Correlation & Regression

Paul Schmidt

2023-11-14

Correlation and simple linear regression (with and without intercept).

Table of contents

This chapter is trying to give you a feeling for what correlation and (simple linear) regression is. I am aware that the example data doesn't have anything to do with agriculture or related fields, but I decided to keep it because it allows for an intuitive conclusion at the end.

```
# (install &) load packages
pacman::p_load(
  broom,
  conflicted,
  modelbased,
  tidyverse)
```

```
# handle function conflicts
conflicts_prefer(dplyr::filter)
conflicts_prefer(dplyr::select)
```

1 Data

This is data I made up: Peter and Max went out multiple evenings and at the end of every evening wrote down how many drinks they had and what the alcohol content in their blood was.

1.1 Import

```
# data is available online:
path <- "https://raw.githubusercontent.com/SchmidtPaul/dsfair_quarto/master/data/DrinksPet
```

```
dat <- read_csv(path) # use path from above
dat
```

```
# A tibble: 20 x 3
   Person drinks blood_alc
   <chr>   <dbl>     <dbl>
 1 Max         1       0.2
 2 Max         2       0.3
 3 Max         3       0.5
 4 Max         3       0.6
 5 Max         4       0.6
 6 Max         4       0.5
 7 Max         4       0.7
 8 Max         5       0.6
 9 Max         7       0.8
10 Max         8       1
11 Peter       1       0.1
12 Peter       1       0.1
13 Peter       1       0.2
14 Peter       1       0.2
15 Peter       1       0.1
16 Peter       3       0.3
17 Peter       5       0.5
```

```
18 Peter       6       0.8
19 Peter       8       0.9
20 Peter       9       1.3
```

1.2 Goal

The goal of this analysis is to answer the question how the number of drinks relates to the blood alcohol level. Note that we can ignore the column `Person`, since we do not care whether data came from Peter or Max. Thus, we only focus on the two *numeric* columns `drinks` and `blood_alc`. For them, we will do a correlation and a regression analysis.

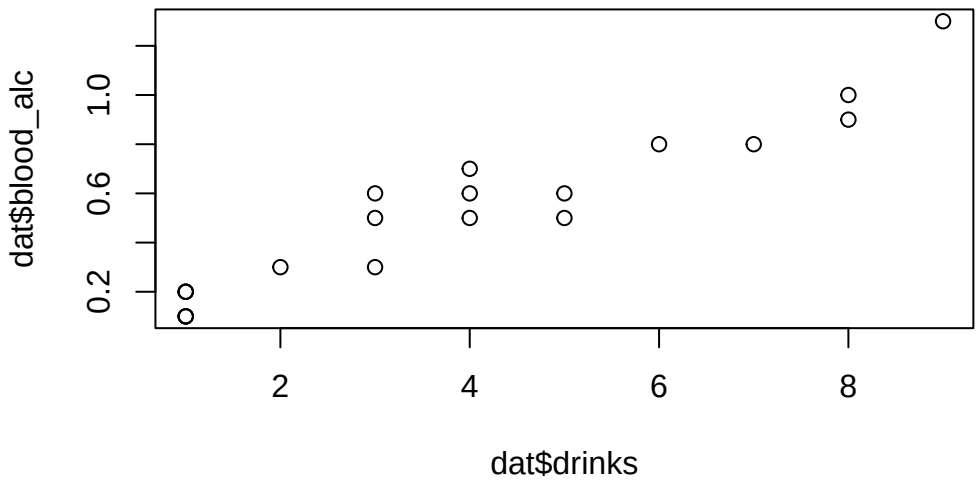
1.3 Exploring

To quickly get a first feeling for this dataset, we can use `summary()` and draw a plot via `plot()` or `ggplot()`.

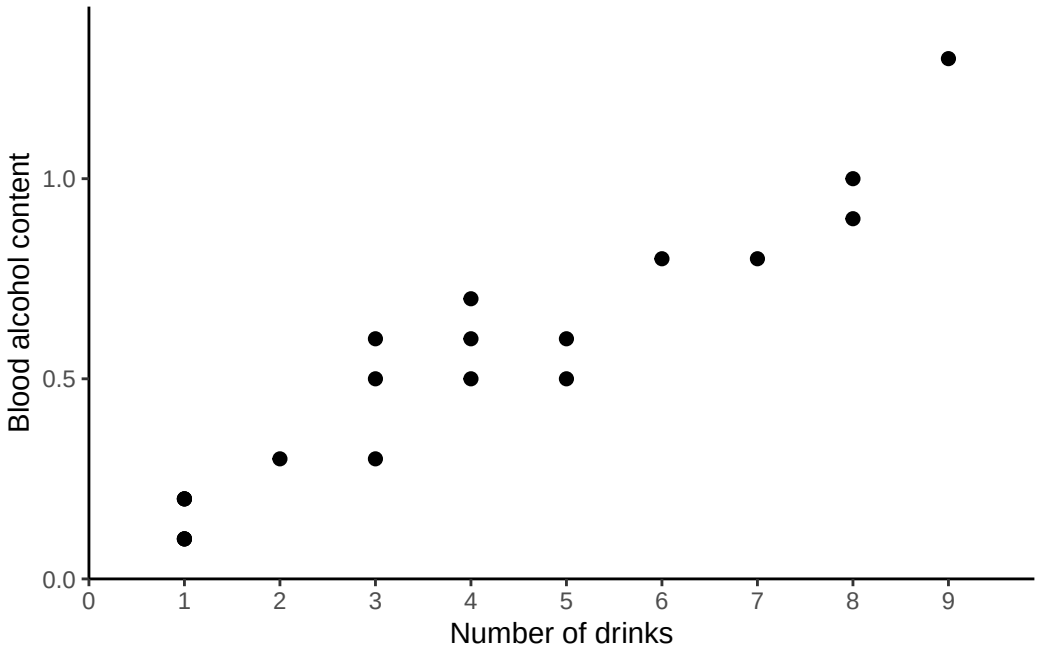
```
summary(dat)
```

```
   Person              drinks       blood_alc
 Length:20          Min.   :1.00   Min.   :0.100
 Class :character   1st Qu.:1.00   1st Qu.:0.200
 Mode  :character   Median :3.50   Median :0.500
                    Mean   :3.85   Mean   :0.515
                    3rd Qu.:5.25   3rd Qu.:0.725
                    Max.   :9.00   Max.   :1.300
```

```
plot(y = dat$blood_alc, x = dat$drinks)
```




```
ggplot(data = dat) +
  aes(x = drinks, y = blood_alc) +
  geom_point(size = 2) +
  scale_x_continuous(
    name = "Number of drinks",
    limits = c(0, 9),
    breaks = seq(0, 9),
    expand = expansion(mult = c(0, 0.1))
  ) +
  scale_y_continuous(
    name = "Blood alcohol content",
    limits = c(0, NA),
    expand = expansion(mult = c(0, 0.1))
  ) +
    theme_classic()
```

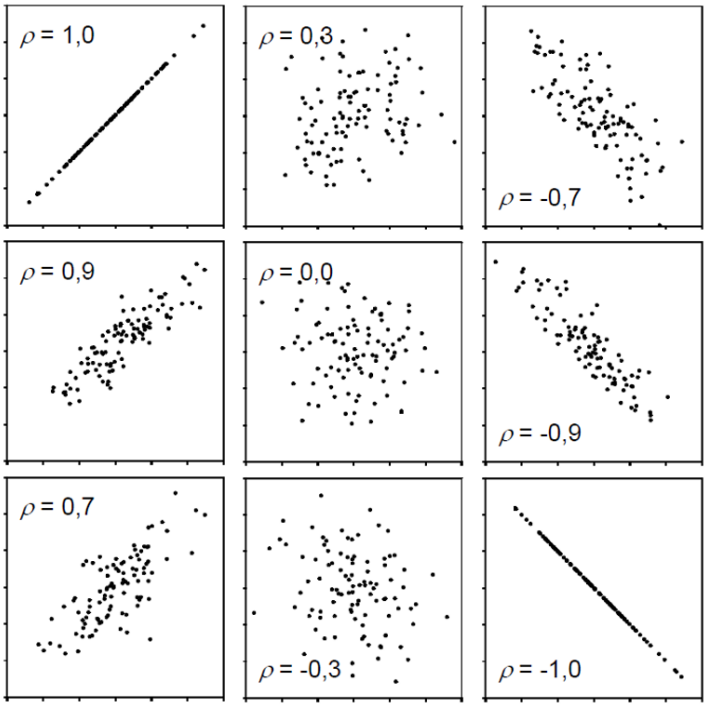


Apparently, the number of drinks ranges from 1 to 9 with a mean of 3.85, while the measured blood alcohol levels range from 0.1 to 1.3 with a mean of 0.515. The plots show a clear trend of increasing blood alcohol levels with a higher number of drinks - which is what we would expect.

2 Correlation

One way of actually putting a number on this relationship is to estimate the correlation. When people talk about correlation (ρ or r) in statistics, they usually refer to the Pearson correlation coefficient, which is a measure of linear correlation between two numeric variables. Correlation can only have values between -1 and 1, where 0 means *no correlation*, while all other possible values are either negative or positive correlations. The farther away from 0, the stronger is the correlation.

Simply put, a positive correlation means *"if one variable gets bigger, the other also gets bigger"* and a negative correlation means *"if one variable gets bigger, the other gets smaller"*. Therefore, it does not matter which of the two variables is the first ("x") or the second ("y") variable. Thus, a correlation estimate is not like a model and it cannot make predictions. Finally, *"correlation does not imply causation"* means that just because you found a (strong) correlation between two things, you cannot conclude that there is a cause-and-effect relationship between the two, which becomes clear when looking at these examples.



2.1 Get it

If you only want to get the actual correlation estimate, you can use the function `cor()` and provide the two numeric variables (as vectors):

```
cor(dat$drinks, dat$blood_alc)
```

```
[1] 0.9559151
```

So the correlation between number of drinks and blood alcohol content in our sample is ca. 0.96 and thus very strong, since it is almost 1.

2.2 Test it

If you would like additional information, such as a confidence interval and a test resulting in a p-value, you can use `cor.test()` instead of `cor()`. We may also use the {broom} package to get the results in a more convenient format.

```
mycor <- cor.test(dat$drinks,
                  dat$blood_alc)
mycor
```

```
    Pearson's product-moment correlation

data:  dat$drinks and dat$blood_alc
t = 13.811, df = 18, p-value = 5.089e-11
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.8897837 0.9827293
sample estimates:
      cor
0.9559151
```

```
tidy(mycor)
```

```
# A tibble: 1 x 8
  estimate statistic  p.value parameter conf.low conf.high method    alternative
     <dbl>     <dbl>    <dbl>     <int>    <dbl>     <dbl> <chr>     <chr>
1    0.956      13.8 5.09e-11        18    0.890     0.983 Pearson'~ two.sided
```

Looking at this longer output, you can see the sample estimate at the bottom, a confidence interval above it and a p-value with the corresponding test hypothesis above that. Run `?cor.test()` and look at the "Details" section for more info. Here, our correlation estimate of 0.96 is significantly different from 0, since the p-value is 0.0000000000509 and therefore < 0.05. Furthermore, the confidence interval is 0.890 - 0.983 meaning that we are 95% sure that the true correlation is somewhere in that range.

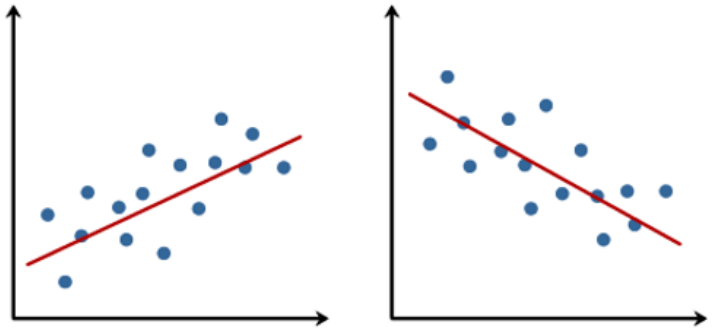
Additional Resources

- {correlation}
- {corrr}
- {ggcorrplot}

3 Simple linear regression

When people talk about regression in statistics, they usually refer to simple linear regression, which - simply put - finds the best straight line that goes through dots in a scatter plot of two

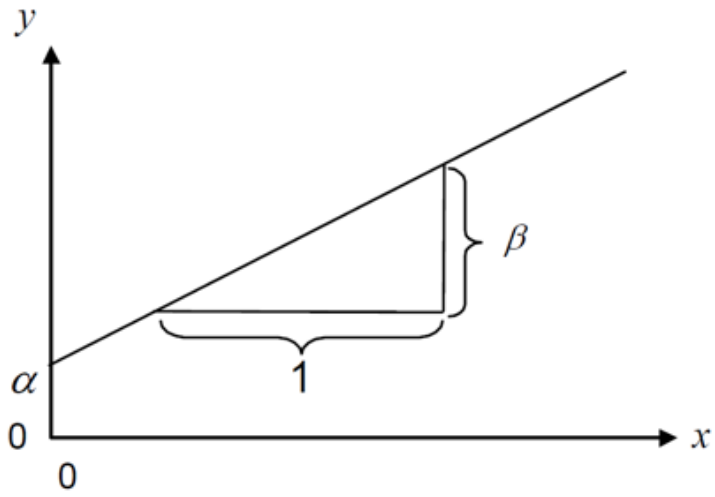
numeric variables:



The linear model behind such a straight line is simply:

$$y = \alpha + \beta x$$

where α or a is the intercept and β or b is the slope, while y and x are our data points. Fitting such a regression is really just finding the optimal estimates for α and β.



In contrast to correlation, a simple linear regression is a model and it therefore matters which variable is y (dependent variable) and which is x (independent), because after fitting the regression, the latter can be used to predict the former.

3.1 Get it

In R, we can use the `lm()` function for fitting linear models so that it fits the simple linear regression equation shown above easily:

```
reg <- lm(formula = blood_alc ~ drinks,
          data = dat)
```

As you can see, we refer to our data object `dat` in the `data =` argument so that in the `formula =` argument we only need to write the names of the respective columns in `dat`. Furthermore, we store the results in the `reg` object. When looking at this object, we get the following results:

```
reg
```

```

Call:
lm(formula = blood_alc ~ drinks, data = dat)

Coefficients:
(Intercept)       drinks
    0.04896      0.12105
```

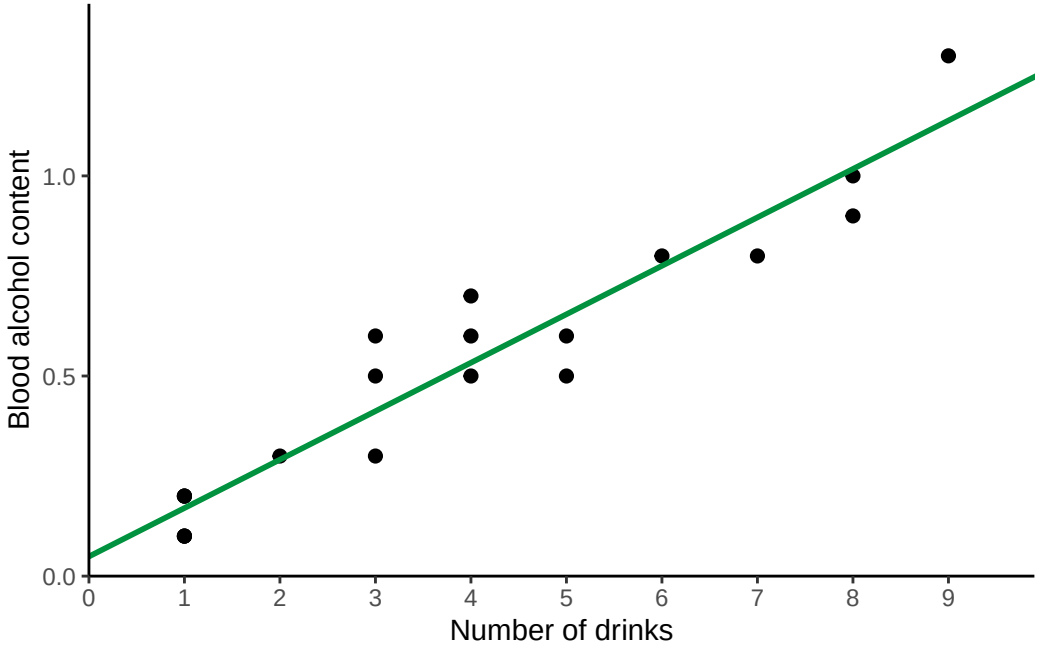
First, our command is repeated and then the "Coefficients" are shown, which are indeed the estimates for a and b. So the best straight line is:

$$bloodalc = 0.049 + 0.121 * drinks$$

which looks like this:

```
ggplot(data = dat) +
  aes(x = drinks, y = blood_alc) +
  geom_point(size = 2) +
  geom_abline(
    intercept = reg$coefficients[1],
    slope = reg$coefficients[2],
    color = "#00923f",
    linewidth = 1
  ) +
  scale_x_continuous(
    name = "Number of drinks",
    limits = c(0, 9),
    breaks = seq(0, 9),
    expand = expansion(mult = c(0, 0.1))
  ) +
  scale_y_continuous(
    name = "Blood alcohol content",
    limits = c(0, NA),
```

```
  expand = expansion(mult = c(0, 0.1))
) +
theme_classic()
```



Here is a little more info why `formula = blood_alc ~ drinks` leads to R estimating the a and b we want: What makes sense is that `blood_alc` is y, `drinks` is x and `~` would therefore be the $=$ in our equation. However, why is it we never had to write anything about a or b? The answer is that (i) when fitting a linear model, there is usually always an intercept ($=a$) by default and (ii) when writing a numeric variable ($=$`drinks`) as on the right side of the equation, it will automatically be assumed to have a slope ($=b$) multiplied with it. Accordingly, `blood_alc ~ drinks` automatically translates to `blood_alc = a + b*drinks` so to speak.

3.2 Is this right?

After fitting a model, you may use it to make predictions. Here is one way of obtaining the expected blood alcohol content for having 0 to 9 drinks according to our simple linear regression via {modelbased}:

```
preddat <- tibble(drinks = seq(0, 9)) %>%
  estimate_expectation(model = reg) %>%
```

```
  as_tibble()

preddat
```

```
# A tibble: 10 x 5
   drinks Predicted     SE  CI_low CI_high
    <int>     <dbl>  <dbl>   <dbl>   <dbl>
 1      0    0.0490 0.0406 -0.0363   0.134
 2      1    0.170  0.0337  0.0993   0.241
 3      2    0.291  0.0278  0.233    0.349
 4      3    0.412  0.0238  0.362    0.462
 5      4    0.533  0.0226  0.486    0.581
 6      5    0.654  0.0247  0.602    0.706
 7      6    0.775  0.0294  0.713    0.837
 8      7    0.896  0.0357  0.821    0.971
 9      8    1.02   0.0428  0.927    1.11
10      9    1.14   0.0505  1.03     1.24
```

You may notice that according to our model, the expected alcohol content in your blood when having 0 drinks is actually 0.049 and thus larger than 0. This is obviously not true in real life. Instead, the true intercept should actually be exactly 0, so what went wrong?

First of all, data will never be perfect in the sense that the when a parameter really is e.g. 42, its estimate based on measured data is also exactly 42.000000… . Instead, there are e.g. measurement errors: Peter and Max may have forgotten a drink or two or their device to measure the alcohol content is not precise enough. In fact, this would most likely be the underlying reason here - but remember that I made the data up.

So I would like you to think about the issue from two other angles:

1. Are the results really saying the intercept is > 0?
2. Did we even ask the right question or should we have fitted a different model?

3.2.1 Are the results really saying the intercept is > 0?

No, they are not. Yes, the sample estimate for the intercept is 0.049, but when looking at more detailed information via e.g. `summary()`. We may also use the {broom} package to get the results in a more convenient format.

```
tidy(reg, conf.int = TRUE)
```

```
# A tibble: 2 x 7
  term        estimate std.error statistic  p.value conf.low conf.high
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>
1 (Intercept)   0.0490   0.0406       1.21 2.43e- 1  -0.0363     0.134
2 drinks        0.121    0.00876     13.8  5.09e-11   0.103      0.139
```

you can see that the p-value for the intercept is 0.243, which is larger than 0.05 and thus saying that we could not find the intercept to be significantly different from 0. A second indication can be found when looking at the confidence interval of the expected value for having 0 drinks in the table above: `[-0.0363, 0.1340]`. This interval actually includes 0 which suggests that the true expected blood alcohol content for having 0 drinks may indeed be 0.

3.2.2 Should we have fitted a different model?

We certainly **could** have and we will actually do it now. It must be clear that statistically speaking there was nothing wrong with our analysis. However, from a biological standpoint or in other words - because of our background knowledge and expertise as scientists - we could have indeed actively decided for a regression analysis that does **not** have an intercept and is thus forced to start 0 in terms of blood alcohol content. After all, statistics is just a tool to help us make conclusions. It is a powerful tool, but it will always be our responsibility to "ask the right questions" i.e. apply expedient methods.

A simple linear regression without an intercept is strictly speaking no longer "simple", since it no longer has the typical equation, but instead this one:

$$y = \beta x$$

To tell `lm()` that it should not estimate the default intercept, we simply add `0 +` right after the `~`. As expected, we only get one estimate for the slope:

```
reg_noint <- lm(formula = blood_alc ~ 0 + drinks, data = dat)
reg_noint
```

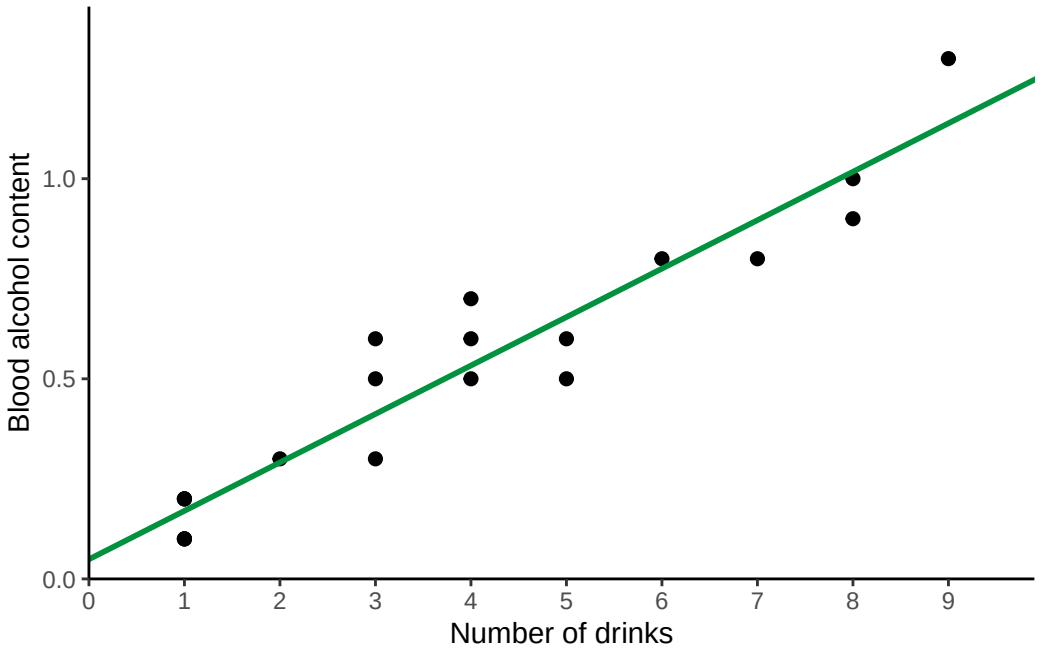
```

Call:
lm(formula = blood_alc ~ 0 + drinks, data = dat)

Coefficients:
drinks
0.1298
```

meaning that this regression with no intercept is estimated as

$$bloodalc = 0.1298 * drinks$$

and must definitely predict 0 `blood_alc` when having 0 `drinks`. As a final result, we can compare both regression lines visually in a ggplot:

```
ggplot(data = dat) +
  aes(x = drinks, y = blood_alc) +
  geom_point(size = 2) +
  geom_abline(
    intercept = reg$coefficients[1],
    slope = reg$coefficients[2],
    color = "#00923f",
    linewidth = 1
  ) +
  geom_abline(
    intercept = reg_noint$coefficients[1],
    slope = reg_noint$coefficients[2],
    color = "#e4572e",
    linewidth = 1
  ) +
  scale_x_continuous(
    name = "Number of drinks",
    limits = c(0, 9),
    breaks = seq(0, 9),
    expand = expansion(mult = c(0, 0.1))
  ) +
  scale_y_continuous(
    name = "Blood alcohol content",
    limits = c(0, NA),
    expand = expansion(mult = c(0, 0.1))
  ) +
  theme_classic()
```


Blood alcohol content
1.0
0.5
0.0
0
1
2
3
4
5
6
7
8
9
Number of drinks