Informative Hypotheses

Herbert Hoijtink, Nikola Sekulovski

2021-11-05

Contents

Chapter 1

Introduction

This is a template GitBook based on *A GitBook Example for Teaching* and *bookdown: Authoring Books and Technical Documents with R Markdown.*

Chapter 2

Monte Carlo Simulations

```
library(tidyverse)
# {-} no chapter number
# see Appendix.Rmd how to make parts in a book
# don't forget library(bookdown)
# R - help -cheatsheets
```

2.1 The Confidence Interval

In this exercise I will try to repeat the example given by Gerko Vink

The main idea of this exercise is to illustrate the nature of the *Confidence Interval* as described by Neyman (1934)

We set a seed to make our results reproducible:

```
set.seed(6465)
```

- The first step is to take 100 samples (in this case of size 800) from a *normal distributuon* with $\mu = 0$ and $\sigma = 1$:

```
samples <-plyr::rlply(100, rnorm(800, 0, 1))
```

- Secondly, we need to calculate for the mean of each sample: the absolute bias; standard error lower bound of the 95% confidence interval and upper bound of the 95% confidence interval.

Table 2.1: Here is a table of the samples

Mean	Bias	Std.Err	Lower	Upper	Covered
-0.0945589	0.0945589	0.0353553	-0.1639592	-0.0251585	0
0.0740058	0.0740058	0.0353553	0.0046055	0.1434062	0

We can construct a function that does this:

```
samp_function <- function(x) {

  m <- mean(x)
  n <- length(x)
  se <- 1/sqrt(n)
  bias <- abs(-0 - m)
  df <- n - 1
  interval <- qt(.975, df) * se
  return(c(m, bias, se, m - interval, m + interval))

}

format <- c("Mean" = 0, "Bias" = 0, "Std.Err" = 0, "Lower" = 0, "Upper" = 0)
```

Now we use the constructed function `samp_function` on all 100 samples contained in the object `samples`. And we also add a new column to the results that indicates which CI of the respective samples does contain μ.

```
results <- samples %>%
  vapply(., samp_function, format) %>%
  t %>%
  as_tibble %>%
  mutate(Covered = ifelse(Lower < 0 & Upper > 0, 1, 0))
```

We can also add a table with the sample statistics of the samples whose CI's do not contain μ.

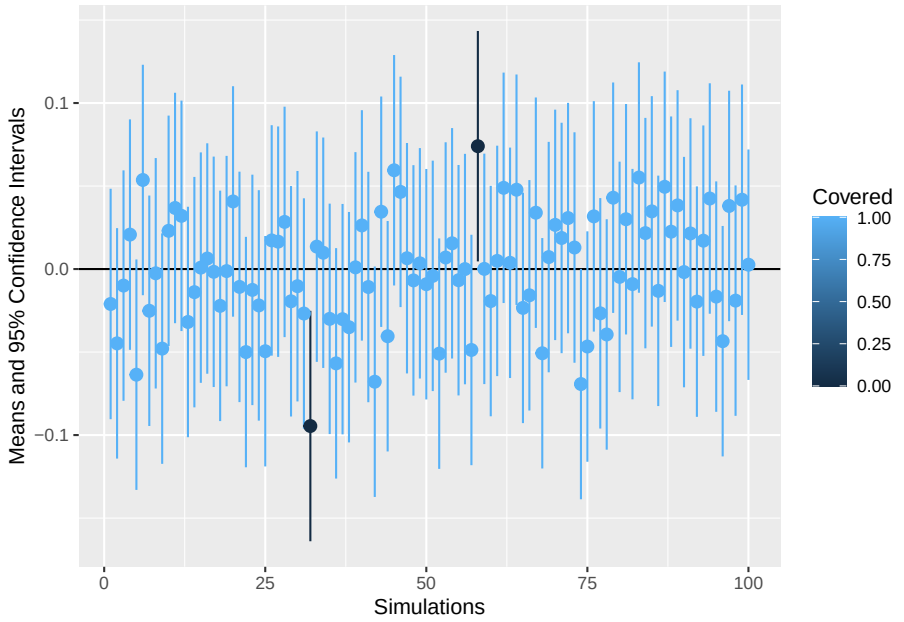
```
# the results contains the table content that is piped into the table
results %>%
  filter(Covered ==0) %>%
  kableExtra::kable(caption = "Here is a table of the samples" )
```

And finally we can also make a nice plot illustrating everything that we did so far.

```
# in the line above also the size of the figure can be adjusted
lims <- aes(ymax = results$Upper, ymin = results$Lower)
ggplot(results, aes(y=Mean, x=1:100, colour = Covered)) +
  geom_hline(aes(yintercept = 0)) +
  geom_pointrange(lims) +
  xlab("Simulations") +
  ylab("Means and 95% Confidence Intervals")
```

```
## Warning: Use of `results$Upper` is discouraged. Use `Upper` instead.
```

```
## Warning: Use of `results$Lower` is discouraged. Use `Lower` instead.
```



In this case only two out of 100 CI's do not include the true population mean.

2.2 The Central Limit Theorem

Here we will also try to illustrate the Central Limit Theorem, in it's most basic form, with a very simple example.

First we draw 1000 samples (again of size 800), form , say, a *Poisson* distribution, of course we could've drawn them from a uniform or an exponential as well.

Now we calculate the mean for each sample:

```
means <- samples_2 %>%
  lapply(., mean) %>%
  as.data.frame() %>%
  t()
```

And now we plot a histogram of the resulting means:

```
hist(t(means))
```

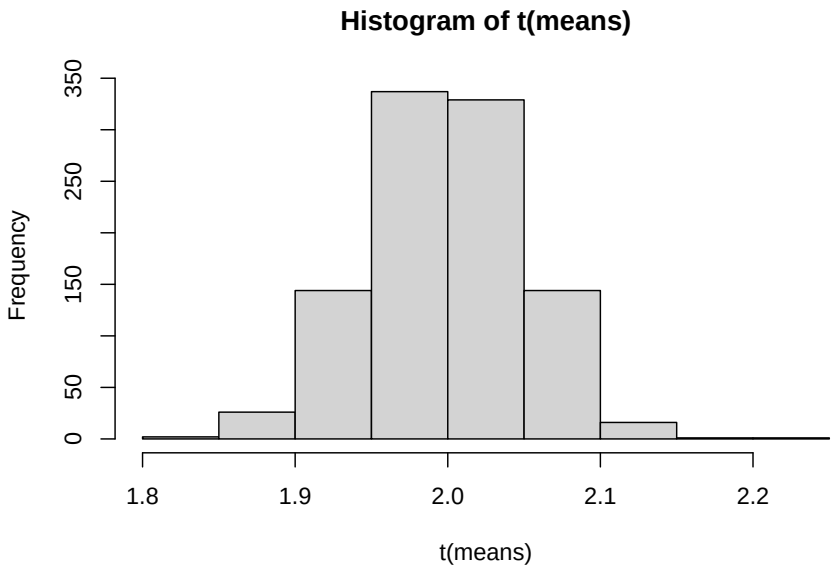


Figure 2.1: Histogram of the sampling distribution of the mean

Chapter 3

Informative hypothesis (Bayesian) testing for multi-level models

3.1 Example for citation

We will be using the *Approximate Adjusted Fractional Bayes factor* proposed by Gu, Mulder, & Hoijtink (2018), which is implemented in the R package `bain` (Gu, Hoijtink, Mulder, & van Lissa, 2021).

We can reference like this Hoijtink, Klugkist, & Boelen (2008) or like this (Hoijtink, Klugkist, & Boelen, 2008)

(PART) `R` package `bain`

Chapter 4

Introduction to `bain`

`bain` is an acronym for **"Bayesian informative hypotheses evaluation"**. It uses the *Bayes factor* (Kass & Raftery, 1995) to evaluate hypotheses specified using *equality* and *inequality* constraints among (linear combinations of) parameters in a wide range of statistical models.

4.1 Available tutorials

Two tutorials are retrievable from the `bain` website under the section **Tutorials**.

The first introducing Bayesian evaluation of informative hypotheses is provided by Hoijtink, Mulder, Lissa, & Gu (2019). By reading the tutorial in combination with executing the analyses contained in `easyBFtutorial.R` and `BFtutorial.R` (available on the aforementioned website) you will quickly learn the basics of Bayesian hypothesis evaluation.

The second containing introduction to Bayesian evaluation of informative hypotheses in the context of structural equation models is provided by Van Lissa et al. (2021)[1].

An overview of all other relevant papers concerning informative hypotheses testing and the package `bain` can be found here.

4.2 Usage

This is how a general call to `bain` looks like:

```
results  <- bain(x, hypothesis, fraction = 1, ...)
```

[1] Users are advised to read these tutorials and this vignette before using `bain`.

4.2.1 Arguments

- `x` An R object containing the outcome of a statistical analysis. Currently, the following objects can be processed:

1) `t_test()` objects (Student's t-test, Welch's t-test, paired samples t-test, one-group t-test, equivalence test). Note that, `t_test` can be used in the same way as `t.test`.
2) `lm()` objects (ANOVA, ANCOVA, multiple regression).
3) `lavaan` objects generated with the `sem()`, `cfa()`, and `growth` functions.
4) A named vector containing the estimates resulting from a statistical analysis. Using this option triggers the `bain.default()` method. Note that, named means that each estimate has to be named such that it can be referred to in hypotheses.
5) Wrapper functions for repeated measures ANOVA(…) and linear two-level models built with `lmer` (TBA when finished).

- `hypothesis` A character string containing the informative hypotheses to evaluate (see section 3.2.2).

- `fraction = 1` A number representing the fraction of information in the data used to construct the prior distribution (see, for example, Gu, Mulder, & Hoijtink (2018)). The default value 1 denotes the minimal fraction, 2 denotes twice the minimal fraction, etc. The examples in chapter 5 show how the `fraction` can be employed to execute a sensitivity analysis. See also Hoijtink, Mulder, Lissa, & Gu (2019) for more details.

- `...` Additional arguments (see next chapter).

4.2.2 The specification of `hypotheses`

`hypotheses` is a character string that specifies which (informative) hypotheses have to be evaluated. A simple example is `hypotheses <- "a > b > c; a = b = c;"` which specifies two hypotheses using three estimates with names "a," "b," and "c," respectively.

The hypotheses specified have to adhere to the following rules (`bain` may still run if you deviate from the rules, however, the output will be nonsense):

- When using `bain` with a `lm` or `t_test` or `lavaan` object, (unique abbreviations of) the names displayed by `coef(x)` have to be used (but see the section 5.3 for additional instructions if multiple group analyses are executed and/or parameters are labeled). If, for example, the names are `cat` and `dog`, `c` and `d` would be unique abbreviations. If, for example, the names are `cato` and `cata` the whole names have to be used.

- When using `bain` with a named vector, parameters are referred to using the names specified in `names()`.
- Linear combinations of parameters must be specified adhering to the following rules:

a) Each parameter name is used at most once.

b) Each parameter name may or may not be pre-multiplied with a number.

c) A constant may be added or subtracted from each parameter name.

d) A linear combination can also be a single number.

Examples are: `3 * a + 5`; `a + 2 * b + 3 * c - 2`; `a - b`; and `5`.

- (Linear combinations of) parameters can be constrained using $<$, $>$, and $=$. For example, `a > 0` or `a > b = 0` or `2 * a < b + c > 5`.
- The ampersand `&` can be used to combine different parts of a hypothesis. For example, `a > b & b > c` which is equivalent to `a > b > c` or `a > 0 & b > 0 & c > 0`.
- Sets of (linear combinations of) parameters subjected to the same constraints can be specified using (). For example, `a > (b,c)` which is equivalent to `a > b & a > c`.
- The specification of a hypothesis is completed by typing ; For example, `hypotheses <- "a > b > c; a = b = c;"`, specifies two hypotheses.
- Hypotheses have to be compatible, non-redundant and possible. What these terms mean will be elaborated below.

The set of hypotheses has to be compatible. For the statistical background of this requirement see Gu, Mulder, & Hoijtink (2018). Usually the sets of hypotheses specified by researchers are compatible, and if not, `bain` will return an error message. The following steps can be used to determine if a set of hypotheses is compatible:

1) Replace a range constraint, e.g., `1 < a1 < 3`, by an equality constraint in which the parameter involved is equated to the midpoint of the range, that is, `a1 = 2`.
2) Replace in each hypothesis the $<$ and $>$ by $=$. For example, `a1 = a2> a3 > a4` becomes `a1 = a2 = a3 = a4`.
3) The hypotheses are compatible if there is at least one solution to the resulting set of equations. For the two hypotheses considered under 1. and 2., the solution is `a1 = a2 = a3 = a4 = 2`. An example of two non-compatible hypotheses is `hypotheses <- "a = 0; a > 2;"` because there is no solution to the equations `a=0` and `a=2`.

Each hypothesis in a set of hypotheses has to be non-redundant. A hypothesis is redundant if it can also be specified with fewer constraints. For

example, `a = b & a > 0 & b > 0` is redundant because it can also be specified as `a = b & a > 0`. `bain` will work correctly if hypotheses specified using only $<$ and $>$ are redundant. `bain` will return an error message if hypotheses specified using at least one $=$ are redundant.

Each hypothesis in a set of hypotheses has to be possible. An hypothesis is impossible if estimates in agreement with the hypothesis do not exist. For example: values for `a, b, c` in agreement with `a > b > c & a < c` do not exist. It is the responsibility of the user to ensure that the hypotheses specified are possible. If not, `bain` will either return an error message or render an output table containing `Inf`'s.

4.2.3 Output

The commands `bain()` or `results<-bain()` followed by `results` or `print(results)` will render the default (most important) output from `bain`. These concern for each hypothesis specified in `hypotheses` the fit, complexity, Bayes factor versus the unconstrained hypothesis, Bayes factor versus its complement, posterior model probability (based on equal prior model probabilities) excluding the unconstrained hypothesis, posterior model probability including the unconstrained hypothesis, and posterior model probability of each hypothesis specified and their joint complement. Note that, all the posterior model probabilities are computed from the Bayes factors of each hypothesis versus the unconstrained hypothesis. In Hoijtink, Mulder, Lissa, & Gu (2019) it is elaborated how these quantities (and the other output presented below) should be interpreted. Additionally, using `summary(results, ci=0.95)`, a descriptives matrix can be obtained in which for each estimate, the name, the value, and a 95% central credibility interval is presented.

The following commands can be used to retrieve the default and additional information from the `bain` output object:

- `results$fit` renders the default output, `results$fit$Fit` contains only the column containing the fit of each hypothesis. In the last command `Fit` can be replaced by `Com`, `BF`, `BF.u`, `BF.c`, `PMPa`, `PMPb`, `PMPc` to obtain the information in the corresponding columns of the default output. Note that, in the columns `BF` and `BF.c` the Bayes factor of the hypothesis at hand versus its complement is displayed. In the column `BF.u` the Bayes factor of the hypothesis at hand versus the unconstrained hypothesis is displayed. `PMPa` renders the posterior model probabilities (based on equal prior model probabilities) of the hypotheses specified. `PMPb` renders the posterior model probabilities (based on equal prior model probabilities) of the hypotheses specified plus the unconstrained hypothesis. `PMPc` renders the posterior model probabilities (based on equal prior model probabilities) of the hypotheses specified plus the complement of the union of these hypotheses. If, in the latter case, the complexity of the complement of

the union of all hypotheses specified is smaller than .05, the hypotheses specified (almost) completely cover the parameter space. In this case `PMPc` is not provided and instead `PMPa` should be used.
- `results$BFmatrix` contains the matrix containing the mutual Bayes factors of the hypotheses specified in `hypotheses`.
- `results$b` contains for each of the groups in the analysis the fraction of information of the data in the group at hand used to specify the covariance matrix of the prior distribution.
- `results$prior` contains the covariance matrix of the prior distribution.
- `results$posterior` contains the covariance matrix of the posterior distribution.
- `results$call` displays the call to `bain`.
- `results$model` displays the named vector or the R object that is input to `bain`.
- `results$n` displays the sample sizes per group.
- `results$independent_restrictions` displays the number of independent constraints in the set of hypotheses under consideration. Note that, in Gu, Mulder, & Hoijtink (2018) and Hoijtink, Gu, & Mulder (2019) the definition given was misprinted (besides R and S also r and s should have been added to the definition).
- `results$fit$Fit_eq` displays the fit of the equality constrained part of each hypothesis. Replacing `Fit_eq` by `Fit_in`, renders the fit of the inequality constrained part of an hypothesis conditional on the fit of the equality constrained part. `Com_eq`, and `Com_in`, respectively, are the complexity counterparts of `Fit_eq`, and `Fit_in`.

Note that, if you have specified two hypotheses that both have a small `BF.u` (close to zero), then there is no support in the data for these hypotheses. In these cases the corresponding entry in `results$BFmatrix` (the Bayes factor comparing both hypotheses) is very unstable and should only be interpreted if repeated analyses using different seeds (see `set.seed()`) render about the same results.

4.2.3.1 Example output

The following example is aimed at illustrating the output from `bain`. The specifics on how to actually use `bain` for testing hypotheses about the parameters of different statistical models are given in the next chapter.

In this example we will use `bain` to test the following two (informative) hypotheses concerning the parameters of a multiple linear regression model, with three (continuous) predictors:

- $H_1 : \beta_1 > 0;\ \beta_2 > 0;\ \beta_3 > 0$
- $H_2 : \beta_1 = \beta_2 < \beta_3$

(1) The default output:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class lm (continuous predi
##
##    Fit   Com   BF.u   BF.c    PMPa  PMPb
## H1 0.871 0.032 27.295 204.167 0.747 0.727
## H2 2.637 0.285 9.247  9.247   0.253 0.246
## Hu                                 0.027
##
## Hypotheses:
##   H1: age>0&peabody>0&prenumb>0
##   H2: age=peabody<prenumb
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

The output consists of several parts:

(a) On the top, it is stated from which type of R object the parameter estimates come from;
(b) The most important part of the output are the results from bain presented in the middle. Here, for each hypothesis, the fit, complexity, the BF of the hypothesis against the unconstrained, the BF of the hypothesis against it's compliment, and the posterior model probabilities are printed in each row. For example, for $H_1 = \beta_1 > 0;\ \beta_2 > 0;\ \beta_3 > 0$, the fit and complexity are equal to 0.871 and 0.032, respectively, resulting in a $BF_{1u} = 27.3$. The value for BF_{1u} suggests that H_1 is 27 times as likely compared to H_u. The value for PMPa suggests that the posterior probability of H_1 *given* the data is around 0.75, assuming equal prior probabilities. The value for PMPb suggests that the posterior probability of $H_1 + H_u$ *given* the data is around 0.73, assuming equal prior probabilities. If we would like to obtain the BF for H_1 against H_2 (BF_{12}), then we just need to take the ratio of the two BF's against the unconstrained hypothesis. In this case,

$$BF_{12} = \frac{27.3}{9.2} \approx 3$$

which means that the support in the data is 3 times larger for H_1 compared to H_2;

(c) The tested hypotheses are printed in the output, just below the results;
(d) The output ends with a note explaining the difference between the two types BF's printed in the main part of the results (b).

(2) Descriptive matrix:

```
summary(results)
```

```
##   Parameter   n   Estimate          lb        ub
## 1       age 240 0.05797594 -0.04043979 0.1563917
## 2   peabody 240 0.14781779  0.03457188 0.2610637
## 3   prenumb 240 0.56740695  0.46067277 0.6741411
```

(3) Obtaining the separate values contained within the `results` object. Here we only present how to obtain the value for the fit, of course, as mentioned above different values contained within the `results` object can be obtained by changing what comes after the `$` in the code below:

```
results$fit
```

```
##      Fit_eq    Com_eq    Fit_in     Com_in       Fit        Com          BF
## H1 1.000000 1.0000000 0.8705764 0.03189549 0.8705764 0.03189549 204.167441
## H2 2.636926 0.5703344 1.0000000 0.50000000 2.6369262 0.28516720   9.246948
## Hu       NA        NA        NA         NA        NA         NA          NA
##         PMPa       PMPb      BF.u       BF.c
## H1 0.7469474 0.72705089 27.294657 204.167441
## H2 0.2530526 0.24631200  9.246948   9.246948
## Hu        NA 0.02663711        NA         NA
```

Chapter 5

Hands-on examples using `bain`

Unless indicated otherwise, the examples that follow below use a simulated data set inspired by the `Sesame Street` data set presented in: Stevens (1996). This data set is included in the `bain` package.

The variables contained in `sesamesim` are subsequently:

- `sex` (1 = boy, 2 = girl) of the child
- `site` (1 = disadvantaged inner city, 2 = advantaged suburban , 3 = advantaged rural, 4 = disadvantaged rural, 5 = disadvantaged Spanish speaking) from which the child originates
- `setting` (1 = at home, 2 = at school) in which the child watches sesame street
- `age` (in months) of the child
- `viewenc` (0 = no, 1 = yes), whether or not the child is encouraged to watch Sesame Street
- `peabody` (mental age) score of the child (higher score is higher mental age)
- `prenumb` (score on a numbers test before watching Sesame Street for a year)
- `postnumb` (score on a numbers test after watching Sesame Street for a year)
- `funumb` (follow up numbers test score measured one year after postnumb)
- `Bb` Knowledge of body parts before
- `Bl` Knowledge of letters before
- `Bf` Knowledge of forms before
- `Bn` Knowledge of numbers before
- `Br` Knowledge of relations before
- `Bc` Knowledge of classifications before

- `Ab` Knowledge of body parts after
- `Al` Knowledge of letters after
- `Af` Knowledge of forms after
- `An` Knowledge of numbers after
- `Ar` Knowledge of relations after
- `Ac` Knowledge of classifications after

The examples that follow below are organized in four categories:

1) running `bain` with a `t_test` object
2) running `bain` with a `lm` object
3) running `bain` with a `lavaan` object
4) running bain with a named vector

Load the `bain` package which includes the simulated `sesamesim` data set:

```
library(bain)
```

5.1 Using `bain` with a `t_test` object

If a `t_test` object is used, the main output table is labeled: *"Bayesian informative hypothesis testing for an objectof class t_test"* (which denotes that a `t-test` was executed).

Options:

a) Bayesian Student's t-test (equal within group variances)
b) Bayesian Welch's t-test (unequal within group variances
c) Bayesian paired samples t-test
d) Bayesian one group t-test
e) Bayesian Equivalence test

Main steps:

1) Execute the analysis with `t_test()`. Note that, `t_test` will apply listwise deletion if there are cases with missing values in the variables used.

2) Call 'bain:

- `set.seed(seed)`. Set `seed` equal to an integer number to create a repeatable random number sequence. `bain` uses sampling to compute Bayes factors and posterior model probabilities. It is therefore recommended to run analyses with two different seeds to ensure stability of the results.

- When `t_test()` is used, hypotheses have to be specified using the names `x`, `y`, or `difference`.

3) `results <- bain(x,hypotheses,fraction = 1)` `fraction = 1`represents the fraction of information in the data used to construct the prior distribution (see, for example, Gu, Mulder, & Hoijtink (2018)). The default value 1 denotes the minimal fraction, 2 denotes twice the minimal fraction, etc.

4) `print(results)` Print the results of an analysis with `bain`.

5) `summary(results, ci=0.95)` Present estimates and credibility intervals for the parameters used to specify the `hypotheses`. `ci` can be used to specify the confidence level of the credibility intervals.

5.1.1 Student's t-test (equal within group variances):

1) Execute the analysis with `t_test()`:

```
# collect the data for the boys in the vector x and for the girls in the
# vector y
x<-sesamesim$postnumb[which(sesamesim$sex==1)]
y<-sesamesim$postnumb[which(sesamesim$sex==2)]
# execute student's t-test (with assumed equal variances)
ttest <- t_test(x,y,paired = FALSE, var.equal = TRUE)
```

2) Call `bain` & 3) Store the output in a `results` object:

```
# set a seed value
set.seed(100)
# test hypotheses with bain. The names of the means are x and y.
results <- bain(ttest, "x = y; x > y; x < y", fraction = 1)
```

3) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class t_test:
##
##    Fit   Com   BF.u   BF.c   PMPa  PMPb
## H1 0.183 0.016 11.583 11.583 0.853 0.794
## H2 0.777 0.500 1.554  3.481  0.114 0.107
## H3 0.223 0.500 0.446  0.287  0.033 0.031
```

```
## Hu                                    0.069
##
## Hypotheses:
##   H1: x=y
##   H2: x>y
##   H3: x<y
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

4) Summary: estimates and credibility intervals:

```
summary(results)
```

```
##   Parameter   n Estimate       lb       ub
## 1         x 115 30.09565 27.79295 32.39836
## 2         y 125 28.85600 26.64732 31.06468
```

5.1.2 Welch's t-test (unequal within group variances):

1) Execute the analysis with `t_test()`:

```
wtest <- t_test(x,y,paired = FALSE, var.equal = FALSE)
```

2) Call `bain` & 3) Store the output in a `results` object:

```
set.seed(100)
results <- bain(wtest, "x = y; x > y; x < y")
```

3) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class t_test:
##
##    Fit   Com   BF.u   BF.c   PMPa  PMPb
## H1 0.183 0.016 11.586 11.586 0.853 0.794
## H2 0.776 0.500 1.552  3.467  0.114 0.106
## H3 0.224 0.500 0.448  0.288  0.033 0.031
## Hu                                 0.069
##
## Hypotheses:
```

```
##   H1: x=y
##   H2: x>y
##   H3: x<y
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothe
```

4) Summary: estimates and credibility intervals:

```
summary(results)
```

```
##   Parameter   n Estimate       lb       ub
## 1         x 115 30.09565 27.70909 32.48221
## 2         y 125 28.85600 26.72394 30.98806
```

5.1.3 Paired samples t-test:

1) Execute the analysis with `t_test()`:

```
pttest <- t_test(sesamesim$prenumb,sesamesim$postnumb,paired = TRUE)
```

2) Call `bain` & 3) Store the output in a `results` object:

```
# set a seed value
set.seed(100)
# test hypotheses with bain. The names of the means are x and y.
results <- bain(pttest, "difference=0; difference>0; difference<0")
```

3) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class t_test:
##
##    Fit   Com   BF.u  BF.c               PMPa  PMPb
## H1 0.000 0.042 0.000 0.000              0.000 0.000
## H2 0.000 0.500 0.000 0.000              0.000 0.000
## H3 1.000 0.500 2.000 22919082073133.293 1.000 0.667
## Hu                                            0.333
##
## Hypotheses:
##   H1: difference=0
```

```
##   H2: difference>0
##   H3: difference<0
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

4) Summary: estimates and credibility intervals:

```
summary(results)
```

```
##    Parameter   n  Estimate        lb        ub
## 1 difference 240 -8.691667 -9.895994 -7.487339
```

5.1.4 One group t-test

1) Execute the analysis with `t_test()`:

```
# compare post measurements with the reference value 30
ttest_one_gr <- t_test(sesamesim$postnumb)
```

2) Call `bain` & 3) Store the output in a `results` object:

```
# set a seed value
set.seed(100)
# test hypotheses with bain versus the reference value 30. Use x to refer to the mean
results <- bain(ttest_one_gr, "x=30; x>30; x<30", fraction = 2)
```

3) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class t_test:
##
##    Fit   Com   BF.u  BF.c  PMPa  PMPb
## H1 0.390 0.045 8.712 8.712 0.813 0.744
## H2 0.249 0.500 0.498 0.332 0.047 0.043
## H3 0.751 0.500 1.502 3.012 0.140 0.128
## Hu                                0.085
##
## Hypotheses:
##   H1: x=30
##   H2: x>30
##   H3: x<30
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

4) Summary: estimates and credibility intervals:

```
summary(results)
```

```
##   Parameter   n Estimate       lb       ub
## 1         x 240    29.45 27.85743 31.04257
```

5.1.5 Equivalence test

1) Execute the analysis with `t_test()`:

```
# collect the data for the boys in the vector x and for the girs in the
# vector y
x<-sesamesim$postnumb[which(sesamesim$sex==1)]
y<-sesamesim$postnumb[which(sesamesim$sex==2)]
# execute student's t-test
ttest_eq <- t_test(x,y,paired = FALSE, var.equal = TRUE)
```

Compute the pooled within standard deviation using the variance of x:

```
# (ttest$v[1]) and y (ttest$v[2])
pwsd <- sqrt(((length(x) -1) * ttest$v[1] + (length(y)-1) * ttest$v[2])/
((length(x) -1) + (length(y) -1)))
# print pwsd in order to be able to include it in the hypothesis. Its value
print(pwsd)
```

```
## [1] 12.59908
```

2) Call `bain` &

3) Store the output in a `results` object. Test hypotheses (the means of boy and girl differ less than .2 * pwsd = 2.52 VERSUS the means differ more than .2 * pwsd = 2.52). Note that, with `bain`, .2 is a value for Cohen's d reflecting a *"small"* effect, that is, the means differ less or more than .2 pwsd. Again, use x and y to refer to the means:

```
# set a seed value
set.seed(100)

results <- bain(ttest_eq, "x - y > -2.52 & x - y < 2.52")
```

3) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class t_test:
##
##    Fit   Com   BF.u  BF.c   PMPa  PMPb
## H1 0.770 0.111 6.967 26.927 1.000 0.874
## Hu                                0.126
##
## Hypotheses:
##   H1: x-y>-2.52&x-y<2.52
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrain
```

4) Summary: estimates and credibility intervals:

```
summary(results)
```

```
##   Parameter   n Estimate       lb       ub
## 1         x 115 30.09565 27.79295 32.39836
## 2         y 125 28.85600 26.64732 31.06468
```

5.2 Using `bain` with a `lm` object

Options:

a) Bayesian ANOVA. The example concerns a one-way ANOVA. Two-way or higher order ANOVAs can only be handled by recoding all factors into one factor. If, for example, there is a factor `sex` with levels `man` and `woman`, and a factor `age` with levels `young` and `old`, these have to be recoded in a new factor `sexage` with levels `manyoung`, `manold`, `womanyoung`, `womanold`. If a `lm` "ANOVA" object is used, the main output table is labeled: "Bayesian informative hypothesis testing for an object of class lm (ANOVA)"

b) Bayesian ANCOVA. The example concerns a one-way ANCOVA. Two-way or higher order ANCOVAs can only be handled by recoding all factors into one factor. **Note that calls to `lm` using functions of the predictors (for example, adding squared predictors to the model using commands like `y ~ x + I(x^2)`) can not be processed by `bain`.** However, one can compute a new variable (for example, a squared predictor) and add this variable to the model specification of `lm`. If a `lm` "ANCOVA" object is used, the main output table is labeled: "Bayesian informative hypothesis testing for an object of class lm (ANCOVA)." **Note that, in the ANCOVA the covariates are centered!**

c) Bayesian multiple regression. **Note that calls to `lm` using functions of the predictors (for example, adding squared predictors to the model using commands like `y ~ x + I(x^2)`) can not be processed by `bain`.** However, one can compute a new variable (for example, a squared predictor) and add this variable to the model specification of `lm`. Note furthermore, that only if `standardize = FALSE` interactions between predictors should be processed, because a standardized interaction is not the same as the interaction between two standardized variables. If a `lm` "regression with only continuous predictors" object is used, the main output table is labeled: "Bayesian informative hypothesis testing for an object of class lm (continuous predictors)." If a `lm` object contains **two or more**factors and, optionally, continuous predictors, the main output table islabeled: "Bayesian informative hypothesis testing for an object of class lm (mixed predictors)." In case of mixed predictors, the one group approach to computing Bayes factors is used (see, Hoijtink, Gu, & Mulder (2019)). This may render inferior results if group sizes are unequal.

d) Bayesian ANOVA. Sensitivity analysis. See Hoijtink, Mulder, Lissa, & Gu (2019) for elaborations.

Main steps:

1) Execute the analysis with `lm()`. Note that, `lm` will apply list-wise deletion if there are cases with missing values in the variables used.

2) Display the estimates and their names using the command `coef(x)`. (Unique abbreviations of) these names will be used to specify `hypotheses`.

3) `set.seed(seed)`. Set `seed` equal to an integer number to create a repeatable random number sequence. `bain` uses sampling to compute Bayes factors and posterior model probabilities. It is therefore recommended to run analyses with two different seeds to ensure stability of the results. And call `bain` by using `results <- bain(x,hypotheses,fraction = 1)` or `results <-bain(x,hypotheses,fraction = 1,standardize = FALSE)`. The first call to `bain` is used in case of `lm` implementations of *ANOVA* and *ANCOVA*. The second call to `bain` is used in case of `lm` implementations of *multiple regression*. With `standardize = TRUE` hypotheses with respect to standardized regression coefficients are evaluated. With `standardize = FALSE` hypotheses with respect to unstandardized regression coefficients are evaluated. `fraction = 1` represents the fraction of information in the data used to construct the prior distribution. The default value 1 denotes the minimal fraction, 2 denotes twice the minimal fraction, etc.

4) `print(results)` Print the results of an analysis with `bain`.

5) `summary(results, ci=0.95)` Present estimates and credibility intervals for the parameters used to specify the `hypotheses`. `ci` can be used to specify the confidence level of the credibility intervals.

5.2.1 ANOVA

1) Execute the analysis with `lm()`:

```
# make a factor of variable site
sesamesim$site <- as.factor(sesamesim$site)
# execute an analysis of variance using lm() which, due to the -1, returns
# estimates of the means per group
anov <- lm(postnumb~site-1,sesamesim)
```

2) Display the estimated coefficients using `coef()`:

```
coef(anov)
```

```
##    site1    site2    site3    site4    site5
## 29.66667 38.98182 23.18750 25.32558 31.72222
```

3) Set seed and call `bain`:

```
set.seed(100)
# test hypotheses with bain
results <- bain(anov, "site1=site2=site3=site4=site5; site2>site5>site1>
site3>site4")
```

4) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class lm (ANOVA):
##
##    Fit   Com   BF.u   BF.c   PMPa  PMPb
## H1 0.000 0.000 0.000  0.000  0.000 0.000
## H2 0.121 0.008 14.559 16.428 1.000 0.936
## Hu                                0.064
##
## Hypotheses:
##   H1: site1=site2=site3=site4=site5
##   H2: site2>site5>site1>site3>site4
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

5) Summary: estimates and credibility intervals:

```
summary(results)
```

```
##   Parameter  n Estimate       lb       ub
## 1     site1 60 29.66667 26.82991 32.50343
## 2     site2 55 38.98182 36.01892 41.94472
## 3     site3 64 23.18750 20.44082 25.93418
## 4     site4 43 25.32558 21.97466 28.67650
## 5     site5 18 31.72222 26.54303 36.90141
```

5.2.2 ANCOVA

1) Execute the analysis with `lm()`:

```
# Center the covariate. If centered the coef() command below displays the
# adjusted means. If not centered the intercepts are displayed.
sesamesim$prenumb <- sesamesim$prenumb - mean(sesamesim$prenumb)
# execute an analysis of covariance using lm() which, due to the -1, returns
# estimates of the adjusted means per group
ancov <- lm(postnumb~site+prenumb-1,sesamesim)
```

2) Display the estimated coefficients using `coef()`:

```
coef(ancov)
```

```
##      site1      site2      site3      site4      site5    prenumb
## 27.4293820 34.9818244 27.6904082 26.9840163 31.4298837  0.7159311
```

3) Set seed and call `bain`:

```
set.seed(100)
# test hypotheses with bain
results <- bain(ancov, "site1=site2=site3=site4=site5;
                        site2>site5>site1>site3>site4")
```

4) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class lm (ANCOVA):
##
##    Fit   Com   BF.u   BF.c   PMPa  PMPb
## H1 0.000 0.000 0.002  0.002  0.000 0.000
## H2 0.183 0.009 19.899 24.126 1.000 0.952
## Hu                                0.048
##
## Hypotheses:
##   H1: site1=site2=site3=site4=site5
##   H2: site2>site5>site1>site3>site4
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

5) Summary: estimates and credibility intervals:

```
summary(results)
```

```
##   Parameter   n   Estimate         lb        ub
## 1     site1  60 27.4293820 25.1603844 29.698380
## 2     site2  55 34.9818244 32.5529993 37.410650
## 3     site3  64 27.6904082 25.4002689 29.980548
## 4     site4  43 26.9840163 24.3250440 29.642989
## 5     site5  18 31.4298837 27.3413936 35.518374
## 6   prenumb 240  0.7159311  0.5986552  0.833207
```

5.2.3 Multiple regression

1) Execute the analysis with `lm()`:

```
# execute a multiple regression using lm()
regr <- lm(postnumb ~ age + peabody + prenumb,sesamesim)
```

2) Display the estimated coefficients using `coef()`:

```
coef(regr)
```

```
## (Intercept)         age     peabody     prenumb
##  18.1160056   0.1159941   0.1157549   0.6727219
```

3) Set seed and call `bain`:

- **Testing unstandardized coefficients.**

```
set.seed(100)
#test hypotheses with bain. Note that standardize = FALSE denotes that the
# hypotheses are in terms of unstandardized regression coefficients
results<-bain(regr, "age = 0 & peab=0 & pre=0 ; age > 0 & peab > 0 & pre > 0",
              standardize = FALSE)
```

- **Testing standardized coefficients:** Since it is only meaningful to compare regression coefficients if they are measured on the same scale, `bain` can also evaluate standardized regression coefficients (based on the `seBeta` function by Jones & Waller (2015)):

```
results<-bain(regr, "age = peab = pre ; pre > age > peab",standardize = TRUE)
```

4) Print the (standardized) results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class lm (continuous predictors):
##
##    Fit   Com   BF.u  BF.c  PMPa  PMPb
## H1 0.000 0.202 0.000 0.000 0.000 0.000
## H2 0.125 0.216 0.579 0.519 1.000 0.367
## Hu                                0.633
##
## Hypotheses:
##   H1: age=peabody=prenumb
##   H2: prenumb>age>peabody
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothe
```

5) Summary: estimates and credibility intervals:

```
summary(results)
```

```
##   Parameter   n   Estimate          lb        ub
## 1       age 240 0.05797594 -0.04043979 0.1563917
## 2   peabody 240 0.14781779  0.03457188 0.2610637
## 3   prenumb 240 0.56740695  0.46067277 0.6741411
```

5.3 Using `bain` with a `lavaan` object

```
suppressPackageStartupMessages(library(lavaan))
```

If a `lavaan` object is used, the main output table is labeled: *"Bayesian informative hypothesis testing for an object of class lavaan"*. See the tutorial by Van Lissa et al. (2021) for further elaborations. Also, visit this link for more `lavaan` mini tutorials, examples and elaborations.

Options:

a) Bayesian confirmatory factor analysis

b) Bayesian latent regression

c) Bayesian multiple group latent regression. Note that, multiple group models with between group restrictions cannot be processed.

Main steps:

1) Specify a `lavaan` model using the `model <- ...` command. In case of multiple group models, only models **without** between group restrictions can be processed by `bain` with a `lavaan` object as input. Secondly, execute an analysis with the sem, cfa, or growth functions (e.g: `x <- sem()` or `x <- cfa()` or `x <- growth()`) implemented in `lavaan`. Note that, by default, `lavaan` will apply list-wise deletion if there are cases with missing values in the variables used. An imputation based method for dealing with missing values tailored to Bayesian hypothesis evaluation is illustrated in section 5.4. (based on Hoijtink, Gu, Mulder, & Rosseel (2019)). If an analysis with `lavaan` is executed using `missing = "fiml"` the sample size is not corrected for the presence of missing values. This will affect (bias) the evaluation of hypotheses specified using (about) equality constraints. Specify a `lavaan` model using the `model <- ...` command. In case of multiple group models, only models **without** between group restrictions can be processed by `bain` with a `lavaan` object as input.

2) Display the estimates and their names using the command `coef(x)`. Only parameters who's names contain `~` (regression coefficients), `~1` (intercepts), or `=~` (factor loadings) can be used in the specification of hypotheses. (Unique abbreviations of) the names can be used to specify `hypotheses`. For multiple group analyses the names have to end with a group label `.grp`. Group labels can be assigned using commands like `sesamesim$sex <- factor(sesamesim$sex, labels = c("boy", "girl"))`. If in a `lavaan` model parameters are labeled, e.g., as in `model <- 'age ~ c(a1, a2)*peabody + c(b1, b2)*1` then the labels have to be used in the specification of hypotheses.

3) `set.seed(seed)`. Set `seed` equal to an integer number to create a repeatable random number sequence. `bain` uses sampling to compute Bayes factors and posterior model probabilities. It is therefore recommended to run analyses with two different seeds to ensure stability of the results. And `results <- bain(x,hypotheses,fraction = 1,standardize = FALSE)`. With `standardize = TRUE` hypotheses with respect to standardized coefficients are evaluated. With `standardize = FALSE` hypotheses with respect to unstandardized coefficients are evaluated. `fraction = 1` represents the fraction of information in the data used to construct the prior distribution. The default value 1 denotes the minimal fraction, 2 denotes twice the minimal fraction, etc.).

4) `print(results)` Print the results of an analysis with `bain`.

5) `summary(results, ci=0.95)` Present estimates and credibility intervals for the parameters used to specify the `hypotheses`. `ci` can be used to specify the confidence level of the credibility intervals.

5.3.1 Confirmatory Factor Analysis (CFA):

1) Speficy the model and execute the analysis:

```
#  Specify and fit the confirmatory factor model
model1 <- '
    A =~ Ab + Al + Af + An + Ar + Ac
    B =~ Bb + Bl + Bf + Bn + Br + Bc
'
# Use the lavaan sem function to execute the confirmatory factor analysis
fit1 <- sem(model1, data = sesamesim, std.lv = TRUE)
```

2) Display the estimates and specify an object containing the hypotheses:

```
coef(fit1)
```

```
##  A=~Ab  A=~Al  A=~Af  A=~An  A=~Ar  A=~Ac  B=~Bb  B=~Bl  B=~Bf  B=~Bn  B=~Br
##  3.885  9.583  3.367 10.906  1.790  4.424  4.433  5.054  3.200  8.588  1.997
##  B=~Bc Ab~~Ab Al~~Al Af~~Af An~~An Ar~~Ar Ac~~Ac Bb~~Bb Bl~~Bl Bf~~Bf Bn~~Bn
##  3.685 14.808 47.655  4.843 25.920  3.370  6.101 13.857 35.349  5.360 19.799
## Br~~Br Bc~~Bc   A~~B
##  3.693  6.219  0.788
```

```
hypotheses1 <-
" A=~Ab > .6 & A=~Al > .6 & A=~Af > .6 & A=~An > .6 & A=~Ar > .6 & A=~Ac >.6 &
B=~Bb > .6 & B=~Bl > .6 & B=~Bf > .6 & B=~Bn > .6 & B=~Br > .6 & B=~Bc >.6"
```

3) Set seed and call `bain`:

```
set.seed(100)
y <- bain(fit1,hypotheses1,fraction=1,standardize=TRUE)
```

4) Print the results:

```
print(y)
```

```
## Bayesian informative hypothesis testing for an object of class lavaan:
##
##    Fit   Com   BF.u   BF.c    PMPa  PMPb
## H1 0.880 0.009 95.519 788.316 1.000 0.990
## Hu                                 0.010
##
## Hypotheses:
##   H1: A=~Ab>.6&A=~Al>.6&A=~Af>.6&A=~An>.6&A=~Ar>.6&A=~Ac>.6&B=~Bb>.6&B=~Bl>.6&B=~Bf
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrain
```

5) Summary: estimates and credibility intervals:

```
summary(y, ci = 0.95)
```

```
##    Parameter   n  Estimate        lb        ub
## 1      A=~Ab 240 0.7104508 0.6433340 0.7775677
## 2      A=~Al 240 0.8114029 0.7632944 0.8595115
## 3      A=~Af 240 0.8370936 0.7941226 0.8800647
## 4      A=~An 240 0.9061333 0.8769727 0.9352938
## 5      A=~Ar 240 0.6980414 0.6287429 0.7673399
## 6      A=~Ac 240 0.8731170 0.8374249 0.9088091
## 7      B=~Bb 240 0.7658397 0.7076188 0.8240605
## 8      B=~Bl 240 0.6476698 0.5687998 0.7265397
## 9      B=~Bf 240 0.8101849 0.7604160 0.8599537
## 10     B=~Bn 240 0.8878987 0.8531156 0.9226818
## 11     B=~Br 240 0.7205677 0.6540778 0.7870576
## 12     B=~Bc 240 0.8282004 0.7819303 0.8744705
```

5.3.2 Latent regression

1) Speficy the model and execute the analysis:

```
#  Specify and fit the confirmatory factor model
model2 <- '
    A  =~ Ab + Al + Af + An + Ar + Ac
    B =~ Bb + Bl + Bf + Bn + Br + Bc

    A ~ B + age + peabody
'
fit2 <- sem(model2, data = sesamesim, std.lv = TRUE)
```

2) Display the estimates and specify an object containing the hypotheses:

```
coef(fit2)
```

```
##     A=~Ab     A=~Al     A=~Af     A=~An     A=~Ar     A=~Ac     B=~Bb     B=~Bl
##     2.392     5.899     2.074     6.713     1.102     2.723     4.434     5.055
##     B=~Bf     B=~Bn     B=~Br     B=~Bc       A~B     A~age A~peabody    Ab~~Ab
##     3.200     8.587     1.997     3.686     1.284     0.000    -0.002    14.803
##    Al~~Al    Af~~Af    An~~An    Ar~~Ar    Ac~~Ac    Bb~~Bb    Bl~~Bl    Bf~~Bf
##    47.683     4.837    25.956     3.370     6.101    13.853    35.341     5.358
##    Bn~~Bn    Br~~Br    Bc~~Bc
##    19.815     3.695     6.216
```

```
hypotheses2 <- "A~B > A~peabody = A~age = 0;
                A~B > A~peabody > A~age = 0;
A~B > A~peabody > A~age > 0"
```

3) Set seed and call `bain`:

```
set.seed(100)
y <- bain(fit2, hypotheses2, fraction = 1, standardize = TRUE)
```

4) Print the results:

```
print(y)
```

```
## Bayesian informative hypothesis testing for an object of class lavaan:
##
##     Fit    Com   BF.u    BF.c    PMPa  PMPb
## H1 69.899 0.463 150.871 150.871 0.792 0.788
## H2 3.045  0.090 33.858  33.858  0.178 0.177
## H3 0.069  0.012 5.850   6.210   0.031 0.031
## Hu                                 0.005
```

```
##
## Hypotheses:
##   H1: A~B>A~peabody=A~age=0
##   H2: A~B>A~peabody>A~age=0
##   H3: A~B>A~peabody>A~age>0
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrain
```

5) Summary: estimates and credibility intervals:

```
summary(y, ci = 0.95)
```

```
##    Parameter   n     Estimate          lb         ub
## 1      A=~Ab 240  0.711205262  0.64424171 0.77816881
## 2      A=~Al 240  0.811775528  0.76376372 0.85978734
## 3      A=~Af 240  0.837770094  0.79496009 0.88058010
## 4      A=~An 240  0.906287871  0.87719307 0.93538268
## 5      A=~Ar 240  0.698701975  0.62953799 0.76786596
## 6      A=~Ac 240  0.873484275  0.83789531 0.90907324
## 7      B=~Bb 240  0.765919628  0.70771871 0.82412054
## 8      B=~Bl 240  0.647765959  0.56891562 0.72661629
## 9      B=~Bf 240  0.810246503  0.76049539 0.85999762
## 10     B=~Bn 240  0.887803097  0.85301152 0.92259468
## 11     B=~Br 240  0.720382073  0.65386367 0.78690047
## 12     B=~Bc 240  0.828276665  0.78202772 0.87452561
## 13       A~B 240  0.788751775  0.72999714 0.84750641
## 14     A~age 240 -0.000478606 -0.09265311 0.09169589
## 15 A~peabody 240 -0.015527151 -0.10769273 0.07663843
```

5.3.3 Multiple group regression

1) Speficy the model and execute the analysis:

```
#  Specify and fit the confirmatory factor model
model3 <- '
    postnumb ~ prenumb + peabody
'
# Assign labels to the groups to be used when formulating
# hypotheses
sesamesim$sex <- factor(sesamesim$sex, labels = c("boy", "girl"))
# Fit the multiple group regression model
fit3 <- sem(model3, data = sesamesim, std.lv = TRUE, group = "sex")
```

2) Display the estimates and specify an object containing the hypotheses:

```
coef(fit3)
```

```
##      postnumb~prenumb      postnumb~peabody    postnumb~~postnumb
##                 0.660                 0.173                85.644
##            postnumb~1   postnumb~prenumb.g2   postnumb~peabody.g2
##                21.645                 0.723                 0.052
## postnumb~~postnumb.g2         postnumb~1.g2
##                80.068                26.715
```

```
hypotheses3 <-
"postnumb~prenumb.boy = postnumb~prenumb.girl & postnumb~peabody.boy = postnumb~peabody.girl;
 postnumb~prenumb.boy < postnumb~prenumb.girl & postnumb~peabody.boy < postnumb~peabody.girl
"
```

3) Set seed and call `bain`:

```
set.seed(100)
y <- bain(fit3, hypotheses3, fraction = 1, standardize = TRUE)
```

4) Print the results:

```
print(y)
```

```
## Bayesian informative hypothesis testing for an object of class lavaan:
##
##    Fit   Com   BF.u   BF.c   PMPa  PMPb
## H1 7.786 0.189 41.199 41.199 0.996 0.972
## H2 0.019 0.106 0.181  0.165  0.004 0.004
## Hu                                0.024
##
## Hypotheses:
##   H1: postnumb~prenumb.boy=postnumb~prenumb.girl&postnumb~peabody.boy=postnumb~peabody.girl
##   H2: postnumb~prenumb.boy<postnumb~prenumb.girl&postnumb~peabody.boy<postnumb~peabody.girl
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothe
```

5) Summary: estimates and credibility intervals:

```
summary(y, ci = 0.95)
```

```
##               Parameter   n   Estimate          lb        ub
```

```
## 1  postnumb~prenumb.boy 115 0.53181375  0.38134118 0.6822863
## 2  postnumb~peabody.boy 115 0.23097972  0.06511841 0.3968410
## 3        postnumb~1.boy 115 1.66487431  1.11918686 2.2105618
## 4 postnumb~prenumb.girl 125 0.63788102  0.51532021 0.7604418
## 5 postnumb~peabody.girl 125 0.06298202 -0.09071110 0.2166751
## 6       postnumb~1.girl 125 2.20546228  1.63305518 2.7778694
```

5.4 Using `bain` with a named vector

If a `named vector` object is used, the main output table is labeled: "Bayesian informative hypothesis testing for an object of class numeric" (which denotes that `named vector` input was used).

Options:

a) Bayesian ANOVA
b) Bayesian Welch's ANOVA (unequal within group variances)
c) Bayesian robust (against non-normality) ANOVA (unequal within group variances)
d) Bayesian ANCOVA
e) Bayesian repeated measures analysis (one within factor)
f) Bayesian repeated measures analysis (within between design)
g) Bayesian one group logistic regression (counterpart of multiple regression)
h) Bayesian multiple group logistic regression (counterpart of ANCOVA)
i) Bayesian multiple regression with missing data
j) Bayesian confirmatory factor analysis . .(add multilevel models here maybe?) .

Main steps:

1) Execute a statistical analysis.

In case of a single group analysis, the following information has to be extracted from the statistical analysis and supplied to `bain`:

a) A vector containing estimates of the parameters used to specify `hypotheses`;
b) A list containing the covariance matrix of these parameters; and,
c) The sample size used for estimation of the parameters. Note that, due to missing values this sample size may be smaller than the total sample size.

In case of a multiple group analysis, the following information has to be extracted from the statistical analysis and supplied to `bain`:

a) A vector containing estimates of the group specific parameters possibly augmented with the estimates of parameters that are shared by the groups. The structure of this vector is [parameters of group 1, parameters of group 2, ..., the parameters that are shared by the groups];
b) A list containing, per group, the covariance matrix of the parameters corresponding to the group at hand and, possibly, the augmented parameters. In the rows and columns of each covariance matrix the parameters of the group at hand come first, possibly followed by the augmented parameters.
c) Per group the sample size used for estimation of the parameters. Note that, due to missing values this sample size may be smaller than the total sample size per group.

2) Assign names to the estimates using `names(estimates)<-c()`. Note that, `names` is a character vector containing new names for the estimates in `estimates`. Each name has to start with a letter, and may consist of "letters," "numbers," "`.`" "`_`," "`:`" "`~`," "`=~`," and "`~1`." These names are used to specify `hypotheses` (see below). An example is `names <- c("a", "b", "c")`.
3) `set.seed(seed)`. Set `seed` equal to an integer number to create a repeatable random number sequence. `bain` uses sampling to compute Bayes factors and posterior model probabilities. It is therefore recommended to run analyses with two different seeds to ensure stability of the results.
4) `results <- bain(estimates, hypotheses, n=., Sigma=., group_parameters = 2, joint_parameters = 0, fraction = 1)` executes `bain` with the following arguments:

a) `estimates` A named vector with parameter estimates.
b) `hypotheses` A character string containing the informative hypotheses to evaluate (the specification is elaborated below).
c) `n` A vector containing the sample size of each group in the analysis. See, Hoijtink, Gu, and Mulder (2019), for an elaboration of the difference between one and multiple group analyses. A multiple group analysis is required when group specific parameters are used to formulate `hypotheses`. Examples are the Student's and Welch's t-test, ANOVA, and ANCOVA. See the Examples section for elaborations of the specification of multiple group analyses when a named vector is input for `bain`.
d) `Sigma` A list of covariance matrices. In case of one group analyses the list contains one covariance matrix. In case of multiple group analyses the list contains one covariance matrix for each group. See the Examples section and Hoijtink, Gu, and Mulder (2019) for further instructions.
e) `group_parameters` The number of group specific parameters. In, for example, an ANOVA with three groups, `estimates` will contain three sample means, `group_parameters = 1` because each group is characterized by one mean, and `joint_parameters = 0` because there are no parameters that apply to each of the groups. In, for example, an ANCOVA with

three groups and two covariates, `estimates` will contain five parameters (first the three adjusted means, followed by the regression coefficients of the two covariates), `group_parameters = 1` because each group is characterized by one adjusted mean, and `joint_parameters = 2` because there are two regression coefficients that apply to each group. In, for example, a repeated measures design with four repeated measures and two groups (a between factor with two levels and a within factor with four levels) `estimates` will contain eight means (first the four for group 1, followed by the four for group 2), `group_parameters = 4` because each group is characterized by four means and `joint_parameters = 0` because there are no parameters that apply to each of the groups.

f) `joint_parameters` In case of one group `joint_parameters = 0`. In case of two or more groups, the number of parameters in `estimates` shared by the groups. In, for example, an ANCOVA, the number of `joint_parameters` equals the number of covariates.

g) `fraction = 1` A number representing the fraction of information in the data used to construct the prior distribution. The default value 1 denotes the minimal fraction, 2 denotes twice the minimal fraction, etc.

5) `print(results)` Print the results of an analysis with `bain`.

6) `summary(results, ci=0.95)` Present estimates and credibility intervals for the parameters used to specify the `hypotheses`. `ci` can be used to specify the confidence level of the credibility intervals.

5.4.1 ANOVA

1) Execute the statistical analysis:

```
# make a factor of variable site
sesamesim$site <- as.factor(sesamesim$site)
# execute an analysis of variance using lm() which, due to the -1, returns
# estimates of the means per group
anov <- lm(postnumb~site-1,sesamesim)
```

a) A vector containing estimates of the parameters:

```
# collect the estimates means in a vector
estimate <- coef(anov)
```

c) Per group the sample size used for estimation of the parameters (Note: the order in this example is deliberately changed):

```
ngroup <- table(sesamesim$site)
```

b) A list containing, per group, the covariance matrix of the parameters:

```
var <- summary(anov)$sigma**2
cov1 <- matrix(var/ngroup[1], nrow=1, ncol=1)
cov2 <- matrix(var/ngroup[2], nrow=1, ncol=1)
cov3 <- matrix(var/ngroup[3], nrow=1, ncol=1)
cov4 <- matrix(var/ngroup[4], nrow=1, ncol=1)
cov5 <- matrix(var/ngroup[5], nrow=1, ncol=1)
covlist <- list(cov1, cov2, cov3, cov4,cov5)
```

2) Assign names to the estimates:

```
names(estimate) <- c("site1", "site2", "site3","site4","site5")
```

3) Set seed and 4) call `bain`:

```
set.seed(100)
results <- bain(estimate,
"site1=site2=site3=site4=site5; site2>site5>site1>site3>site4",
n=ngroup,Sigma=covlist,group_parameters=1,joint_parameters = 0)
```

5) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class numeric:
##
##    Fit   Com   BF.u   BF.c   PMPa  PMPb
## H1 0.000 0.000 0.000  0.000  0.000 0.000
## H2 0.121 0.008 14.559 16.428 1.000 0.936
## Hu                                0.064
##
## Hypotheses:
##   H1: site1=site2=site3=site4=site5
##   H2: site2>site5>site1>site3>site4
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothe
```

6) Summary: estimates and credibility intervals:

```
summary(results, ci = 0.95)
```

```
##   Parameter  n Estimate       lb       ub
## 1     site1 60 29.66667 26.82991 32.50343
## 2     site2 55 38.98182 36.01892 41.94472
## 3     site3 64 23.18750 20.44082 25.93418
## 4     site4 43 25.32558 21.97466 28.67650
## 5     site5 18 31.72222 26.54303 36.90141
```

5.4.2 Welch's ANOVA (equal variances not assumed)

1) Execute the statistical analysis:

a) A vector containing estimates of the parameters:

```
sesamesim$site <- as.factor(sesamesim$site)
# collect the estimated means in a vector
estimates <- aggregate(sesamesim$postnumb,list(sesamesim$site),mean)[,2]
```

b) A list containing, per group, the covariance matrix of the parameters:

```
vars <- aggregate(sesamesim$postnumb,list(sesamesim$site),var)
vargroup <- vars[,2]/ngroup
# collect the variances of the means in a covariance listcov1 <- matrix(vargroup[1], n
cov2 <- matrix(vargroup[2], nrow=1, ncol=1)
cov3 <- matrix(vargroup[3], nrow=1, ncol=1)
cov4 <- matrix(vargroup[4], nrow=1, ncol=1)
cov5 <- matrix(vargroup[5], nrow=1, ncol=1)
covlist <- list(cov1, cov2, cov3, cov4,cov5)
```

c) Per group the sample size used for estimation of the parameters:

```
ngroup <- table(sesamesim$site)
```

2) Assign names to the estimates:

```
names(estimates) <- c("site1", "site2", "site3","site4","site5")
```

3) Set seed and 4) call `bain`:

```
set.seed(100)
# test hypotheses using bain. Note that there are multiple groups
# characterized by one mean, therefore group_parameters=1. Note that
# there are no joint parameters, therefore, joint_parameters=0.
results <- bain(estimates,
                "site1=site2=site3=site4=site5;site2>site5>site1>site4>site3",
                n=ngroup,Sigma=covlist,group_parameters=1,joint_parameters = 0)
```

5) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class numeric:
##
##    Fit   Com   BF.u   BF.c    PMPa  PMPb
## H1 0.000 0.000 0.000  0.000   0.000 0.000
## H2 0.656 0.011 59.258 170.335 1.000 0.983
## Hu                                  0.017
##
## Hypotheses:
##   H1: site1=site2=site3=site4=site5
##   H2: site2>site5>site1>site4>site3
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothe
```

6) Summary: estimates and credibility intervals:

```
summary(results, ci = 0.95)
```

```
##   Parameter  n Estimate       lb       ub
## 1     site1 60 29.66667 26.82991 32.50343
## 2     site2 55 38.98182 35.54861 42.41503
## 3     site3 64 23.18750 20.40407 25.97093
## 4     site4 43 25.32558 22.65323 27.99793
## 5     site5 18 31.72222 27.79013 35.65432
```

5.4.3 Robust ANOVA (against non-normality)

Load the `WRS2` package which renders the trimmed sample mean and corresponding standard error:

```
library(WRS2)
```

1) Execute the statistical analysis:

a) A vector containing estimates of the parameters:

```
sesamesim$site <- as.factor(sesamesim$site)
# Compute the 20\% sample trimmed mean for each site
estimates <- c(mean(sesamesim$postnumb[sesamesim$site == 1], tr = 0.2),
               mean(sesamesim$postnumb[sesamesim$site == 2], tr = 0.2),
               mean(sesamesim$postnumb[sesamesim$site == 3], tr = 0.2),
               mean(sesamesim$postnumb[sesamesim$site == 4], tr = 0.2),
               mean(sesamesim$postnumb[sesamesim$site == 5], tr = 0.2))
```

b) A list containing, per group, the covariance matrix of the parameters:

```
# Compute the sample trimmed mean standard error for each site
se <- c(trimse(sesamesim$postnumb[sesamesim$site == 1]),
        trimse(sesamesim$postnumb[sesamesim$site == 2]),
        trimse(sesamesim$postnumb[sesamesim$site == 3]),
        trimse(sesamesim$postnumb[sesamesim$site == 4]),
        trimse(sesamesim$postnumb[sesamesim$site == 5]))
# Square the standard errors to obtain the variances of the sample
# trimmed means
var <- se^2
# Store the variances in a list of matrices
covlist <- list(matrix(var[1]),matrix(var[2]),
matrix(var[3]),matrix(var[4]), matrix(var[5]))
```

c) Per group the sample size used for estimation of the parameters:

```
ngroup <- table(sesamesim$site)
```

2) Assign names to the estimates:

```
names(estimates) <- c("s1", "s2", "s3","s4","s5")
```

3) Set seed and 4) call `bain`:

```
set.seed(100)
# test hypotheses with bain
results <- bain(estimates,"s1=s2=s3=s4=s5;s2>s5>s1>s3>s4",
n=ngroup,Sigma=covlist,group_parameters=1,joint_parameters= 0)
```

5) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class numeric:
##
##    Fit   Com   BF.u   BF.c   PMPa  PMPb
## H1 0.000 0.000 0.000  0.000  0.000 0.000
## H2 0.076 0.007 10.407 11.185 1.000 0.912
## Hu                                 0.088
##
## Hypotheses:
##   H1: s1=s2=s3=s4=s5
##   H2: s2>s5>s1>s3>s4
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothe
```

6) Summary: estimates and credibility intervals:

```
summary(results, ci = 0.95)
```

```
##   Parameter  n Estimate       lb       ub
## 1        s1 60 29.69444 27.11542 32.27347
## 2        s2 55 39.63636 35.93878 43.33394
## 3        s3 64 22.95000 20.04036 25.85964
## 4        s4 43 25.29630 22.98708 27.60551
## 5        s5 18 31.75000 26.06594 37.43406
```

5.4.4 ANCOVA

1) Execute the statistical analysis:

```
# make a factor of variable site
sesamesim$site <- as.factor(sesamesim$site)
# center the covariate. If centered the coef() command below displays the
# adjusted means. If not centered the intercepts are displayed.
```

```
sesamesim$prenumb <- sesamesim$prenumb - mean(sesamesim$prenumb)
# execute an analysis of covariance using lm() which, due to the -1, returns
# estimates of the adjusted means per group
ancov2 <- lm(postnumb~site+prenumb-1,sesamesim)
```

a) A vector containing estimates of the parameters:

```
estimates <- coef(ancov2)
```

b) A list containing, per group, the covariance matrix of the parameters:

```
var <- (summary(ancov2)$sigma)**2
# below, for each group, the covariance matrix of the adjusted mean and
# covariate is computed
# see Hoijtink, Gu, and Mulder (2019) for further explanation and elaboration
cat1 <- subset(cbind(sesamesim$site,sesamesim$prenumb), sesamesim$site == 1)
cat1[,1] <- 1
cat1 <- as.matrix(cat1)
cov1 <- var * solve(t(cat1) %*% cat1)
#
cat2 <- subset(cbind(sesamesim$site,sesamesim$prenumb), sesamesim$site == 2)
cat2[,1] <- 1
cat2 <- as.matrix(cat2)
cov2 <- var * solve(t(cat2) %*% cat2)
#
cat3 <- subset(cbind(sesamesim$site,sesamesim$prenumb), sesamesim$site == 3)
cat3[,1] <- 1
cat3 <- as.matrix(cat3)
cov3 <- var * solve(t(cat3) %*% cat3)
#
cat4 <- subset(cbind(sesamesim$site,sesamesim$prenumb), sesamesim$site == 4)
cat4[,1] <- 1
cat4 <- as.matrix(cat4)
cov4 <- var * solve(t(cat4) %*% cat4)
#
cat5 <- subset(cbind(sesamesim$site,sesamesim$prenumb), sesamesim$site == 5)
cat5[,1] <- 1
cat5 <- as.matrix(cat5)
cov5 <- var * solve(t(cat5) %*% cat5)
#
covariances <- list(cov1, cov2, cov3, cov4,cov5)
```

c) Per group the sample size used for estimation of the parameters:

```
ngroup <- table(sesamesim$site)
```

2) Assign names to the estimates:

```
names(estimates)<- c("v.1", "v.2", "v.3", "v.4","v.5", "pre")
```

3) Set seed and 4) call `bain`:

```
set.seed(100)
results2<-bain(estimates,"v.1=v.2=v.3=v.4=v.5;v.2 > v.5 > v.1 > v.3 >v.4;",
n=ngroup,Sigma=covariances,group_parameters=1,joint_parameters = 1)
```

5) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class numeric:
##
##    Fit   Com   BF.u   BF.c   PMPa  PMPb
## H1 0.000 0.000 0.000  0.000  0.000 0.000
## H2 0.076 0.007 10.407 11.185 1.000 0.912
## Hu                                 0.088
##
## Hypotheses:
##   H1: s1=s2=s3=s4=s5
##   H2: s2>s5>s1>s3>s4
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothe
```

6) Summary: estimates and credibility intervals:

```
summary(results, ci = 0.95)
```

```
##   Parameter  n Estimate       lb       ub
## 1        s1 60 29.69444 27.11542 32.27347
## 2        s2 55 39.63636 35.93878 43.33394
## 3        s3 64 22.95000 20.04036 25.85964
## 4        s4 43 25.29630 22.98708 27.60551
## 5        s5 18 31.75000 26.06594 37.43406
```

5.4.5 One within factor repeated measures analysis

1) Execute the statistical analysis:

```
within <- lm(cbind(prenumb,postnumb,funumb)~1, data=sesamesim)
```

a) A vector containing estimates of the parameters:

```
# collect the estimates means in a vector
estimate <- coef(within)[1:3]
```

b) A list containing, per group, the covariance matrix of the parameters:

```
covmatr <- list(vcov(within))
```

c) Per group the sample size used for estimation of the parameters:

```
ngroup <- nrow(sesamesim)
```

2) Assign names to the estimates:

```
names(estimate) <- c("pre", "post", "fu")
```

3) Set seed and 4) call `bain`:

```
set.seed(100)
results <- bain(estimate,"pre = post = fu; pre < post < fu", n=ngroup,
Sigma=covmatr, group_parameters=3, joint_parameters = 0)
```

5) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class numeric:
##
##    Fit   Com   BF.u  BF.c       PMPa  PMPb
## H1 0.000 0.002 0.000 0.000      0.000 0.000
## H2 1.000 0.243 4.115 237946.392 1.000 0.805
## Hu                                     0.195
##
```

```
## Hypotheses:
##   H1: pre=post=fu
##   H2: pre<post<fu
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothe
```

6) Summary: estimates and credibility intervals:

```
summary(results, ci = 0.95)
```

```
##   Parameter   n     Estimate        lb        ub
## 1       pre 240 4.586534e-16 -1.343254  1.343254
## 2      post 240 2.945000e+01 27.857428 31.042572
## 3        fu 240 3.432083e+01 31.750376 36.891290
```

5.4.6 One between-within factor repeated measures analysis

1) Execute the statistical analysis:

```
# make a factor of the variable sex
sesamesim$sex <- factor(sesamesim$sex)
# estimate the means of prenumb, postnumb, and funumb for boys and girls
# the -1 denotes that the means have to estimated
bw <- lm(cbind(prenumb, postnumb, funumb)~sex-1, data=sesamesim)
```

a) A vector containing estimates of the parameters:

```
est1 <-coef(bw)[1,1:3] # the three means for sex = 1
est2 <-coef(bw)[2,1:3] # the three means for sex = 2
estimate <- c(est1,est2)
```

b) A list containing, per group, the covariance matrix of the parameters:

```
# cov1 has to contain the covariance matrix of the three means in group 1.
# cov2 has to contain the covariance matrix in group 2
# typing vcov(bw) in the console pane highlights the structure of
# the covariance matrix of all 3+3=6 means
# it has to be dissected in to cov1 and cov2
cov1 <- c(vcov(bw)[1,1],vcov(bw)[1,3],vcov(bw)[1,5],vcov(bw)[3,1],
vcov(bw)[3,3],vcov(bw)[3,5],vcov(bw)[5,1],vcov(bw)[5,3],vcov(bw)[5,5])
cov1 <- matrix(cov1,3,3)
```

```
cov2 <- c(vcov(bw)[2,2],vcov(bw)[2,4],vcov(bw)[2,6],vcov(bw)[4,2],
vcov(bw)[4,4],vcov(bw)[4,6],vcov(bw)[6,2],vcov(bw)[6,4],vcov(bw)[6,6])
cov2 <- matrix(cov2,3,3)
covariance<-list(cov1,cov2)
```

c) Per group the sample size used for estimation of the parameters:

```
ngroup<-table(sesamesim$sex)
```

2) Assign names to the estimates:

```
names(estimate) <- c("pre1", "post1", "fu1","pre2", "post2", "fu2")
```

3) Set seed and 4) call `bain`:

```
set.seed(100)
results <-bain(estimate, "pre1 - pre2 = post1 - post2 = fu1 -fu2;
pre1 - pre2 > post1 - post2 > fu1 -fu2"  , n=ngroup, Sigma=covariance,
group_parameters=3, joint_parameters = 0)
```

5) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class numeric:
##
##    Fit   Com   BF.u    BF.c    PMPa  PMPb
## H1 0.052 0.000 111.145 111.145 0.994 0.985
## H2 0.175 0.243 0.722   0.663   0.006 0.006
## Hu                                   0.009
##
## Hypotheses:
##   H1: pre1-pre2=post1-post2=fu1-fu2
##   H2: pre1-pre2>post1-post2>fu1-fu2
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

6) Summary: estimates and credibility intervals:

```
summary(results, ci = 0.95)
```

```
##   Parameter   n   Estimate        lb        ub
## 1      pre1 115  0.3981884 -1.545126  2.341503
## 2     post1 115 30.0956522 27.792949 32.398355
## 3       fu1 115 34.9043478 31.184613 38.624082
## 4      pre2 125 -0.3663333 -2.230295  1.497629
## 5     post2 125 28.8560000 26.647325 31.064675
## 6       fu2 125 33.7840000 30.216156 37.351844
```

5.4.7 Logistic regression (1)

This example is presented as a counterpart of multiple regression.

Regression coefficients can only be mutually compared if the corresponding predictors are measured on the same scale, therefore the predictors are standardized

```
sesamesim$age <- (sesamesim$age - mean(sesamesim$age))/sd(sesamesim$age)
sesamesim$peabody <- (sesamesim$peabody - mean(sesamesim$peabody))/
sd(sesamesim$peabody)
sesamesim$prenumb <- (sesamesim$prenumb - mean(sesamesim$prenumb))/
sd(sesamesim$prenumb)
```

1) Execute the statistical analysis:

```
logreg <- glm(viewenc ~ age + peabody + prenumb, family=binomial,
data=sesamesim)
```

a) A vector containing estimates of the parameters:

```
estimate <- coef(logreg)
```

b) A list containing, per group, the covariance matrix of the parameters:

```
covmatr <- list(vcov(logreg))
```

c) Per group the sample size used for estimation of the parameters:

```
ngroup <- nrow(sesamesim)
```

2) Assign names to the estimates:

```
names(estimate) <- c("int", "age", "peab" ,"pre" )
```

3) Set seed and 4) call `bain`:

```
set.seed(100)
results <- bain(estimate, "age = peab = pre; age > pre > peab", n=ngroup,
Sigma=covmatr, group_parameters=4, joint_parameters = 0)
```

5) Print the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class numeric:
##
##    Fit   Com   BF.u  BF.c   PMPa  PMPb
## H1 0.176 0.018 9.865 9.865  0.682 0.638
## H2 0.611 0.133 4.605 10.257 0.318 0.298
## Hu                                0.065
##
## Hypotheses:
##   H1: age=peab=pre
##   H2: age>pre>peab
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

6) Summary: estimates and credibility intervals:

```
summary(results, ci = 0.95)
```

```
##   Parameter   n    Estimate         lb            ub
## 1       int 240 -1.18057044 -1.4856839 -0.8754569440
## 2       age 240  0.19968233 -0.1224419  0.5218065630
## 3      peab 240 -0.38378044 -0.7684559  0.0008950488
## 4       pre 240  0.01367146 -0.3745181  0.4018610168
```

5.4.8 Logistic regression (2)

This example is presented as a counterpart of an ANCOVA, based on Hoijtink, Gu, & Mulder (2019).

The research question concerns the probability of encouragement to view sesame street (a dichotomous 0/1 meaning no/yes dependent variable) for boys (sex=1)

and girls (sex=2) adjusted for their age. This is a kind of ANCOVA with a dichotomous instead of a continuous dependent variable.

Load the numDeriv package which will be used to compute the covariance matrix of the adjusted group coefficients and regression coefficient of age for the boys and the girls # using the estimates obtained using the data for the boys AND the girls.

```
# make a factor of the variable sex
sesamesim$sex <- factor(sesamesim$sex)
# center the covariate age
sesamesim$age <- sesamesim$age - mean(sesamesim$age)
```

1) Execute the statistical analysis:

c) Per group the sample size used for estimation of the parameters

```
# determine sample size per sex group
# (in case of missing values in the variables used, the command
# below has to be modified such that only the cases without
# missing values are counted)
ngroup <- table(sesamesim$sex)
# execute the logistic regression, -1 ensures that the coefficients
# for boys and girl are estimated adjusted for the covariate age
anal <- glm(viewenc ~ sex + age - 1, family=binomial, data=sesamesim)
```

a) A vector containing estimates of the parameters and add the names:

```
estimates <-coef(anal)
names(estimates) <- c("boys", "girls", "age")
```

b) A list containing, per group, the covariance matrix of the parameters:

Use numDeriv to compute the Hessian matrix and subsequently the covariance matrix for each of the two (boys and girls) groups. The vector f should contain the regression coefficient of the group at hand and the regression coefficient of the covariate.

- The first group:

```
data <- subset(cbind(sesamesim$sex,sesamesim$age,sesamesim$viewenc),
sesamesim$sex==1)
f <- 1
f[1] <- estimates[1] # the regression coefficient of boys
f[2] <- estimates[3] # the regression coefficient of age
```

Within the for loop below the log likelihood of the logistic regression model is computed using the data for the boys:

```
logist1 <- function(x){
  out <- 0
  for (i in 1:ngroup[1]){
    out <- out + data[i,3]*(x[1] + x[2]*data[i,2]) - log (1 +
    exp(x[1] + x[2]*data[i,2]))
  }
  return(out)
}
ngroup <- table(sesamesim$sex)
hes1 <- hessian(func=logist1, x=f)
# multiply with -1 and invert to obtain the covariance matrix for the
# first group
cov1 <- -1 * solve(hes1)
```

- The second group (repeat the same):

```
data <- subset(cbind(sesamesim$sex,sesamesim$age,sesamesim$viewenc),
sesamesim$sex==2)
f[1] <- estimates[2] # the regression coefficient of girls
f[2] <- estimates[3] # the regression coefficient of age
```

```
logist2 <- function(x){
  out <- 0
  for (i in 1:ngroup[2]){
    out <- out + data[i,3]*(x[1] + x[2]*data[i,2]) - log (1 +
    exp(x[1] + x[2]*data[i,2]))
  }
  return(out)
}
hes2 <- hessian(func=logist2, x=f)
# multiply with -1 and invert to obtain the covariance matrix
cov2 <- -1 * solve(hes2)
```

Combine the covariance matrices:

```
covariance<-list(cov1,cov2)
```

2) Assign names to the estimates:

```
names(estimates) <- c("boys", "girls", "age")
```

3) Set seed and 4) call `bain`:

```
set.seed(100)
results <- bain(estimates, "boys < girls & age > 0", n=ngroup,
Sigma=covariance, group_parameters=1, joint_parameters = 1)
```

5) Print the results:

```
print(results)
```

6) Summary: estimates and credibility intervals:

```
summary(results, ci = 0.95)
```

5.4.9 CFA

This option uses `bain.default()`, so the user must manually extract the required information. See the tutorial by Van Lissa et al. (2021) for further elaborations.

First, we fit a structural equation model using `lavaan`:

```
model1 <- 'A =~ Ab + Al + Af + An + Ar + Ac
           B =~ Bb + Bl + Bf + Bn + Br + Bc'
fit1 <- sem(model1, data = sesamesim, std.lv = TRUE)
```

We want to specify the following hypotheses:

```
hypotheses <- "(Ab, Al, Af, An, Ar, Ac) >.6 &
               (Bb, Bl, Bf, Bn, Br, Bc) >.6"
```

Next, we extract the required information from the `lavaan` model object, in the order of the parameter table above:

```
# Extract standardized parameter estimates (argument x)
PE1 <- parameterEstimates(fit1, standardized = TRUE)
# Identify which parameter estimates are factor loadings
loadings <- which(PE1$op == "=~")
# Collect the standardized "std.all" factor loadings "=~"
```

```
estimates <- PE1$std.all[loadings]
# Assign names to the standardized factor loadings
names(estimates) <- c("Ab", "Al", "Af", "An", "Ar", "Ac",
                      "Bb", "Bl", "Bf", "Bn", "Br", "Bc")

# Extract sample size (argument n)
n <- nobs(fit1)

# Extract the standardized model parameter covariance matrix,
# selecting only the rows and columns for the loadings
covariance <- lavInspect(fit1, "vcov.std.all")[loadings, loadings]

# Give the covariance matrix the same names as the estimate
dimnames(covariance) <- list(names(estimates), names(estimates))

# Put the covariance matrix in a list
covariance <- list(covariance)

# Specify number of group parameters; this simply means that there
# are 12 parameters in the estimate.
group_parameters <- 12

# Then, the number of joint parameters. This is always 0 for evaluations of
# SEM hypotheses:
joint_parameters <- 0
```

Now, we can evaluate the hypotheses using `bain.default()`:

```
res <- bain(x = estimates,
            hypothesis = hypotheses,
            n = n,
            Sigma = covariance,
            group_parameters = group_parameters,
            joint_parameters = joint_parameters)
res
```

```
## Bayesian informative hypothesis testing for an object of class numeric:
##
##    Fit   Com   BF.u   BF.c    PMPa  PMPb
## H1 0.880 0.009 93.351 768.863 1.000 0.989
## Hu                                 0.011
##
## Hypotheses:
##   H1: (Ab,Al,Af,An,Ar,Ac)>.6&(Bb,Bl,Bf,Bn,Br,Bc)>.6
##
```

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

Finally, we print the results of the bain analysis and obtain the estimates and 95% central credibility intervals:

```
sres <- summary(res, ci = 0.95)
sres
```

5.5 Using bain with missing data[1]:

Load mice (multiple imputation of missing data) inspect the mice help file to obtain further information:

```
library(mice)
```

```
##
## Attaching package: 'mice'

## The following object is masked from 'package:stats':
##
##     filter

## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(psych)
```

```
##
## Attaching package: 'psych'

## The following object is masked from 'package:lavaan':
##
##     cor2cov
```

```
library(MASS)
```

"Create" missing values in four variables from the sesamesim data set:

[1] Running this example on your machine will take about 60 seconds

```
sesamesim <- cbind(sesamesim$prenumb,sesamesim$postnumb,sesamesim$funumb,sesamesim$peal
colnames(sesamesim) <- c("prenumb","postnumb","funumb","peabody")
sesamesim <- as.data.frame(sesamesim)

set.seed(1)
pmis1<-1
pmis2<-1
pmis3<-1

for (i in 1:240){
  pmis1[i] <- .80
  pmis2[i]<- .80
  pmis3[i]<- .80

  uni<-runif(1)
  if (pmis1[i] < uni) {
    sesamesim$funumb[i]<-NaN
  }
  uni<-runif(1)
  if (pmis2[i] < uni) {
    sesamesim$prenumb[i]<-NaN
    sesamesim$postnumb[i]<-NaN
  }
  uni<-runif(1)
  if (pmis3[i] < uni) {
    sesamesim$peabody[i]<-NaN
  }
}
# print data summaries - note the missing values (NAs)
print(describe(sesamesim))
```

```
##          vars   n  mean    sd median trimmed   mad   min   max range skew
## prenumb     1 200  0.04  0.98  -0.02   -0.01  0.98 -1.86  2.94  4.80 0.53
## postnumb    2 200 29.55 12.70  28.00   29.30 11.86  0.00 63.00 63.00 0.23
## funumb      3 190 34.17 20.16  33.50   33.12 21.50  0.00 91.00 91.00 0.38
## peabody     4 190  0.06  1.04  -0.05    0.04  1.11 -2.60  2.81  5.41 0.18
##          kurtosis   se
## prenumb     -0.02 0.07
## postnumb    -0.21 0.90
## funumb      -0.41 1.46
## peabody     -0.38 0.08
```

1) Use `mice` to create 1000 imputed data matrices. Note that, the approach used below is only one manner in which mice can be instructed. Many other options are available:

```
M <- 1000
out <- mice(data = sesamesim, m = M, seed=999, meth=c("norm","norm","norm","norm"), diagnostics =
# create matrices in which 1000 vectors with estimates can be stored and in which a covariance ma
mulest <- matrix(0,nrow=1000,ncol=2)
covwithin <- matrix(0,nrow=2,ncol=2)
```

2) Create matrices in which 1000 vectors with estimates can be stored and in which a covariance matrix can be stored:

```
mulest <- matrix(0,nrow=1000,ncol=2)
covwithin <- matrix(0,nrow=2,ncol=2)
```

3) Execute 1000 multiple regressions using the imputed data matrices and store the estimates of only the regression coefficients of `funumb` on `prenumb` and `postnumband`and the average of the 1000 covariance matrices. See Hoijtink, Gu, Mulder, & Rosseel (2019) for an explanation of the latter.

```
for(i in 1:M) {
  mulres <- lm(funumb~prenumb+postnumb,complete(out,i))
  mulest[i,]<-coef(mulres)[2:3]
  covwithin<-covwithin + 1/M * vcov(mulres)[2:3,2:3]
}
```

4) Compute the average of the estimates and assign names, the between and total covariance matrix. See Hoijtink, Gu, Mulder, & Rosseel (2019) for an explanation.

```
estimates <- colMeans(mulest)
names(estimates) <- c("prenumb", "postnumb")
covbetween <- cov(mulest)
covariance <- covwithin + (1+1/M)*covbetween
# determine the sample size
samp <- nrow(sesamesim)
```

HERE 5) compute the effective sample size. See Hoijtink, Gu, Mulder, & Rosseel (2019).

```
nucom<-samp-length(estimates)
lam <- (1+1/M)*(1/length(estimates))* sum(diag(covbetween %*% ginv(covariance)))
nuold<-(M-1)/(lam^2)
nuobs<-(nucom+1)/(nucom+3)*nucom*(1-lam)
nu<- nuold*nuobs/(nuold+nuobs)
fracmis <- (nu+1)/(nu+3)*lam + 2/(nu+3)
neff<-samp-samp*fracmis
covariance<-list(covariance)
```

6. Use `bain`:

```
set.seed(100)
results <- bain(estimates,"prenumb=postnumb=0",n=neff,Sigma=covariance,group_parameters
```

7. Display the results:

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class numeric:
##
##    Fit   Com   BF.u  BF.c  PMPa  PMPb
## H1 0.000 0.010 0.000 0.000 1.000 0.000
## Hu                                1.000
##
## Hypotheses:
##   H1: prenumb=postnumb=0
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrain
```

8. Obtain the descriptive table:

```
summary(results, ci = 0.95)
```

```
##   Parameter        n   Estimate         lb       ub
## 1   prenumb 172.0185 -1.8492141 -5.2953273 1.596899
## 2  postnumb 172.0185  0.9692542  0.6928871 1.245621
```

Chapter 6

Sensitivity analysis using `bain`

Load `bain`:

```
library(bain)
```

Please see Hoijtink, Mulder, Lissa, & Gu (2019) for further elaborations.

Note that, most the code below can be replaced by a call to the function `bain_sensitivity`. See the help file for this function for further instructions (call `?bain_sensitivity`).

6.1 The `t-test` example from 3.1.1

First, we test the same hypotheses as in 3.1.1 using the minimal fraction (*J=1*) which is by default implemented in `bain`.

```
# set a seed value
set.seed(100)
# test hypotheses with bain. The names of the means are x and y.
results <- bain(ttest, "x = y; x > y; x < y", fraction = 1)
```

```
print(results)
```

```
## Bayesian informative hypothesis testing for an object of class t_test:
##
```

```
##    Fit   Com   BF.u   BF.c   PMPa  PMPb
## H1 0.183 0.016 11.583 11.583 0.853 0.794
## H2 0.777 0.500 1.554  3.481  0.114 0.107
## H3 0.223 0.500 0.446  0.287  0.033 0.031
## Hu                                0.069
##
## Hypotheses:
##   H1: x=y
##   H2: x>y
##   H3: x<y
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

Now we execute the sensitivity analysis by interactively changing the `fraction` argument to be equal to 2 and 3, respectively:

```
results2 <- bain(ttest, "x = y; x > y; x < y", fraction = 2)
results3 <- bain(ttest, "x = y; x > y; x < y", fraction = 3)
```

```
print(results2)
```

```
## Bayesian informative hypothesis testing for an object of class t_test:
##
##    Fit   Com   BF.u  BF.c  PMPa  PMPb
## H1 0.183 0.022 8.190 8.190 0.804 0.732
## H2 0.777 0.500 1.554 3.481 0.152 0.139
## H3 0.223 0.500 0.446 0.287 0.044 0.040
## Hu                                0.089
##
## Hypotheses:
##   H1: x=y
##   H2: x>y
##   H3: x<y
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrai
```

```
print(results3)
```

```
## Bayesian informative hypothesis testing for an object of class t_test:
##
##    Fit   Com   BF.u  BF.c  PMPa  PMPb
## H1 0.183 0.027 6.687 6.687 0.770 0.690
## H2 0.777 0.500 1.554 3.481 0.179 0.160
## H3 0.223 0.500 0.446 0.287 0.051 0.046
```

```
## Hu                                0.103
##
## Hypotheses:
##   H1: x=y
##   H2: x>y
##   H3: x<y
##
## Note: BF.u denotes the Bayes factor of the hypothesis at hand versus the unconstrained hypothe
```

As it is clear by looking at the results, the BF_{0u} changes from 11.58 when `fraction` = 1 to 8.19 and 6.68, when `fraction` = 2 and 3, respectively. This stems from the fact that the value of the **complexity** (which is the proportion of the prior distribution that is supported by the (*null*) hypothesis at hand) changes, thus changing the resulting BF's.

Literature

Gu, X., Hoijtink, H., Mulder, J., & van Lissa, C. J. (2021). *Bain: Bayes factors for informative hypotheses*. Retrieved from https://CRAN.R-project.org/package=bain

Gu, X., Mulder, J., & Hoijtink, H. (2018). Approximated adjusted fractional bayes factors: A general method for testing informative hypotheses. *British Journal of Mathematical and Statistical Psychology*, *71*, 229–261.

Hoijtink, H., Gu, X., & Mulder, J. (2019). Bayesian evaluation of informative hypotheses for multiple populations. *British Journal of Mathematical and Statistical Psychology*, *72*, 219–243.

Hoijtink, H., Gu, X., Mulder, J., & Rosseel, Y. (2019). Computing bayes factors from data with missing values. *Psychological Methods*, *24*, 253.

Hoijtink, H., Klugkist, I., & Boelen, P. A. (2008). *Bayesian evaluation of informative hypotheses*. Springer.

Hoijtink, H., Mulder, J., Lissa, C. van, & Gu, X. (2019). A tutorial on testing hypotheses using the bayes factor. *Psychological Methods*, *24*, 539.

Jones, J. A., & Waller, N. G. (2015). The normal-theory and asymptotic distribution-free (ADF) covariance matrix of standardized regression coefficients: Theoretical extensions and finite sample behavior. *Psychometrika*, *80*, 365–378.

Kass, R. E., & Raftery, A. E. (1995). Bayes factors. *Journal of the American Statistical Association*, *90*, 773–795.

Stevens, J. (1996). *Applied multivariate statistics for the social sciences. Malwah, NJ: Lawerence erlbaum associates*. Inc.

Van Lissa, C. J., Gu, X., Mulder, J., Rosseel, Y., Van Zundert, C., & Hoijtink, H. (2021). Teacher's corner: Evaluating informative hypotheses using the bayes factor in structural equation models. *Structural Equation Modeling: A Multidisciplinary Journal*, *28*, 292–301.

Appendix A