Intro to GORIC in R

Rebecca M. Kuiper

22 January 2025

Contents

Preliminaries

Checking your working directory

The working directory is the folder R uses to gather all the files it uses; e.g., data, scripts, etc.

Before we start doing any analyses, it is convenient to check that the working directory is the same as the location of the data files. To retrieve your current working directory, you can run the following command in your R script.

```
getwd()
```

If you have opened Rstudio by opening the .R file, then your working directory is that folder and the code in the sections below should work.
When you, at some point, start working with R, checkout the footnote regarding making projects.[1]

[1]It can be wise to keep files that are associated with a single project together. This can be done by means of an R-project

Loading your data

In this practical, you will need the Sesame Street dataset. Information about the variables in the Sesame street dataset can, for instance, be found here.

Here, we will use a `.txt` file.
You can download it from Blackboard as a part of a .zip file, or open the .txt file and copy-paste it in Notepad (Dutch: Kladblok) to save it yourself as a `.txt` file. If you place this file into your working directory, R can find the file by itself.

To load a .txt file, you can use a function called `read.table`.

```
data <- read.table("data/sesameR.txt", header = TRUE,
                   na.string = c("-999", "-99999"))

data <- data.frame(data)
```

The argument `header = TRUE` tells R that the first line of the `.txt` file contains headers, typically variable names. With the option `na.string = ...`, you can specify how your missing values are labelled.

See the Section 'Miscellaneous' (which is extra material, not exam material), for saving adjusted data and rendered output, for a bit more information on missing data, and for loading data in case of other data formats (namely, SPSS and Excel).

Preview your data

Inspect your data

R has several functions that might be convenient when you want to inspect your data. For example, the command `View(data)` will open an SPSS-like data viewer, where you can see everyone's score on every variable. Other useful commands are specified in the command lines below.

```
dim(data)          #dimensions of your data, in this case,
```

```
[1] 240  20
```

```
                   #the number of rows and columns
head(data, n = 6) #shows the first six rows of your dataset
```

```
  id site sex age viewcat setting treat prebody prelet preform prenumb prerelat preclasf postbody postlo
1  1    1   1  66       1       2     1      16     23      12      40        14       20       18       3
2  2    1   2  67       3       2     1      30     26       9      39        16       22       30       3
3  3    1   1  56       3       2     2      22     14       9       9         9        8       21       4
4  4    1   1  49       1       2     2      23     11      10      14         9       13       21       1
5  5    1   1  69       4       2     2      32     47      15      51        17       22       32       5
6  6    1   2  54       3       2     2      29     26      10      33        14       14       27       3
```

```
tail(data, n = 6) #shows the last six rows of your dataset
```

```
     id site sex age viewcat setting treat prebody prelet preform prenumb prerelat preclasf postbody pos
235 235    5   1  53       4       1     1      26     25      14      36        13       13       30
236 236    5   2  51       2       1     1      30     15       8      12        10       10       30
237 237    5   1  49       4       1     1      17     16      12      15         8       15       25
238 238    5   1  43       2       1     1      16     13       6      11         8        9       22
239 239    5   2  60       3       1     1      23     16       9      33        14       16       29
240 240    5   1  51       4       1     1      21     11      10      27        10       12       25
```

(File > New Project). The directory name you choose will be the project name (of an .Rproj files); it is convenient to click the box *Open in new session*. If you locate your data in the same folder as your R-project, you will not have to specify the working directory anymore, since this is done by creating the R-project within a specific folder.

If you are, for example, interested in the data on the fifth row of your data (because you want to inspect this person's data), you can use the following command:

```
data[5, ]
```

```
  id site sex age viewcat setting treat prebody prelet preform prenumb prerelat preclasf postbody postle
5  5    1   1  69       4       2     2      32     47      15      51       17       22       32      5
```

This command shows you the fifth row and all columns (indicated by the blank space after the trailing comma) of the object called `data`. You might as well be interested in all scores on the second column (i.e., all the scores on / observations on a specific variable). In this case, you can use the following command.

```
data[, 2]
```

```
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

This shows you all values (indicated by the blank space before the comma) in the second column (indicated by the `2` after the comma). Although you can see all scores on all variables that you are interested in, the specifications of columns is not very insightful.

If you want to obtain the names of the columns in your data, that is, the variable names, you can use the following command:

```
names(data)
```

```
 [1] "id"        "site"      "sex"       "age"       "viewcat"   "setting"   "treat"     "prebody"   "pr
```

If you are interested in a specific variable, it is often more convenient to identify it using its name, instead of its column number. This helps in case you inspect your code after some time and might not remember which column contains which variable; or in case others read your code. Hence, the following command is more insightful:

```
data$viewcat
```

```
  [1] 1 3 3 1 4 3 3 2 4 3 2 4 1 2 2 3 2 4 3 3 4 4 3 4 2 2 4 4 3 2 3 2 3 3 4 2 3 1 2 2 3 2 3 1 2 4 3 3 4
```

This asks for the variable `viewcat` within the object called `data`.

Notably, the following command will *not* work since there is no object called `viewcat` in your R-environment:

```
viewcat
```

Descriptive statistics

Now, we know how to call certain columns / named variables, rows and combinations of the two. We can also obtain descriptive information about the data using existing (base) functions in R.

```
str(data)     #gives you the structure of the variables in the data
```

```
'data.frame':   240 obs. of  20 variables:
 $ id       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ site     : int  1 1 1 1 1 1 1 1 1 1 ...
 $ sex      : int  1 2 1 1 1 2 2 1 1 2 ...
 $ age      : int  66 67 56 49 69 54 47 51 69 53 ...
 $ viewcat  : int  1 3 3 1 4 3 3 2 4 3 ...
 $ setting  : int  2 2 2 2 2 2 2 2 2 2 ...
 $ treat    : int  1 1 2 2 2 2 2 1 1 1 ...
 $ prebody  : int  16 30 22 23 32 29 23 32 27 30 ...
 $ prelet   : int  23 26 14 11 47 26 12 48 44 38 ...
 $ preform  : int  12 9 9 10 15 10 11 19 18 17 ...
 $ prenumb  : int  40 39 9 14 51 33 13 52 42 31 ...
 $ prerelat : int  14 16 9 9 17 14 11 15 15 10 ...
```

```
 $ preclasf : int  20 22 8 13 22 14 12 23 20 17 ...
 $ postbody : int  18 30 21 21 32 27 22 31 32 32 ...
 $ postlet  : int  30 37 46 14 53 36 45 47 50 52 ...
 $ postform : int  14 17 15 13 18 14 12 18 17 19 ...
 $ postnumb : int  44 39 40 19 54 39 44 51 48 52 ...
 $ postrelat: int  14 14 9 8 14 16 12 17 14 17 ...
 $ postclasf: int  23 22 19 15 21 24 15 23 24 24 ...
 $ peabody  : int  62 80 32 27 71 32 28 38 49 32 ...
```

```
summary(data) #gives you some summary statistics of all variables in your data
```

```
       id              site           sex             age           viewcat         setting           tr
 Min.   :  1.00   Min.   :1.00   Min.   :1.000   Min.   :34.00   Min.   :1.000   Min.   :1.000   Min.
 1st Qu.: 60.75   1st Qu.:1.75   1st Qu.:1.000   1st Qu.:48.00   1st Qu.:2.000   1st Qu.:1.000   1st Qu
 Median :120.50   Median :3.00   Median :2.000   Median :52.00   Median :3.000   Median :1.000   Median
 Mean   :120.50   Mean   :2.60   Mean   :1.521   Mean   :51.52   Mean   :2.558   Mean   :1.404   Mean
 3rd Qu.:180.25   3rd Qu.:4.00   3rd Qu.:2.000   3rd Qu.:56.00   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu
 Max.   :240.00   Max.   :5.00   Max.   :2.000   Max.   :69.00   Max.   :4.000   Max.   :2.000   Max.
```

We can see that, currently, the variables "sex" and "site" are not factors / grouping variables, but continuous variables. Note that we used a .txt file for the data in which you cannot specify the measurement level as you can do in SPSS). We can change the measurement level of these variables by the following commands:

```
data$sex <- as.factor(data$sex)
data$site <- as.factor(data$site)
```

To check whether the data is now coded accordingly, we can once again ask for a summary of the data.

```
summary(data)
```

```
       id         site   sex          age           viewcat         setting          treat          pre
 Min.   :  1.00   1:60   1:115   Min.   :34.00   Min.   :1.000   Min.   :1.000   Min.   :1.000   Min.
 1st Qu.: 60.75   2:55   2:125   1st Qu.:48.00   1st Qu.:2.000   1st Qu.:1.000   1st Qu.:1.000   1st Qu
 Median :120.50   3:64           Median :52.00   Median :3.000   Median :1.000   Median :1.000   Median
 Mean   :120.50   4:43           Mean   :51.52   Mean   :2.558   Mean   :1.404   Mean   :1.367   Mean
 3rd Qu.:180.25   5:18           3rd Qu.:56.00   3rd Qu.:4.000   3rd Qu.:2.000   3rd Qu.:2.000   3rd Qu
 Max.   :240.00                  Max.   :69.00   Max.   :4.000   Max.   :2.000   Max.   :2.000   Max.
```

Now, the descriptive statistics of "sex" and "site" are suited for a grouping variable.

If we are not interested in all these statistics of all these variables, we can ask for a specific descriptive of a single variable. If we are interested in, for example, the standard deviation (SD) of the variable `postnumb`, we can use the command for the standard deviation to achieve this:

```
sd(data$postnumb)
```

```
[1] 12.84577
```

Notably, if there was a missing value in the data (which is made below for illustrative purposes), we would receive NA ("Not Available"):

```
value <- data$postnumb[1] # store value, to afterwards restore data like it was
# Make missing value (NA = "Not Available") for illustrative purposes:
data$postnumb[1] <- NA

# Ask for SD of postnumb
sd(data$postnumb)
```

```
[1] NA
```

R usually needs explicit instructions about how to deal with missing data. If we tell R to remove the missing values (using the argument 'na.rm = TRUE'), it will give us the standard deviation over all observed values:

```
sd(data$postnumb, na.rm = TRUE)
```

```
[1] 12.84082
```

Furthermore, we could ask for the mean or the median of the variable **postnumb**; or the Pearson correlation coefficient between the variables **prenumb** and **postnumb**. Bear in mind that we still have to specify how to handle missing data.

```
mean(data$postnumb, na.rm = TRUE) #mean
```

```
[1] 29.99582
```

```
median(data$postnumb, na.rm = TRUE) #median
```

```
[1] 29
```

```
cor(data$prenumb, data$postnumb, use = "complete.obs")
```

```
[1] 0.6735467
```

Note: Using "?", you can learn more about any function, what options you can set, and what are its defaults. With '?cor', you can learn more about the cor function (which also tells you how to estimate Spearman instead of Pearson correlations).

Data visualization

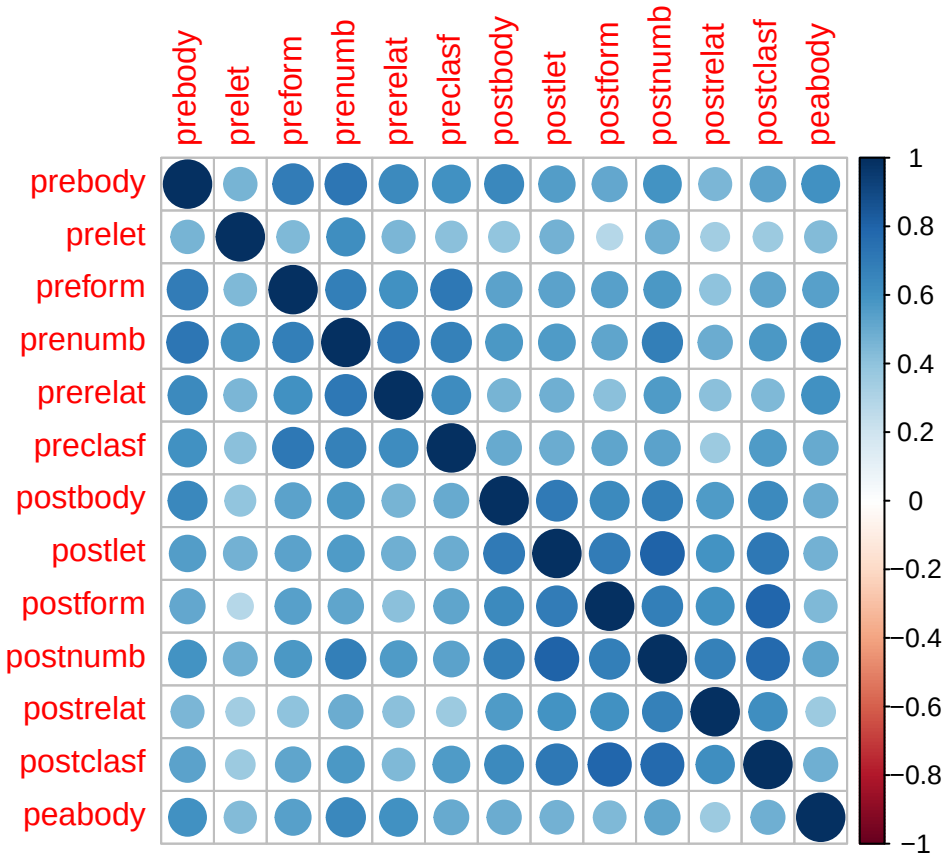
There are several plot options in R. A nice visualization of several correlations is given by:

```
if (!require("corrplot")) install.packages("corrplot")
```

```
Loading required package: corrplot
```

```
corrplot 0.95 loaded
```

```
library(corrplot) # load package
correlations <- cor(data[, 8:20], use = "complete.obs", method = "spearman")
# data[, 8:20]: the pre and post score variables
corrplot(correlations, method = "circle")
```



Now, restore the value we extracted from the data, such that we end up with the original data again:

```
data$postnumb[1] <- value
```

Statistical Modelling in R

Example 1: ANOVA

Recall that an ANOVA model is equivalent to a regression model. Both are based on the linear model. You can find ANOVA assumptions on the following websites:

- https://sites.ualberta.ca/~lkgray/uploads/7/3/6/2/7362679/slides_-_anova_assumptions.pdf
- https://statistics.laerd.com/spss-tutorials/one-way-anova-using-spss-statistics.php
- https://statistics.laerd.com/statistical-guides/one-way-anova-statistical-guide-3.php

There are also several online examples of how to do an ANOVA in R, see below.

- https://www.statmethods.net/stats/anova.html
- http://www.sthda.com/english/wiki/one-way-anova-test-in-r
- http://www.sthda.com/english/wiki/two-way-anova-test-in-r
- https://www.r-bloggers.com/one-way-analysis-of-variance-anova/

In the next example, we will continue with the Sesame Street data. We will use the function `lm`, which stands

for "linear model". We can learn more about the `lm` function by '?lm'.

Let us assume that we are interested in the increase in the knowledge of numbers after watching Sesame Street. Since this is not a variable in the data, we need to create a postnumb-prenumb variable:

```
data <- data.frame(data)
data$IncreaseNumb <- data$postnumb-data$prenumb
```

In the first example, we choose `IncreaseNumb` (increase in the knowledge of numbers after watching Sesame Street) as the dependent variable and `viewcat` as the grouping variable (frequency of viewing; 1 = rarely watched the show, 2 = once or twice a week, 3 = three to five times a week, 4 = watched the show on average more than 5 times a week). Since `viewcat` is currently a numeric variable (for more details see the 'Be aware' Section), we need to tell R that `viewcat` is a factor / grouping variable:

```
data$viewcat <- factor(data$viewcat)
#> levels(sesamdata$viewcat)
#[1] "1" "2" "3" "4"
```

Next, we create an object called anova1 to store the ANOVA output in. We will use the function `lm`, in which we must specify the dependent variable before the `~`, and the independent variable(s) after the `~`, as well as that the variables should come from our dataset `data`. Once we created the object anova1 containing the results, we can access the results by means of the function `summary`:

```
anova1 <- lm(IncreaseNumb ~ viewcat, data = data)
summary(anova1)
```

```
Call:
lm(formula = IncreaseNumb ~ viewcat, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-45.906  -5.671  -0.230   6.407  24.717

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.593      1.267   3.623 0.000356 ***
viewcat2       3.691      1.747   2.112 0.035697 *
viewcat3       6.314      1.721   3.668 0.000302 ***
viewcat4       7.585      1.734   4.375 1.82e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.314 on 236 degrees of freedom
Multiple R-squared:  0.08624,   Adjusted R-squared:  0.07462
F-statistic: 7.424 on 3 and 236 DF,  p-value: 8.996e-05
```

Interpreting the output

The lm output has several parts.

Call tells us what we asked the lm function to do. Here, we can see that we wanted to predict `IncreaseNumb` using `viewcat` which are variables from the dataset called `data`.

Residuals describe how the model residuals are distributed. It gives the minimum and maximum values, as well as the median and the first and third quartile. Ideally, we want the residuals to be normally distributed.

Coefficients are the model parameter estimates. Often, these are of main interest.

The columns contain the (regression coefficients) estimates themselves, standard errors, t statistics, and p-values. In other words, we obtain the regression coefficients and several measures of the preciseness with which we estimated them.
The rows correspond to the various predictors. In this example, the first row is for the intercept (i.e., the mean of the dependent variable when all the predictors are set to 0), that is, the mean for the reference category (here, viewcat = 1). The other rows are the differences in group means for the corresponding group versus the reference category/group. stated otherwise, the average increase in knowledge of numbers for someone who rarely watches Sesame Street is 4.593. The difference in means between rarely watching and watching the show once or twice a week is 3.691; that is, the average increase in knowledge of numbers for someone who watches Sesame Street once or twice a week is 4.593 + 3.691. Et cetera.
Furthermore, we can see that all three differences in means for the groups compared to the reference groups are all significant: for all three, it would be very unlikely to observe this regression coefficient if it was really zero in population ($p < .05$).

Finally, we receive estimates of the model's accuracy. For example, the R-squared denotes the proportion of variance explained by the model. In this case, we explain only about 8%.

ANOVA table

To also obtain the ANOVA table, we can run the following command.

```
anova(anova1)
```

```
Analysis of Variance Table

Response: IncreaseNumb
           Df  Sum Sq Mean Sq F value    Pr(>F)
viewcat     3  1932.3  644.09  7.4244 8.996e-05 ***
Residuals 236 20473.7   86.75
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the F-statistic can be computed as follows:

$$\frac{MS_{viewcat}}{MS_{residuals}} = (anova(anova1)[,3])[1]/(anova(anova1)[,3])[2] = 7.42.$$

We can see, from the ANOVA table but also the anova1 output, that the p-value is smaller than 0.05 ($p < .001$). This means that you can reject the null hypothesis, with H_0: the three group mean differences are 0, which implies that the mean of IncreaseNumb is equal for all four categories, that is, $\mu_1 = \mu_2 = \mu_3 = \mu_4$.

Pairwise, post-hoc tests

Above, we performed the so-called *omnibus* ANOVA (i.e., testing for the overall effect of the grouping variable on the outcome) and we concluded that the main effect for `viewcat` was statistically significant.

If you want to know more about the difference between the 4 groups, you can use pairwise comparison using t-tests. E.g.:

```
pairwise.t.test(data$IncreaseNumb, data$viewcat, p.adj = 'none')
```

```
    Pairwise comparisons using t tests with pooled SD

data:  data$IncreaseNumb and data$viewcat

  1      2      3
2 0.0357 -      -
```

```
3 0.0003  0.1184 -
4 1.8e-05 0.0218 0.4445

P value adjustment method: none
```

The output returns p-values for each of the group comparisons. Usually, one would look for values below .05 to assess whether there is a difference between that pair of groups (e.g. 1 and 3 with $p = .0003$).

What is the problem with doing comparisons like these?

One of them is that we increase the risk of Type I error (i.e., detecting a non-existing difference, that is, rejecting the null while it is not true), because we do 6 of these tests. One way to address this is using a p-value correction:

```
pairwise.t.test(data$IncreaseNumb, data$viewcat, p.adj = 'bonferroni')
```

```
    Pairwise comparisons using t tests with pooled SD

data:  data$IncreaseNumb and data$viewcat

  1       2       3
2 0.21418 -       -
3 0.00181 0.71059 -
4 0.00011 0.13097 1.00000

P value adjustment method: bonferroni
```

Now, the p-values have increased such that our Type 1 error rate remains the same (namely, 5%). Notice that this specific correction (called Bonferroni) simply multiplies your p-values by the number of tests you do. After doing this, you would conclude that the group 1 differs from 3 and 4.

Informative hypotheses in ANOVA

We can also evaluate theory-based hypotheses in R. Let us assume that beforehand, we expect that the mean of `IncreaseNumb` increases with the categories of `viewcat` (frequency of viewing).
Note that we do not per se assume a linear trend (since we created a factor), but only an increase (i.e., $\mu_1 < \mu_2 < \mu_3$).

One can use Bayesian model selection to evaluate such a hypothesis, but one could also use model selection using information criteria, like the GORIC (Generalized Order Restricted Information Criterion). Both have the same goals but GORIC does not need priors.
- To use **Bayesian model selection**, we can use the `bain` function within the R-package `bain`.
- The (non-Bayesian) **information criteria method** can be executed by the `goric` function in the R-package `restriktor`.

With both functions, you can evaluate theory-based / informative / inequality constrained hypotheses. This means that, instead of the classical null hypothesis (e.g., $H_0 : \mu_1 = \mu_2 = \mu_3$) and the classical alternative hypothesis (e.g., $H_A : \mu_1, \mu_2, \mu_3$), one can evaluate an informative hypothesis (e.g., $H_1 : \mu_1 < \mu_2 < \mu_3$). This thus goes beyond null hypothesis (significance) testing.

Both methods will be shown below. For each, we first load the required package, then we specify the hypothesis of interest (i.e., $H_1 : \mu_1 < \mu_2 < \mu_3 < \mu_4$) - where the specification uses variables names (e.g., viewcat1) and not population parameter names (e.g., μ_1). Subsequently, we can execute the `bain` or `goric` command and ask for the results. In this example, we use the complement of H_1 (= not H_1) as the alternative (safeguard) hypotheses. That hypothesis reflects all ordering other than the one we specified in H_1, which is more powerful than the classical alternative hypothesis (H_A: every ordering is allowed including the one of interest).

Note that it is easiest to specify the hypothesis for a model that gives the estimates for all group means (for more details see the 'Be aware' Section):

```
anova2 <- lm(IncreaseNumb ~ -1 + viewcat, data = data) # model 2
# Note: -1 means suppressing the intercept.
summary(anova2)
```

```
Call:
lm(formula = IncreaseNumb ~ -1 + viewcat, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-45.906  -5.671  -0.230   6.407  24.717

Coefficients:
         Estimate Std. Error t value Pr(>|t|)
viewcat1    4.593      1.267   3.623 0.000356 ***
viewcat2    8.283      1.202   6.889 5.07e-11 ***
viewcat3   10.906      1.164   9.367  < 2e-16 ***
viewcat4   12.177      1.183  10.295  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.314 on 236 degrees of freedom
Multiple R-squared:  0.5187,    Adjusted R-squared:  0.5105
F-statistic: 63.58 on 4 and 236 DF,  p-value: < 2.2e-16
```

bain The code for evaluating the informative hypothesis $H_1 : \mu_1 < \mu_2 < \mu_3 < \mu_4$ with `bain` is:

```
if (!require("bain")) install.packages("bain")
```

```
Loading required package: bain
```

```
library(bain)
names(coef(anova2)) #check coefficient names
```

```
[1] "viewcat1" "viewcat2" "viewcat3" "viewcat4"
```

```
H1 <- "viewcat1 < viewcat2 < viewcat3 < viewcat4"
set.seed(123) # to ensure every time same result
              # (and to have the option to test the sensitivity of the results)
bain(anova2, H1)
```

```
Bayesian informative hypothesis testing for an object of class lm (ANOVA):

   Fit   Com   BF.u   BF.c   PMPa  PMPb  PMPc
H1 0.695 0.044 15.863 49.784 1.000 0.941 0.980
Hu                                0.059
Hc 0.305 0.956 0.319                 0.020

Hypotheses:
  H1: viewcat1<viewcat2<viewcat3<viewcat4

Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothesis Hu. Bl
```

In the bain output, the rows contain the hypotheses. In this case, you have H1 (the one you specified), Hu

(the unconstrained one), and Hc (the complement). The columns give us several ways to express the degree of support we obtained towards H1 compared to the other hypotheses. For instance, BF.u shows how likely H1 is compared to the unconstrained hypothesis. BF.c compares H1 to its complement.

From this, we can conclude that the order-restricted hypothesis H1 has BF.c times more support than its complement, that is, all other possible orderings / theories.

GORIC The code for evaluating the informative hypothesis $H_1 : \mu_1 < \mu_2 < \mu_3 < \mu_4$ with `goric` is:

```
if (!require("restriktor")) install.packages("restriktor") #install package
```

```
Loading required package: restriktor
```

```
This is restriktor 0.6-10
```

```
Please report any bugs to info@restriktor.org
```

```
library(restriktor) #load package

names(coef(anova2)) #check coefficient names
```

```
[1] "viewcat1" "viewcat2" "viewcat3" "viewcat4"
```

```
H1 <- "viewcat1 < viewcat2 < viewcat3 < viewcat4" # Hypothesis of interest

set.seed(123) # to ensure every time same result
              # (and to have the option to test the sensitivity of the results)
out <- goric(anova2,
             hypotheses = list(H1), comparison = "complement")
             # Default: type = "goric"
#summary(out) # renders all output
out           # renders nice overview of important output
```

```
restriktor (0.6-10): generalized order-restricted information criterion:

Results:
        model    loglik  penalty     goric  loglik.weights  penalty.weights  goric.weights
1          H1  -874.096    3.095  1754.381           0.574            0.855          0.888
2  complement  -874.394    4.870  1758.528           0.426            0.145          0.112

Conclusion:
The order-restricted hypothesis 'H1' has 7.95 times more support than its complement.
```

The GORIC output shows us that the order-restricted hypothesis H_1 is approximately 8 (i.e., .888/.112) times more support than its complement.

Note that the goric.weights resemble the posterior model probabilities of Bayesian model selection and that the ratio of these weights resemble Bayes factors. The model/hypothesis with the highest goric.weight (or smallest goric value) is the best of the set. Moreover, the ratio of goric.weights quantifies how much more support one hypothesis has over another one.

In practice, one should also check the closeness of loglik values (cf. the guidelines), but this is beyond the scope of the ARMS course.

Extra: GORICA The GORICA (an approximation of the the GORIC) can be used using the 'type' argument:

```
set.seed(123) # to ensure the same result every time
              # (and to have the option to test the sensitivity of the results)
```

```
out_gorica <- goric(anova2,
                    hypotheses = list(H1), comparison = "complement",
                    type = "gorica") # Now, you obtain GORICA output
#summary(out_gorica) # renders all output
out_gorica           # renders nice overview of important output
```

```
restriktor (0.6-10): generalized order-restricted information criterion approximation:

Results:
        model  loglik  penalty  gorica  loglik.weights  penalty.weights  gorica.weights
1          H1  -4.417    2.095  13.023           0.573            0.855           0.888
2  complement  -4.710    3.870  17.161           0.427            0.145           0.112

Conclusion:
The order-restricted hypothesis 'H1' has 7.92 times more support than its complement.
```

The GORICA will render (approximately) the same weights as the GORIC does.

Note: The GORICA can be used for any type of statistical model (thus, including repeated measures analyses, logistic regression models, and SEM), while the GORIC can only be applied to normal linear models, like ANOVA and regression.

Example 2: Linear Regression

Now you have seen some of the things you can do in R. If you want to dive a little deeper, you can go through the following example. It showcases linear regression, another linear model, but you will experience more complex scenarios (such as controlling for multiple predictors).

Linear regression has several assumptions as well. Visit the links below to learn more:

- http://r-statistics.co/Assumptions-of-Linear-Regression.html
- https://www.statisticssolutions.com/assumptions-of-linear-regression/

To learn how to run a linear regression in R, you can visit the following links:

- http://r-statistics.co/Linear-Regression.html
- https://www.statmethods.net/stats/regression.html
- https://www.r-bloggers.com/simple-linear-regression-2/

In this example, we will continue with the Sesame Street dataset. To run a linear regression in R, we once again need the function `lm`, in which we must specify the dependent variable before the `~`, and the independent variable(s) after the `~`, as well as the data. Once we created the object containing the results of the linear regression, we can access the results by means of the function `summary`.

Output

Next, we will predict the knowledge of number after watching Sesame Street from the knowledge of number before watching the show:

```
lm1 <- lm(postnumb ~ prenumb, data = data)
summary(lm1)
```

```
Call:
lm(formula = postnumb ~ prenumb, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
```

```
-39.253  -6.283  -0.394   5.796  22.170

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 13.08508    1.34795   9.707   <2e-16 ***
prenumb      0.81208    0.05746  14.133   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.492 on 238 degrees of freedom
Multiple R-squared:  0.4563,    Adjusted R-squared:  0.454
F-statistic: 199.7 on 1 and 238 DF,  p-value: < 2.2e-16
```

As a reminder: The R^2 tells us the proportion of variation in the dependent variable that has been explained by this model and R^2_{adj} additionally penalizes the total number of predictors in your model.

From the output, you can see that knowledge of numbers before watching (`prenumb`) is positively related to `postnumb`, knowledge of numbers after watching the show. Moreover, as `prenumb` increases with 1 point, `postnumb` increases with 0.812 points.
When the p-value of the coefficient is less than the significance level (in this case, if $p < .05$), we can safely reject the null hypothesis that the coefficient β of the predictor is zero (using a t-test). The regression equation in this example is:

$$postnumb = \beta_0 + \beta_1 * prenumb,$$

where β is the estimated effect (regression coefficient) of `prenumb`. Note, however, to obtain an interpretation of the effect of the variable `prenumb`, one should also take the range of scores (minimum and maximum values) on this variable into account.

```r
min(data$prenumb)
```

```
[1] 1
```

```r
max(data$prenumb)
```

```
[1] 52
```

Extra: Additional output

Other useful functions regarding the created regression model are `coef` and `confint`, for the model coefficients and the confidence intervals for the model parameters, respectively:

```r
coef(lm1)
```

```
(Intercept)     prenumb
 13.0850832   0.8120798
```

```r
confint(lm1, level = .95)
```

```
                2.5 %    97.5 %
(Intercept) 10.4296516 15.740515
prenumb      0.6988865  0.925273
```

Furthermore, there are several functions (of which the output is not shown below) that are probably very useful when you estimated a regression model in R. These functions are stated below, including a short description of what they do.

```r
fitted(lm1)    # predicted values
residuals(lm1) # residuals, for example for testing normality assumptions
anova(lm1)     # anova table with output
               # (containing a significance test of the model)
```

```
vcov(lm1)       # covariance matrix for the model parameters
influence(lm1) # regression diagnostics
```

Extra: Plot

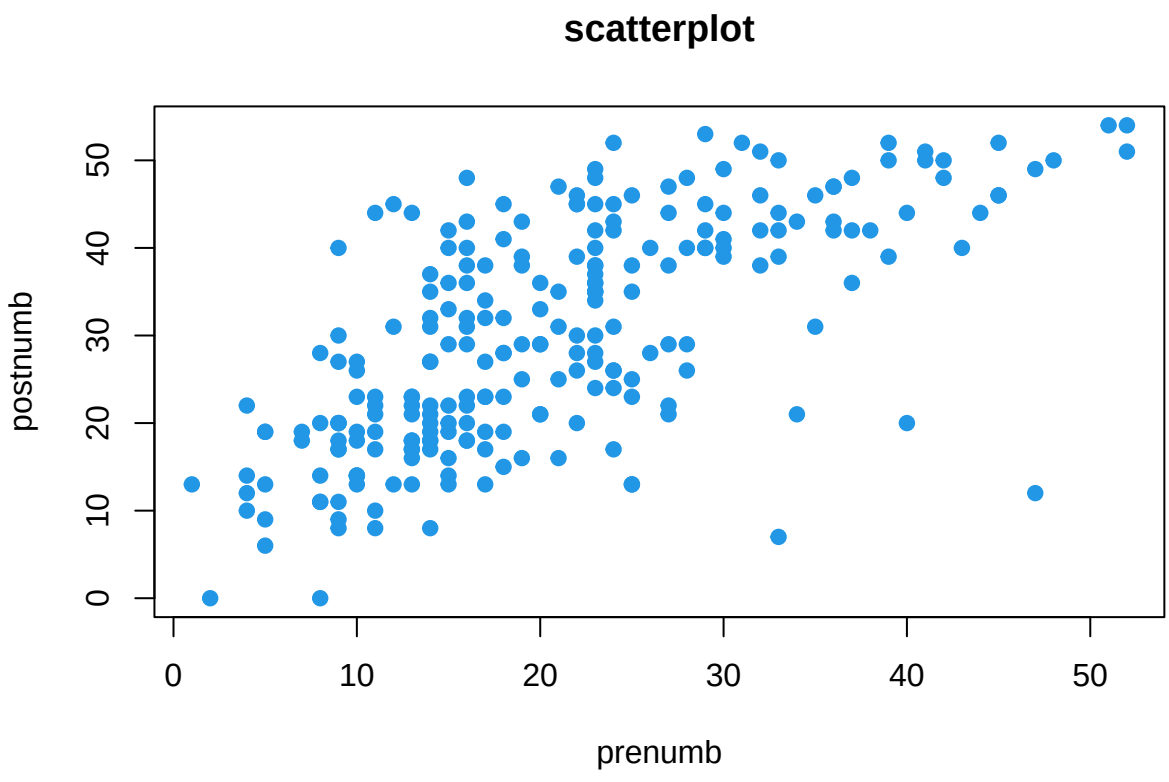
We can plot the relation between the dependent and independent variable in a scatterplot. In this, we specify the label for the x-axis (`xlab`) and the label for the y-axis (`ylab`), a main title for the plot (`main`), a plotting character (`pch`), and a color (`col`).

First, we set the graphical parameters in such a way that two plots can be shown below each other:

```
par(mfrow = c(2,1))
```

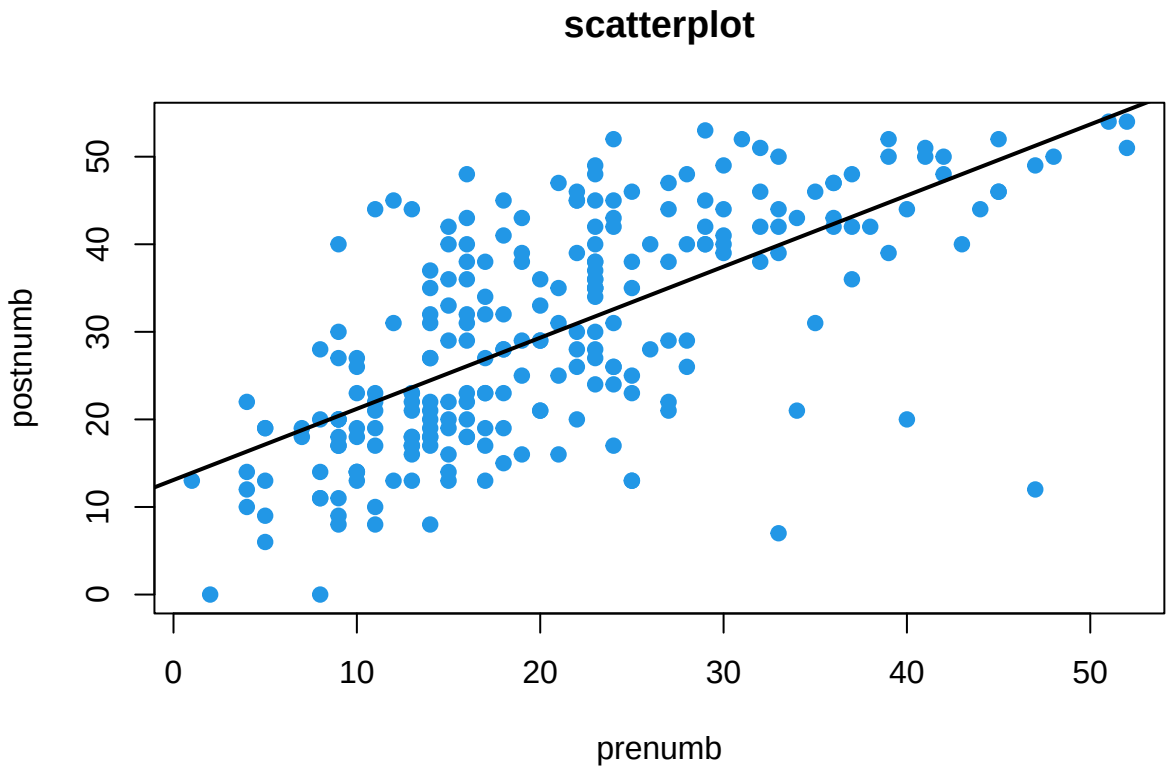
The plot is rendered by:

```
plot(data$prenumb, data$postnumb,
     xlab = "prenumb", ylab = "postnumb",
     main = "scatterplot", pch = 19, col = 4)
```



Now, we can also plot a regression line, to visualize the relation between both variables (where we use a line width (lwd) that is thicker than the default one):

```r
abline(lm1, lwd = 2)
```



Example 3: Multiple linear regression

We are now going to run a multiple regression model in which `postnumb` is regressed on the combination of `prenumb`, `age`, and `sex`. From the Section 'Unstandardized vs centered vs standardized', we can see that it can be helpful for interpreting the regression parameter estimates to use centered continuous variables:

```r
age_c <- scale(data$age, center = TRUE, scale = FALSE)
prenumb_c <- scale(data$prenumb, center = TRUE, scale = FALSE)

lm2_cc <- lm(postnumb ~ prenumb_c + age_c + sex, data = data)
summary(lm2_cc)
```

```
Call:
lm(formula = postnumb ~ prenumb_c + age_c + sex, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-37.846  -6.465  -0.516   5.696  22.851

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 29.79620    0.88835  33.541   <2e-16 ***
prenumb_c    0.78031    0.06383  12.224   <2e-16 ***
```

```
age_c        0.12666    0.10905   1.162    0.247
sex2         0.49529    1.23371   0.401    0.688
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.503 on 236 degrees of freedom
Multiple R-squared:  0.4596,    Adjusted R-squared:  0.4527
F-statistic:  66.9 on 3 and 236 DF,  p-value: < 2.2e-16
```

Now, the intercept is equal to 29.8, which is the mean `postnumb` value for a boy with an average `prenumb` value and an average `age` value. The analogue mean for girls is: 29.8 + 0.5.
Additionally, if `age` increases with 1 point (in this case, 1 month), then `postnumb` increases with the regression coefficient of `age`: 0.13. Likewise, if `prenumb` increases with 1 point, then `postnumb` increases with the regression coefficient of `prenumb`: 0.78.
Stated otherwise, the regression equation for boys now

$$Y_{postnumb} = 29.8 + 0.78 * X_{prenumb_c} + 0.13 * X_{age_c},$$

while the regression line for girls equals

$$Y_{postnumb} = 29.8 + 0.5 + 0.78 * X_{prenumb_c} + 0.13 * X_{age_c}.$$

Informative hypotheses in linear regression

We can also evaluate informative hypotheses in the case of multiple linear regression, like in the ANOVA example. Here, we will only inspect the second method, that is, the GORIC.

Let us assume that we expect beforehand that `prenumb` is the most important predictor (so, better than age); and that girls have a higher mean of `postnumb` than boys.

Note that, if you want to compare regression coefficients of continuous predictors (`prenumb` versus `age`), you have to standardize the variables (such that they are on the same scale and thus comparable):

```
postnumb_s <- scale(data$postnumb)
prenumb_s <- scale(data$prenumb)
age_s <- scale(data$age)
```

Next, we regress standardized `postnumb` on the combination of standardized `prenumb`, standardized `age` and **unstandardized** sex; and we ask for the summary of the model:

```
lm2_s <- lm(postnumb_s ~ prenumb_s + age_s + sex, data = data)
summary(lm2_s)
```

```
Call:
lm(formula = postnumb_s ~ prenumb_s + age_s + sex, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.94615 -0.50326 -0.04019  0.44341  1.77884

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.02008    0.06915  -0.290    0.772
prenumb_s    0.64908    0.05310  12.224   <2e-16 ***
age_s        0.06193    0.05332   1.162    0.247
sex2         0.03856    0.09604   0.401    0.688
---
```

```
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7398 on 236 degrees of freedom
Multiple R-squared:  0.4596,    Adjusted R-squared:  0.4527
F-statistic:  66.9 on 3 and 236 DF,  p-value: < 2.2e-16
```

To specify the informative hypothesis/-es to be evaluated with the GORIC, we first check the names of the coefficients. Subsequently, we specify the hypothesis using these names:

```
coef(lm2_s)
```

```
(Intercept)   prenumb_s       age_s        sex2
-0.02008171  0.64907518  0.06193487  0.03855688
```

```
H1_lm <- "prenumb_s > age_s; sex2 > 0"
```

If you have not installed and loaded the R-package `restriktor`, you will have to do this before you can use the `goric` function:

```
if (!require("restriktor")) install.packages("restriktor")
library(restriktor)
```

```
out_lm <- goric(lm2_s,
                hypotheses = list(H1_lm = H1_lm), comparison = "complement")
#summary(out_lm)
out_lm
```

```
restriktor (0.6-10): generalized order-restricted information criterion:

Results:
        model    loglik  penalty    goric  loglik.weights  penalty.weights  goric.weights
1       H1_lm  -266.192    3.978  540.340           0.520            0.633          0.652
2  complement  -266.274    4.522  541.593           0.480            0.367          0.348

Conclusion:
The order-restricted hypothesis 'H1_lm' has 1.87 times more support than its complement.
```

Here, we find that the order-restricted hypothesis H_1 receives approximately 1.87 times more support than its complement.

As another example, let us suppose that we only hypothesize that the variable `prenumb` is more important than the variable `age`. We can specify this hypothesis as shown below.

```
H2_lm <- "prenumb_s > age_s"
out2_lm <- goric(lm2_s,
                 hypotheses = list(H2_lm = H2_lm), comparison = "complement")
#summary(out2_lm)
out2_lm
```

```
restriktor (0.6-10): generalized order-restricted information criterion:

Results:
        model    loglik  penalty    goric  loglik.weights  penalty.weights  goric.weights
1       H2_lm  -266.192    4.500  541.385           1.000            0.500          1.000
2  complement  -286.056    4.500  581.112           0.000            0.500          0.000

Conclusion:
The order-restricted hypothesis 'H2_lm' has 423224262.76 times more support than its complement.
```

From the output, one can conclude that the order-restricted hypothesis H_2 has many times more support than its complement.

Alternatively, one can could be interested in comparing competing hypotheses; for instance, when the literature shows two or more theories. In that case, one should evaluate both/all of them, together with either the unconstrained (to make sure that none of them is weak) of their complement (as another competing theory). When one of the hypotheses is not weak, one can compare the competing theories/hypotheses with each other and conclude which one is the best and how much better.

Extra: Be aware (ANOVA or regression)

Here, we start from the beginning again: We are interested in the increase in the knowledge of numbers after watching Sesame Street:

```
data <- read.table("data/sesameR.txt", header = TRUE,
                   na.string = c("-999", "-99999"))

data <- data.frame(data)
```

Since 'the increase in the knowledge of numbers after watching Sesame Street' is not a variable in the data, we need to create a postnumb-prenumb variable:

```
data <- data.frame(data)
data$IncreaseNumb <- data$postnumb-data$prenumb
```

Here, `IncreaseNumb` (increase in the knowledge of numbers after watching Sesame Street) is the dependent variable and `viewcat` is the predictor (frequency of viewing; 1 = rarely watched the show, 2 = once or twice a week, 3 = three to five times a week, 4 = watched the show on average more than 5 times a week).

Next, we create an object called anova3 to store the analysis output in, and we specify that the variables should come from our dataset `data`:

```
anova3 <- lm(IncreaseNumb ~ viewcat, data = data)
summary(anova3)
```

```
Call:
lm(formula = IncreaseNumb ~ viewcat, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-45.270  -5.753  -0.253   6.214  25.247

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   2.7201     1.5172   1.793   0.0743 .
viewcat       2.5166     0.5447   4.620 6.29e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.295 on 238 degrees of freedom
Multiple R-squared:  0.0823,    Adjusted R-squared:  0.07845
F-statistic: 21.35 on 1 and 238 DF,  p-value: 6.286e-06
```

Interpreting the output

The lm output has several parts.

Call tells us what we asked the lm function to do. Here, we can see that we wanted to predict `IncreaseNumb` using `viewcat` which are variables from the dataset called `data`.

Residuals describe how the model residuals are distributed. It gives the minimum and maximum values, as well as the median and the first and third quartile. Ideally, we want the residuals to be normally distributed.

Coefficients are the model parameter estimates. Often, these are of main interest.
The columns contain the (regression coefficients) estimates themselves, standard errors, t statistics, and p-values. In other words, we obtain the regression coefficients and several measures of the preciseness with which we estimated them.
The rows correspond to the various predictors. In this example, the first row is for the intercept (i.e., the mean of the dependent variable when all the predictors are set to 0), and the second row is for our single predictor (see below).

At the bottom, we receive estimates of the model's accuracy. For example, the R-squared denotes the proportion of variance explained by the model. In this case, we explain only about 8%.

From this analysis, we conclude that with a one point increase of `viewcat`, `IncreaseNumb` increases with 2.52. Additionally, it would be very unlikely to observe this regression coefficient if it was really zero in population ($p < .001$), that is, it is significantly different from 0.

Can we now conclude that `IncreaseNumb` is predicted by different groups? Unfortunately, no. This is not the case, since (according to R) `viewcat` is not a factor (i.e., grouping variable):

```
is.factor(data$viewcat)
```

```
[1] FALSE
```

```
#> levels(sesamdata$viewcat)
#NULL
```

We can ask for the class of the variable `viewcat`:

```
class(data$viewcat)
```

```
[1] "integer"
```

It is a numeric variable. Consequently, the function `lm` will treat it as a continuous predictor. We need to tell R that `viewcat` is a factor, as we do in the next subsection.

ANOVA: using a grouping variable

Tell R that `viewcat` is a factor:

```
data$viewcat <- factor(data$viewcat)
#> levels(sesamdata$viewcat)
#[1] "1" "2" "3" "4"
```

Now, you can run the planned ANOVA:

```
anova1 <- lm(IncreaseNumb ~ viewcat, data = data)
summary(anova1)
```

```
Call:
lm(formula = IncreaseNumb ~ viewcat, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-45.906  -5.671  -0.230   6.407  24.717
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)    4.593      1.267   3.623 0.000356 ***
viewcat2       3.691      1.747   2.112 0.035697 *
viewcat3       6.314      1.721   3.668 0.000302 ***
viewcat4       7.585      1.734   4.375 1.82e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.314 on 236 degrees of freedom
Multiple R-squared:  0.08624,   Adjusted R-squared:  0.07462
F-statistic: 7.424 on 3 and 236 DF,  p-value: 8.996e-05
```

Alternatively, you could use:

```
anova2 <- lm(IncreaseNumb ~ -1 + viewcat, data = data) # model 2
# Note: -1 means suppressing the intercept.
summary(anova2)
```

```
Call:
lm(formula = IncreaseNumb ~ -1 + viewcat, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-45.906  -5.671  -0.230   6.407  24.717

Coefficients:
         Estimate Std. Error t value Pr(>|t|)
viewcat1    4.593      1.267   3.623 0.000356 ***
viewcat2    8.283      1.202   6.889 5.07e-11 ***
viewcat3   10.906      1.164   9.367  < 2e-16 ***
viewcat4   12.177      1.183  10.295  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.314 on 236 degrees of freedom
Multiple R-squared:  0.5187,    Adjusted R-squared:  0.5105
F-statistic: 63.58 on 4 and 236 DF,  p-value: < 2.2e-16
```

What is the difference between model 1 (`anova1`) and model 2 (`anova2`), and how do the results relate? By default, as in `anova1`, an intercept is included in the model. Then, the intercept is the mean for the reference category and the other estimates are the differences in group means w.r.t. the reference category. In `anova2` the intercept is suppressed (by adding `-1`). In this model, you obtain the means per category / group, that is, you obtain all four group means (and the tests are w.r.t. those means).

Note on ANOVA table

To obtain the anova table of model 2, we can run the following command.

```
anova(anova2)
```

```
Analysis of Variance Table

Response: IncreaseNumb
           Df Sum Sq Mean Sq F value    Pr(>F)
viewcat     4  22062  5515.6  63.578 < 2.2e-16 ***
```

```
Residuals 236  20474    86.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the F-statistic can be computed as follows:

$$\frac{MS_{viewcat}}{MS_{residuals}} = (anova(anova2)[,3])[1]/(anova(anova2)[,3])[2] = 63.58.$$

We can see, from the ANOVA table or the anova2 output, that the p-value is smaller than 0.05 ($p < .001$), which means you can reject the null hypothesis. Note that, here, the null is H_0: all means are equal to 0, that is, $\mu_1 = \mu_2 = \mu_3 = \mu_4 = 0$; while in Model 1 (anova1), the null stated that the mean of IncreaseNumb is equal for all four categories, that is, $\mu_1 = \mu_2 = \mu_3 = \mu_4$ and not per se also equal to 0.

Hence, when you are interested in the null stating the mean of IncreaseNumb is equal for all four categories, you should use anova1. In case the interest lies in evaluating informative hypotheses, it is often more helpful to use anova2, as can be seen in teh next subsection.

Note on informative hypotheses

When you would use model 2 (anova 2), you are interested in "mu1 < mu2 < mu3 < mu4", that is, H1 <- "viewcat1 < viewcat2 < viewcat3 < viewcat4"
when you want to evaluate whether the group means increase (when watching more often).

If you would use anova2, H1 should be changed accordingly; where we will use the following:
viewcat2 = mu2 - mu1
viewcat3 = mu3 - mu1
viewcat4 = mu4 - mu1
That is, every estimate denotes the difference in mean versus reference category 1.
Then, "mu1 < mu2 < mu3 < mu4" is written as:
H1_2 <- "viewcat2 > 0; viewcat2 < viewcat3 < viewcat4"
that is:
(mu2 - mu1) > 0, (mu2 - mu1) < (mu3 - mu1) < (mu4 - mu1).
(mu2 - mu1) > 0, mu2 < mu3 < mu4.
mu2 > mu1, mu2 < mu3 < mu4.
mu1 < mu2, mu2 < mu3 < mu4.
mu1 < mu2 < mu3 < mu4.

If we would run this, we indeed obtain the same results:

```
H1_2 <- "viewcat2 > 0; viewcat2 < viewcat3 < viewcat4"
set.seed(123) # to ensure every time same result
              # (and to have the option to test the sensitivity of the results)
goric(anova1, hypotheses = list(H1_2 = H1_2), comparison = "complement")
```

```
restriktor (0.6-10): generalized order-restricted information criterion:

Results:
        model    loglik  penalty     goric  loglik.weights  penalty.weights  goric.weights
1        H1_2  -874.096    3.095  1754.381           0.574            0.855          0.888
2  complement  -874.394    4.870  1758.528           0.426            0.145          0.112

Conclusion:
The order-restricted hypothesis 'H1_2' has 7.95 times more support than its complement.
```

```
#
out # see Section 3.2
```

```
restriktor (0.6-10): generalized order-restricted information criterion:

Results:
        model    loglik  penalty     goric  loglik.weights  penalty.weights  goric.weights
1          H1  -874.096    3.095  1754.381           0.574            0.855          0.888
2  complement  -874.394    4.870  1758.528           0.426            0.145          0.112

Conclusion:
The order-restricted hypothesis 'H1' has 7.95 times more support than its complement.
```

Unstandardized vs centered vs standardized

Unstandardized coefficients

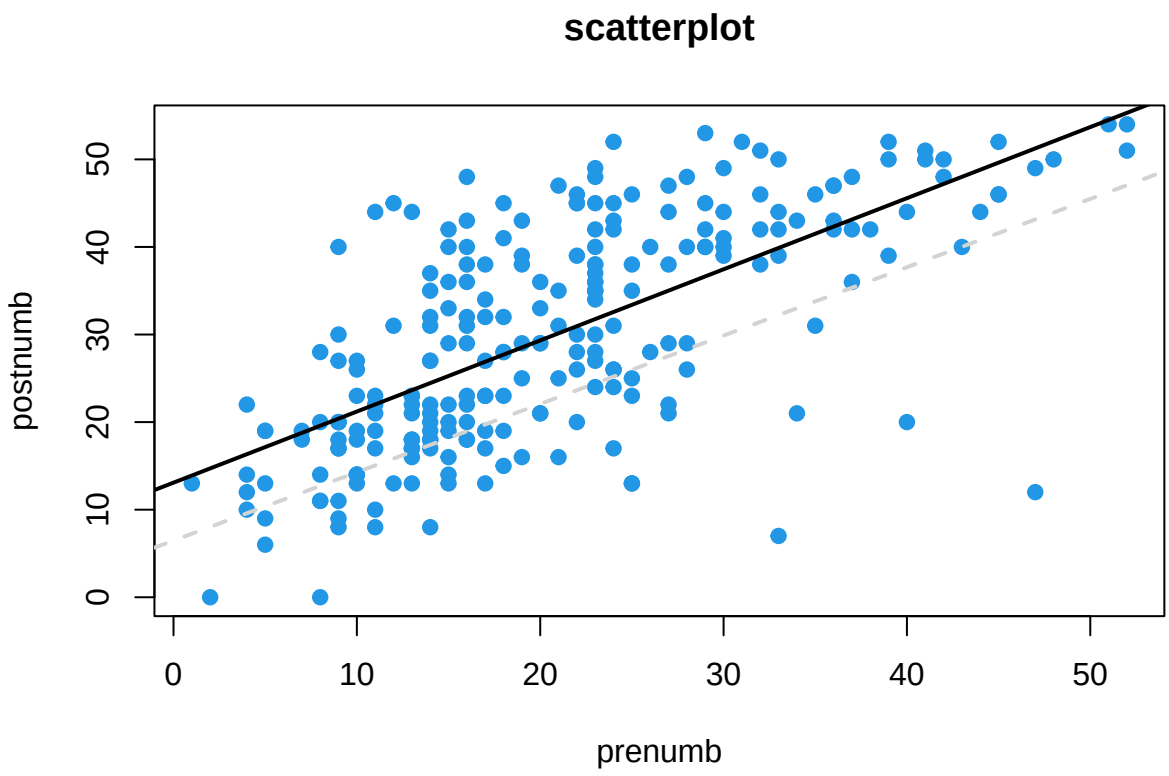
We are now going to run a multiple regression model in which `postnumb` is regressed on the combination of `prenumb`, `age`, and `sex` (the original, un-centered, and unstandardized variables).

```
lm2 <- lm(postnumb ~ prenumb + age + sex, data = data)
summary(lm2)
```

```
Call:
lm(formula = postnumb ~ prenumb + age + sex, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-37.846  -6.465  -0.516   5.696  22.851

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.46957    5.71302   1.132    0.259
prenumb      0.78031    0.06383  12.224   <2e-16 ***
age          0.12666    0.10905   1.162    0.247
sex          0.49529    1.23371   0.401    0.688
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.503 on 236 degrees of freedom
Multiple R-squared:  0.4596,    Adjusted R-squared:  0.4527
F-statistic:  66.9 on 3 and 236 DF,  p-value: < 2.2e-16
```

Based on this, we can add another regression line of `postnumb ~ prenumb` to the scatterplot: In this case, this relation is conditional on the other predictor(s)! To distinguish this line from the previously drawn black line (where there were no controlling variables), we are going to make the line type (`lty`) dashed, and the line color (`col`) light grey:

```
abline(a = coefficients(lm2)[1],
       b = coefficients(lm2)[2],
       lwd = 2, lty = 2, col = "lightgrey")
```



Since this effect is conditional on the other predictors (i.e., age and sex), this is the effect of `prenumb` on `postnumb` when someone's age is equal to 0, and sex is equal to 0 which indicates boys (the reference category).

Unstandardized coefficients for centered data

It might be convenient to center age, such that we obtain the effect of prenumb on postnumb for a boy with an age equal to the mean age, which is 51.525 months.

```
age_c <- scale(data$age, center = TRUE, scale = FALSE)
```

Then, we again specify the regression model and ask for the output.

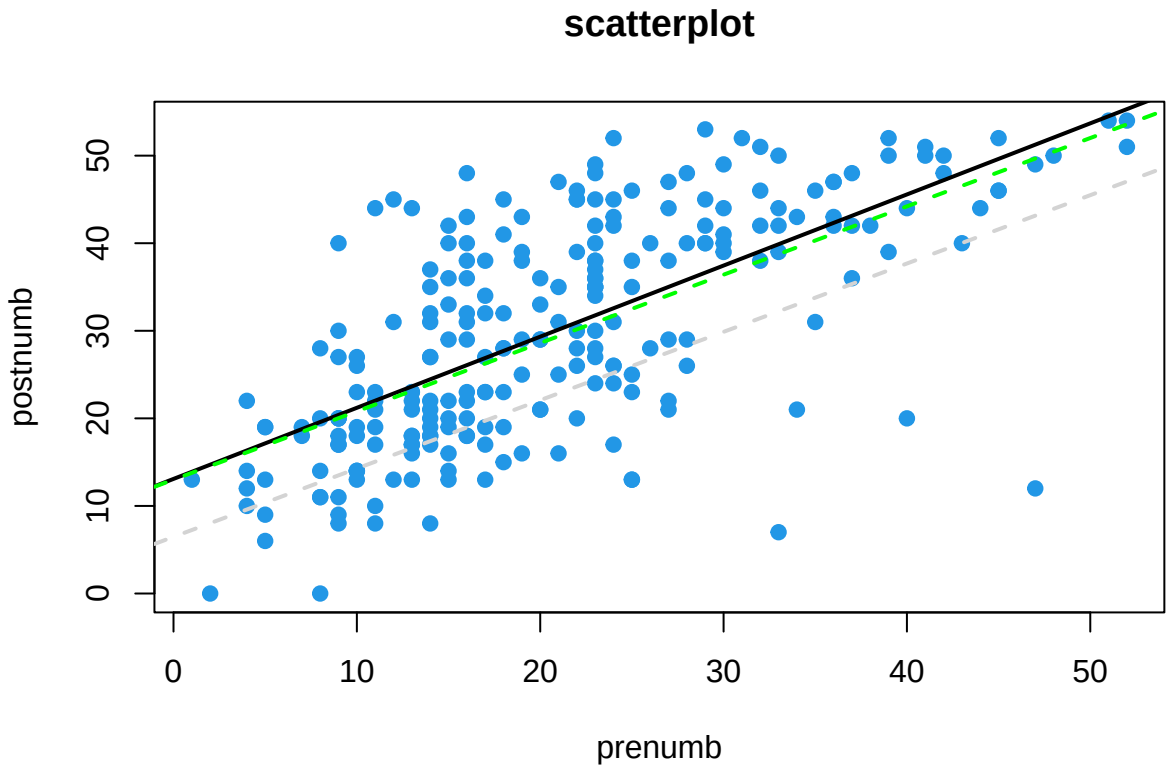
```
lm2_c <- lm(postnumb ~ prenumb + age_c + sex, data = data)
summary(lm2_c)
```

```
Call:
lm(formula = postnumb ~ prenumb + age_c + sex, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-37.846  -6.465  -0.516   5.696  22.851

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 12.99576    2.35751   5.512 9.25e-08 ***
prenumb      0.78031    0.06383  12.224  < 2e-16 ***
age_c        0.12666    0.10905   1.162    0.247
```

```
sex          0.49529    1.23371   0.401    0.688
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.503 on 236 degrees of freedom
Multiple R-squared:  0.4596,    Adjusted R-squared:  0.4527
F-statistic:  66.9 on 3 and 236 DF,  p-value: < 2.2e-16
```

To update our plot, we add the new regression line to our existing plot. This time, we make it a green dashed line:

```
abline(a = coefficients(lm2_c)[1], b = coefficients(lm2_c)[2],
       lwd = 2, lty = 2, col = "green")
```



The green line now indicates the regression line for boys (sex = male/boys), with the average age in the sample, while before, it was the line for boys with age = 0 months. Notably, since age and sex are not significant, this new line is close to the one from the unconditional/first model (lm1).

The regression equation for boys is now

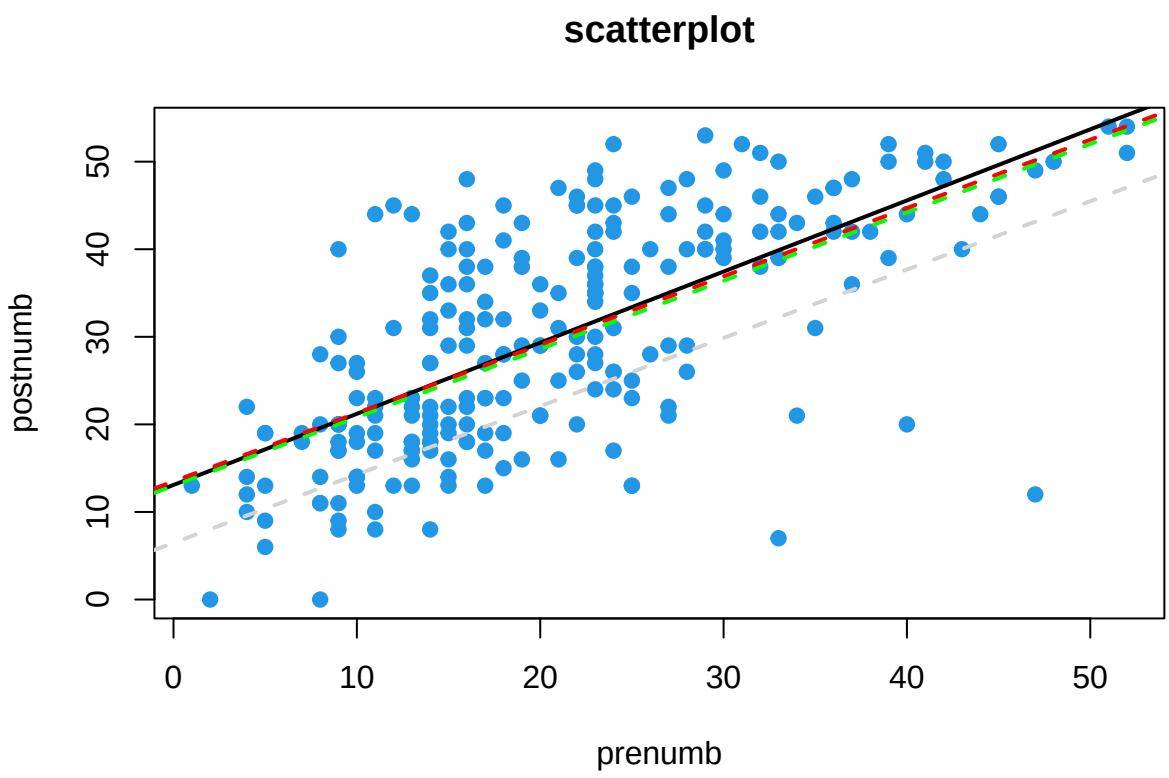
$$Y_{postnumb} = 13 + 0.78 * X_{prenumb} + 0.13 * X_{age},$$

while the regression line for girls is now equal to

$$Y_{postnumb} = 13 + 0.5 + 0.78 * X_{prenumb} + 0.13 * X_{age}.$$

The regression line for girls (in red) can also be added to the plot by means of the following code:

```
abline(a = (coefficients(lm2_c)[1] + coefficients(lm2_c)[4]),
       b = coefficients(lm2_c)[2],
       lwd = 2, lty = 2, col = "red")
```



By including interactions, the slopes can also differ between boys and girls. Since we did not include any interaction term in the model, only the intercept differs.

In addition, it might be helpful to center the variable `prenumb` as well, since this might ease the interpretation of the intercept (since the intercept is then equal to the predicted score for a boy with the mean score on the variable `prenumb` and the mean score on the variable `age`). Then, we again model the relations and ask for the output:

```
prenumb_c <- scale(data$prenumb, center = TRUE, scale = FALSE)

lm2_cc <- lm(postnumb ~ prenumb_c + age_c + sex, data = data)
summary(lm2_cc)
```

```
Call:
lm(formula = postnumb ~ prenumb_c + age_c + sex, data = data)

Residuals:
    Min      1Q  Median      3Q     Max
-37.846  -6.465  -0.516   5.696  22.851

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

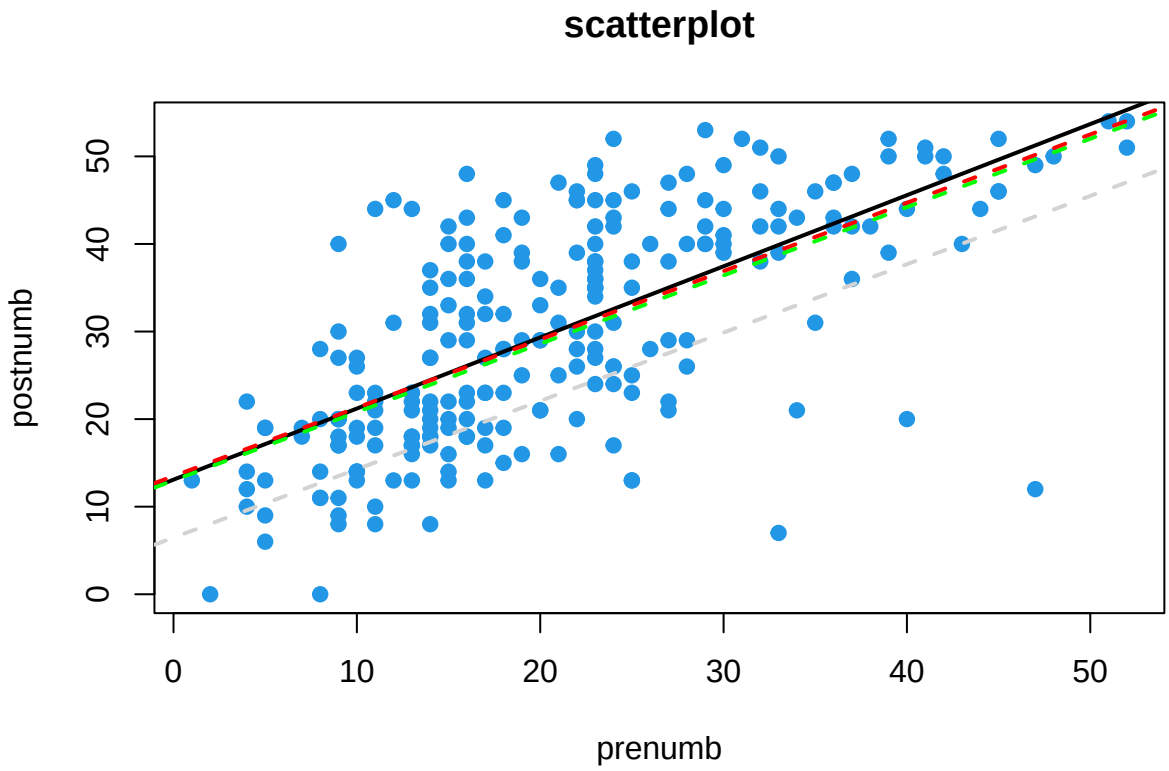
```
(Intercept) 29.30091    1.97399  14.843   <2e-16 ***
prenumb_c    0.78031    0.06383  12.224   <2e-16 ***
age_c        0.12666    0.10905   1.162    0.247
sex          0.49529    1.23371   0.401    0.688
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9.503 on 236 degrees of freedom
Multiple R-squared:  0.4596,    Adjusted R-squared:  0.4527
F-statistic:  66.9 on 3 and 236 DF,  p-value: < 2.2e-16
```

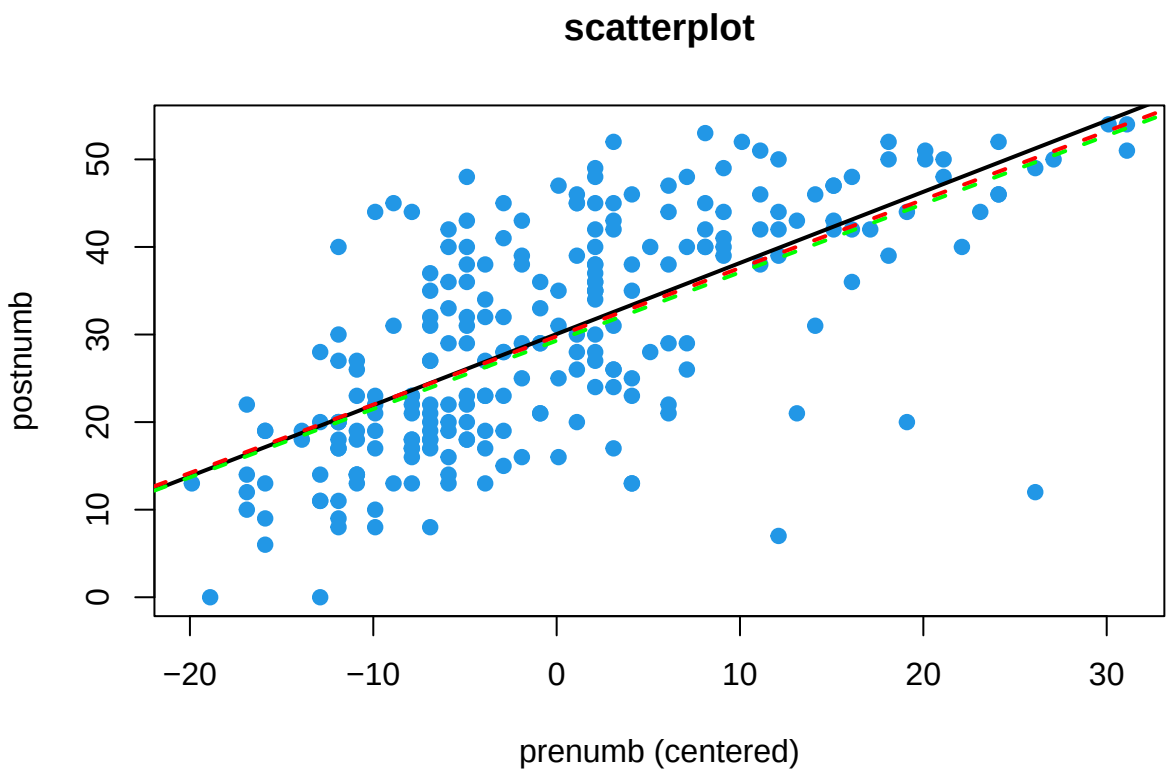
Now, the intercept is equal to 29.3, which is the mean `postnumb` value for a boy with an average `prenumb` value and an average `age` value.

We can plot this relationship by means of the code below. To ease the interpretation of the visualized relationships, we plot the regression lines for the data as a whole (in black), the relationships for boys (in green), and the relationships for girls (in red):

```r
plot(prenumb_c, data$postnumb,
     xlab = "prenumb (centered)",
     ylab = "postnumb",
     main = "scatterplot",
     pch = 19, col = 4)

lm1_c <- lm(postnumb ~ prenumb_c, data = data)
# for the data as a whole
abline(lm1_c, lwd = 2)

# for boys
abline(a = coefficients(lm2_cc)[1],
       b = coefficients(lm2_cc)[2],
       lwd = 2, lty = 2, col = "green")
# for girls
abline(a = (coefficients(lm2_cc)[1] + coefficients(lm2_cc)[4]),
       b = coefficients(lm2_cc)[2],
       lwd = 2, lty = 2, col = "red")
```

Next, you will see such a plot for both the original prenumb variable and the centered one.


scatterplot
postnumb
prenumb
0
10
20
30
40
50



Note that the only thing that happened is a shift, since we centered the variable prenumb (and age). That is, it only changes the intercept (not the slopes).

To prevent that our next plot will also be of the size (2,1), we reset the graphical device:

```
par(mfrow = c(1,1))
```

Standardized coefficients

Before, we used unstandardized regression coefficients, since these are easily interpretable. If, for example, `age` increases with 1 point (in this case, 1 month), then `postnumb` increases with the regression coefficient of `age`. However, if we standardize the variables by means of the standard deviations of the variables, we can easily compare the strength of the parameter estimates, since these are now all on the same scale (namely with a mean of zero and a standard deviation of 1). Bear in mind that it does not make sense do this in the case of categorical variables.

First, we are going to standardize the variables that are not categorical:

```
postnumb_s <- scale(data$postnumb)
prenumb_s <- scale(data$prenumb)
age_s <- scale(data$age)
```

Second, we regress standardized `postnumb` on the combination of standardized `prenumb`, standardized `age` and **unstandardized** sex. Third, we ask for the summary of the model.

```
lm2_s <- lm(postnumb_s ~ prenumb_s + age_s + sex, data = data)
summary(lm2_s)
```

```
Call:
lm(formula = postnumb_s ~ prenumb_s + age_s + sex, data = data)

Residuals:
     Min       1Q   Median       3Q      Max
-2.94615 -0.50326 -0.04019  0.44341  1.77884

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -0.05864    0.15367  -0.382    0.703
prenumb_s    0.64908    0.05310  12.224   <2e-16 ***
age_s        0.06193    0.05332   1.162    0.247
sex          0.03856    0.09604   0.401    0.688
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7398 on 236 degrees of freedom
Multiple R-squared:  0.4596,    Adjusted R-squared:  0.4527
F-statistic:  66.9 on 3 and 236 DF,  p-value: < 2.2e-16
```

The interpretation of the regression coefficients is more difficult, since these are in terms of standard deviations. But: Now, it is now possible to compare the strength of the standardized regression coefficients (betas). That is, we can now conclude that prenumb is a more important predictor than age or that the predictive strength of prenumb is higher than that of age.
The standardized estimates (of continuous variables) are also needed when you evaluate an informative hypothesis in which you compare the parameters of continuous variables; e.g., when you compare the predictive strength of two continuous variables.

These standardized values could also be obtained based on the model with unstandardized estimates; e.g.,

```
coef(lm2)[2]*sd(data$prenumb)/sd(data$postnumb)
```

```
  prenumb
0.6490752
```

Indeed, this gives the same estimate as `coef(lm2_s)[2]` $= 0.649$. Note that in the case of missing data, this does not need to be the same.

Extra: Miscellaneous

Saving data in R

We can save our data by means of the following command, such that we can load it the next time we need the dataset without having to perform the same modifications as we did this time.

```
save(sesamdata, file = "sesamdata_analyses.Rda")

load("sesamdata_analyses.Rda") # for next time, when we want to load it
```

We can also save the entire workspace (that is, all output and objects we created), such that we can load the complete environment the next time we want to use it. This can be done by means of the following command:

```
save(list = ls(),file = "sesamdata_all_analyses.RData")

load("sesamdata_all_analyses.RData") # for next time, when we want to load it
```

Missing data

Currently, we did nothing with missing data, that is, we perform complete case analysis (which is the same as listwise deletion). Missing data analysis is a complete course on its own.

When data in SPSS are not coded as missing accordingly, or when you read in another file in which you cannot assign coding (like in txt files), you can code your missing values in R manually. In case you have values of "99" and "999" in your data that represent missing values, the code below will recode these values to `NA` values in R. First, let us ask for a summary of the variable `peabody`.

```
summary(data$peabody)
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  27.00   35.00   42.00   46.77   58.00   99.00
```

```
is.numeric(data$peabody)
```

```
[1] TRUE
```

We see that the variable is a numeric variable, and that there is at least one value of 99 that should be coded as missing values (note that 99 seems to be out of range and if you know your data, you know this coding of course). To recode observations with a value of 99 and 999, you could use the following function and code below.

```
if (!require("car")) install.packages("car")
```

```
Loading required package: car
```

```
Loading required package: carData
```

```
library(car)
RecodeMissing <- function(x) {
  if(is.numeric(x)) {
    x <- car::recode(x, "99=NaN; 999=NaN",
                     as.factor = FALSE, as.numeric = TRUE)}
  else{
    x <- car::recode(x, "'99'=NA; 'unknown'=NA",
                     as.factor = TRUE, as.numeric = FALSE)}
}
```

This function can be applied to the data. We will inspect the number of missings in the variable `peabody` before and after we apply the function, to see whether recoding the values has been done correctly.

```
sum(is.na(data$peabody))
```

```
[1] 0
```

Before applying the function, we see that there are no missing values.

Next, we apply our function. Here, we make use of the function `lapply`, which applies this function to all columns of the data that you specify.

```
data <- lapply(data, RecodeMissing)
data <- data.frame(data)
# Note: RecodeMissing is the function we made before
sum(is.na(data$peabody))
```

```
[1] 1
```

The function `sapply` applies this function to all columns of the data that you specify. After applying the function, we see that one value of 99 has been recoded into a NA.

Then, we should also check whether peabody is still a numeric variable.

```
is.numeric(data$peabody)
```

```
[1] TRUE
```

And we see that it is still a numeric variable, despite recoding values of 99 into a missing value.

Loading other files types

In the examples above, we loaded a txt file. But what if you have another file type?

Loading SPSS files

If you want to use an SPSS (`.sav`) file, you will have to install and load an external package called **haven**. Packages like **haven** are additional functions that are not contained in base R, but might be needed for your analyses.

After you install a package, it is stored on your PC/laptop, but every time you open R, you will have to load the package again. You can install and load the package **haven** by means of the following commands.

```
if (!require("haven")) install.packages("haven")  # installs package
```

```
Loading required package: haven
```

```
library(haven)                                    # loads the package
```

When you have the required package, you can load `.sav` data using the function `read_sav`. Usually, this package reads in all variables well, **BUT** it will often not recognize categorical variables, so you will have to specify these variables as factors yourself.

```
data_test <- read_sav("name_of_data.sav")
```

You can also use the package `foreign`. It can handle categorical variables pretty well, but might have difficulties with reading in other variables. Therefore, it is advisable to always check whether loading in your data went okay.

```
## to read in data with value labels (such as categorical data)
data_test <- read.spss(file = "name_of_data.sav", to.data.frame = TRUE)

## if you don't want to use the labels, but read the data as if it was numeric data
data_test <- read.spss(file = "name_of_data.sav", to.data.frame = TRUE, use.value.labels = FALSE)
```

Loading Excel files

For **Excel**, it is advisable to save your Excel file within Excel as a tab delimited `.txt` file, and open this file in R. Reading in a tab delimited `.txt` file, can be done as follows.

Note the default for decimal separators is ".", while you might have used a comma (","). If you have used a comma as decimal separator, you could simply specify `dec = ","` within the command line, such that R recognizes the commas as decimal separators.

```
data_test <- read.csv("name_of_data.txt", header = TRUE, sep = "\t")
```