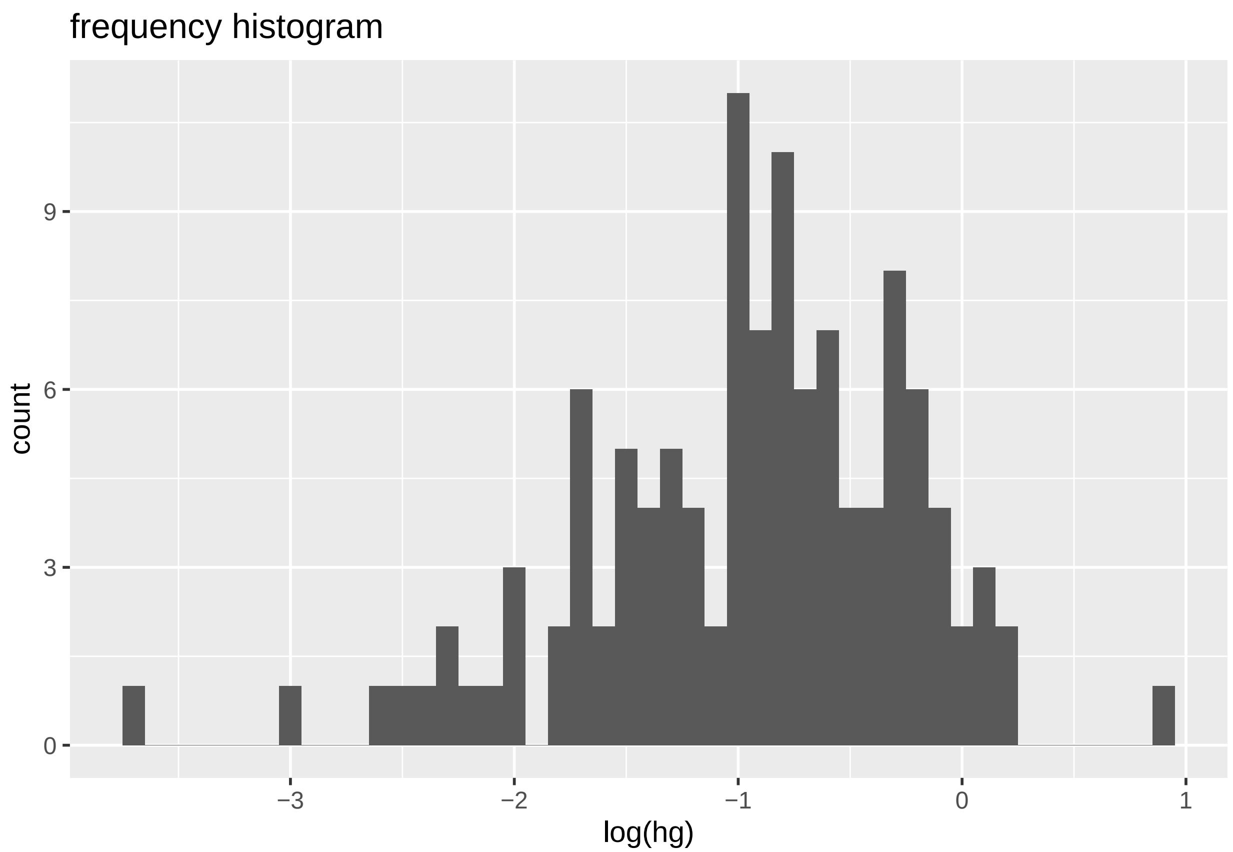


4.2 Create frequency histogram with density curve

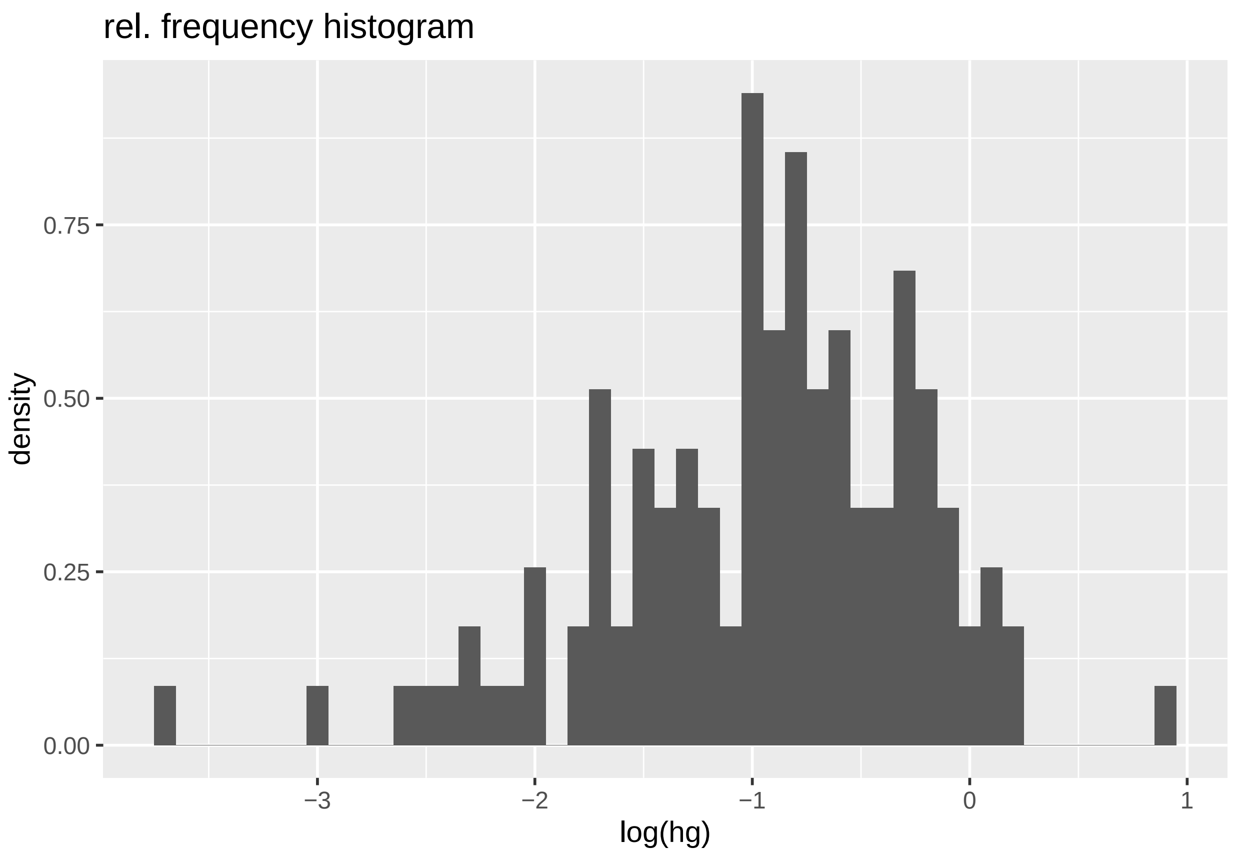
In this section, we use the data set `fish`.

Construct a frequency and relative frequency histogram overlayed with a density curve

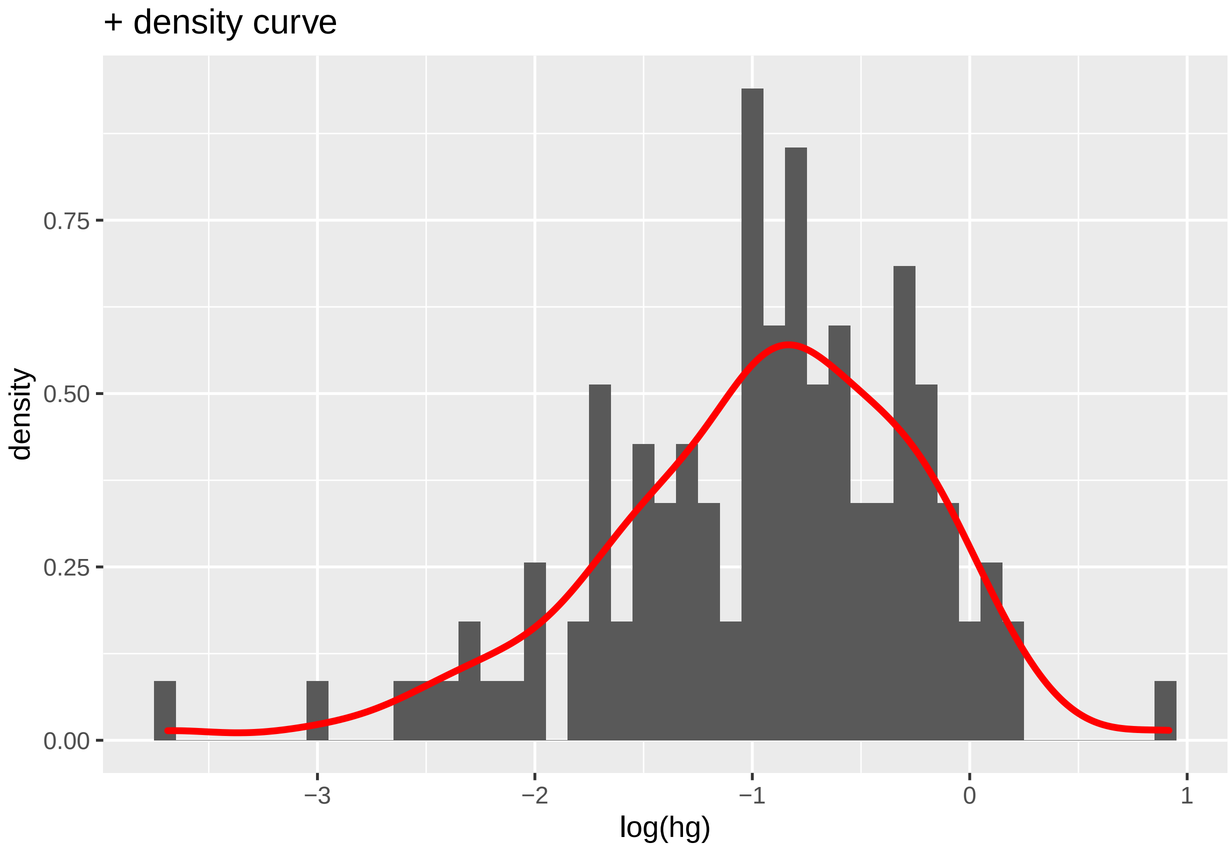
```
pl1 <- ggplot(fish, aes(log(hg)))
pl2 <- pl1 + geom_histogram(binwidth = 0.1) + labs(title = "frequency histogram")
pl2
```



```
pl3 <- pl1 + geom_histogram(aes(y = ..density..), binwidth = 0.1) +
  labs(title = "rel. frequency histogram")
pl3
```

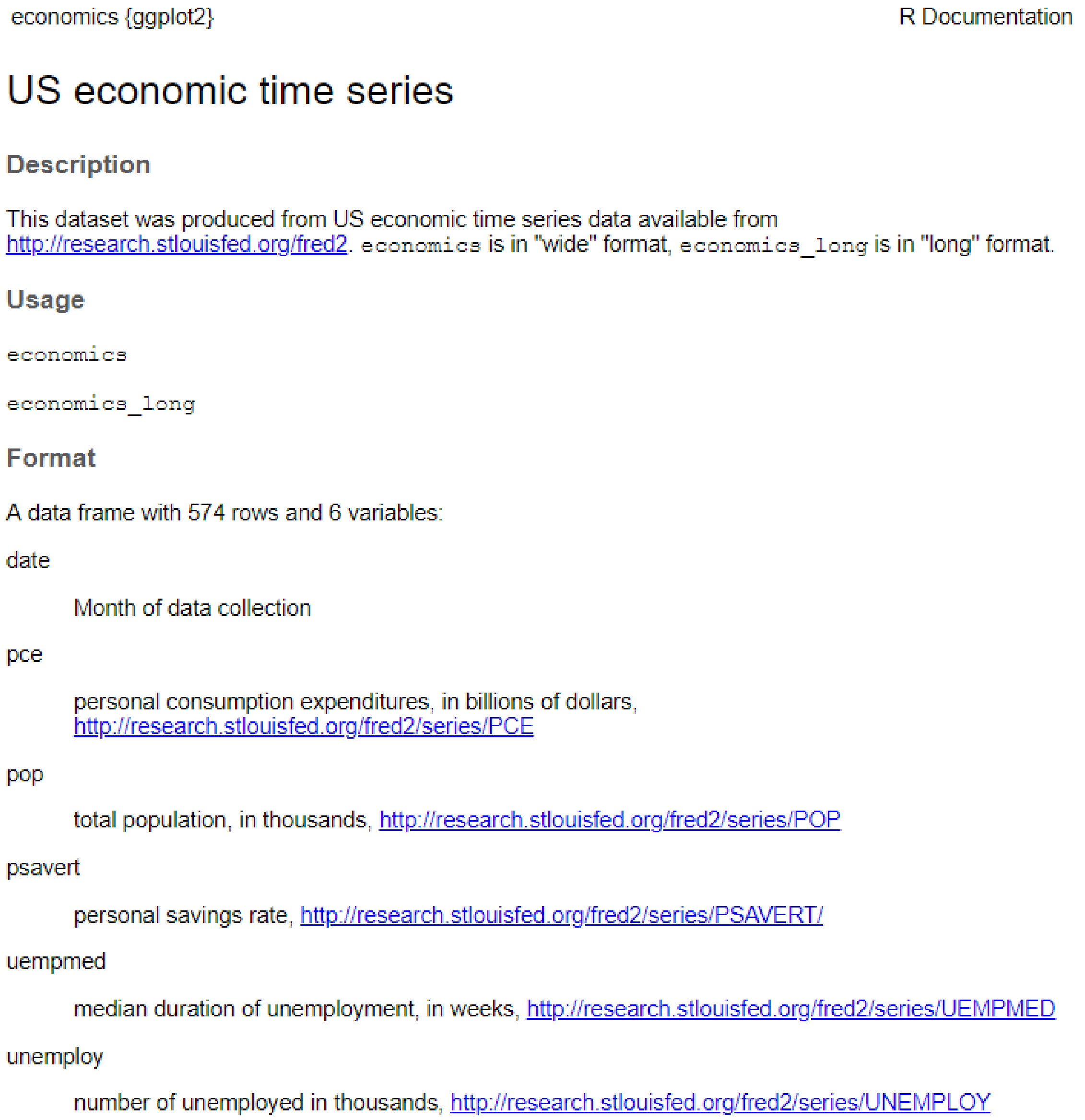


```
pl4 <- pl3 + geom_density(colour = "red", size = 1.2) + labs(title = "+ density curve")
pl4
```



4.3 Visualize multiple variables on same plot (e.g. time series)

In this section, we use the data set `economics` from the package `ggplot2`.



economics {ggplot2} R Documentation

US economic time series

Description

This dataset was produced from US economic time series data available from http://research.stlouisfed.org/fred2. `economics` is in "wide" format, `economics_long` is in "long" format.

Usage

```
economics

economics_long
```

Format

A data frame with 574 rows and 6 variables:

date

Month of data collection

pce

personal consumption expenditures, in billions of dollars, http://research.stlouisfed.org/fred2/series/PCE

pop

total population, in thousands, http://research.stlouisfed.org/fred2/series/POP

psavert

personal savings rate, http://research.stlouisfed.org/fred2/series/PSAVERT/

uempmed

median duration of unemployment, in weeks, http://research.stlouisfed.org/fred2/series/UEMPMED

unemploy

number of unemployed in thousands, http://research.stlouisfed.org/fred2/series/UNEMPLOY

```
head(economics, 10)
```

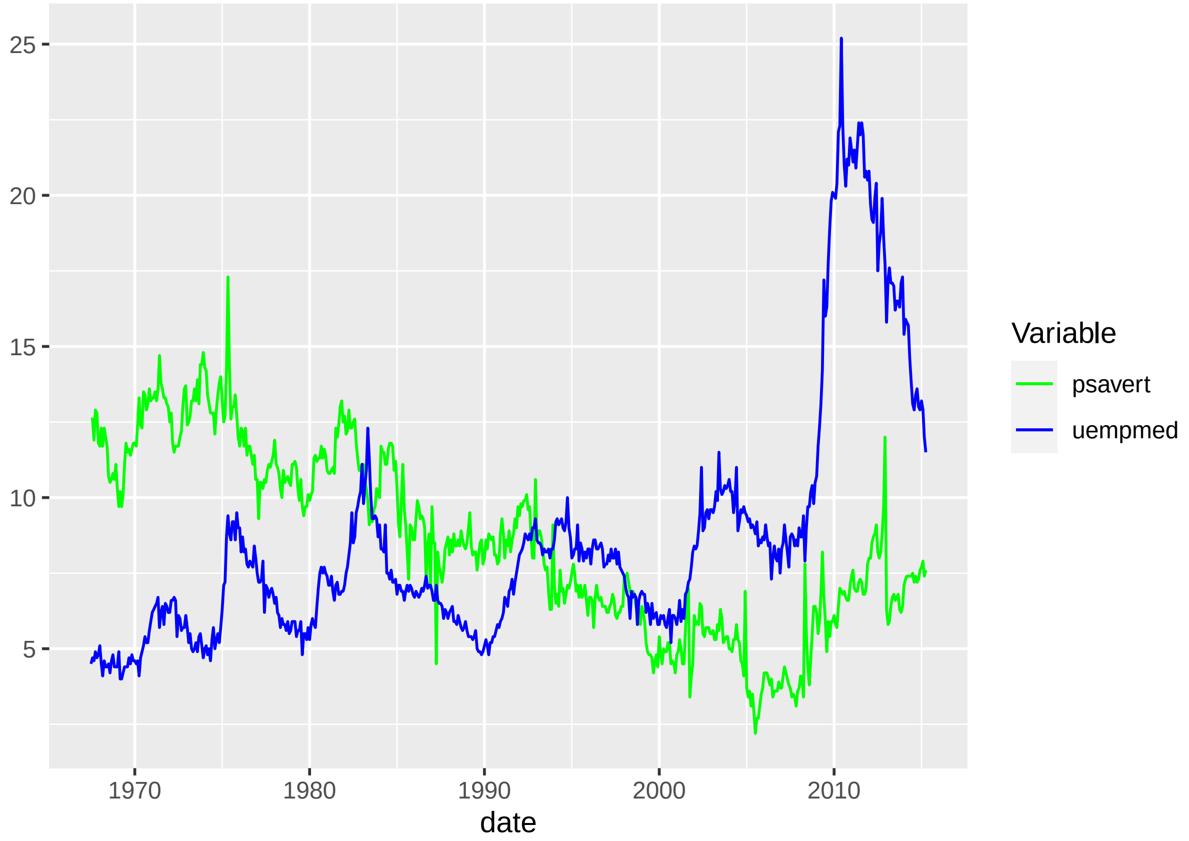
```
## # A tibble: 10 x 6
##    date         pce    pop psavert uempmed unemploy
##    <date>     <dbl>  <dbl>   <dbl>   <dbl>    <dbl>
##  1 1967-07-01  507. 198712    12.6     4.5     2944
##  2 1967-08-01  510. 198911    12.6     4.7     2945
##  3 1967-09-01  516. 199113    11.9     4.6     2958
##  4 1967-10-01  512. 199311    12.9     4.9     3143
##  5 1967-11-01  517. 199498    12.8     4.7     3066
##  6 1967-12-01  525. 199657    11.8     4.8     3018
##  7 1968-01-01  531. 199808    11.7     5.1     2878
##  8 1968-02-01  534. 199920    12.3     4.5     3001
##  9 1968-03-01  544. 200056    11.7     4.1     2877
## 10 1968-04-01  544  200208    12.3     4.6     2709
```

We want to visualize the personal savings rate (`psavert`) and median duration of unemployment (`uempmed`) versus `date` (from 1970 till recent).

```
# Overlaying  lines
ts1 <- ggplot(economics, aes(date))
ts2 <- ts1 + geom_line(aes(y=psavert, colour="psavert")) +
  geom_line(aes(y=uempmed, colour="uempmed"))

# To omit the labeling of the Y axis
ts3 <-  ts2 + ylab(" ")

# To adapt the coloring of the lines + add nice heading to the legend
ts4 <- ts3 + scale_color_manual(name="Variable", values=c("green","blue"))
ts4
```



5 Adding statistical summaries

Using data set `fish`, visualize the `log(hg)` value and its average value for every lake type (`lt`).

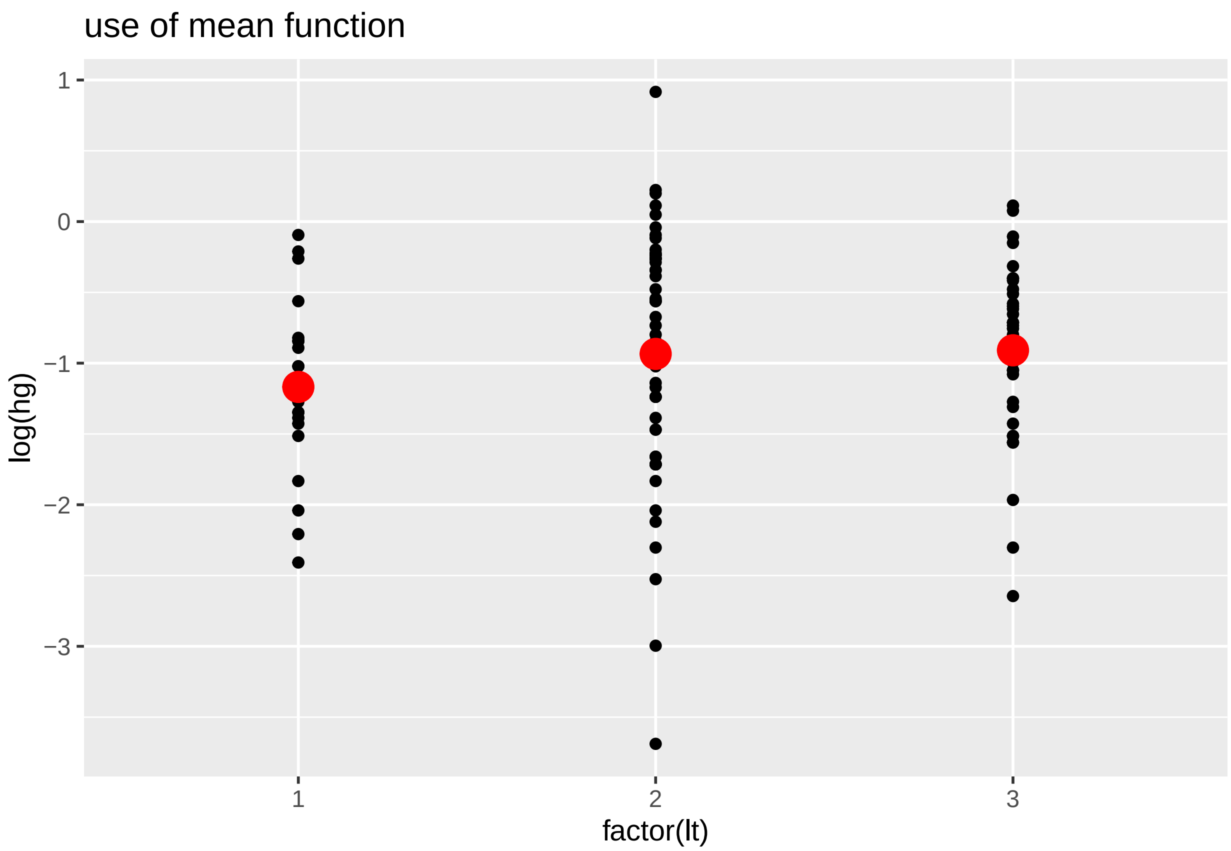
Summarize y values at every value of x: **`stat_summary()`**

5.1 Individual summary functions

You can use the functions `fun` to create simple numeric summary functions. You can use e.g. `mean()`, `median()`,...

```
install.packages("Hmisc")
library(Hmisc)
```

```
pl1 <- ggplot(fish, aes(factor(lt), log(hg)))
pl2 <- pl1 + geom_point()
pl3 <- pl2 + stat_summary(fun = "mean", geom = "point", size = 5, colour = "red")
pl4 <- pl3 + labs(title = "use of mean function")
pl4
```

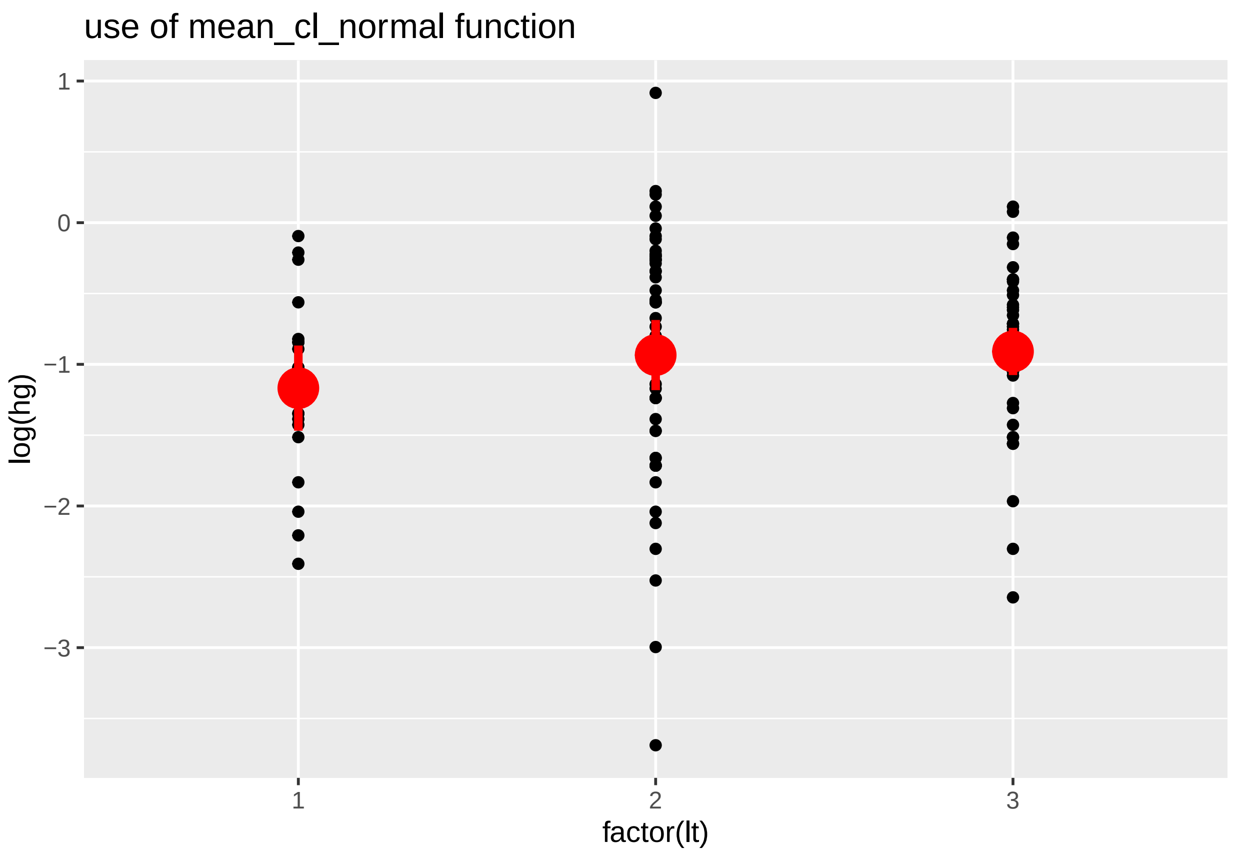


5.2 Single summary functions

`fun.data` can be used with more complex summary functions (you can write also your own summary function!).

`fun.data = ...`	middle	range
`mean_cl_normal()`	mean	se from normal approx.
`mean_cl_boot()`	mean	se from bootstrap
`median_hilow()`	median	25th and 75th percentile

```
pl1 <- ggplot(fish, aes(factor(lt), log(hg)))
pl2 <- pl1 + geom_point()
pl5 <- pl2 + stat_summary(fun.data = "mean_cl_normal", colour = "red", size = 1.5)
pl6 <- pl5 + labs(title = "use of mean_cl_normal function")
pl6
```

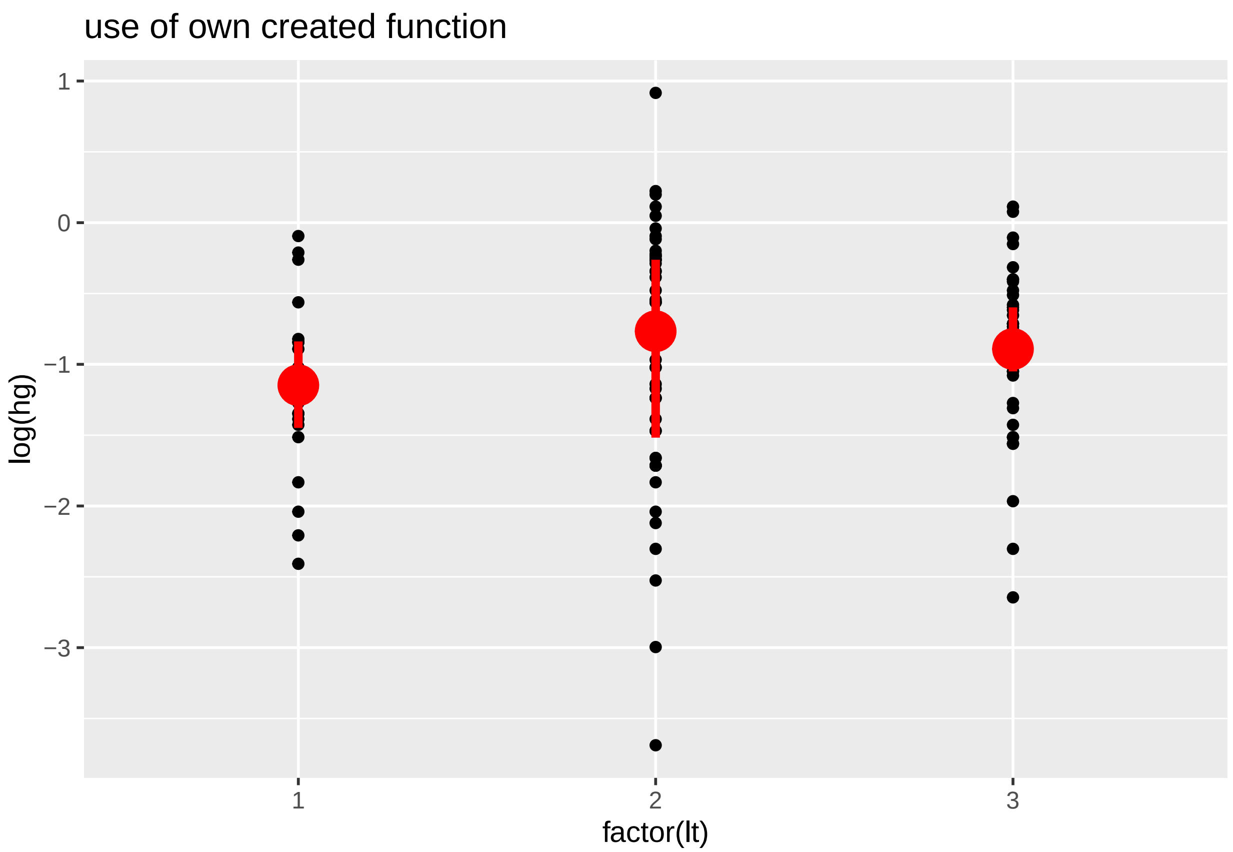


Write your own function:
Assume we want to show median with Q1 and Q3 as lower and upper bound

```
quant <- function(x)
{ q1 <- quantile(x, 0.25)
  q2 <- quantile(x, 0.50)
  q3 <- quantile(x, 0.75)
  qs <- c(q1, q2, q3)
  names(qs) <- c("ymin", "y", "ymax")
  qs}
tapply(log(fish$hg), fish$lt,quant)
```

```
## $`1`
##       ymin          y       ymax
## -1.4488692 -1.1473085 -0.8382227
##
## $`2`
##       ymin          y       ymax
## -1.5174398 -0.7662384 -0.2613648
##
## $`3`
##       ymin          y       ymax
## -1.0498221 -0.8915981 -0.5978370
```

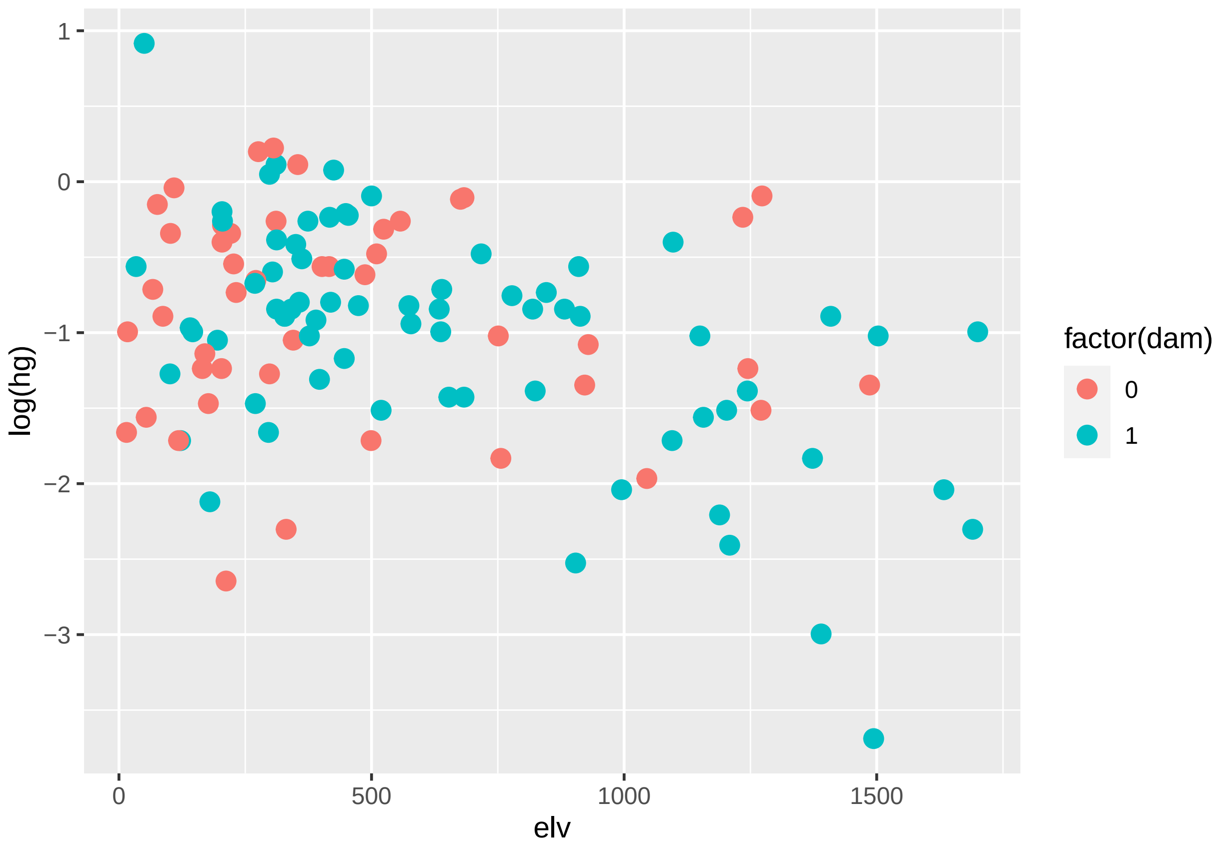
```
pl7 <- pl2 + stat_summary(fun.data = "quant", colour = "red", size = 1.5)
pl7 + labs(title = "use of own created function")
```



6 Animated graph

```
library(gganimate)
library(gifski)
```

```
g1 <- ggplot(fish, aes(elv, log(hg))) + geom_point(aes(colour = factor(dam)), size = 3)
g1
```



```r
animo1 <- g1 + transition_states(factor(dam))
animo2 <- animo1 + enter_fade() + exit_shrink()
# The command 'animo2' will now give an animated graph.
```

7 Exercises

7.1 tips data

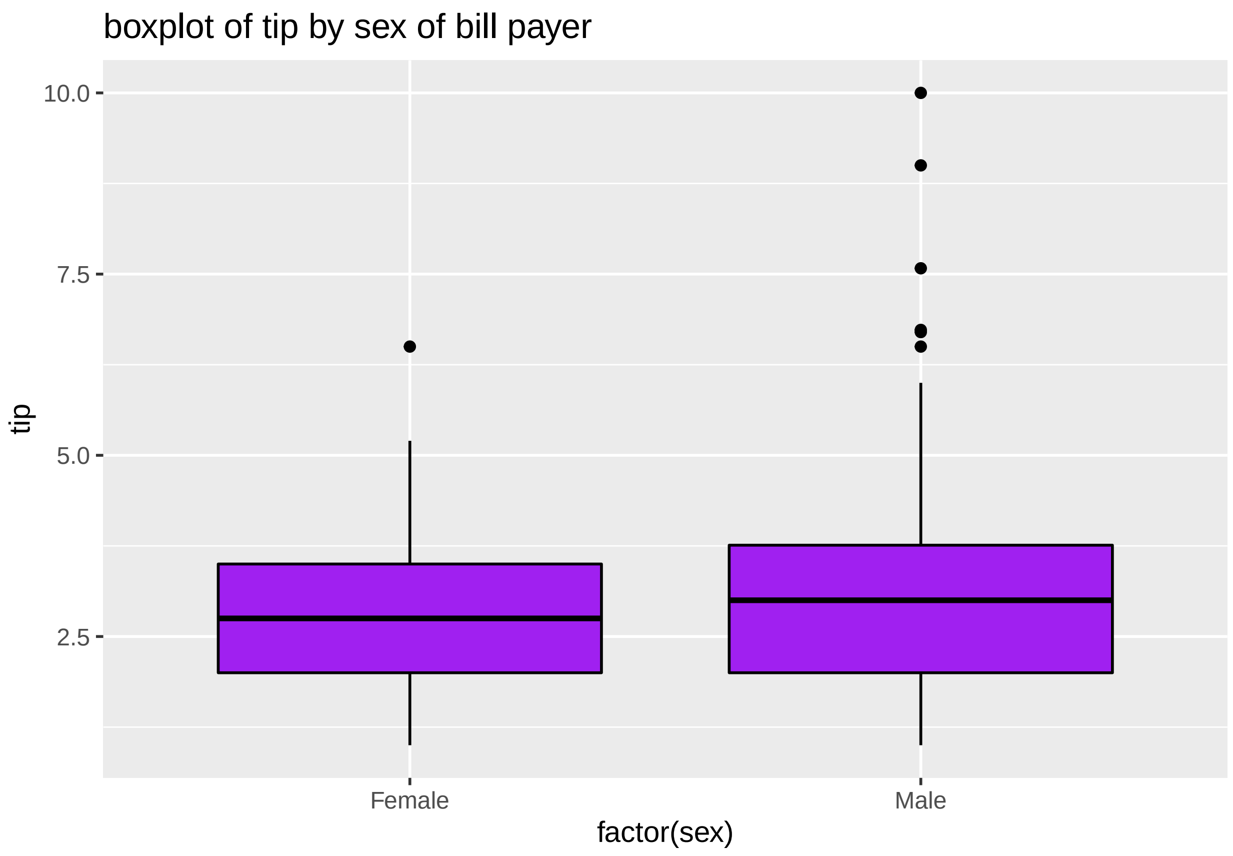
In this exercise, the data set `tips` from the package `reshape` will be used.

One waiter recorded information about each tip he received over a period of a few months working in one restaurant. He collected several variables:

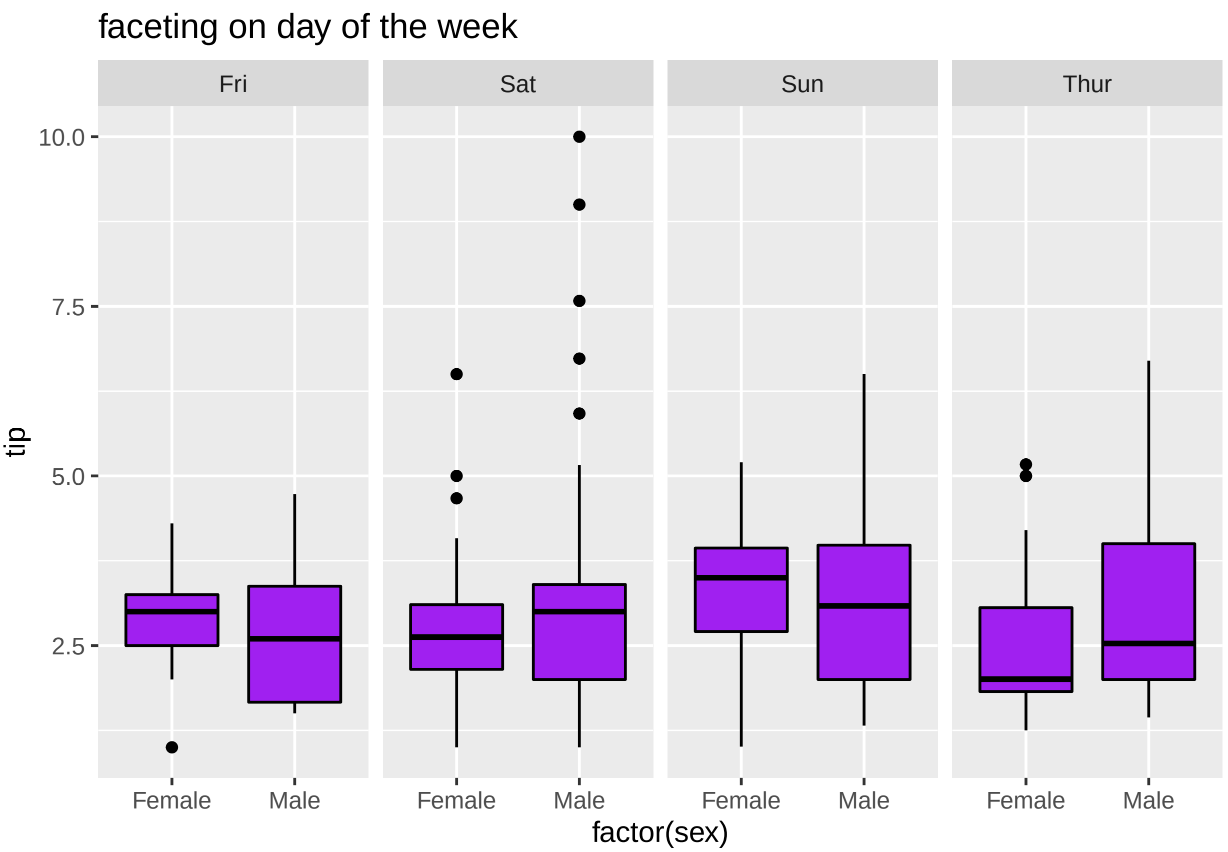
- `tip`: tip in dollars
- `total_bill`: bill in dollars
- `sex`: sex of the bill payer
- `smoker`: whether there were smokers in the party
- `day`: day of the week
- `time`: time of the day
- `size`: size of the party

In all he recorded 244 tips.

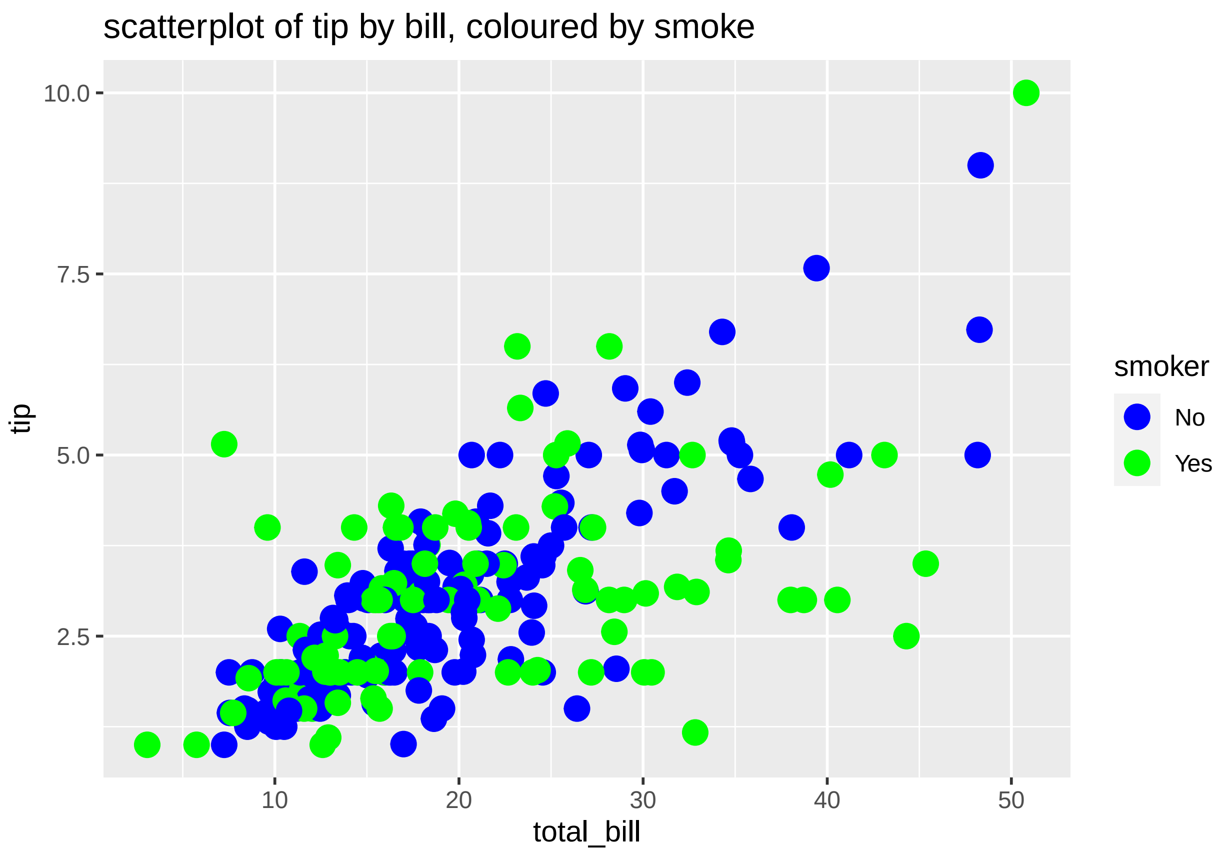
1. Create a boxplot for the tip by sex of the bill payer (use a nice color).



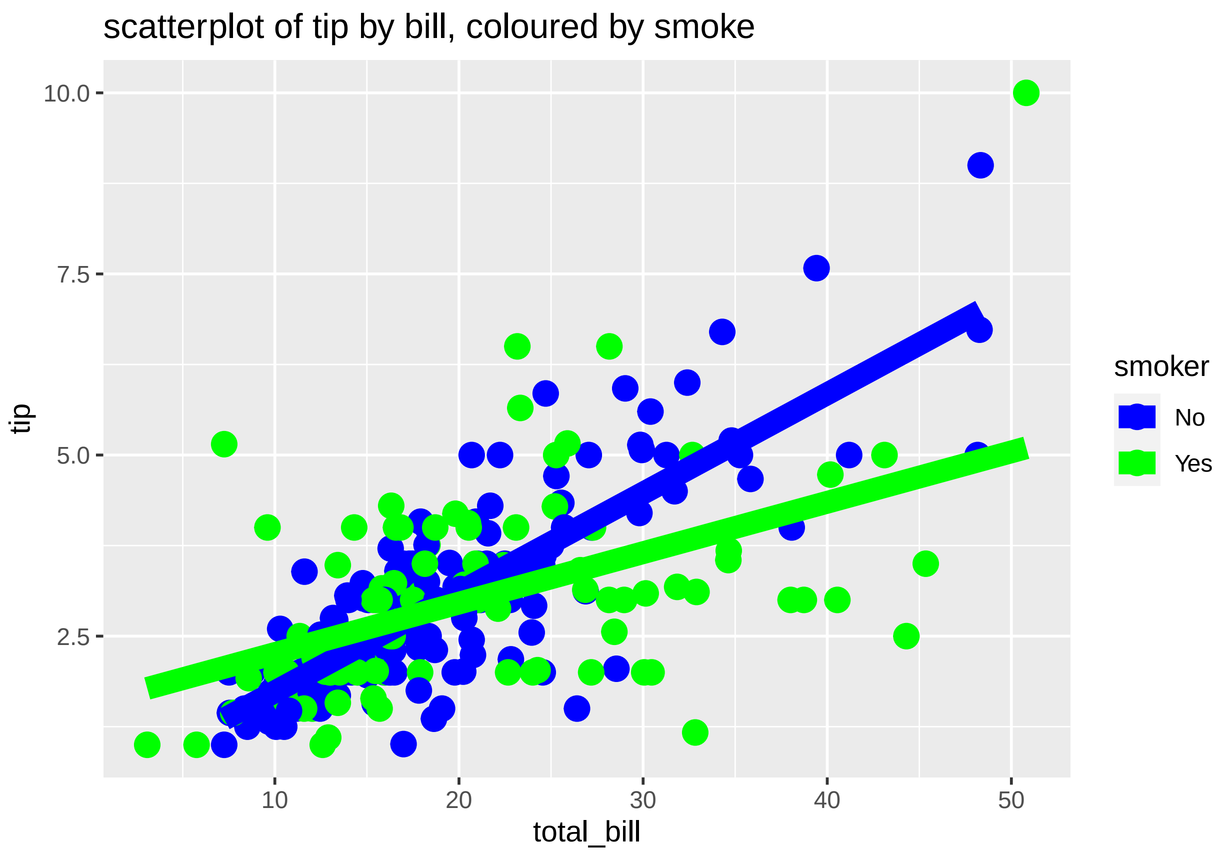
2. Create a matrix of panels of boxplots by using as faceting variable day of the week.



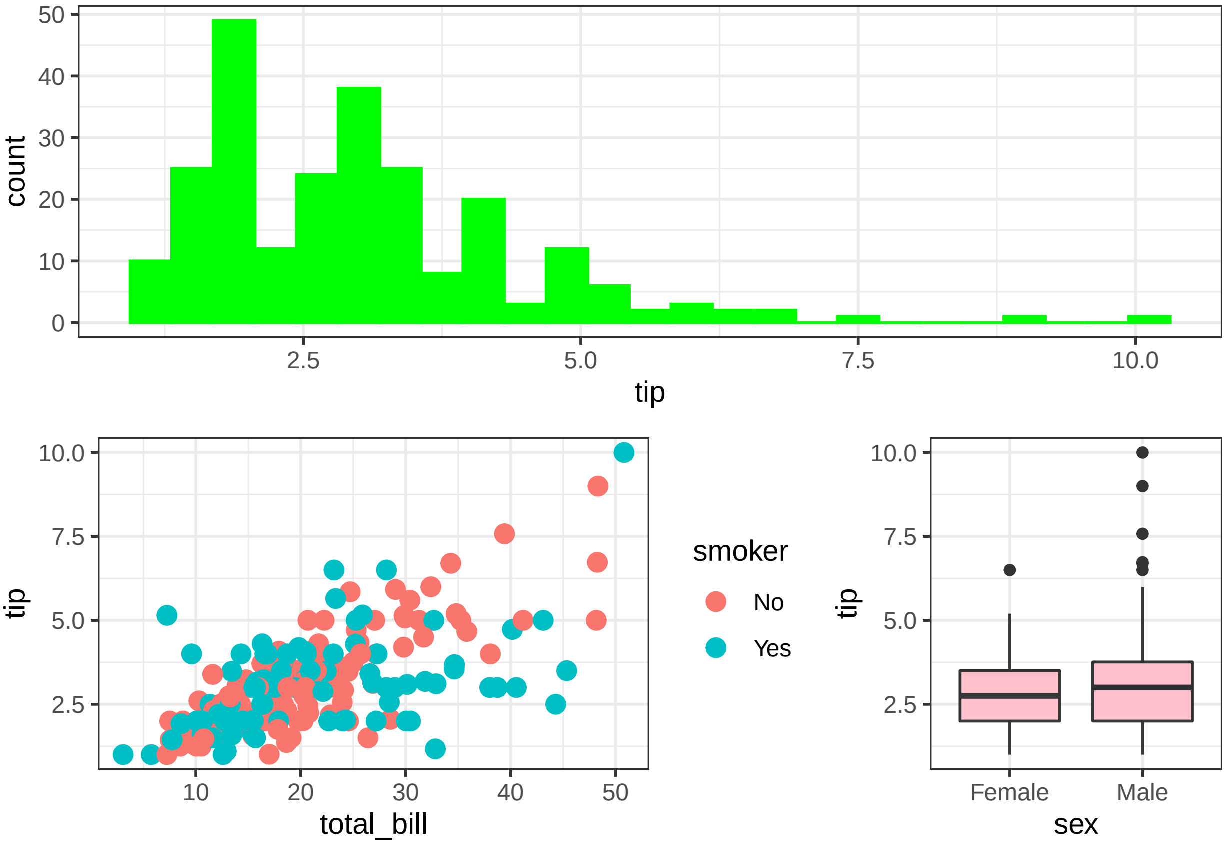
3. Create scatterplot of amount of tip by total amount of the bill. Use different colours for smokers (= green) and non-smokers (=blue).



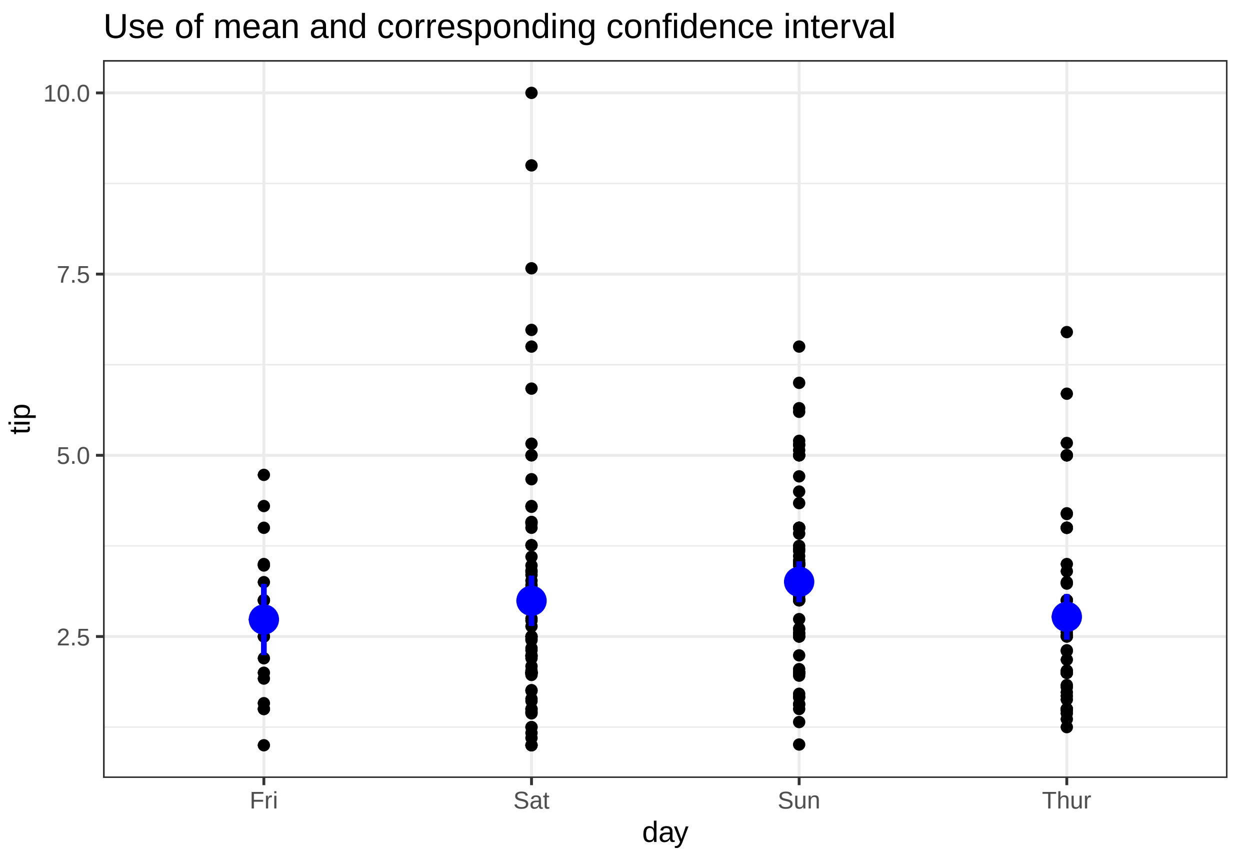
4. Based on previous scatterplot, add two regression lines.



5. Make
   - A histogram of `tip` (in green). Give this object the name `hist1`
   - A boxplot of `tip` by `sex.` Give this object the name `boxplot1`
   - Give the scatterplot of `tip` by `total_bill` and give it the name `scatter1`
   - Plot all of these 3 plots on one and the same graphical window as below



6.
- Make a scatterplot of `tip` (on Y axis) versus `day` (on X axis).
- Add the mean tip value and corresponding confidence interval (from bootstrap). Use a blue color.
- Add a title ‘Use of mean and corresponding confidence interval’.
- Use black and withe background.



7.2 `galileo` data

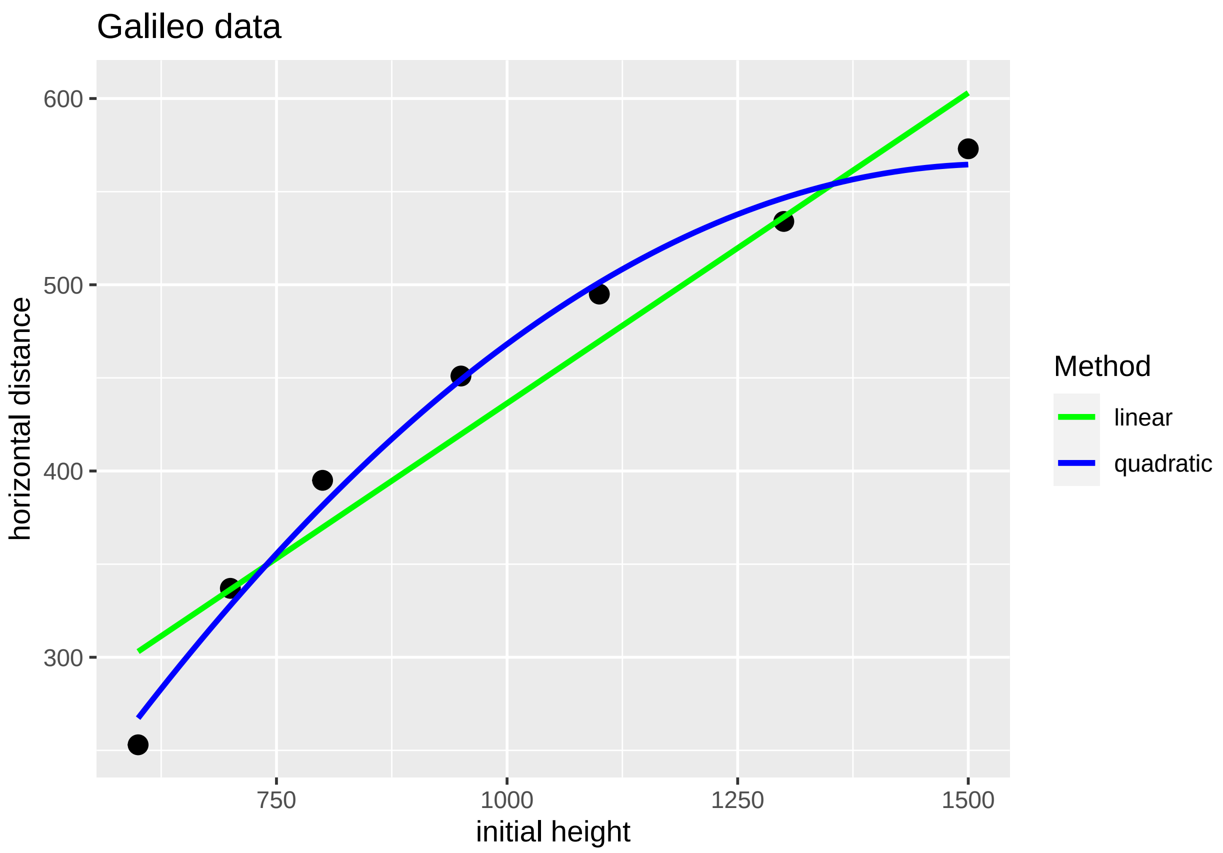
In this exercise, we use the `galileo` data set from the package `UsingR`.
This data frame has 7 observation on the following 2 variables

- `init.h`: Initial height of ball
- `h.d`: Horizontal distance traveled.

```
galileo
```

```
##   init.h h.d
## 1    600 253
## 2    700 337
## 3    800 395
## 4    950 451
## 5   1100 495
## 6   1300 534
## 7   1500 573
```

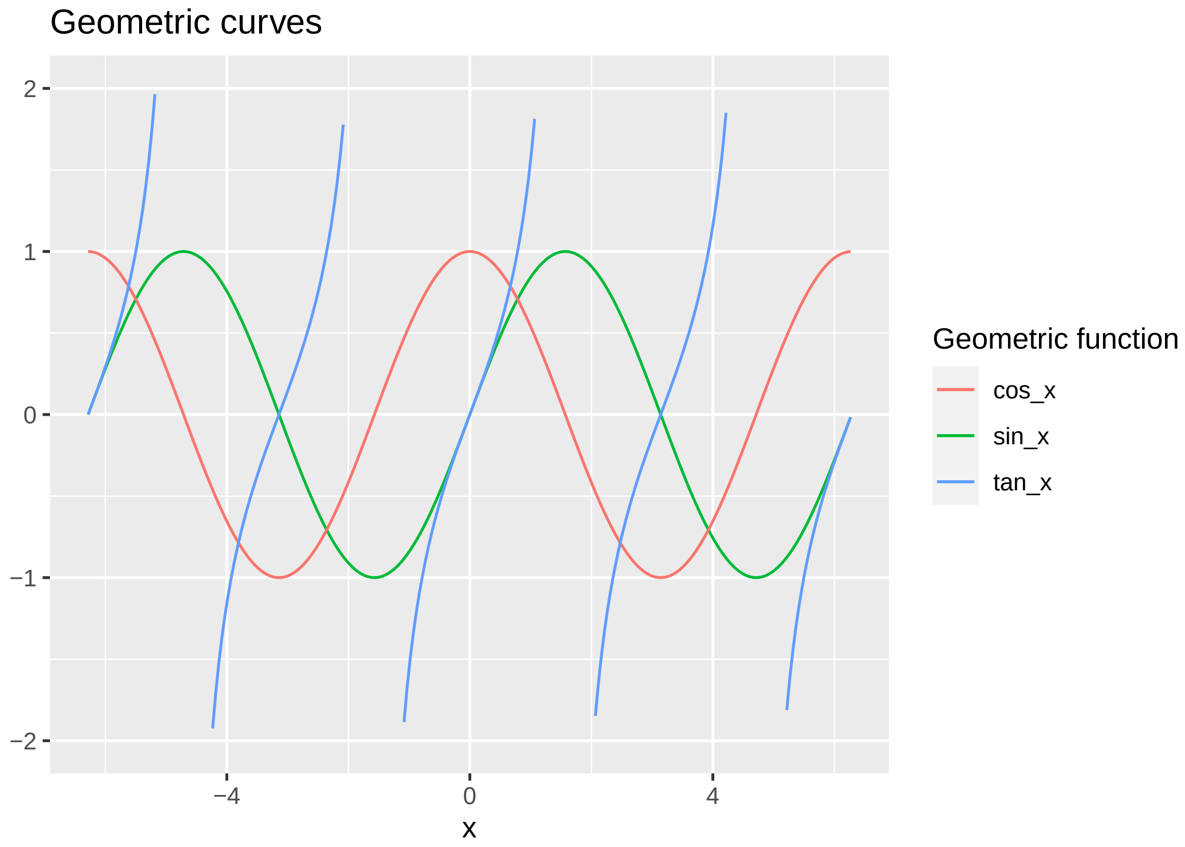
1. Make a scatterplot of initial height (X) versus horizontal distance (Y)
2. Add linear regression line
3. Add quadratic regression line (use *formula* $= y \sim x + I(x\hat{}2)$ for this)
4. Add title, labels, legend in order to obtain the following graph



7.3 Geometric curves

We want to overlay a cosine, sine and tangent function for a sequence of x values (from -2π to $+2\pi$) and create the following plot:

- Generate a set of x values from $-2pi$ to $+2\pi$. (The value π is a known constant in R).
- Construct 3 vectors with the values for $\sin(x)$, $\cos(x)$ and $\tan(x)$
- Put all these vectors in one data frame.
- Create the plot


Geometric curves
2
1
0
−1
−2
−4
0
4
x
Geometric function
cos_x
sin_x
tan_x