A GitBook template

Nikola Sekulovski

2021-10-27

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.*

This is how we would reference a book (Xie, 2015) or Xie (2015).

This is how we would reference articles Gu, Mulder, & Hoijtink (2018) or (Hoijtink, Klugkist, & Boelen, 2008).

Chapter 2

Monte Carlo Simulations

```
library(tidyverse)
```

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.

We can construct a function that does this:

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

```
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.

```
results %>%
  filter(Covered ==0) %>%
  kableExtra::kable(caption = "Here is a table of the samples" )
```

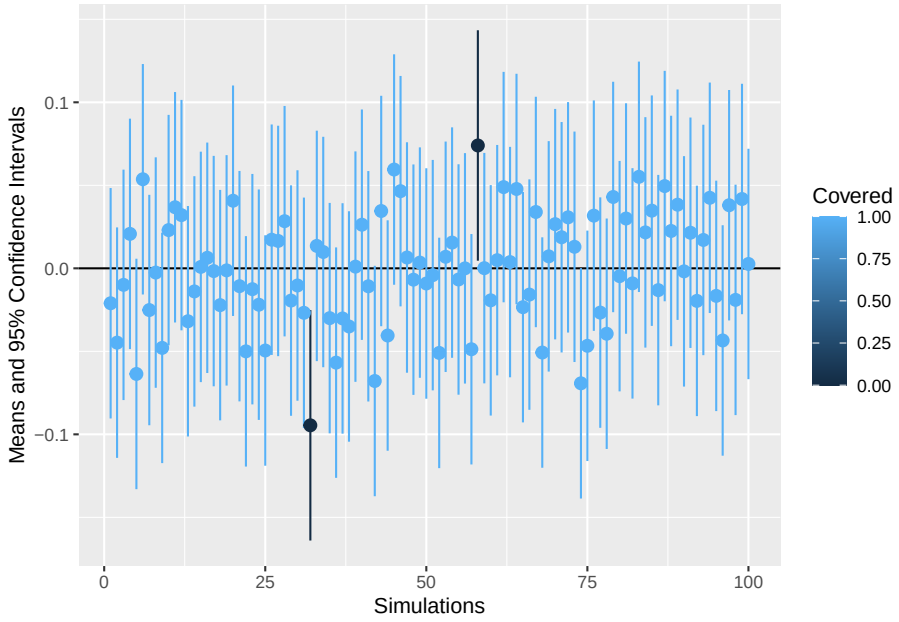
And finally we can also make a nice plot illustrating everything that we did so far.

```
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.

```
samples_2 <- samples <-plyr::rlply(1000, rpois(800, 2))
```

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))
```

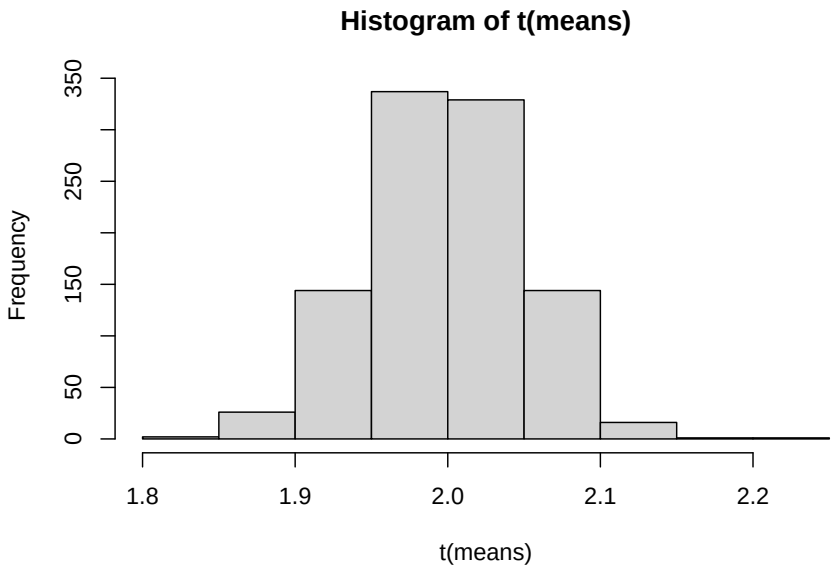


Figure 2.1: Histogram of the sampling distribution of the mean

Chapter 3

Equations & Picture

3.1 Bayes' theorem

$$p(\theta|D) = \frac{p(D|\theta)p(\theta)}{p(D)} \tag{3.1}$$

3.2 Normal PDF

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right) \tag{3.2}$$

This is how we refer to equations: -see equation (3.2)

3.3 Picture

SMBC Comics

(PART) `R` package `bain`

An 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.

3.4 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, van Lissa, & Gu (2019). By reading the tutorial in combination with executing the analyses contained in easyBFtutorial.R and BFtutorial.R (available on the abovementioned 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, C., Gu, X., Mulder, J., Rosseel, Y., van Zundert, C. and Hoijtink, H. (2020), Structural Equation Modelling, 28, 292-301[1].

Users are advised to read these tutorials and this vignette before using bain.

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

3.5 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.

3.5.1 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 below the section on **"Specification of Hypotheses"**).

`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 the next chapter show how the `fraction` can be employed to execute a sensitivity analysis. See also Hoijtink, Mulder, van Lissa, & Gu (2019) for more details.

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

3.5.2 Using `bain` with a `lm` or `t_test` object

The following steps need to be executed:

1) `x <- lm()` or `x <- t_test()`. Execute an analysis with `lm` or `t_test`. See the Examples section below for a complete elaboration of the calls to `lm` and `t_test` that can be processed by `bain`. Note that, `lm` and `t_test` will apply list-wise deletion if there are cases with missing values in the variables used.
2) If `lm()` is used, display the estimates and their names using the command `coef(x)`. (Unique abbreviations of) the names will be used to specify `hypotheses`. If `t_test()` is used, hypotheses have to be specified using the names `x`, `y`, and `difference` (see Examples 1a throught 1e which can be found below).
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(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, ANCOVA, and `t_test`. 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. Example 2d. below shows how `fraction` can be employed to execute a sensitivity analysis. See also Hoijtink, Mulder, van Lissa, and Gu, 2019).
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.

3.5.3 Using `bain` with a `lavaan` object

The following steps need to be executed:

1) `x <- sem()` or `x <- cfa()` or `x <- growth()`. Execute an analysis with the sem, cfa, or growth functions 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 4 "Examples Using a Named Vector" in Example h. (based on Hoijtink, Gu, Mulder, and 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.

2) 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.
3) 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.

4) `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.
5) `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. Example 2d. below shows how `fraction` can be employed to execute a sensitivity analysis. See also Hoijtink, Mulder, van Lissa, and Gu, 2019).
6) `print(results)` Print the results of an analysis with `bain`.
7) `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.

3.5.4 Using `bain` with a named vector

The following steps need to be executed:

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. Example 2d. below shows how `fraction` can be employed to execute a sensitivity analysis. See also Hoijtink, Mulder, van Lissa, and Gu, 2019).

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.

3.5.5 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 "Using bain with a lavaan object" 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 (2019). 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.

3.5.6 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, van Lissa, and 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$independent_restrictions` displays the number of independent constraints in the set of hypotheses under consideration. Note that, in Gu, Mulder, and Hoijtink (2018) en Hoijtink, Gu, and Mulder (2019) the definition given was misprinted (besides R and S also r and s should have been added to the definition).
- `results$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.

Chapter 4

Examples using `bain`

Note that, each of the examples given below can be run by 1) copy pasting them into the Source screen of `RStudio`. 2) by highlighting them followed by `ctrl-enter` or `cmd-enter`.

Unless indicated otherwise, the examples that follow below use a simulated data set inspired by the Sesame Street data set from: 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

References

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

Hoijtink, H., Mulder, J., van Lissa, C., & Gu, X. (2019). A tutorial on testing hypotheses using the bayes factor. *Psychological Methods*, *24*, 539–556.

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.

Xie, Y. (2015). *Dynamic documents with R and knitr* (2nd ed.). Boca Raton, Florida: Chapman; Hall/CRC.

Appendix A