Bayesian Analysis for Nonlinear Regression Models: a leaf growth curve

Bendetta Candelori

Contents

1 Purpose of the project

The main purpose of this project is to show how Bayesian inference could be exploited to fit non-linear models for growth curve data.

Three widely used models are used to fit growth curve data: Gompertz, Richards and Logistic models which use few parameters having a biological interpretation to describe the entire growth process[Archontoulis et al.[1]].

This study aims to estimate these parameters both with the classical and with the Bayesian approach and compare their results.

On one side, the Frequentist approach uses the nonlinear least squares (NLS) method, while on the other side, estimation of parameters with a Bayesian approach exploits the Markov Chain Monte Carlo (MCMC) methods.

2 Dataset

The dataset for this analysis is available in one of the R library called `NRAIA`. Thus, firstly we need to install the package and then load the library.

```r
library(NRAIA)
```

```
## Loading required package: lattice
```

```
## Registered S3 method overwritten by 'NRAIA':
##   method           from
##   plot.profile.nls stats
```

Our dataset (called `Leaves`) contains information about the growth of the leaves over time and contains 15 rows and 2 variables:

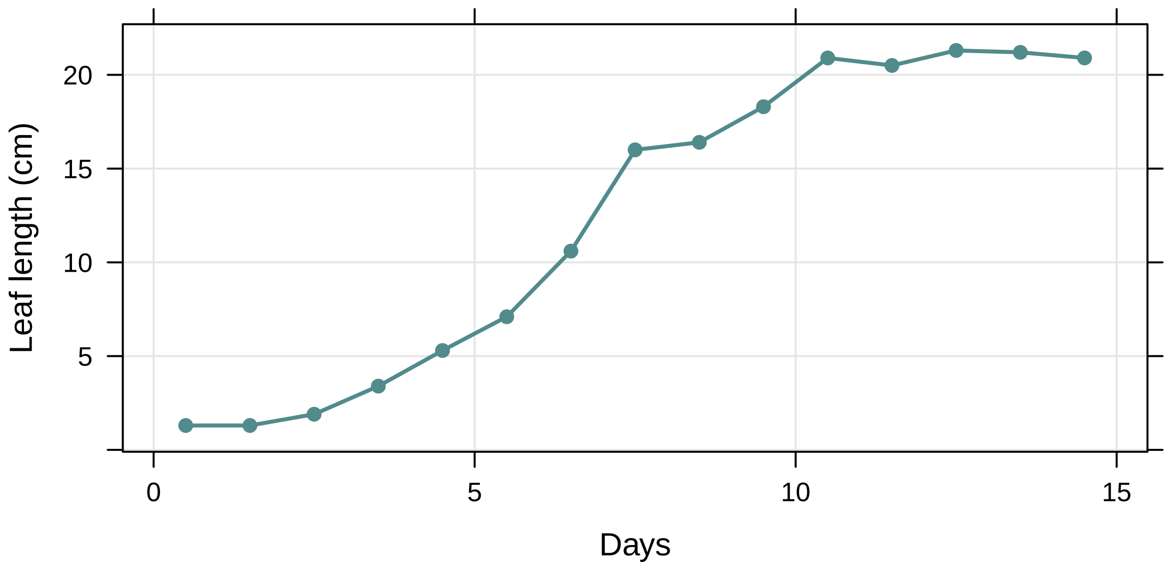
- Time: time from initial emergence (days);
- Length: leaf length (cm).

Here an overview of the data:

```
##   Time Length
## 1  0.5    1.3
## 2  1.5    1.3
## 3  2.5    1.9
## 4  3.5    3.4
## 5  4.5    5.3
```

The following plot shows the variation of the length of leaves overtime. In addition, we can clearly observe from that the growth curve is not linear: firstly (in the early days) the length of the curve is small and increase quite rapidly after an *inflation* point, then the growth of the leaves slows down again in the last days.

[1] Archontoulis, S.V. & Miguez, Fernando. (2013). Nonlinear Regression Models and Applications in Agricultural Research. Agronomy Journal. 105. 1. 10.2134/agronj2012.0506.



3 Model definition

A growth model for a single variable can be formalized as:

$$y_i = f(t_i, \theta) + \epsilon_i, \qquad i = 1, \dots, n$$

where y_i is the observed length n is total number of observations; θ is the vector of unknown parameters and t_i is the time at which the *i-th* observation was taken.

Moreover, $\epsilon_i \sim N(0, \sigma^2)$ is independent random error of y_i and $f(t_i, \theta)$ is characteristic for each model.

1. Gompertz model:

$$f_1(t_i, \theta_1) = y_{asym} \exp\{-\exp[-k(t_i - t_m)]\}$$

where $\theta_1 = (y_{asym}, k, t_m)$

2. Richards model:

$$f_2(t_i, \theta_2) = \frac{y_{asym}}{\Big(1 + v \cdot \exp[-k(t_i - t_m)]\Big)^{1/v}}$$

where $\theta_1 = (y_{asym}, k, t_m, v)$

3. Logistic model:

$$f_3(t_i, \theta_3) = \frac{y_{asym}}{1 + \exp[-k(t_i - t_m)]}$$

where $\theta_3 = (y_{asym}, k, t_m)$

In the above models, y_{asym} represents the asymptotic leaf length, k controls the steepness of the curve, t_m is the inflection point at which the growth rate is maximized and v deals with the asymmetric growth[Archontoulis et al.[2]].

[2] Archontoulis, S.V. & Miguez, Fernando. (2013). Nonlinear Regression Models and Applications in Agricultural Research. Agronomy Journal. 105. 1. 10.2134/agronj2012.0506.

Therefore we assume that:

$$Y_i \mid \theta_k, t_i \sim N(\mu_k, \sigma^2)$$

with $\mu_k = f_k(t_i, \theta_k)$, $k = 1, 2, 3$.

As a consequence, the likelihood function will be:

$$L(y \mid \theta_k, t, \sigma^2) = \frac{1}{(2\pi)^{n/2}\sigma^n} \exp\left[-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \Big(y_i - f_k(t_i, \theta_k) \Big)^2 \right]$$

4 Frequentist approach

Now we are interested to estimate parameters of the models above with the classical approach using NLS methods.

We start with logistic model: we are going to use a `selfStart` model which evaluates the logistic function and its gradient in an efficient way.

Note that the `scal` parameter (implemented in the `R` function) is the inverse of our parameter k.

```
logistic_fit =  nls(Length ~ SSlogis(Time, Asym, tm, scal), data = Leaves)
summary(logistic_fit)
```

```
##
## Formula: Length ~ SSlogis(Time, Asym, tm, scal)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## Asym  21.5089     0.4154   51.78 1.77e-15 ***
## tm     6.3604     0.1388   45.81 7.63e-15 ***
## scal   1.6072     0.1152   13.95 8.89e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7194 on 12 degrees of freedom
##
## Number of iterations to convergence: 0
## Achieved convergence tolerance: 4.384e-06
```

We are going to exploit the logistic starting parameters (implemented due to `SSlogis` function) in the remaining models.

Then, we fit Richard model:

```
richard_fit = nls(Length ~ Asym/(1+exp(-(Time - tm)/scal))^exp(-v), data = Leaves,
                  start = c(coef(logistic_fit),c(v = 0)))
summary(richard_fit)
```

```
##
## Formula: Length ~ Asym/(1 + exp(-(Time - tm)/scal))^exp(-v)
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## Asym  21.2040     0.4436  47.800 4.12e-14 ***
## tm     7.3234     0.7298  10.035 7.14e-07 ***
## scal   1.2866     0.2953   4.357  0.00114 **
```

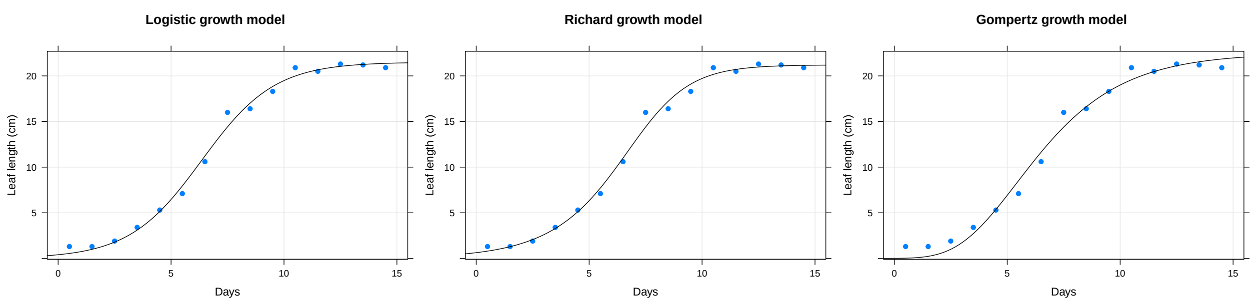
```
## v      0.4817     0.4110   1.172  0.26596
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7086 on 11 degrees of freedom
##
## Number of iterations to convergence: 9
## Achieved convergence tolerance: 2.872e-06
```

Finally, Gompertz model:

```
gompertz_fit = nls(Length ~ Asym*exp(-exp(-(Time - tm)/scal)), data = Leaves,
                   start = coef(logistic_fit))
summary(gompertz_fit)
```

```
##
## Formula: Length ~ Asym * exp(-exp(-(Time - tm)/scal))
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## Asym  22.5066     0.8372  26.882 4.32e-12 ***
## tm     5.4268     0.1944  27.917 2.76e-12 ***
## scal   2.5765     0.3047   8.455 2.12e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.024 on 12 degrees of freedom
##
## Number of iterations to convergence: 10
## Achieved convergence tolerance: 3.97e-06
```

Here a summary plots that show how the curves with those estimated parameters fit the data points.



We may use the AIC and BIC[3] criterion for the models' evaluation:

	df	AIC	BIC
Logistic	4	37.34040	40.17260
Richard	5	37.58090	41.12116
Gompertz	4	47.94153	50.77373

As we can see from the outputs above all the models leads to similar results: residual standard errors are low (about 0.70) for logistic and for Richard model, while they are over 1 for the last model. All parameters are statistically significant except for parameter v in Richard model (it has a high standard error).

[3] AIC: Akaike information criterion; BIC: Bayesian information criterion.

Moreover also from the plots we can see that all the models seem to fit well the data points, however it is possible to see how the fitted curve obtained with Gompertz model is worse than the others specifically for leaf length in the initial days.

To sum up, we may conclude to prefer Logistic and Richard models and also AIC and BIC criteria confirm that.

Table 2: NLS estimation results for the three growth functions

Model	Parameter	Estimate	Standard Error (SE)	Residual standard error	AIC	BIC
Logistic	Asym	21.5089	0.4154	0.7194	37.34040	40.17260
	tm	6.3604	0.1388			
	scal	1.6072	0.1152			
Richard	Asym	21.2040	0.4436	0.7086	37.58090	41.12116
	tm	7.3234	0.7298			
	scal	1.2866	0.2953			
	v	0.4817	0.4110			
Gompertz	Asym	22.5066	0.8372	1.024	47.94153	50.77373
	tm	5.4268	0.1944			
	scal	2.5765	0.3047			

5 Bayesian approach

In order to develop a Bayesian analysis and make inference for our parameters, we need to firstly define some main ingredients.

We need to define and propose prior distributions for our parameters y_{asym}, k, t_m and, if necessary, v and choose respectively hyperparameters for their distribution.

The following non-informative prior distributions are proposed:

- $y_{asym} \sim N(0, 1000) \cdot \mathbb{I}(1, \infty)$
- $k \sim Unif(0, 1)$
- $t_m \sim N(0, 1000) \cdot \mathbb{I}(1, \infty)$
- $v \sim Unif(0, 1)$

Lastly, for the error variances σ^2 we may choose an inverse gamma distribution: $\sigma^2 \sim InvGamma(0.01, 0.01)$

Now we are ready to proceed with the implentation of Gibbs sampling algorithm using `JAGS` tool in order to estimate our parameters for each model.

We have to prepare data for jags:

```
library(R2jags)
N = nrow(Leaves)
data4jags = list('Y'=Leaves$Length, 'N'=N, 't'=Leaves$Time)
```

5.1 Logistic model

Model definition:

```
cat('model {
  for( i in 1:N ) {
  Y[i] ~ dnorm(mu[i], precision)
  mu[i] <- Asym / (1+exp(-k*(t[i]-tm)))
  }
  Asym ~ dnorm(0.0, 1.0E-3)I(1.0,)
    tm ~ dnorm(0.0, 1.0E-3)I(1.0,)
    k ~ dbeta(1.0, 1.0)
    precision ~ dgamma(0.01, 0.01)

    sigma <- 1 / sqrt(precision)

}',file = 'logistic_model.txt')
```

```
inits1=list(Asym = 10, tm = 10, k = .1)
inits2=list(Asym = 1, tm = 1, k = .5)
inits=list(inits1,inits2)
param = c("Asym","tm","k","sigma")
```

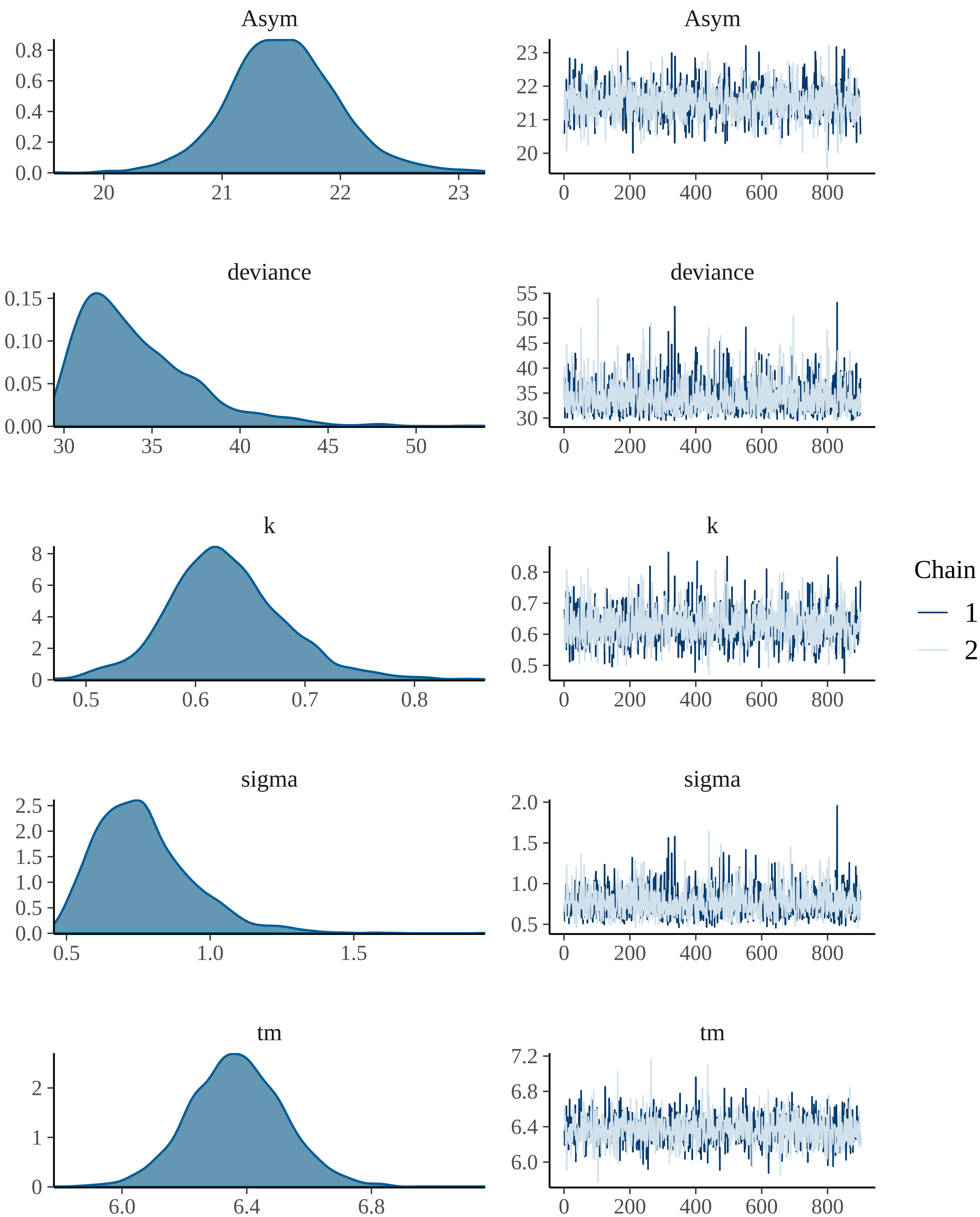
```
logistic_jags = jags(data=data4jags, parameters.to.save=param,
                     model.file='logistic_model.txt', n.chains=2,
                     n.iter=10000,n.burnin = 1000,n.thin = 10)
```

```
logistic_jags
```

```
## Inference for Bugs model at "logistic_model.txt", fit using jags,
##  2 chains, each with 10000 iterations (first 1000 discarded), n.thin = 10
##  n.sims = 1800 iterations saved
##          mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
## Asym      21.522   0.470 20.602 21.221 21.507 21.814 22.518 1.002  1400
## k          0.628   0.053  0.526  0.593  0.624  0.659  0.746 1.003   630
## sigma      0.769   0.171  0.517  0.647  0.747  0.859  1.179 1.001  1800
## tm         6.369   0.154  6.072  6.266  6.367  6.468  6.678 1.001  1800
## deviance  34.133   3.376 29.937 31.647 33.318 35.792 42.808 1.000  1800
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 5.7 and DIC = 39.8
## DIC is an estimate of expected predictive error (lower deviance is better).
```

Below, it is reported a summary of the parameter chains (traceplots on the right column) and the density plots of their distribution.

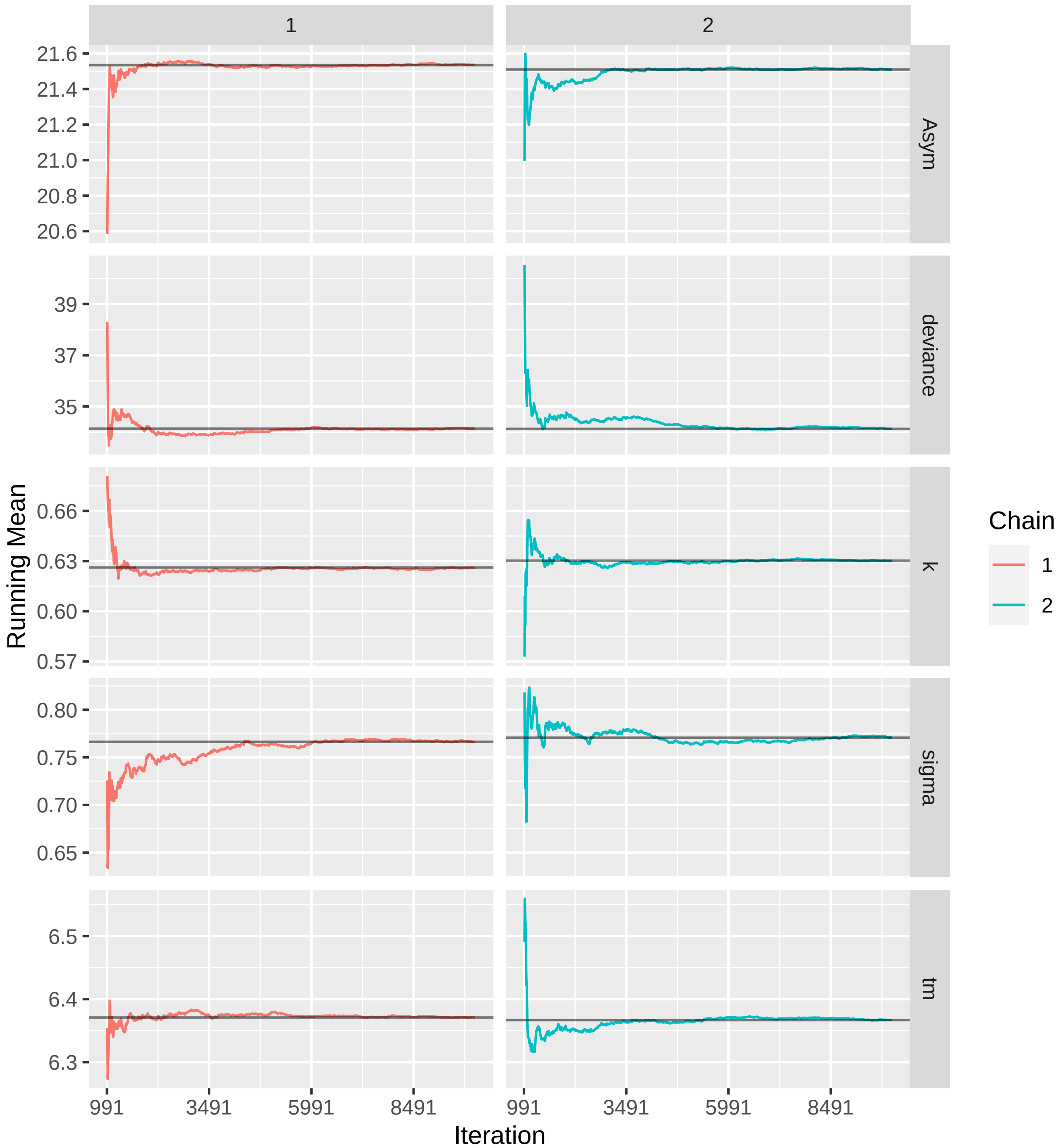
```r
chain = logistic_jags$BUGSoutput$sims.array
mcmc_combo(chain)
```



Running means

With the plot below, we are able to see the behavior of the average for each parameter through iterations. We can observe that all the estimated means converge after a large number of iteration, no matter of the starting point of the Gibbs sampler.

```
coda_logistic = as.mcmc(logistic_jags)
ggs_chain = ggs(coda_logistic)
ggs_running(ggs_chain)
```



5.2 Richard model

Model definition:

```
cat('model {
  for( i in 1:N ) {
  Y[i] ~ dnorm(mu[i], precision)
  mu[i] <- Asym / pow(1+v*exp(-k*(t[i]-tm)), (1/v))
  }
  Asym ~ dnorm(0.0, 1.0E-3)I(1.0,)
    tm ~ dnorm(0.0, 1.0E-3)I(1.0,)
    v ~ dbeta(1.0, 1.0)
    k ~ dbeta(1.0, 1.0)
    precision ~ dgamma(0.01, 0.01)

    sigma <- 1 / sqrt(precision)

}',file = 'richard_model.txt')
```

```
inits1=list(Asym = 10, tm = 10, k = .1, v = .1)
inits2=list(Asym = 1, tm = 1, k = .5, v = .5)
inits=list(inits1,inits2)
param = c("Asym","tm","k","sigma","v")
```

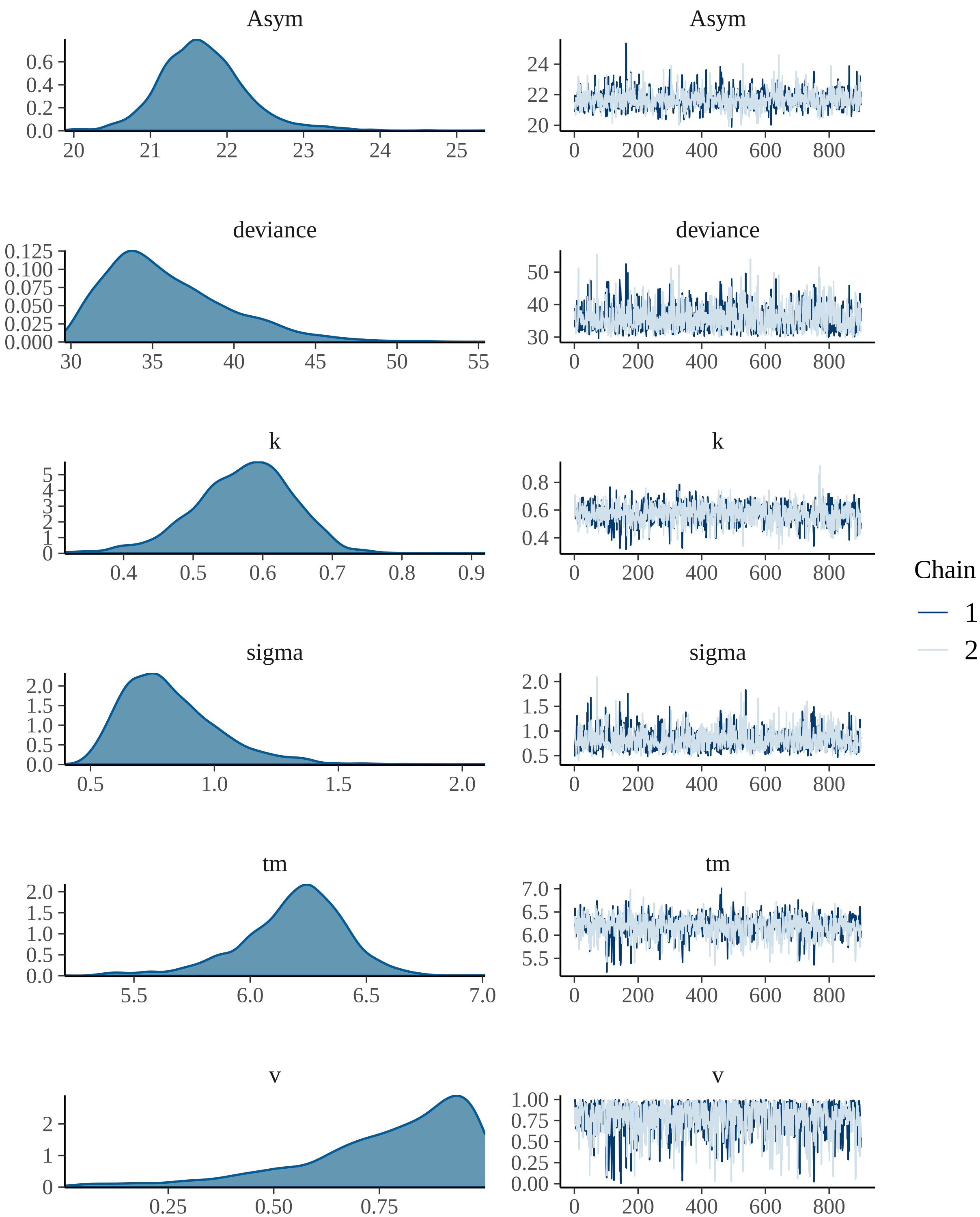
```
richard_jags = jags(data=data4jags, parameters.to.save=param,
                    model.file='richard_model.txt', n.chains=2,
                    n.iter=10000,n.burnin = 1000,n.thin = 10)
```

```
richard_jags
```

```
## Inference for Bugs model at "richard_model.txt", fit using jags,
##  2 chains, each with 10000 iterations (first 1000 discarded), n.thin = 10
##  n.sims = 1800 iterations saved
##          mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
## Asym      21.693   0.576 20.674 21.319 21.648 22.001 23.024 1.000  1800
## k          0.573   0.071  0.415  0.528  0.578  0.622  0.694 1.001  1800
## sigma      0.823   0.199  0.539  0.678  0.785  0.929  1.314 1.002  1300
## tm         6.190   0.226  5.663  6.071  6.217  6.335  6.580 1.003   570
## v          0.769   0.199  0.245  0.671  0.820  0.923  0.994 1.002  1300
## deviance  35.940   3.846 30.621 33.115 35.156 38.095 45.121 1.002   920
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 7.4 and DIC = 43.3
## DIC is an estimate of expected predictive error (lower deviance is better).
```

In the following plots there are shown density plots of the parameter distributions (on the left) and the respective traceplots (on the right).

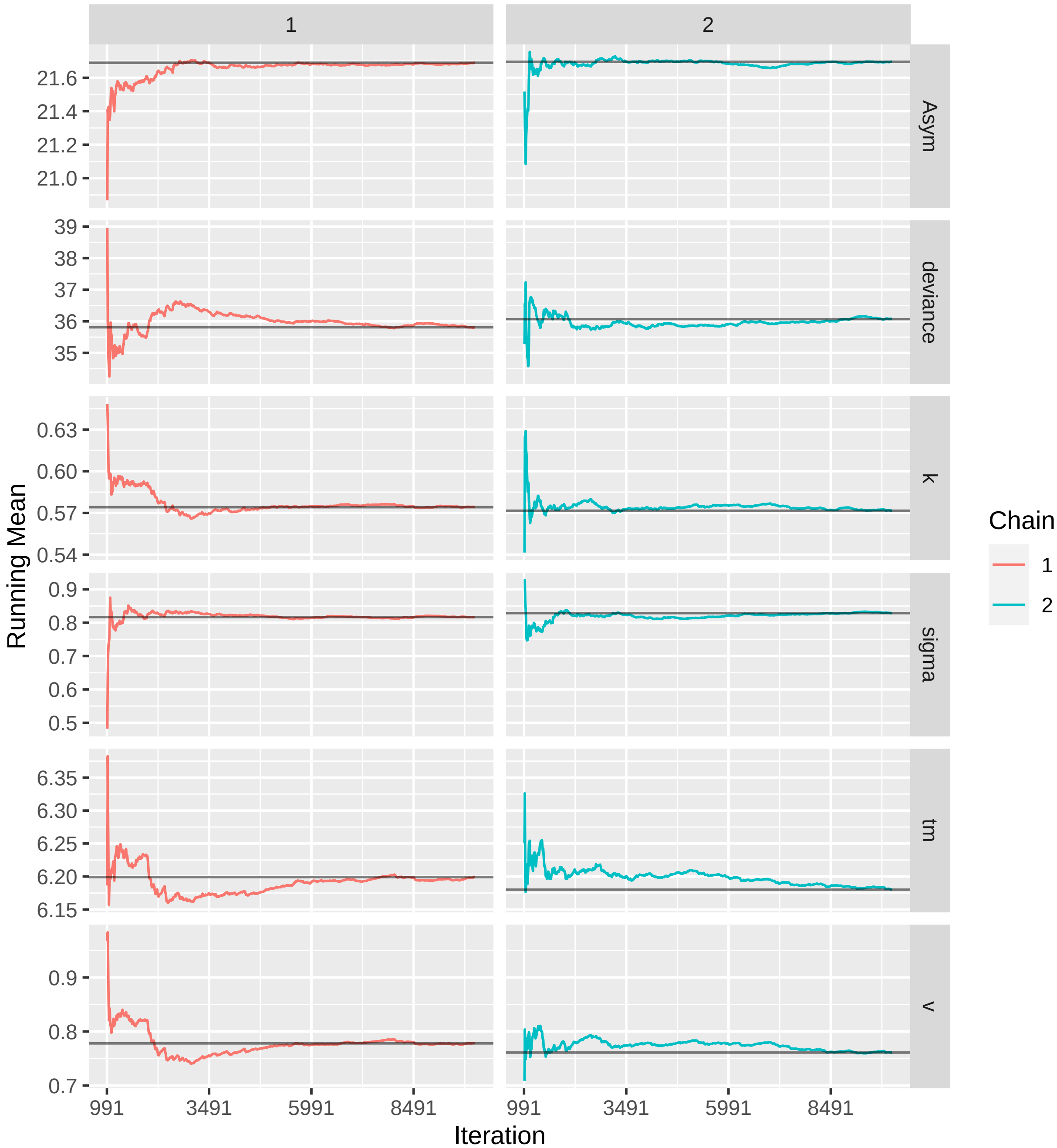
```
chain = richard_jags$BUGSoutput$sims.array
mcmc_combo(chain)
```



Running means

Here we can see the behavior of the running means and from that we can state that all the estimated means converge after a large number of iteration, independently from the starting point of the Gibbs algorithm.

```
coda_richard = as.mcmc(richard_jags)
ggs_chain = ggs(coda_richard)
ggs_running(ggs_chain)
```



5.3 Gompertz model

Model definition:

```
cat('model {
  for( i in 1:N ) {
  Y[i] ~ dnorm(mu[i], precision)
  mu[i] <- Asym * exp(-exp(-k*(t[i]-tm)))
  }
  Asym ~ dnorm(0.0, 1.0E-3)I(1.0,)
    tm ~ dnorm(0.0, 1.0E-3)I(1.0,)
    k ~ dbeta(1.0, 1.0)
    precision ~ dgamma(0.01, 0.01)

    sigma <- 1 / sqrt(precision)

}',file = 'gompertz_model.txt')
```

```
inits1=list(Asym = 10, tm = 10, k = .1)
inits2=list(Asym = 1, tm = 1, k = .5)
inits=list(inits1,inits2)
param = c("Asym","tm","k","sigma")
```

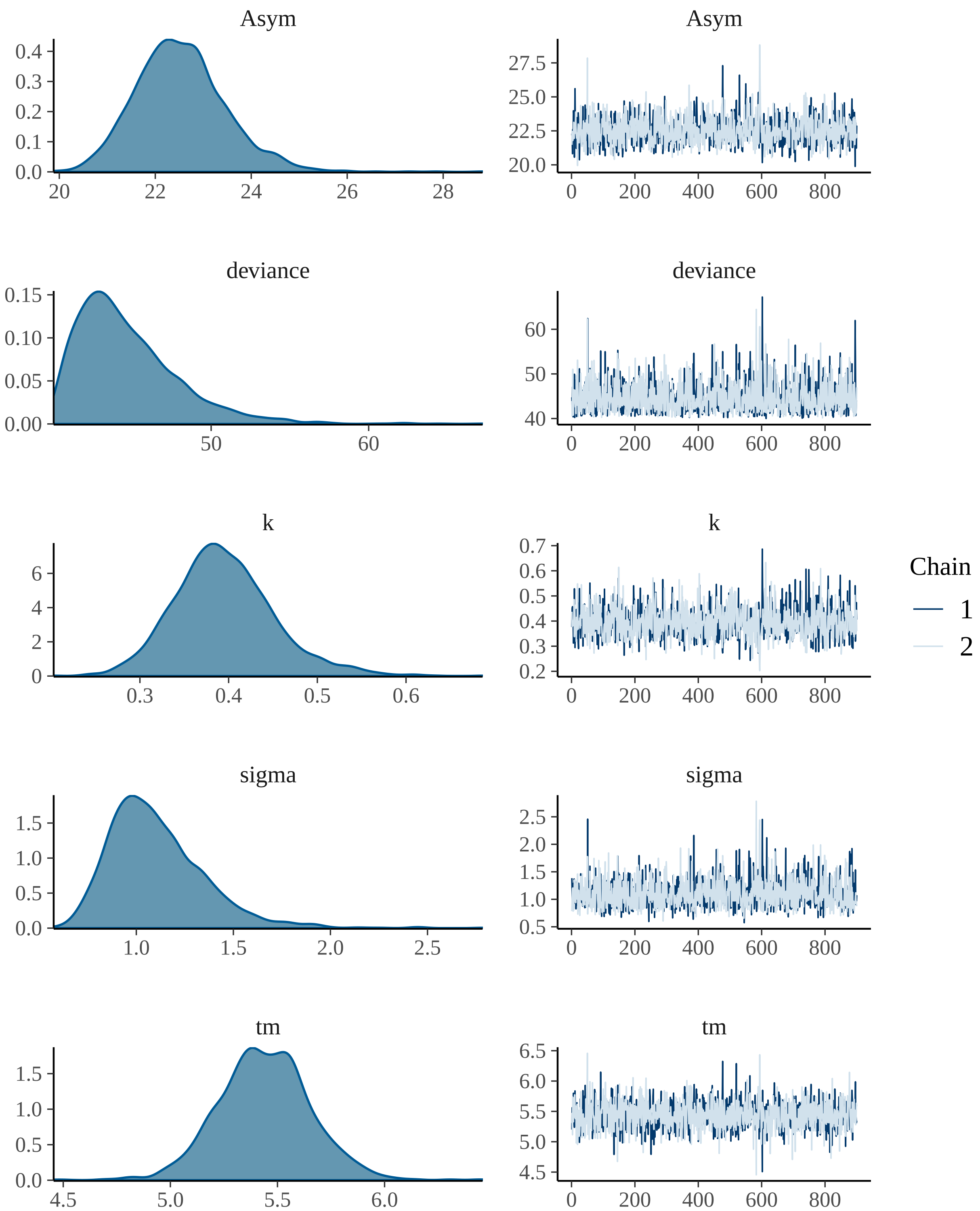
```
gompertz_jags= jags(data=data4jags, parameters.to.save=param,
                    model.file='gompertz_model.txt', n.chains=2,
                    n.iter=10000,n.burnin = 1000,n.thin = 10)
```

```
gompertz_jags
```

```
## Inference for Bugs model at "gompertz_model.txt", fit using jags,
##  2 chains, each with 10000 iterations (first 1000 discarded), n.thin = 10
##  n.sims = 1800 iterations saved
##          mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
## Asym      22.536   0.944 20.854 21.898 22.478 23.069 24.536 1.001  1800
## k          0.395   0.057  0.294  0.359  0.391  0.428  0.527 1.001  1800
## sigma      1.104   0.246  0.742  0.931  1.065  1.231  1.682 1.001  1800
## tm         5.440   0.217  5.025  5.302  5.438  5.575  5.867 1.001  1800
## deviance  44.692   3.266 40.564 42.341 43.970 46.301 52.778 1.001  1800
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 5.3 and DIC = 50.0
## DIC is an estimate of expected predictive error (lower deviance is better).
```

There is reported a summary of the parameter chains (traceplots on the right column) and the density plots of their distribution.

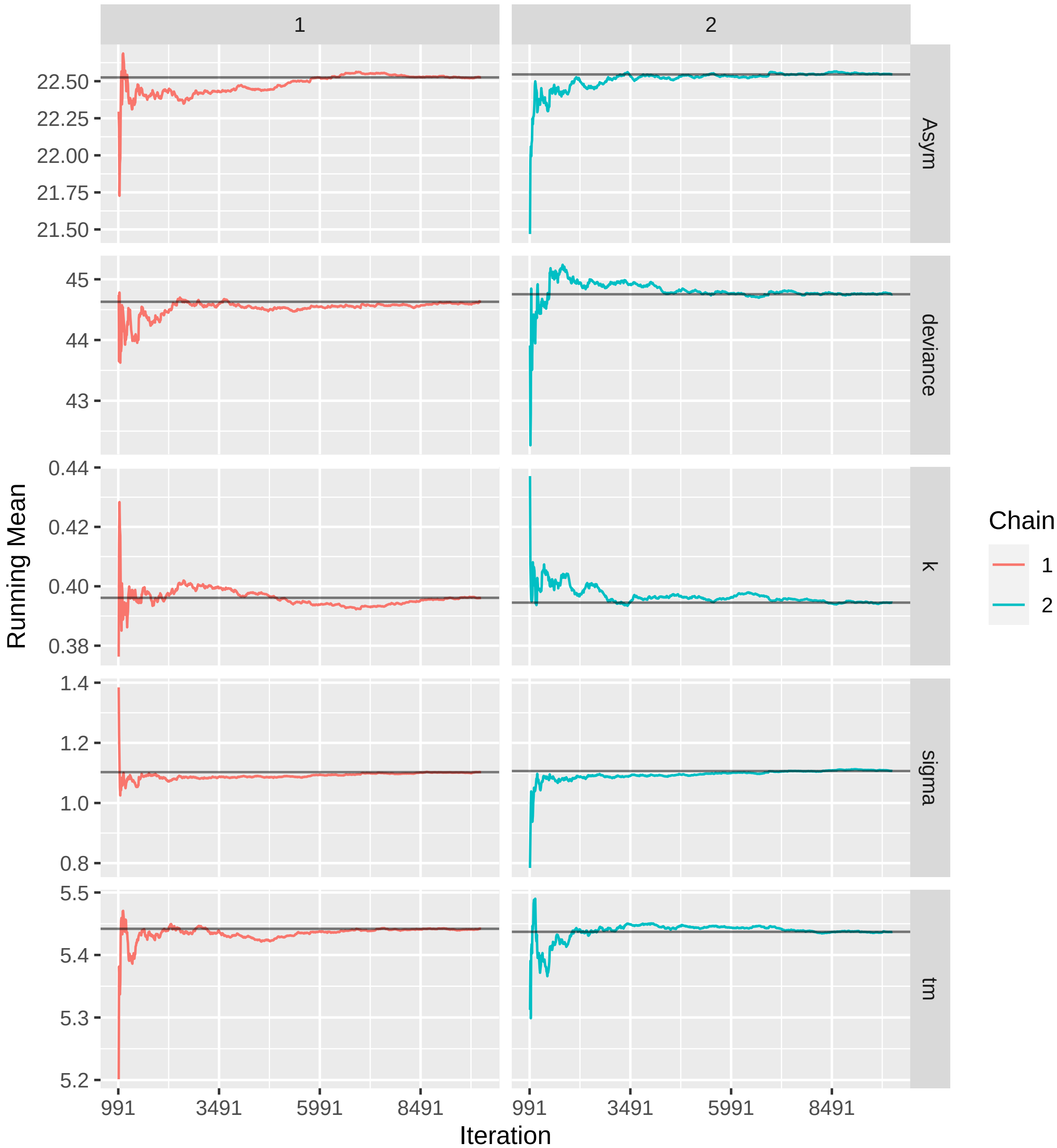
```
chain = gompertz_jags$BUGSoutput$sims.array
mcmc_combo(chain)
```



Running means

Here running means are shown below and from that we can observe that all the estimated means converge after a large number of iteration, whatever was the starting point of the Gibbs sampler.

```
coda_gompertz = as.mcmc(gompertz_jags)
ggs_chain = ggs(coda_gompertz)
ggs_running(ggs_chain)
```



5.4 Approximation error

In order to evaluate the accuracy of the estimates we need to quantify the approximation error for each estimated parameter. To do that we may compute the variance of $\hat{I}_t$, taking into account the dependence structure of MCMC.

From the theory it can be derived that the variance of the empirical mean can be approximated by the Monte Carlo error ($Error_{MC} = \frac{\sigma^2}{T}$) times the inefficiency factor of the Markov Chain (which is equal to $1 + 2\sum_{h=1}^{\infty} \rho_h$). The inefficiency factor is used to compute the effective sample size $ESS = \frac{T}{1+2\sum_{h=1}^{\infty} \rho_h}$.

Thus, it follows that:

$$\mathbb{V}(\hat{I}_t) = \frac{\sigma^2}{T}\Big[1 + 2\sum_{h=1}^{T-1} \frac{T-h}{T}\rho_h\Big] \approx \frac{\sigma^2}{T}\Big[1 + 2\sum_{h=1}^{\infty} \rho_h\Big] = \frac{\mathbb{V}(h(X))}{ESS}$$

where ρ_h is the autocorrelation function.

Moreover, the error effects the accuracy of parameter estimates: if the chain is too dependent, the ESS will be small, the error will be large and resulting estimates will not be accurate (and viceversa).

Let's see the MC errors for all the models that we have fitted.

Logistic model

```
##
##  MC error of Asym : 0.3228708
##  MC error of tm : 2.321569
##  MC error of k : 0.03672653
##  MC error of sigma : 0.1173601
```

Richard model

```
##
##  MC error of Asym : 0.3704092
##  MC error of tm : 2.474598
##  MC error of k : 0.04582617
##  MC error of sigma : 0.1282971
##  MC error of v : 0.145674
```

Gompertz model

```
##
##  MC error of Asym : 0.6400041
##  MC error of tm : 2.214969
##  MC error of k : 0.03846529
##  MC error of sigma : 0.1665983
```

5.5 Comparison between 3 models

There exist a variety of methodologies to compare models for a given data set and to select the one that best fits the data. In this case, we are going to compare models using a Bayesian measure of fit, DIC[4] criterion, which is a tool that is used for model assessment and provides a Bayesian alternative to classical criteria AIC and BIC[5].

This statistic takes into account the number of unknown parameters in the model and it can be seen as a generalization of the AIC.

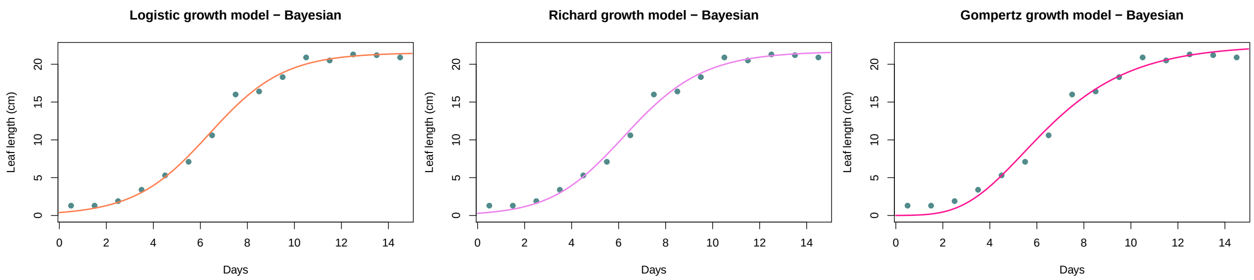
[4] DIC: deviance information criterion.
[5] AIC: Akaike information criterion; BIC: Bayesian information criterion.

$$DIC = 2\bar{D} - D(\bar{\theta}_i)$$

where $\bar{D} = -2\int \log[p(y \mid \theta_i)]p(\theta_i \mid y)d\theta_i$ and $D(\bar{\theta_i})) = -2\log[p(y \mid \hat{\theta}_i)]$.

According to this criteria the model that fits best the data is the Logistic model (with $DIC \simeq 40$), followed by Richards and Gompertz.

Moreover, it could be interesting also to check visually how the growth curves fit the data, exploiting posterior mean of parameters to get the curves.



Also from the plots above we are able to see the difference in the fitting curve between the one of the Gompertz model and the two first ones (that would be overlapped).

Thus, we can clearly state that we prefer Logistic and Richard models, but in particular the first one, according to DIC criterion.

Table 3: Posterior summary statistics for the three growth functions obtained with Gibs sampling algorithm

Model	Parameter	Posterior Mean	Sd	MC Error	95% Credible Interval [2.50%,97.50%]	DIC
Logistic	Asym	21.506	0.468	0.32435	[20.612, 22.427]	40.0
	tm	6.359	0.154	2.3005	[6.055, 6.666]	
	k	0.629	0.053	0.03701	[0.533, 0.745]	
	σ^2	0.775	0.178	0.11547	[0.516, 1.177]	
Richard	Asym	21.684	0.581	0.3232	[20.675, 22.929]	44.0
	tm	6.197	0.229	2.2225	[5.649, 6.590]	
	k	0.575	0.072	0.04125	[0.424, 0.705]	
	v	0.769	0.194	0.11292	[0.271, 0.991]	
	σ^2	0.829	0.200	0.1353	[0.551, 1.310]	
Gompertz	Asym	22.536	0.946	0.642	[20.826, 24.598]	49.8
	tm	5.441	0.220	2.2804	[5.018, 5.886]	
	k	0.396	0.057	0.03907	[0.303, 0.530]	
	σ^2	1.093	0.244	0.1690	[0.733, 1.657]	

6 Diagnostics

In order to verify the correctness of the MCMC structure produced by the Gibbs sampler (`jags` in our case), we may decide to run 2 different tests: Geweke Diagnostic Test and Heidelberger & Welch Diagnostic Test

We may exploit the functions in R available in the `coda` package.

6.1 Geweke Diagnostic Test

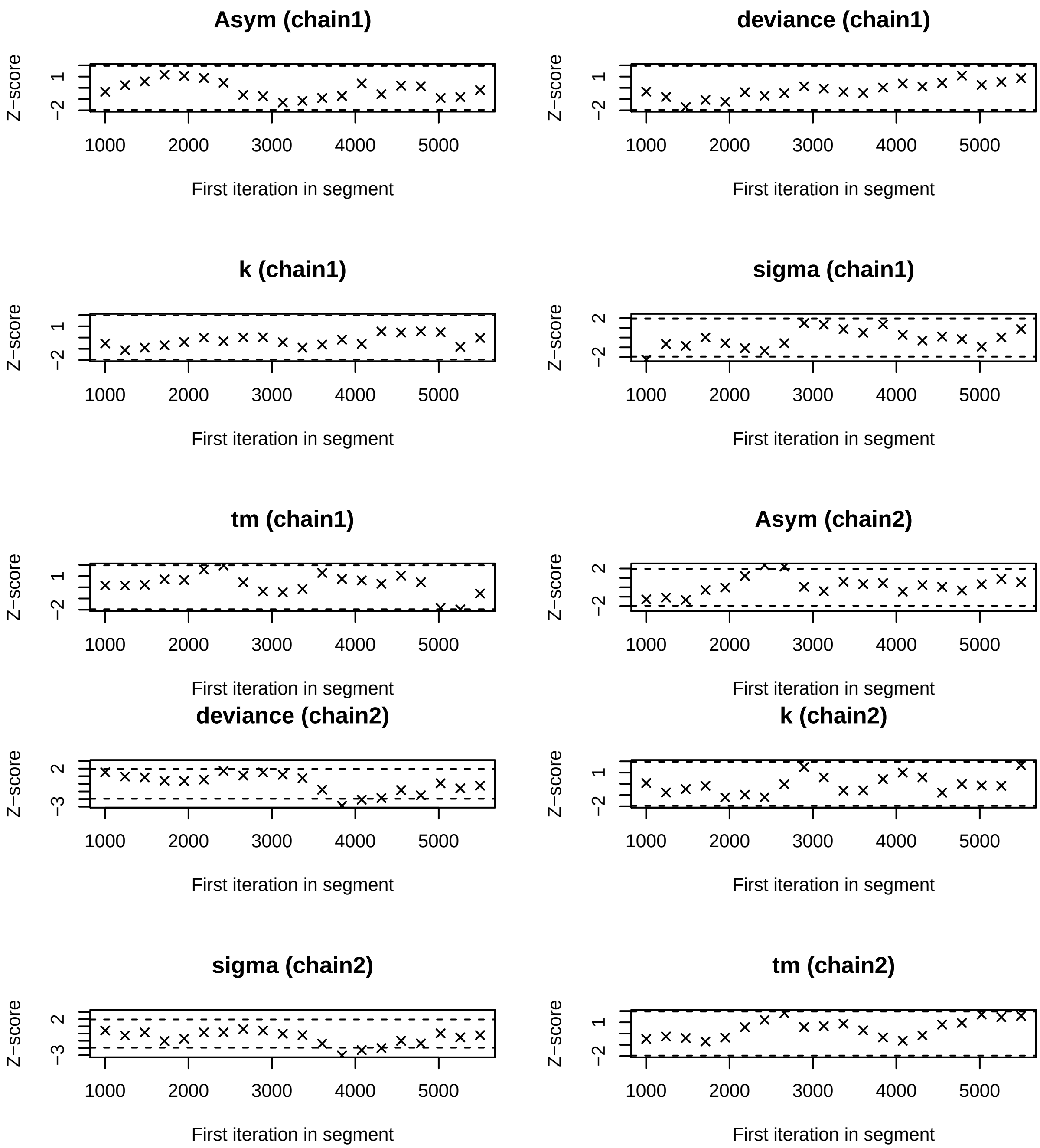
Geweke test is a convergence diagnostic for Markov chains. The aim of the test proposed by Geweke is to compare the means of the first and last part of a Markov chain (by default the first 10% and the last 50%). If the samples are drawn from a stationary distribution of the chain (as we would like to be!), then the two means are equal. Therefore, the test statistic that has been used in this case is a standard Z-score with the assumption of asymptotically independence of the two parts of the chain.

Logistic model

```
geweke.diag(coda_logistic)
```

```
## [[1]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     Asym deviance        k    sigma       tm
##  -0.3534  -0.3407  -0.5266  -2.2575   0.1697
##
##
## [[2]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     Asym deviance        k    sigma       tm
## -1.28659  1.51623  0.06954  0.41988 -0.47598
```

```
geweke.plot(coda_logistic)
```



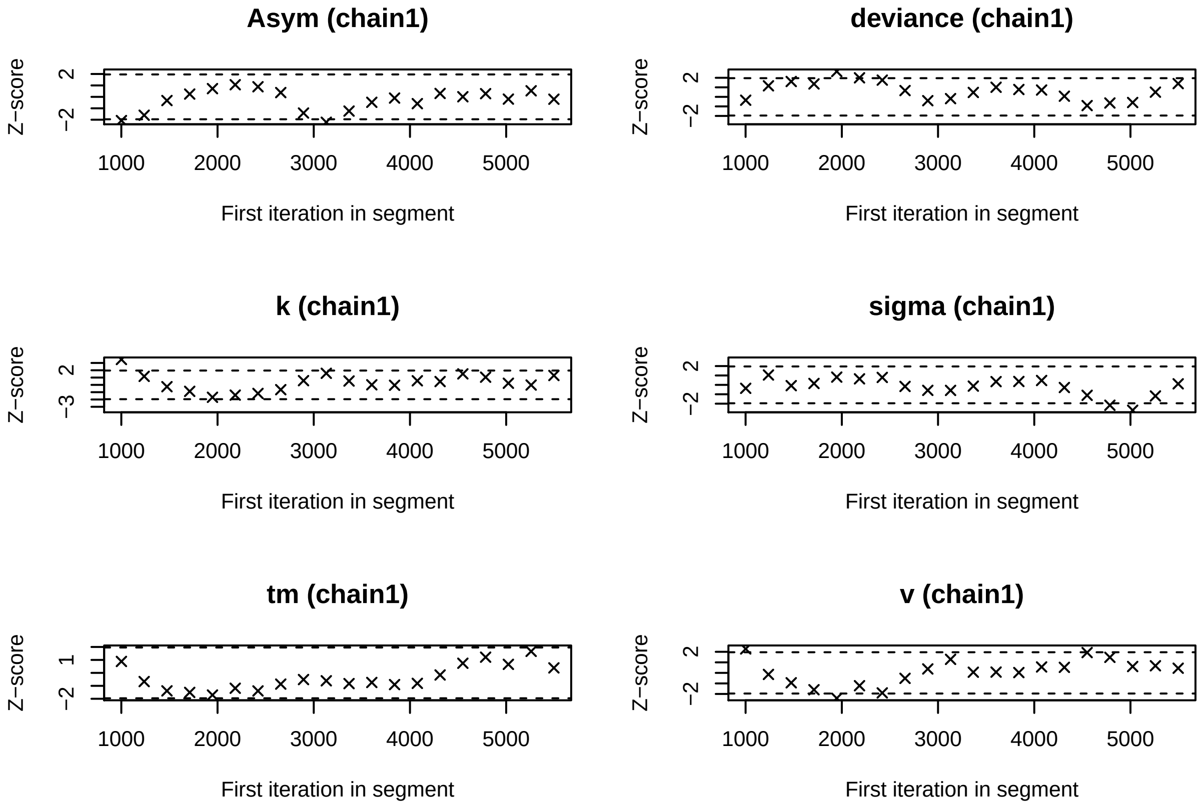
Richard model

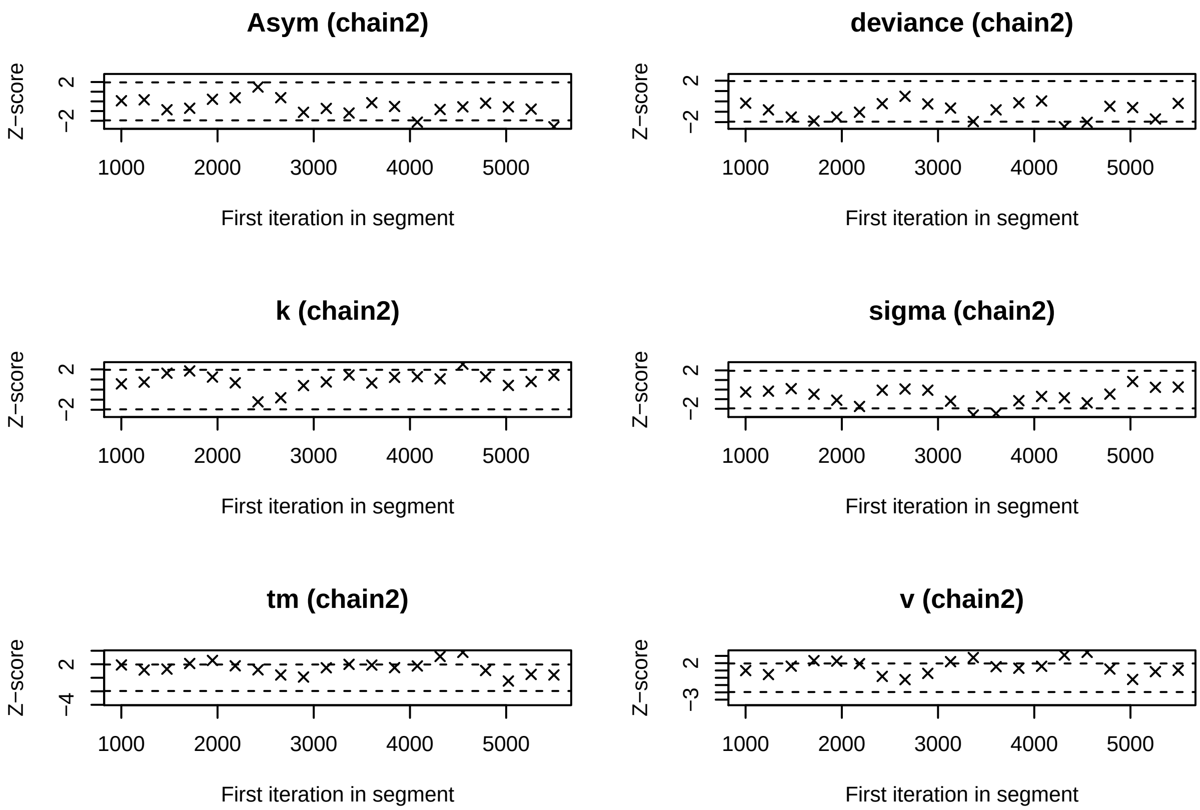
```
geweke.diag(coda_richard)
```

```
## [[1]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     Asym deviance        k    sigma       tm        v
##  -2.0746  -0.3522   3.4738  -0.3669   0.8821   2.2746
##
##
```

```
## [[2]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     Asym deviance        k    sigma       tm        v
##  0.06462 -0.16865  0.56629 -0.25311  1.89508  0.98305
```

```
geweke.plot(coda_richard)
```



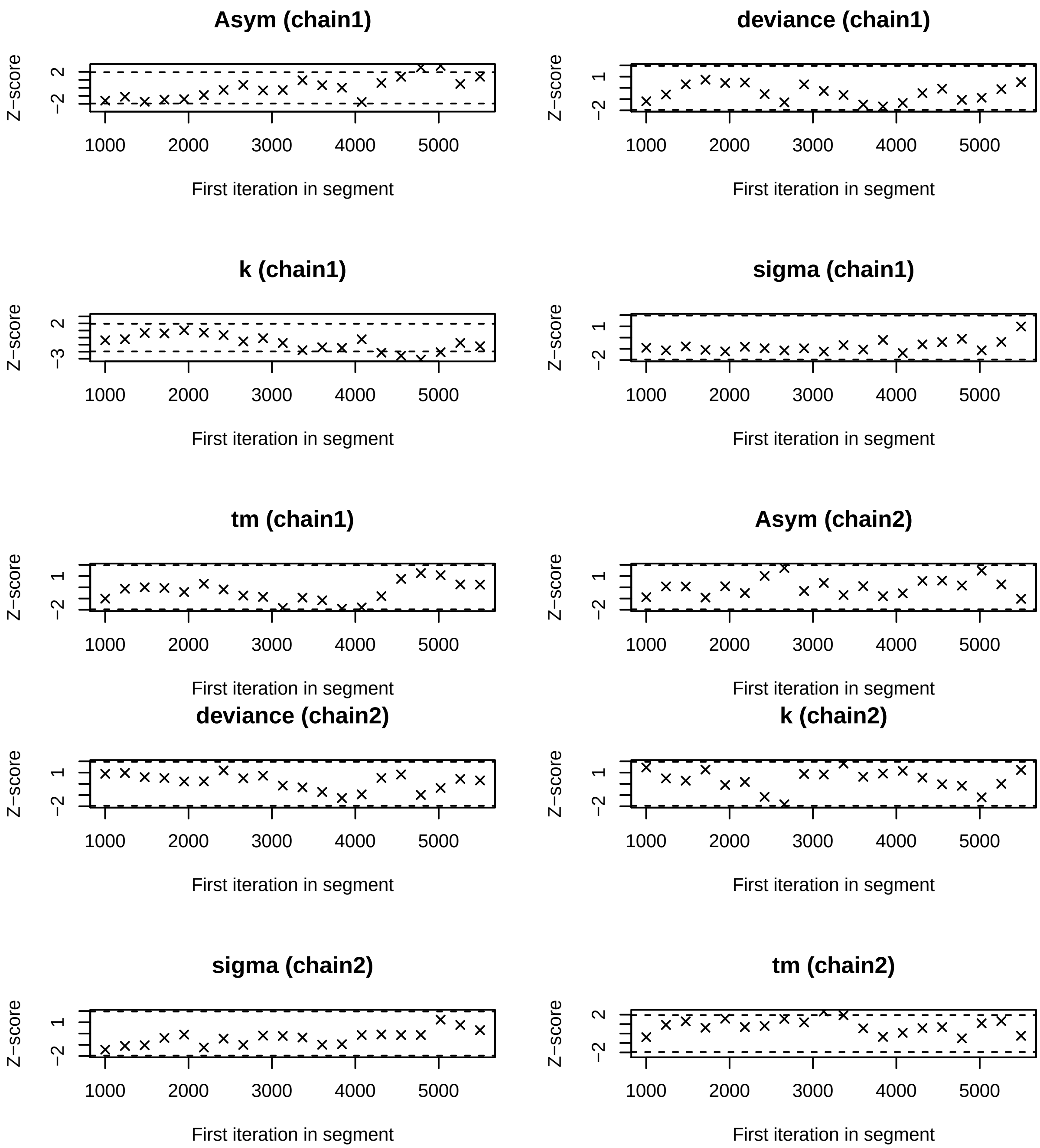


Gompertz model

```
geweke.diag(coda_gompertz)
```

```
## [[1]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     Asym deviance        k    sigma       tm
##  -1.6001  -1.1940  -0.3793  -0.9074  -1.0079
##
##
## [[2]]
##
## Fraction in 1st window = 0.1
## Fraction in 2nd window = 0.5
##
##     Asym deviance        k    sigma       tm
##  -0.8819   0.8951   1.4546  -1.4350  -0.3850
```

```
geweke.plot(coda_gompertz)
```



6.2 Heidelberger & Welch Diagnostic Test

This diagnostic test is divided in two steps.

1. The convergence test uses the Cramer-von-Mises statistic to test the null hypothesis that the sampled values come from a stationary distribution. In particular the same test is performed sequencially: firstly for the whole chain, then after discarding the first 10%, 20%, . . . of the chain until either the null hypothesis is accepted. If we need to discard the first 50% of the chain to pass the test, it means that the chain is not stationary and indicates that a longer MCMC run is needed.

2. The half-width test calculates a 95% confidence interval for the mean, using the portion of the chain which passed the stationary test. The test is passed if the ratio between the half-width and the mean is lower than *eps*.

Logistic model

```r
heidel.diag(coda_logistic)
```

```
## [[1]]
##
##          Stationarity start     p-value
##          test         iteration
## Asym     passed       1         0.941
## deviance passed       1         0.652
## k        passed       1         0.735
## sigma    passed       1         0.501
## tm       passed       1         0.728
##
##          Halfwidth Mean   Halfwidth
##          test
## Asym     passed    21.535 0.03274
## deviance passed    34.139 0.22022
## k        passed     0.626 0.00354
## sigma    passed     0.766 0.01132
## tm       passed     6.371 0.01075
##
## [[2]]
##
##          Stationarity start     p-value
##          test         iteration
## Asym     passed       1         0.681
## deviance passed       1         0.137
## k        passed       1         0.848
## sigma    passed       1         0.649
## tm       passed       1         0.455
##
##          Halfwidth Mean   Halfwidth
##          test
## Asym     passed    21.510 0.03064
## deviance passed    34.128 0.22108
## k        passed     0.630 0.00343
## sigma    passed     0.771 0.01098
## tm       passed     6.367 0.01013
```

Richard model

```r
heidel.diag(coda_richard)
```

```
## [[1]]
##
##          Stationarity start     p-value
##          test         iteration
## Asym     passed       1         0.476
## deviance passed       1         0.062
## k        passed       1         0.690
## sigma    passed       1         0.719
## tm       passed       1         0.152
## v        passed       1         0.339
##
##          Halfwidth Mean   Halfwidth
```

```
##          test
## Asym     passed    21.690 0.03960
## deviance passed    35.811 0.33455
## k        passed     0.574 0.00617
## sigma    passed     0.817 0.01417
## tm       passed     6.199 0.01987
## v        passed     0.778 0.01978
##
## [[2]]
##
##          Stationarity start     p-value
##          test         iteration
## Asym     passed       1         0.3461
## deviance passed       1         0.4700
## k        passed       1         0.4383
## sigma    passed       1         0.1859
## tm       passed       1         0.0501
## v        passed       1         0.1323
##
##          Halfwidth Mean   Halfwidth
##          test
## Asym     passed    21.696 0.03882
## deviance passed    36.069 0.31461
## k        passed     0.572 0.00578
## sigma    passed     0.829 0.01331
## tm       passed     6.180 0.02106
## v        passed     0.761 0.01852
```

Gompertz model

```
heidel.diag(coda_gompertz)
```

```
## [[1]]
##
##          Stationarity start     p-value
##          test         iteration
## Asym     passed       1         0.253
## deviance passed       1         0.640
## k        passed       1         0.288
## sigma    passed       1         0.189
## tm       passed       1         0.333
##
##          Halfwidth Mean   Halfwidth
##          test
## Asym     passed    22.525 0.07350
## deviance passed    44.629 0.21047
## k        passed     0.396 0.00424
## sigma    passed     1.102 0.01598
## tm       passed     5.442 0.01488
##
## [[2]]
##
##          Stationarity start     p-value
##          test         iteration
## Asym     passed       1         0.932
```

```
## deviance passed       1         0.856
## k        passed       1         0.609
## sigma    passed       1         0.404
## tm       passed       1         0.435
##
##          Halfwidth Mean   Halfwidth
##          test
## Asym     passed    22.546 0.07050
## deviance passed    44.754 0.21635
## k        passed     0.395 0.00418
## sigma    passed     1.106 0.01505
## tm       passed     5.437 0.01446
```

7 Evaluation of predictive perfomance of the three models

In this last section, we are interested in evaluating predictive performance of our three models.

In order to do that, we firstly need to modify the model definition in `jags` to run the Gibbs algorithm.

Here there is shown the new model for logistic model.

```
cat('model {
      for( i in 1:N ) {
      Y[i] ~ dnorm(mu[i], precision)
      mu[i] <- Asym / (1+exp(-k*(t[i]-tm)))

      Ypred[i] ~ dnorm(condexp[i], precision)
        condexp[i]  <- Asym / (1+exp(-k*(t[i]-tm)))
      }

      Asym ~ dnorm(0.0, 1.0E-3)I(1.0,)
        tm ~ dnorm(0.0, 1.0E-3)I(1.0,)
        k ~ dbeta(1.0, 1.0)
        precision ~ dgamma(0.01, 0.01)

        sigma <- 1 / sqrt(precision)

  }',file = 'logistic_model_pred.txt')
```

```
## Inference for Bugs model at "logistic_model_pred.txt", fit using jags,
##  2 chains, each with 10000 iterations (first 1000 discarded), n.thin = 10
##  n.sims = 1800 iterations saved
##             mu.vect sd.vect   2.5%    25%    50%    75%  97.5%  Rhat n.eff
## Asym         21.521   0.451 20.683 21.230 21.505 21.811 22.413 1.001  1800
## Ypred[1]      0.538   0.824 -1.078  0.006  0.503  1.078  2.196 1.001  1800
## Ypred[2]      0.997   0.814 -0.562  0.468  0.994  1.509  2.673 1.001  1800
## Ypred[3]      1.795   0.819  0.165  1.265  1.826  2.303  3.502 1.001  1800
## Ypred[4]      3.128   0.898  1.342  2.574  3.103  3.700  4.926 1.001  1800
## Ypred[5]      5.133   0.873  3.421  4.578  5.130  5.689  6.891 1.007   630
## Ypred[6]      7.964   0.887  6.187  7.416  7.947  8.526  9.720 1.002  1400
## Ypred[7]     11.247   0.893  9.463 10.704 11.243 11.793 13.074 1.003  1800
## Ypred[8]     14.433   0.895 12.715 13.832 14.432 15.024 16.251 1.000  1800
## Ypred[9]     17.003   0.879 15.197 16.465 16.996 17.567 18.686 1.000  1800
## Ypred[10]    18.852   0.871 17.089 18.317 18.858 19.387 20.633 1.001  1800
## Ypred[11]    19.981   0.826 18.342 19.465 19.970 20.513 21.629 1.001  1800
## Ypred[12]    20.657   0.876 18.860 20.126 20.658 21.210 22.398 1.001  1800
## Ypred[13]    21.043   0.853 19.329 20.524 21.068 21.567 22.714 1.007   320
## Ypred[14]    21.280   0.880 19.505 20.729 21.272 21.822 23.009 1.001  1600
## Ypred[15]    21.399   0.878 19.649 20.864 21.401 21.935 23.125 1.000  1800
## condexp[1]    0.558   0.154  0.297  0.454  0.545  0.646  0.899 1.002  1100
## condexp[2]    1.007   0.226  0.596  0.857  0.994  1.142  1.492 1.002  1100
## condexp[3]    1.792   0.313  1.203  1.594  1.780  1.986  2.427 1.002  1000
## condexp[4]    3.104   0.392  2.334  2.860  3.099  3.355  3.889 1.002   910
## condexp[5]    5.136   0.429  4.265  4.873  5.141  5.412  5.971 1.002   840
## condexp[6]    7.937   0.412  7.104  7.693  7.942  8.186  8.746 1.002   960
## condexp[7]   11.224   0.407 10.428 10.967 11.220 11.471 12.072 1.001  1800
## condexp[8]   14.421   0.429 13.577 14.151 14.406 14.688 15.299 1.000  1800
## condexp[9]   17.013   0.406 16.189 16.768 17.013 17.258 17.855 1.000  1800
## condexp[10]  18.827   0.342 18.078 18.620 18.825 19.029 19.509 1.001  1800
```

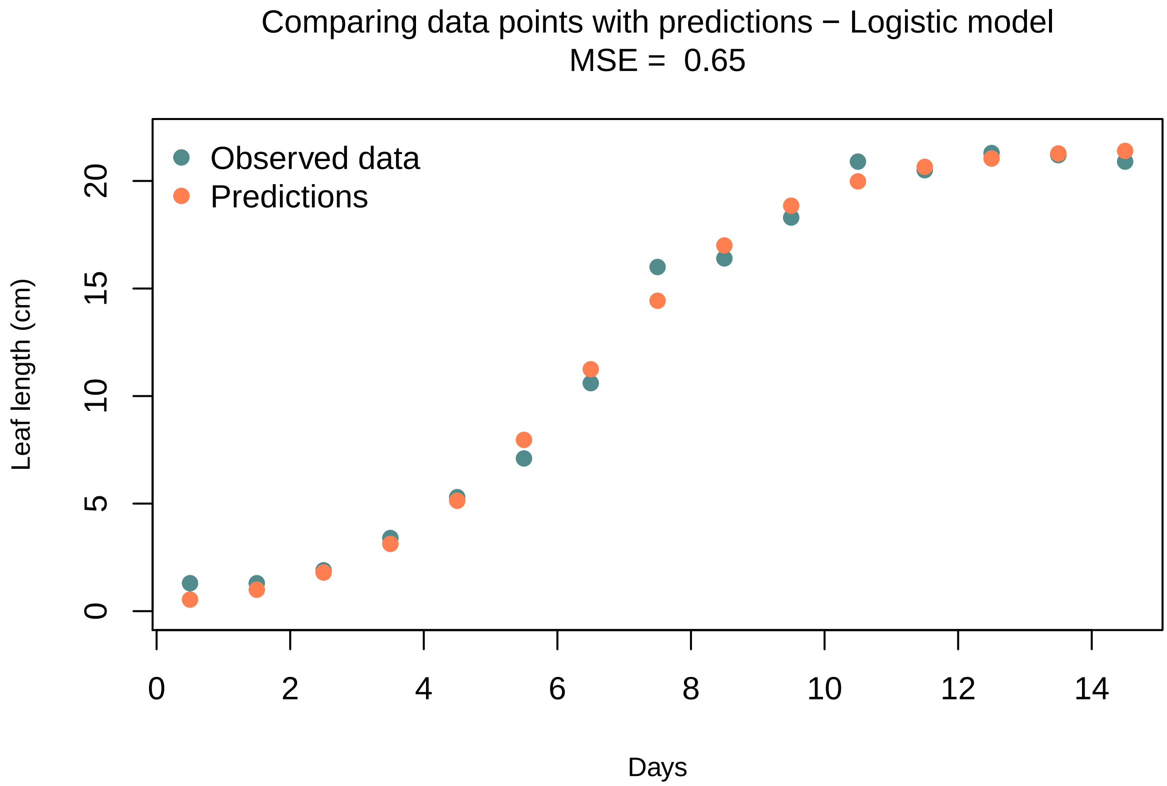
```
## condexp[11]  19.973   0.303 19.360 19.788 19.971 20.161 20.558 1.001  1800
## condexp[12]  20.651   0.314 20.021 20.462 20.652 20.848 21.271 1.000  1800
## condexp[13]  21.039   0.348 20.339 20.821 21.037 21.269 21.737 1.000  1800
## condexp[14]  21.255   0.381 20.505 21.015 21.253 21.505 22.019 1.000  1800
## condexp[15]  21.375   0.406 20.588 21.119 21.368 21.632 22.183 1.000  1800
## k             0.625   0.051  0.532  0.591  0.624  0.655  0.733 1.001  1600
## sigma         0.769   0.177  0.516  0.640  0.741  0.860  1.178 1.000  1800
## tm            6.364   0.154  6.056  6.267  6.360  6.468  6.667 1.001  1800
## deviance     34.021   3.370 29.929 31.608 33.209 35.506 42.458 1.004  1800
##
## For each parameter, n.eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor (at convergence, Rhat=1).
##
## DIC info (using the rule, pD = var(deviance)/2)
## pD = 5.7 and DIC = 39.7
## DIC is an estimate of expected predictive error (lower deviance is better).
```

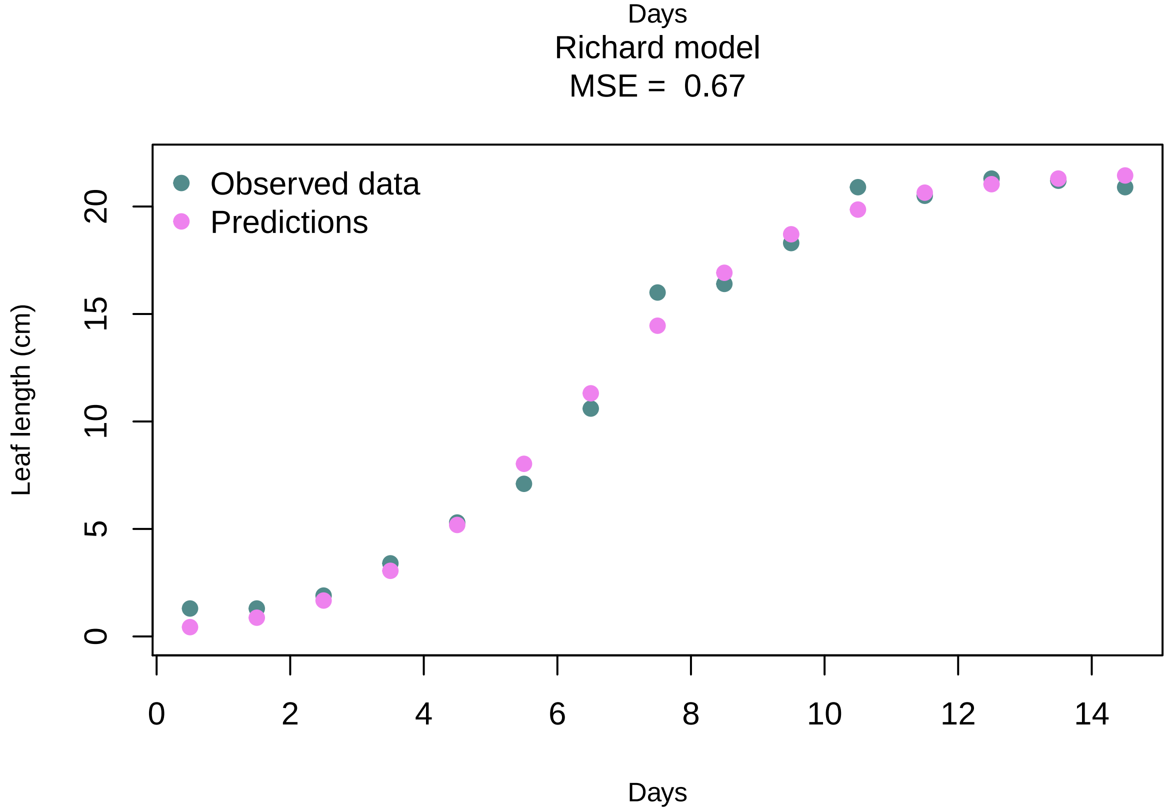
We may do the same procedure also for the other two models.

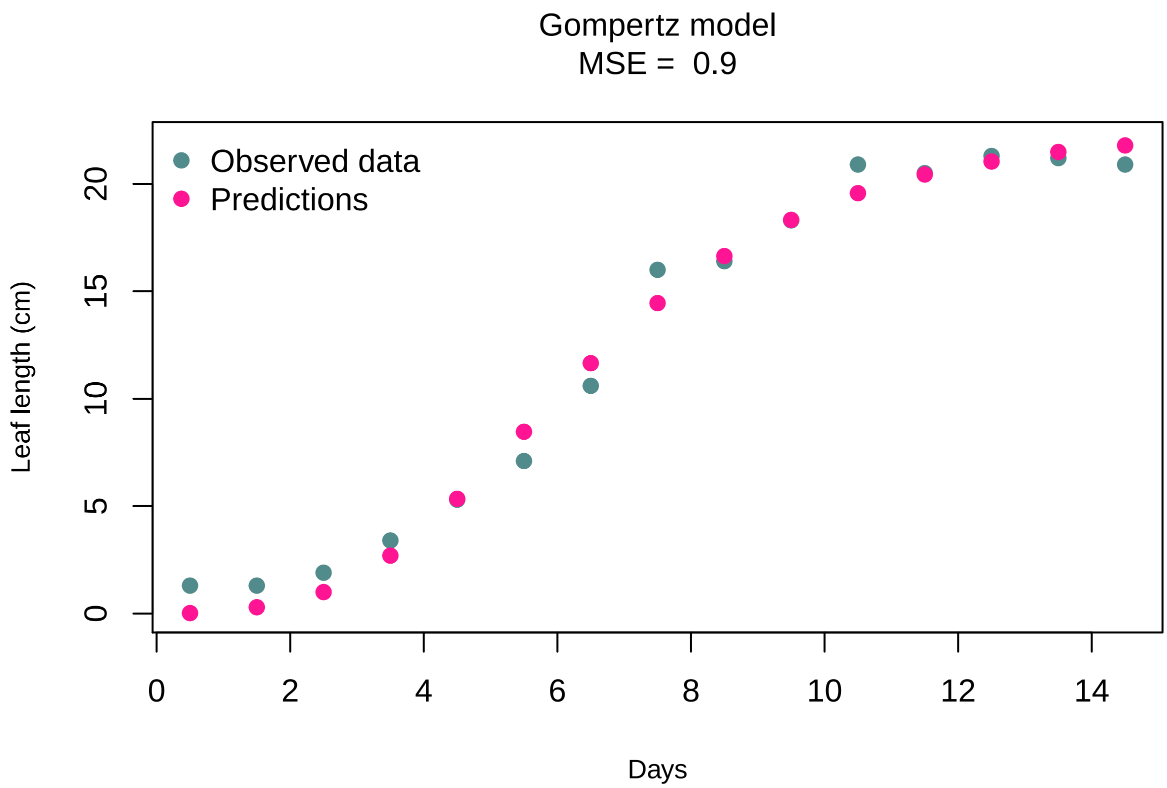
```
cat('model {
  for( i in 1:N ) {
    Y[i] ~ dnorm(mu[i], precision)
    mu[i] <- Asym / pow(1+v*exp(-k*(t[i]-tm)), (1/v))

    Ypred[i] ~ dnorm(condexp[i], precision)
    condexp[i]  <-  Asym / pow(1+v*exp(-k*(t[i]-tm)), (1/v))
  }

  Asym ~ dnorm(0.0, 1.0E-3)I(1.0,)
    tm ~ dnorm(0.0, 1.0E-3)I(1.0,)
    v ~ dbeta(1.0, 1.0)
    k ~ dbeta(1.0, 1.0)
    precision ~ dgamma(0.01, 0.01)

    sigma <- 1 / sqrt(precision)

}', file = 'richard_model_pred.txt')

cat('model {
  for( i in 1:N ) {
    Y[i] ~ dnorm(mu[i], precision)
    mu[i] <- Asym * exp(-exp(-k*(t[i]-tm)))

    Ypred[i] ~ dnorm(condexp[i], precision)
    condexp[i]  <- Asym * exp(-exp(-k*(t[i]-tm)))
  }

  Asym ~ dnorm(0.0, 1.0E-3)I(1.0,)
    tm ~ dnorm(0.0, 1.0E-3)I(1.0,)
    k ~ dbeta(1.0, 1.0)
    precision ~ dgamma(0.01, 0.01)

    sigma <- 1 / sqrt(precision)

}', file = 'gompertz_model_pred.txt')
```

Then, run the `jags`.

The plots above compare the observed data points with the predicted leaf length with respect to the time (in days) for the three different models. Predicted data points are obtained with the point estimates of the parameter Y_{pred} of the Gibbs.







We may make some statements by observing the plots:

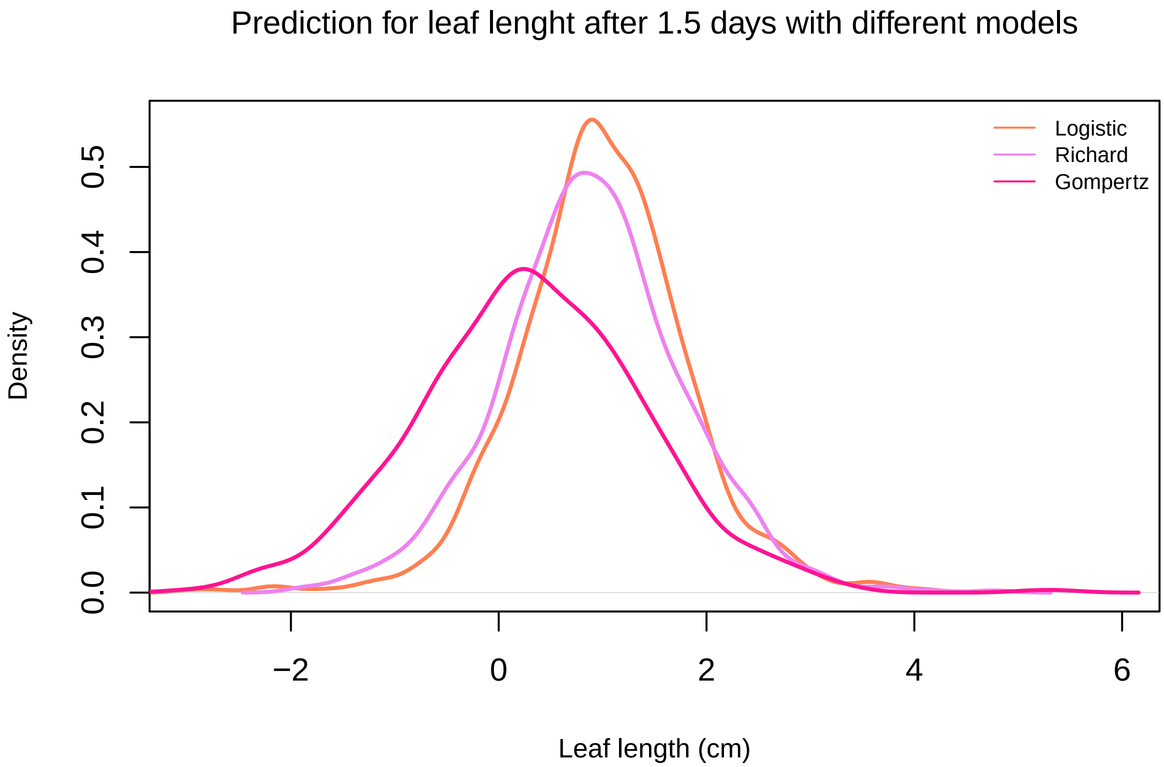
- Logistic and Richard model seem to lead to similar predictions, thus the MSE is between is smaller than 0.7 for both. On the other side MSE of Gompertz's predictions is a little bit higher (about 0.9).
- Gompertz model fails more to predict the leaf lengths in the first and in the last period. Specifically, it tends to underestimate lengths in first days, while they are overestimated after 14 days.

These aspects could be double checked with computing the approximation errors of estimates in different time.

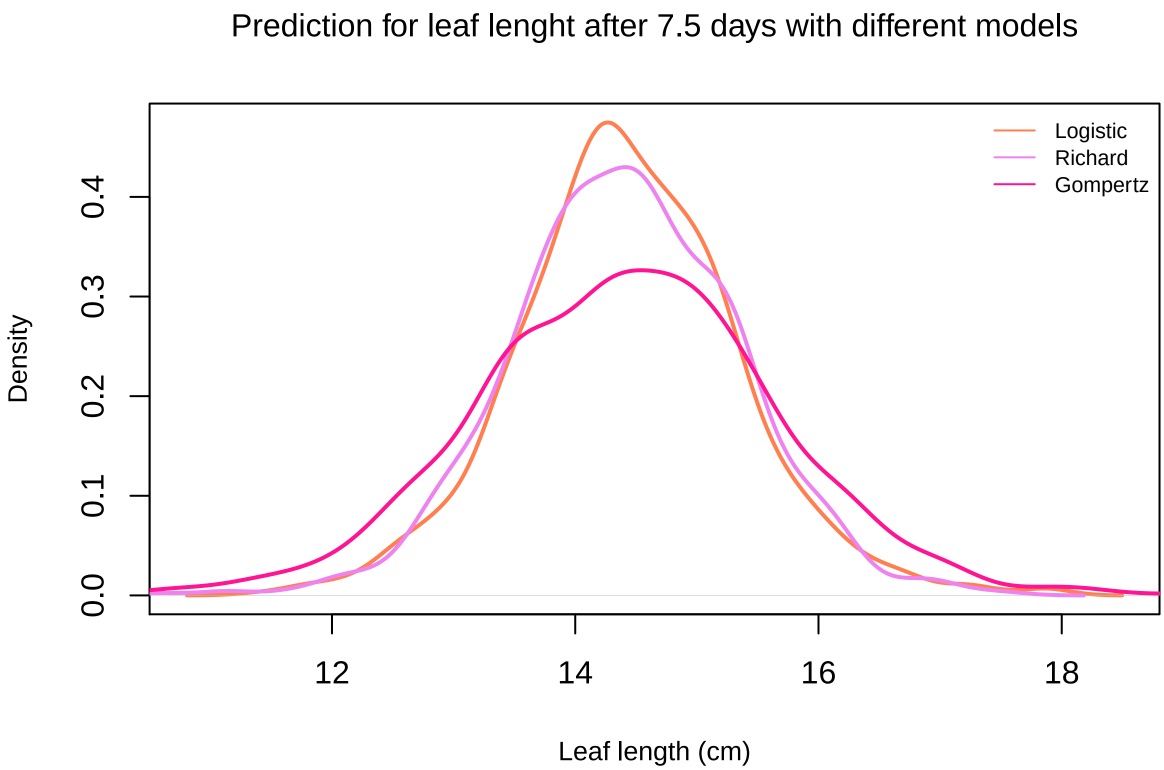
For that, we are interested in doing prediction on the length of the leaf after a fixed number of days and see the error associated to that prediction.

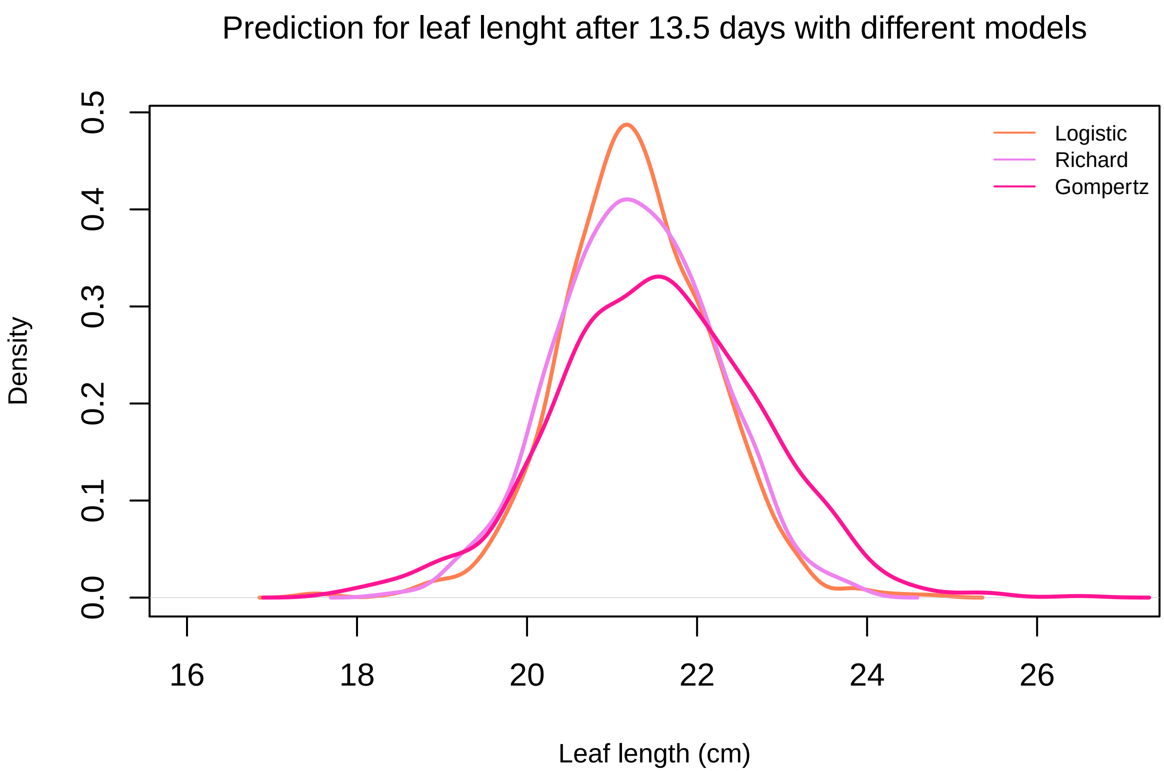
Let's select three times (days) after which we would like to predict the leaf length:

1. at the beginning of the measurement: t = 1.5 (the second measurement)
2. approximately near the *inflation* point: t = 7.5 (the central measurement)
3. in the last period: t = 13.5 (second last measurement)



Now we are going to repeat the same steps in order to evaluate predictions in the other other two fixed times of the growth.





Here there are shown all the MC error for the previous predictions. As we may expected, Gompertz MC errors are higher for all the predictions and it performs worst in predicting the leaf length in t = 13.5. Logistic model is confirmed again to be the best one.

	Logistic model	Richard model	Gompertz model
t = 1.5	0.0276771	0.0311395	0.0366330
t = 7.5	0.0306264	0.0335451	0.0441841
t = 13.5	0.0294579	0.0307878	0.0406393

As we could imagine from the previous results, both in the evaluation of the models and in the prediction plots above, the models that lead to smallest errors are Logistic and Richard ones, while Gompertz model is the one that provides largest errors of the predictions (at some fixed time). This fact can be observed also from the posterior predictive density plots, where the curves that represent Logistic and Richard are more concentrated around the posterior mean of the prediction, while those ones of Gompertz are more 'flatted' and asymmetric too.

Moreover, predictions of the length of the leaf after few days (t = 1.5) are more accurate and this is true for all the models.

8 Last remarks

To conclude, we could state that Logistic and Richard growth models provide equivalent results and work better than Gompertz function both in the classical (NLS) and in the Bayesian approach. Moreover, Logistic model obtained the best predictive performances.

From the two summary tables (Table 1 and Table 2) we may highlight some aspects:

- in the Frequentist approach the parameter t_m (which represents the inflection point at which the growth rate is maximized) has a small SE for all the models, while the MC error is quite large in the Bayesian set up;

- the parameter v in Richard model has been estimated with a small approximation error (0.14371) with Gibbs, while it is not statistically significant (with a large SE) in the classical approach.