Bayesian Data Analysis

Daniel Paulin & Nicolò Margaritella

University of Edinburgh

Semester 2, 2020/2021

With thanks to Jonathan Gair, Ruben Amoros-Salvador, Ken Newman, Vanda Inácio and Natalia Bochkina for much of the material.

Outline

The BUGS language

- ↪ We will learn now the basics of the BUGS language.
- ↪ BUGS stands for **B**ayesian Inference **U**sing **G**ibbs **S**ampling and began in 1989.
- ↪ Its basic philosophy was to separate the language used to describe a model from the actual programs used to carry out the computations (the *engine*).
- ↪ This approach has two attractive advantages:
  - ↪ One can easily specify complex models without requiring extensive knowledge of Bayesian computational tools.
  - ↪ The language has remained stable and consistent over time, even as the underlying programs which actually do the sampling are constantly changing.

The BUGS language

- ↪ The syntax of the BUGS language is relatively straightforward. Every line does one of the following things:
  - ↪ Defines a stochastic node
    `theta~dbeta(1,1)`
  - ↪ Defines a deterministic node
    `tau=pow(sigma,-2)`
  - ↪ Provides a comment
    `## Prior`
  - ↪ Defines a loop
    `for (i in 1:n)`
- ↪ Common distributions in the BUGS language are `dbin`, `dpois`, `dunif`, `dnorm`, `dgamma`.
- ↪ The normal is parameterised in terms of its mean and precision (=1/ variance).
- ↪ Functions cannot be used as arguments in distributions (one needs to create new nodes).
- ↪ For more details on the distributions and functions available in BUGS, check the online manual.

The BUGS language

- Historically, the most widely used BUGS engine has been a program called WinBUGS. It began in the early 1980s under David Spiegelhalter at the Medical Research Council Biostatistics Unit in Cambridge, UK.
- Further developments by the developers of WinBUGS, however, have focused on a somewhat different program called OpenBUGS; the features and appearance of the programs are similar, but OpenBUGS is more compatible with different operating systems.
- Information about WinBUGS and OpenBUGS is available at

  `https://www.mrc-bsu.cam.ac.uk/software/bugs/`

- A separate project, JAGS (Just Another Gibbs Sampler), can be thought of as an engine for running BUGS models, although strictly speaking, the language syntax for the two programs is not always identical.

  `http://mcmc-jags.sourceforge.net/`

The BUGS language

↪ Another two recent engines are `STAN` and `NIMBLE`, which can be found, respectively, at

`http://mc-stan.org/`

`https://r-nimble.org`

The BUGS language

↪ All engines carry out the following steps for fitting BUGS models:

1. Checking model syntax.
2. Reading in data.
3. Compiling the model.
4. Initialising the simulation.
5. Sampling.
6. Reporting results.

The BUGS language

Running JAGS via `rjags`

↪ In general, the steps to using JAGS to produce samples are:

1. Download JAGS at

   `http://sourceforge.net/projects/mcmc-jags/files/`

2. Install the `rjags` package in `R`, which should automatically find your JAGS installation.
3. Specify the statistical model (likelihood and prior) using the model command.
4. Compile the model using `jags.model`.
5. Generate samples using `update` and `coda.samples`.

↪ As it was already mentioned, there are, at least, 3 `R` interfaces for JAGS. The model syntax is the same for the three interfaces, what changes is how relevant information is extracted. We will stick with `rjags`, although you are free to choose your preferred interface.

↪ As we are using Kaggle.com for the workshops in this course, the code for loading JAGS and `rjags` is already included in the workshop notebook file. So no installation is required on your personal computer.

The BUGS language

Running JAGS via `rjags`

- We will learn `rjags` as we go! As an introductory example, we will go back to the example where we have normally distributed data where both the mean and the variance are unknown, i.e., $Y_i \overset{\text{iid}}{\sim} \text{N}(\mu, \sigma^2)$, with μ and σ^2 unknown (see `R` script for Lecture 2, JAGS example unknown mean and variance).

- We note again that the BUGS language uses the precision, instead of the variance, in the parameterisation of the normal distribution. Therefore, we assume $\mu \sim \text{N}(\mu_0, \sigma_0^2)$ and $\sigma^{-2} \sim \text{Gamma}(a, b)$, with $\mu_0 = 0$, $\sigma_0^2 = 100$, and $a = b = 0.1$.

- First we must load the `rjags` package

```
require(rjags)
```

The BUGS language
Running JAGS via `rjags`

↪ The program specifying the model (BUGS) code must be put in a separate file which is then read by JAGS. When working in `R` this is most conveniently done by saving the model in an object, e.g., `model_string="model{...}"` and then using the `R` function `textConnection` when passing the model to the `jags.model` function.

```
model_string <- "model{

  # Likelihood
  for(i in 1:n){
    y[i]~dnorm(mu,inv.var)
  }

  # Prior for mu
  mu~dnorm(mu0,inv.var0)
  inv.var0=1/sigma02

  # Prior for the inverse variance
  inv.var~dgamma(a, b)

  # Compute the variance
  sigma2=1/inv.var
}"
```

The BUGS language
Running JAGS via rjags

↪ Alternatively we can use the R function cat, to put the model in a file, say, m1.jag

```
cat("model{

# Likelihood
for(i in 1:n){
y[i]~dnorm(mu,inv.var)
}

# Prior for mu
mu~dnorm(mu0,inv.var0)
inv.var0=1/sigma02

# Prior for the inverse variance
inv.var~dgamma(a, b)

# Compute the variance
sigma2=1/inv.var
}",
file="m1.jag")
```

The BUGS language

Running JAGS via `rjags`

- ↪ The BUGS language is declarative, i.e., it is not executed as the program runs. Instead it is a specification of the model structure, and after the model is set up JAGS will decide how best to go about the MCMC simulation.
- ↪ We now specify the parameters we will feed into JAGS (data and priors)

```
data=list(y=y,n=n,mu0=mu0,sigma02=sigma02,a=a,b=b)
```

- ↪ Starting values can be supplied, although JAGS will be able to generate them in most cases. If we want to run several chains, for example to monitor convergence, we must supply starting values for each chain.
- ↪ In this example, we use three chains, hence the initial values are a list of three lists. Each of these lists has one named value for each parameter – in this case there are two parameters: `mu` (mean) and `inv.var` (precision).

```
inits=list(list(mu=mean(y),inv.var=1/var(y)), list(mu=0,inv.var=1), list(mu=10,inv.var=0.1))
```

The BUGS language

Running JAGS via `rjags`

- Once the model, the data, and the initial values (in case we want to supply them) have been specified, we use the function `jags.model` to compile and initialise the model.

```
model=jags.model(textConnection(model_string),n.chains=3,data=data,inits=inits)
```

- As this step, we obtain the following

```
Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 150
   Unobserved stochastic nodes: 2
   Total graph size: 160

Initializing model
```

- In case we have used the function `cat` to define the model and saved it as file `m1.jag`, we would instead write

```
model=jags.model("m1.jag",n.chains=3,data=data,inits=inits)
```

The BUGS language

Running JAGS via `rjags`

- ↪ To get samples from the posterior distribution of the parameters, we use the `coda.samples` function after first using the `update` function to run the Markov Chain for a burn-in period of a number of specified iterations (in this case 1000).

- ↪ For `coda.samples` we must specify

  - ↪ The variables (nodes) that we want to monitor in the subsequent cycles of the chain. This is done using the argument variable.names. In our case the variables we want to monitor are called `mu` and `sigma2`.

  - ↪ How many iterations to run the chain (n.iter).

  - ↪ How often we sample the specified parameters and retain the results in memory (thin, by default equal to one).

The BUGS language

Running JAGS via `rjags`

↪ We thus type

```
update(model,1000,progress.bar="none")
res=coda.samples(model,variable.names=c("mu","sigma2"),n.iter=10000, progress.bar="none")
```

↪ As always in `R`, the most useful overview comes from the `summary` function

```
> summary(res)

Iterations = 12001:22000
Thinning interval = 1
Number of chains = 3
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:

        Mean     SD Naive SE Time-series SE
mu     1.926 0.2336 0.001348       0.001336
sigma2 8.215 0.9591 0.005537       0.005640

2. Quantiles for each variable:

        2.5%   25%   50%   75%  97.5%
mu     1.468 1.767 1.926 2.083  2.384
sigma2 6.541 7.541 8.144 8.812 10.283
```

The BUGS language

Running JAGS via `rjags`

↪ We can also type `plot(res)` to visualise the traceplots and densityplots.

↪ For inspection of the autocorrelation function, type `autocorr.plot(res)` and for the effective sample size use `effectiveSize(res)`. The Gelman-Rubin statistic is obtained by typing `gelman.plot(res)` (requires that the number of chains is equal or greater than 2).

↪ We can also extract results for each chain and each parameter using the object `res`. For instance, to obtain the values of `mu` of the first chain, we type `res[[1]][,1]`, while for obtaining the `sigma2` values for chain 3, we use `res[[3]][,2]`.

↪ We can also collect all the posterior samples from the different chains in one matrix by typing `resmat=as.matrix(res)`.

Regression

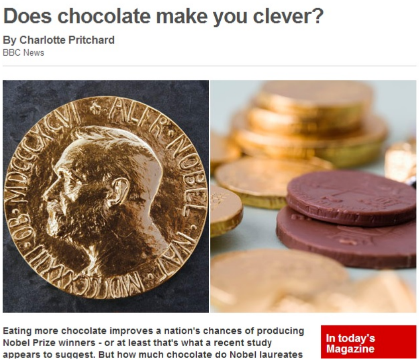
Linear regression: general context

- ↪ Linear regression is, perhaps, the most widely used statistical modelling tool.
- ↪ It addresses the following question: How does a quantity of primary interest, y, vary with (depend upon) another quantity, or set of quantities, $\mathbf{x}$?
- ↪ The quantity y is called the response or outcome variable. Some people simply refer to it as the dependent variable.
- ↪ The variable(s) $\mathbf{x} = (x_1, x_2, \ldots, x_p)$ are called explanatory variables, covariates or simply independent variables.
- ↪ For example, y might be the final mark in the Bayesian data analysis class. Variable x_1 might be the number of workshops attended, x_2 the amount of time spent studying, and so on. Regression allows us to test the effect of, say, the number of workshops attended, but accounting for the time spent studying.

Regression

Linear regression: examples

Some examples (of varying quality!) in the media:

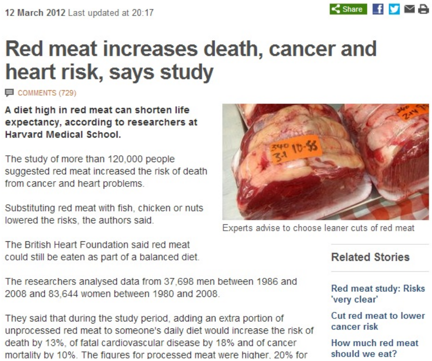


Does chocolate make you clever?

By Charlotte Pritchard
BBC News

Eating more chocolate improves a nation's chances of producing Nobel Prize winners - or at least that's what a recent study appears to suggest. But how much chocolate do Nobel laureates

In today's Magazine



12 March 2012 Last updated at 20:17

Share

Red meat increases death, cancer and heart risk, says study

COMMENTS (729)

A diet high in red meat can shorten life expectancy, according to researchers at Harvard Medical School.

The study of more than 120,000 people suggested red meat increased the risk of death from cancer and heart problems.

Substituting red meat with fish, chicken or nuts lowered the risks, the authors said.

The British Heart Foundation said red meat could still be eaten as part of a balanced diet.

The researchers analysed data from 37,698 men between 1986 and 2008 and 83,644 women between 1980 and 2008.

They said that during the study period, adding an extra portion of unprocessed red meat to someone's daily diet would increase the risk of death by 13%, of fatal cardiovascular disease by 18% and of cancer mortality by 10%. The figures for processed meat were higher, 20% for

Experts advise to choose leaner cuts of red meat

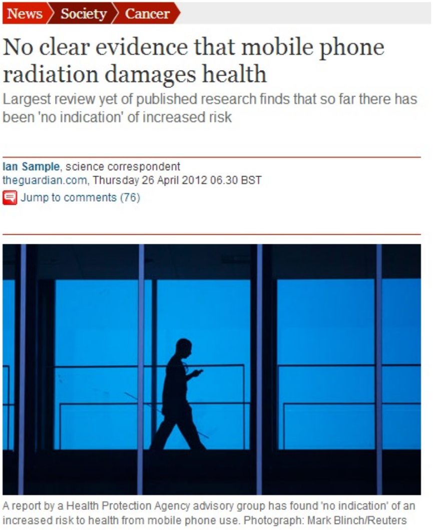
Related Stories

Red meat study: Risks 'very clear'

Cut red meat to lower cancer risk

How much red meat should we eat?

Linear regression



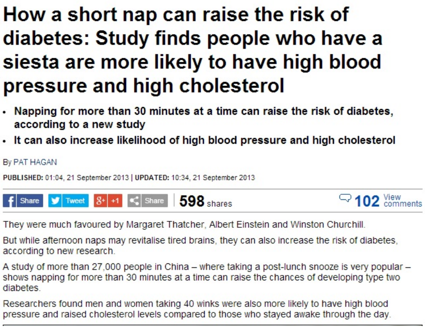
News › Society › Cancer

No clear evidence that mobile phone radiation damages health

Largest review yet of published research finds that so far there has been 'no indication' of increased risk

Ian Sample, science correspondent
theguardian.com, Thursday 26 April 2012 06.30 BST
Jump to comments (76)

A report by a Health Protection Agency advisory group has found 'no indication' of an increased risk to health from mobile phone use. Photograph: Mark Blinch/Reuters



How a short nap can raise the risk of diabetes: Study finds people who have a siesta are more likely to have high blood pressure and high cholesterol

- Napping for more than 30 minutes at a time can raise the risk of diabetes, according to a new study
- It can also increase likelihood of high blood pressure and high cholesterol

By PAT HAGAN

PUBLISHED: 01:04, 21 September 2013 | UPDATED: 10:34, 21 September 2013

Share Tweet +1 Share 598 shares 102 View comments

They were much favoured by Margaret Thatcher, Albert Einstein and Winston Churchill.

But while afternoon naps may revitalise tired brains, they can also increase the risk of diabetes, according to new research.

A study of more than 27,000 people in China – where taking a post-lunch snooze is very popular – shows napping for more than 30 minutes at a time can raise the chances of developing type two diabetes.

Researchers found men and women taking 40 winks were also more likely to have high blood pressure and raised cholesterol levels compared to those who stayed awake through the day.

Regression

Linear regression: examples

↪ The two main aims of regression analysis are

1. **Estimation**: the estimation of relationships between the explanatory and response variables with the object of both identifying and quantifying this relationship.
2. **Prediction**: the prediction of new values of y from values of $x_1, \ldots x_p$.

Regression

Linear regression: formulation

- $\hookrightarrow$ We will start by considering regression models of the form

$$y_i \overset{\text{ind}}{\sim} \mathrm{N}(\mu_i, \sigma^2), \qquad i = 1, \ldots, n \tag{1}$$

where

$$\mu_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip}.$$

- $\hookrightarrow$ The variables $\beta_0, \beta_1, \ldots \beta_p$ are an unknown set of regression coefficients (to be estimated from the data).

- $\hookrightarrow$ Equation (1) can also be written as

$$y_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_p x_{ip} + \epsilon_i, \qquad i = 1, \ldots, n$$

where $\epsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$.

Regression

Linear regression: formulation

$\hookrightarrow$ This is because, since the x_{ij} are known, we can write

$$\begin{aligned}
\mathbb{E}\left(y_i \mid x_{i1}, \dots x_{ip}\right) &= \mathbb{E}\left(\beta_0 + \beta_1 x_{i1} + \dots \beta_p x_{ip} + \epsilon_i\right) \\
&= \beta_0 + \dots + \beta_p x_{ip} \\
&= \mu_i
\end{aligned}$$

$$\begin{aligned}
\operatorname{var}(y_i \mid x_{1i}, \dots x_{pi}) &= \operatorname{var}(\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \epsilon_i) \\
&= \operatorname{var}(\epsilon_i) \\
&= \sigma^2
\end{aligned}$$

$\hookrightarrow$ In short, $y_i = \mathbf{x}_i^T \boldsymbol{\beta} + \varepsilon_i$, where $x_i = (1, x_{i1}, \dots, x_{ip})^T$ and $\boldsymbol{\beta} = (\beta_0, \beta_1, \dots, \beta_p)^T$.

Regression

Linear regression: formulation

$\hookrightarrow$ In matrix notation, we have

$$\mathbf{y} = \mathbf{x}\boldsymbol{\beta} + \boldsymbol{\varepsilon},$$

where $\mathbf{y} = (y_1, \ldots, y_n)^T$, $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n)^T$,

$$\mathbf{x} = \begin{bmatrix} 1 & x_{11} & x_{12} & \ldots & x_{1p} \\ 1 & x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix},$$

and $\boldsymbol{\varepsilon} \sim \mathrm{N}(\mathbf{0}, \sigma^2 I_n)$.

Regression

Linear regression: formulation

- There are $p+2$ unknown parameters to be estimated:
  - $\beta = (\beta_0, \beta_1, \ldots \beta_p)^T$
  - σ^2
- The data $(y_i, x_{i1}, x_{i2}, \ldots x_{ip})$ for $i \in \{1, \ldots n\}$ are used to obtain estimates.
- Under the classical setting, the estimator for $\boldsymbol{\beta}$ is given by

$$\hat{\boldsymbol{\beta}} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y},$$

  assuming that $(\mathbf{x}^T\mathbf{x})$ is invertible. Since we are assuming normal errors, both maximum likelihood and ordinary least squares give rise to this estimator.

Regression

Linear regression: Bayesian formulation

- We will now turn our attention to Bayesian estimation of $\boldsymbol{\beta}$ and σ^2.
- The Bayesian linear model is given by

$$\begin{aligned} y_i \mid \mu_i, \sigma^2, \mathbf{x}_i &\overset{\text{ind}}{\sim} \mathrm{N}(\mu_i, \sigma^2), \qquad i = 1, \ldots, n, \\ \mu_i &= \mathbf{x}_i^T \boldsymbol{\beta}, \\ \boldsymbol{\beta}, \sigma^2 &\sim \pi(\boldsymbol{\beta}, \sigma^2). \end{aligned}$$

- $\pi(\boldsymbol{\beta}, \sigma^2)$ is the joint prior on the parameters. There are several approaches to model the joint prior distribution.
- All inference proceeds from $p(\boldsymbol{\beta}, \sigma^2 \mid \mathbf{y})$.
- The full conditionals $p(\beta_j \mid \tau, \mathbf{y})$ are not difficult, but tedious, to obtain. The full conditional distribution $p(\tau \mid \boldsymbol{\beta}, \mathbf{y})$, where $\tau = \sigma^{-2}$, is easily obtained.

Regression

Linear regression: Bayesian formulation

- In normal regression models, the simplest approach is to assume that all parameters are a priori independent having the structure

$$\pi(\boldsymbol{\beta}, \tau) = \prod_{j=0}^{p} \pi_j(\beta_j)\pi(\tau), \qquad \tau = \sigma^{-2},$$
$$\beta_j \sim \mathrm{N}(\mu_{\beta_j}, \sigma^2_{\beta_j}), \quad j = 0, \ldots, p,$$
$$\tau \sim \mathrm{Gamma}(a, b).$$

- In this prior setup, again, we have substituted the variance σ^2 by the corresponding precision parameter τ in order to make it compatible to the BUGS notation.

Regression

Linear regression: Bayesian formulation

- When no information is available, a usual choice for the prior mean is the zero value ($\mu_{\beta_j} = 0$).

- This choice centers our prior beliefs around zero, which corresponds to the assumption of no effect of x_j on y.

- In this way, we express our prior doubts about the effect of x_j on y, prompting Spiegelhalter et al. (2004, pp. 90, 158-160) to call this a “sceptical” prior.

- The prior variance $\sigma^2_{\beta_j}$ of the effect β_j is set equal to a large value (e.g., 10^4) to represent high uncertainty or prior ignorance.

- A very popular prior for τ, especially among BUGS users, is to use equally low prior parameter values, e.g., $a = b = 0.1$, or $a = b = 0.01$. Both prior configurations lead to a peak in 0^+. Their use is controversial. For a good discussion on this topic, see

  `http://www.stat.columbia.edu/~gelman/research/published/taumain.pdf`

Regression

Linear regression: Bayesian formulation

- Each regression coefficient represents the effect of explanatory variable x_j on the expectation of the response variable y adjusted for the remaining covariates.
- To ascertain whether the effect of x_j is important for the prediction or description of y, we initially focus on examining whether the posterior distribution of β_j is scattered around zero (or not).
- Posterior distributions far away from the zero value indicate an important contribution of x_j to the prediction of the response variable.
- Although formal Bayesian hypothesis testing is not based on simply examining the posterior distribution and derived credible intervals, such an analysis offers a first and reliable tool for identifying important variables.

Regression

Linear regression: Bayesian formulation

- The magnitude of the effect of variable x_j on y is given by the posterior distribution of β_j ($j = 1, \ldots, p$) since

$$\begin{aligned}
&\mathbb{E}(y \mid x_1, \ldots, x_{j-1}, x_j = x+1, x_{j+1}, \ldots, x_p) - \mathbb{E}(y \mid x_1, \ldots, x_{j-1}, x_j = x, x_{j+1}, \ldots, x_p)\\
=&\beta_0 + \beta_j(x+1) + \sum_{k\neq j, k=1}^{p} \beta_k x_k - \beta_0 - \beta_j x - \sum_{k\neq j, k=1}^{p} \beta_k x_k\\
=&\beta_j
\end{aligned}$$

- Hence the posterior mean or median of β_j will correspond to the corresponding posterior measures of the expected change of the response variable y.

- An increase of one unit of x_j, given that the remaining covariates remain unchanged, induces an a posteriori average change on the expectation of y equal to the posterior mean of β_j.

Regression

Linear regression: Bayesian formulation

- The interpretation of the intercept parameter, β_0, is as the expected value of the response variable y when the observed values of all covariates are equal to zero.
- Frequently such a combination lies outside the range of the observed covariate values and so, in many cases, direct interpretation of β_0 does not lead to realistic and sensible interpretations.
- An alternative is to center around zero all explanatory variables x_j, by subtracting their sample mean. In this case, the constant β_0 represents the expected value of y when all covariates are equal to the sample means, representing in this way the expected response y for an “average” or “typical” subject within our sample.
- Note that this does not change the meaning of β_j, $j = 1, \ldots, p$.
- Centering (or centering and scaling) has some further advantages: it usually helps the mixing of the MCMC chains and also facilitates prior specification.

Regression

Linear regression

↪ Let us look at an example. The following dataset (the `mtcars` dataset available on `R`) gives the fuel consumption of cars, together with three aspects of car construction and performance.

	mpg	Rear axle ratio	Weight (lb/1000)	1/4 mile time
Mazda RX4	21.0	3.90	2.620	16.46
Mazda RX4 Wag	21.0	3.90	2.875	17.02
Datsun 710	22.8	3.85	2.320	18.61
Hornet 4 Drive	21.4	3.08	3.215	19.44
Hornet Sportabout	18.7	3.15	3.440	17.02
Valiant	18.1	2.76	3.460	20.22
Duster 360	14.3	3.21	3.570	15.84
⋮	⋮	⋮	⋮	⋮

↪ Based on these data we are interested in estimating the fuel consumption of a car based on the rear axle ratio, the weight, and the 1/4 mile time.

Regression

Linear regression

- We will fit a Bayesian multiple linear regression model to the `mtcars` dataset.
- The model we will fit is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \varepsilon_i, \quad \varepsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2), \quad i = 1, \dots 32,$$

  where

  - y_i is the miles driven per gallon for the i-th car.
  - x_{i1} is the rear axle ratio for the i-th car.
  - x_{i2} is the weight of the i-th car (lb/1000).
  - x_{i3} is the 1/4 mile time for the i-th car.

- We will consider $\beta_j \sim \text{N}(0, 1000)$ (variance of 1000 or precision of 0.001) and $\tau \sim \text{Gamma}(0.1, 0.1)$.

Regression

Linear regression

↪ We start by fitting a frequentist regression model (using ordinary least squares).

```
data(mtcars)
help(mtcars)

#creating a new dataframe with only mpg,drat,wt, and qsec as variables
vars=names(mtcars)%in%c("cyl", "disp", "hp", "vs","am","gear","carb") #variables to exclude
mtcars1=mtcars[!vars]

fit=lm(mpg~drat+wt+qsec,data=mtcars1)
summary(fit)

Call:
lm(formula = mpg ~ drat + wt + qsec, data = mtcars1)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1152 -1.8273 -0.2696  1.0502  5.5010

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  11.3945     8.0689   1.412  0.16892
drat          1.6561     1.2269   1.350  0.18789
wt           -4.3978     0.6781  -6.485 5.01e-07 ***
qsec          0.9462     0.2616   3.616  0.00116 **

Residual standard error: 2.56 on 28 degrees of freedom
Multiple R-squared:  0.837,Adjusted R-squared:  0.8196
F-statistic: 47.93 on 3 and 28 DF,  p-value: 3.723e-11
```

Regression

Linear regression

↪ We now implement the model using the package `rjags`. We need to specify the likelihood of our normal multiple linear regression model.

```
require(rjags)
n=nrow(mtcars1)
#covariates and response
y=mtcars1$mpg; drat=mtcars1$drat; wt=mtcars1$wt; qsec=mtcars1$qsec

model_string <- "model{

# Likelihood
for(i in 1:n){
y[i]~dnorm(mu[i],tau)
mu[i]=beta[1]+beta[2]*drat[i]+beta[3]*wt[i]+beta[4]*qsec[i]
}

# Prior for beta
for(j in 1:4){
beta[j]~dnorm(mu0,tau0)
}
tau0=1/sigma02

# Prior for the precision
tau~dgamma(a, b)

# Compute the variance
sigma2=1/tau
}"
```

Regression

Linear regression

↪ We will now pass all information needed and compile the model.

```
#hyperparameters for the betas and tau
mu0=0; sigma02=1000; a=0.1; b=0.1

# list with data and hyperparameters
data=list(y=y,drat=drat,wt=wt,qsec=qsec,n=n,mu0=mu0,sigma02=sigma02,a=a,b=b)

#Compile the model
model=jags.model(textConnection(model_string),n.chains=1,data=data)

Compiling model graph
   Resolving undeclared variables
   Allocating nodes
Graph information:
   Observed stochastic nodes: 32
   Unobserved stochastic nodes: 5
   Total graph size: 254

Initializing model
```

↪ Note that we have 5 unobserved stochastic nodes (=variables): β_0, β_1, β_2, β_3, and σ^2. The 32 observed stochastic nodes correspond to the 32 y's.

Regression

Linear regression

↪ We will do a burn-in of 100 000 iterations, run the model for 500 000 iterations, and do a thinning of 50.

```
update(model,100000,progress.bar="none")
res=coda.samples(model,variable.names=c("beta","sigma2"),n.iter=500000,thin=50,progress.bar=
summary(res)

Iterations = 100050:6e+05
Thinning interval = 50
Number of chains = 1
Sample size per chain = 10000

1. Empirical mean and standard deviation for each variable,
   plus standard error of the mean:
I
           Mean     SD Naive SE Time-series SE
beta[1] 10.3694 8.0796 0.080796       0.234179
beta[2]  1.7771 1.2427 0.012427       0.029145
beta[3] -4.3354 0.6834 0.006834       0.015359
beta[4]  0.9677 0.2675 0.002675       0.005713
sigma2   6.9785 1.9893 0.019893       0.020370

2. Quantiles for each variable:

           2.5%     25%     50%    75%  97.5%
beta[1] -5.0983  5.0167 10.3319 15.617 26.349
beta[2] -0.7206  0.9583  1.7916  2.622  4.166
beta[3] -5.7024 -4.7828 -4.3340 -3.885 -2.995
beta[4]  0.4487  0.7894  0.9673  1.149  1.493
sigma2   4.1601  5.5733  6.6201  8.016 11.729
```

Regression

Linear regression

- ↪ Note that the above results might slightly change when you run the model on your computer (stochastic nature of the MCMC).
- ↪ The chains for the regression coefficients were highly correlated, and that is why we have thinned them.
- ↪ The effective sample sizes for each parameter (β_0, β_1, β_2, β_3, and σ^2) are: 1190, 1817, 1979, 2192, and 9536, respectively.
- ↪ All convergence diagnostics seem ok!
- ↪ As an exercise, try centering and scaling each covariate, i.e., for each covariate subtract its mean and divide by its standard deviation. This can be done by redefining the covariates, e.g.,

  ```
  drat=(mtcars1$drat-mean(mtcars1$drat))/sqrt(var(mtcars1$drat));
  ```
- ↪ Look at autocorrelations again (no thinning). Does centring and scaling help increase the effective sample size?

Regression

Linear regression

$\hookrightarrow$ Under the frequentist analysis we have

Parameter	Point Estimate (95% confidence interval)
β_0	11.395 $(-5.134, 27.922)$
β_1	1.750 $(-0.857, 4.169)$
β_2	-4.347 $(-5.787, -3.009)$
β_3	0.946 $(0.410, 1.482)$
σ^2	6.554

$\hookrightarrow$ For the Bayesian version results are as follows

Parameter	Posterior mean (95% credible interval)
β_0	10.369 $(-5.098, 26.349)$
β_1	1.777 $(-0.721, 4.166)$
β_2	-4.335 $(-5.702, -2.995)$
β_3	0.968 $(0.449, 1.493)$
σ^2	6.978 $(4.160, 11.729)$

$\hookrightarrow$ Results are pretty similar! However, interpretation of the intervals are completely different.

Regression

Linear regression

- The regression model described earlier can be defined in JAGS using vectors and matrices instead of scalar nodes, by using the function `inprod`.

```
x=cbind(rep(1,n),drat,wt,qsec)

# Likelihood
for(i in 1:n){
y[i]~dnorm(mu[i],tau)
mu[i]=inprod(beta[],x[i,])
}

# Prior for beta
for(j in 1:4){
beta[j]~dnorm(mu0,tau0)
}
tau0=1/sigma02

# Prior for the inverse variance
tau~dgamma(a, b)

# Compute the variance
sigma2=1/tau
}"

mu0=0; sigma02=1000; a=0.1; b=0.1
data=list(y=y,x=x,n=n,mu0=mu0,sigma02=sigma02,a=a,b=b)
```

- Fitting the model in JAGS then proceeds as before.

Regression

Linear regression

- We can also use multivariate normal priors for the regression coefficients $\boldsymbol{\beta}_{p+1} \sim \mathrm{N}(\boldsymbol{\mu}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\beta}})$, using the function `dmnorm`.

- The prior we used before is a particular case of this more general prior, in which the off-diagonal elements are all zero).

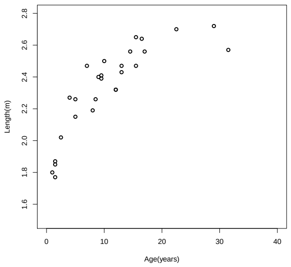
```
model_string <- "model{

# Likelihood
for(i in 1:n){
y[i]~dnorm(mu[i],inv.var)
mu[i]=inprod(beta[],x[i,])
}

beta~dmnorm(mu.beta,tau.beta)

# Prior for the inverse variance
inv.var~dgamma(a, b)

# Compute the variance
sigma2=1/inv.var
}"

#this coincides with the previous prior, as off diagonal entries are zero
mu.beta=rep(0,4); tau.beta=diag(0.0001,4); a=0.001; b=0.001
data=list(y=y,x=x,n=n,mu.beta=mu.beta,tau.beta=tau.beta,a=a,b=b)
```

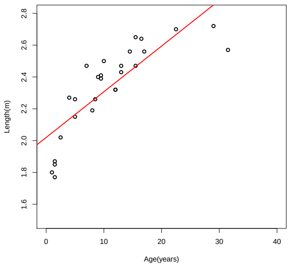
Regression

Nonlinear regression

$\hookrightarrow$ Another desirable extension is the ability to fit a nonlinear model.

$\hookrightarrow$ Carlin and Gelfand (1991) consider data on length (y_i) and age (x_i) measurements for $i = 1, \ldots, n = 27$ dugongs (sea cows) captured off the coast of Queensland.





$\hookrightarrow$ A linear fit to this dataset (shown on the right) iclearly does not fit the behaviour for young or old dugongs.

Regression

Nonlinear regression - Dugongs

Regression

Nonlinear regression

- The following nonlinear growth model has been suggested

$$y_i = \alpha - \beta\gamma^{x_i} + \varepsilon_i, \quad i = 1, \ldots, n,$$

  where $\alpha > 0$, $\beta > 0$, $0 \leq \gamma \leq 1$ and $\varepsilon_i \overset{\text{iid}}{\sim} \text{N}(0, \sigma^2)$.

- In this model, as explained in Carlin and Louis (2009, p.174), α corresponds to the average length of a fully grown dugong, $(\alpha - \beta)$ is the length of a dugong at birth, and γ determines the growth rate.

- Specifically, lower values of γ produce an initially steep growth curve (rapid progression to adulthood immediately after birth), while higher γ values lead to much more gradual, almost linear growth.

- As stated by these authors, flat priors are suitable for the two "endpoint" parameters α and β, but the harder-to-estimate growth parameter γ benefits from a tighter (albeit uniform) specification.

Regression

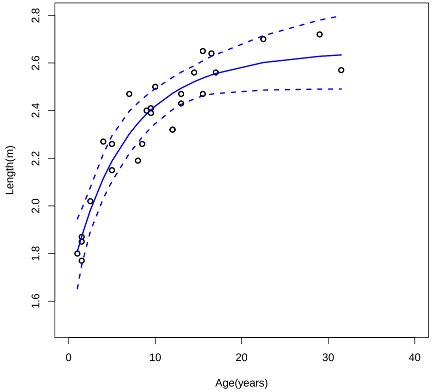
Nonlinear regression

↪ The code to specify the model is

```
model_string <- "model{
# Likelihood
for(i in 1:n){
y[i]~dnorm(mu[i],tau)
mu[i]=alpha-beta*pow(gamma,x[i])
}

alpha~dunif(0,10)

beta~dunif(0,10)

gamma~dunif(0.5,1)

# Prior for the inverse variance
tau~dgamma(a, b)

# Compute the variance
sigma2=1/tau
}"
```

Regression

Nonlinear regression



↪ **Additional reading**: see Sections 6.6.2-6.6.5 of Core Statistics by Simon Wood for more examples on JAGS.

Model checking

Linear regression: model checking

- The conclusions of any (Bayesian) analysis are conditional on the appropriateness of the assumed statistical model, so we need to be satisfied that our assumptions are a reasonable approximation to reality, even though we do not generally believe any model is actually "true" (Lunn et al. 2012).

- In defining the multiple linear regression model we have assumed

  1. **Normality**: the error terms $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ are normally distributed, with mean 0 and variance σ^2.
  2. **Independence**: the error terms $\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_n$ are independent of one another.
  3. **Constant variability**: the value of σ^2 does not depend on the values x_{ij} (this is known as homoskedasticity).
  4. **Linearity**: the mean μ_i is a linear function of $x_{i1}, \ldots, x_{ip}$.

- All of these assumptions can be (informally) checked using the Studentized residuals.

Model checking

Linear regression: model checking

- Studentized residuals are often used in frequentist analyses and are defined by

$$\widehat{\varepsilon}_i = \frac{y_i - \hat{y}_i}{\hat{\sigma}\sqrt{1 - h_{ii}}}, \qquad i = 1, \dots, n$$

  where $\hat{y}_i = \sum_{j=0}^{p} \widehat{\beta}_j x_{ij}$ and h_{ii} is the i-th diagonal entry of the 'hat' matrix $H = \mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T$.

- This is effectively a rescaling, as $\hat{\sigma}\sqrt{1 - h_{ii}}$ is an estimate of the standard deviation of the residual $y_i - \hat{y}_i$ ($\hat{y}_i$ depends on y_i, hence the correction term $\sqrt{1 - h_{ii}}$).

- Under the assumption of normality of the error terms ε_i, we have the approximation

$$\{\widehat{\varepsilon}_i\} \sim \mathrm{N}(0, 1)$$

  Note: this is an approximation. Although the errors $\{\varepsilon_i\}$ should be independent under our model assumptions, the Studentized residuals $\{\widehat{\varepsilon}_i\}$ are not necessarily independent, so tests of Normality are not applied rigorously - it is usually enough to inspect graphs, etc.

Model checking

Linear regression: model checking

- $\hookrightarrow$ In the Bayesian context, the point estimates $\hat{\beta}_j$ and $\hat{\sigma}$ are replaced by distributions so the Studentized residuals are not single-valued. Lunn et al. (2012, p.141) suggest

  "If we want to create a single-valued residual rather than a random quantity, alternative possibilities include using a single draw $\theta^{(j)} = (\boldsymbol{\beta}^{(j)}, \sigma^{(j)})$, plugging in the posterior means of θ, or using the posterior mean residual."

- $\hookrightarrow$ Here, we opt for the last option (the posterior mean of the residual), that is, for each draw $(\boldsymbol{\beta}^{(j)}, \sigma^{(j)})$ from the posterior distribution, we compute the Studentized residuals and from this ensemble the posterior mean (or median if preferred) is computed.

- $\hookrightarrow$ The same option is taken by Wang et al. (Bayesian Regression Modeling with INLA, 2018, CRC/Chapman & Hall).

Model checking

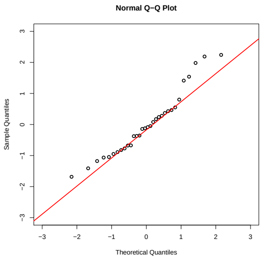
Linear regression: model checking

- $\hookrightarrow$ To assess the normality assumption, a Q-Q plot of the Bayesian Studentized residuals must be constructed.
- $\hookrightarrow$ To assess the independence of the error terms, a plot of the Bayesian Studentized residual against the index, i, that labels the order in which the observations were gathered, can be used. Any pattern suggests that the errors $\{\varepsilon_i\}_{i=1}^n$ are not independent.
- $\hookrightarrow$ To check the assumptions of linearity and constant variance we can plot the Bayesian Studentized residuals against the posterior mean (or median) of the fitted values. This graph should look random, with no obvious patterns, if these two assumptions are valid.
- $\hookrightarrow$ Interpreting these plots is, of course, very subjective...

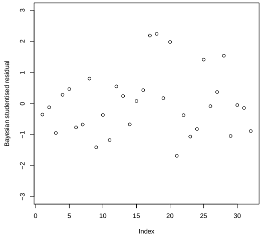
Model checking

Linear regression: model checking

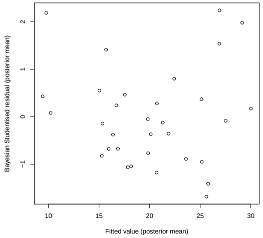
↪ For the mtcars datastet, the following figures were produced



(a)



(b)



(c)

↪ Figure (a) shows a normal Q-Q plot, checking **Normality** of the studentized residuals. Figure (b) plots the studentized residuals versus their index, showing a random behaviour with mean close to 0, meaning that the **Linearity** and **Independence** assumptions seem to hold. Figure (c) plots the posterior mean of the fitted value (of the response variable) versus the posterior mean of the studentized residual. Residuals do not show clear dependence on the fitted value, justifying **Constant variability** assumption.

Model checking

Linear regression: model checking

- Predictive checks are also very popular in Bayesian statistics. Basically, for an overall assessment of model fit we can generate replicate data sets (of the same size as the original data) from the posterior predictive distribution, and compare to the real data using specific test quantities (Gelman et al. 2013, chapter 6).

- Let $\mathbf{y}_{\text{rep}}$ denotes the replicated datasets. The distribution of $\mathbf{y}_{\text{rep}}$ is the posterior predictive distribution
$$p(\mathbf{y}_{\text{rep}} \mid \mathbf{y}) = \int p(\mathbf{y}_{\text{rep}} \mid \theta) p(\theta \mid \mathbf{y}) d\theta.$$

- We do not usually evaluate this integral but instead draw samples from $p(\mathbf{y}_{\text{rep}} \mid \mathbf{y})$:

  1. Draw $\theta^{(j)}$ from $p(\theta \mid \mathbf{y})$, $j = 1, \ldots, M$.
  2. Draw $\mathbf{y}_{\text{rep}}^{(j)} \sim p(\mathbf{y}_{\text{rep}} \mid \theta^{(j)})$, $j = 1, \ldots, M$,

  where M is the number of MCMC iterations.

- The chosen test statistics are then computed for each replicated dataset and compared to the one derived from the observed data.

Model checking

Linear regression: model checking

$\hookrightarrow$ We distinguish between the replicated data $\mathbf{y}_{\text{rep}}$ and the predictive outcomes $\tilde{\mathbf{y}}$.

$\hookrightarrow$ The variable $\tilde{\mathbf{y}}$ is any *future* observable value of the outcome. For example, in a linear regression model $\tilde{\mathbf{y}}$ can have its own set of explanatory variables $\tilde{\mathbf{x}}$.

$\hookrightarrow$ On the other hand, $\mathbf{y}_{\text{rep}}$ must have the same explanatory variables $\mathbf{x}$ as those used in the model for the observed data $\mathbf{y}$. In this sense, $\mathbf{y}_{\text{rep}}$ is similar to "predicting the observed data".

Model checking

Linear regression: model checking

↪ A very good and concise discussion about this topic can be found in the paper

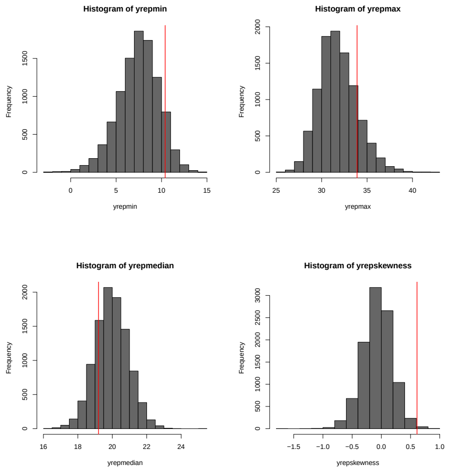
`https://arxiv.org/pdf/1709.01449.pdf`

↪ As the authors of the aforementioned paper state: "*Posterior predictive checking makes use of the data twice, once for the fitting and once for the checking. Therefore it is a good idea to choose statistics that are orthogonal to the model parameters. If the test statistic is related to one of the model parameters, e.g., if the mean statistic is used for a Gaussian model with a location parameter, the posterior predictive checks may be less able to detect conflicts between the data and the model.*"

↪ For the `mtcars` example, since we are using a location-scale normal model, let us pick up as test statistics the minimum, the maximum, the median, and the skewness.

Model checking

Linear regression: model checking



$\hookrightarrow$ The solid red vertical lines indicate the corresponding statistic computed from the observed data.

Model checking

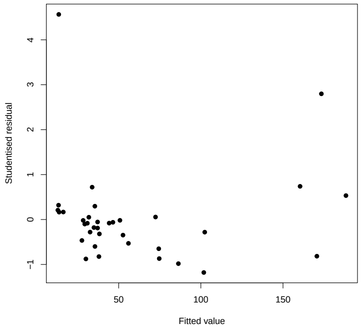
Robustifying linear regression

- Let us now consider robust regression. The normality assumption of the error terms renders the model quite sensitive to outliers, so we now consider modifications to that assumption.
- A classic data set in the literature pertains to hill racing (a somewhat popular sport here in Scotland).
- The data set `hills.txt` contains information on the winning times (in minutes) in 1984 for 35 Scottish hill races, as well as two factors which presumably influence the duration of the race:
  - dist: The distance of the race (in miles).
  - climb: The elevation change (in feet).

Model checking

Robustifying linear regression

$\hookrightarrow$ Fitting a simple linear regression model for time assuming additive linear relationships for dist and climb, we obtain the following residuals



$\hookrightarrow$ The point at the top left corner, seems suspicious

Model checking

Robustifying linear regression

- $\hookrightarrow$ To reduce the impact of the outlier on the fit of the model, we can replace the normal distribution with a fat-tailed distribution.
- $\hookrightarrow$ A natural choice is the *t*-distribution, which can be implemented by simply replacing the normal likelihood with

```
y[i]~dt(mu[i],tau,nu)
```

- $\hookrightarrow$ Recall that as $\nu \to \infty$, the t-distribution resembles the normal, but for small ν has considerably fatter tails.
- $\hookrightarrow$ Let us start by trying $\nu = 5$.

Model checking
Robustifying linear regression

↪ The code for defining this model is

```
hills=read.table("hills.txt",header=TRUE)
y=hills$time; climb=hills$climb; dist=hills$dist

model_string <- "model{

# Likelihood
for(i in 1:n){
y[i]~dt(mu[i],tau,5)
mu[i]=beta[1]+beta[2]*climb[i]+beta[3]*dist[i]
}

# Prior for beta
for(j in 1:3){
beta[j]~dnorm(mu0,tau0)
}
tau0=1/sigma02

# Prior for the inverse variance
tau~dgamma(a, b)

# Compute the variance
sigma2=1/tau
}"
```

Model checking

Robustifying linear regression

- Contrasting the results from a normal model versus a t_5 model (posterior mean and standard deviation)

	Climb	Dist
Normal	0.011 (0.0021)	6.204 (0.6150)
t_5	0.0081 (0.0015)	6.565 (0.3089)

- There is a modest change in the posterior means (the posterior mean for distance goes up, while the one for climb goes down), and a sizeable drop (roughly 2-fold) in the posterior standard deviation.

Model checking

Robustifying linear regression

- $\hookrightarrow$ Of course, one may ask, why a t_5 distribution?
- $\hookrightarrow$ Since we do not actually know ν, it would be more reasonable to include ν as a parameter in our model; the only condition is that we must place a prior on it.
- $\hookrightarrow$ There are several possible options. For instance we could use a similar prior as the one used for the precision.

Model checking

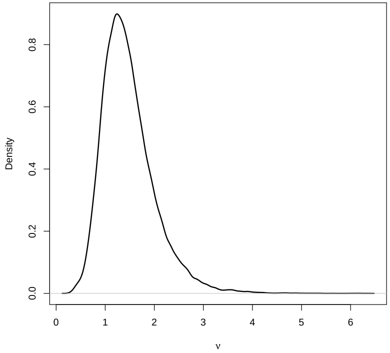
Robustifying linear regression

↪ The modified model definition is now

```
model_string <- "model{
# Likelihood
for(i in 1:n){
y[i]~dt(mu[i],tau,nu)
mu[i]=beta[1]+beta[2]*climb[i]+beta[3]*dist[i]
}

# Prior for beta
for(j in 1:3){
beta[j]~dnorm(mu0,tau0)
}
tau0=1/sigma02

# Prior for the inverse variance
tau~dgamma(a, b)

# Compute the variance
sigma2=1/tau

#Prior for nu
nu~dgamma(c,d)
}"

mu0=0; sigma02=1000; a=0.1; b=0.1; c=0.1; d=0.1
data=list(y=y,climb=climb,dist=dist,n=n,mu0=mu0,sigma02=sigma02,a=a,b=b,c=c,d=d)
```

Model checking

Robustifying linear regression



- ↪ The plot on the left shows the posterior density of the degrees of freedom parameter ν.
- ↪ The posterior mean (standard deviation) for regression coefficient of climb is now 0.0069 (0.0011).
- ↪ The posterior mean (standard deviation) for regression coefficient of dist is now 6.543 (0.300).

↪ **Additional reading**: see Section 7.2 of Core Statistics by Simon Wood and Sections 6-7 of Bayesian Data Analysis by Gelman et al. for various approaches to model checking and model comparison.