Lab 4 - Computational Statistics

Gustav Sternelöv

February, 29, 2016

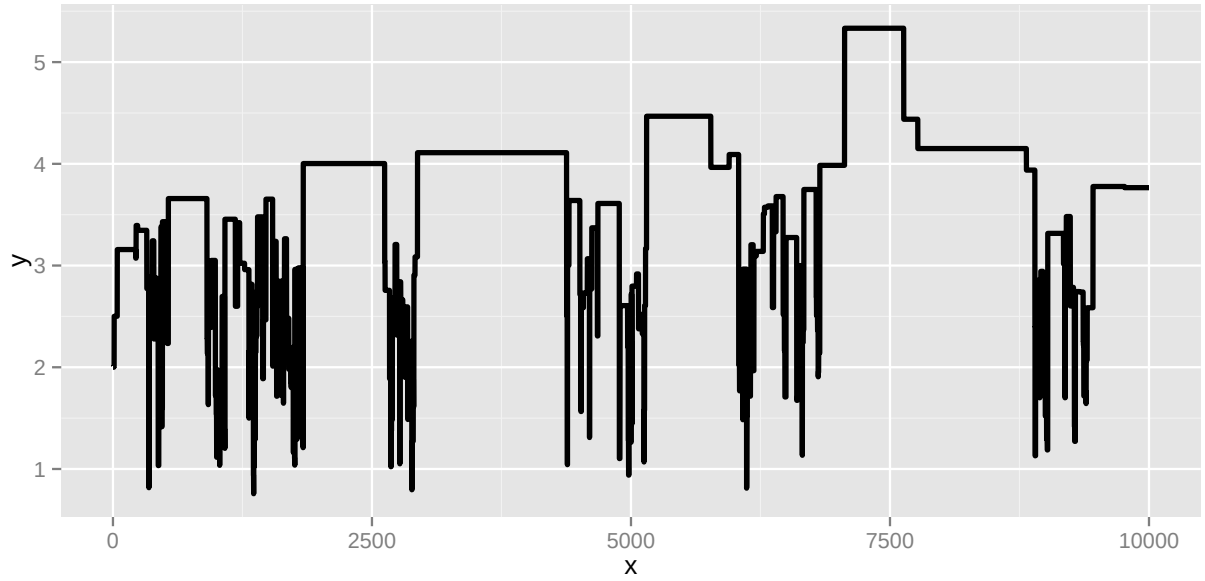
Assignment 1

1.1

The Metropolis-Hastings algorithm is used for generating values from the following distribution:

$$f(x) \propto x^5 e^{-x}$$

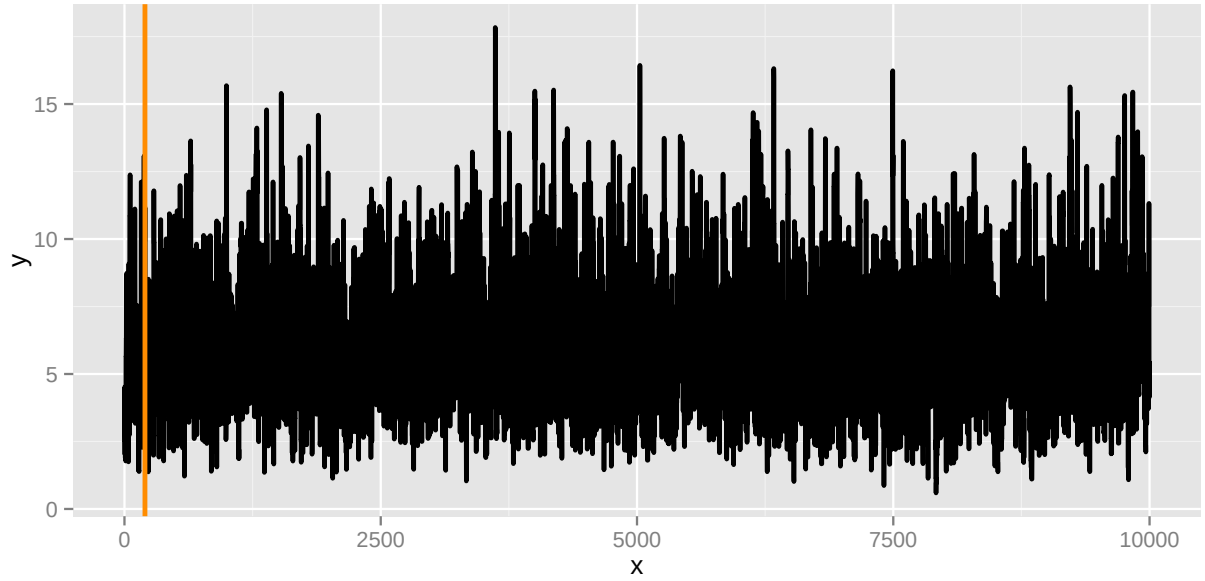
In the first step is the log-normal distribution the chosen proposal distribution and the chosen starting point is $X_0 = 2$. A time-series plot of the chain is generated to analyse the convergence of the chain and if there is a burn-in period.



The chain never converges, the patterin is irregular over the whole time-series, and therefore there is no burn-in period to analyse.

1.2

In the second step is it the same distribution as in *1.1* that the algorithm is supposed to generate values from. The difference is the proposal distribution which now is a chi-square distribution, $\chi^2(floor(X_t + 1))$, where *floor(x)* means that the integer part of x is the inserted value. The starting point is $X_0 = 2$.
The chain is again plotted as a time-series in order to analyse the convergence and the eventual burn-in period.



With the mentioned chi-square distribution as proposal distribution does the chain converge very quickly. Meaning that the chain has converged to the target density.

The burn-in period is not very evident by an visual eximination. Theoretically, according to Martinez(*Computational Statistics Handbook with Matlab*, 2001), the burn-in period can be assumed to be around 1-2 % of n if n is large enough.
In the example above is n set to 10000 and the burn-in period would then be around 100-200 of the first values in the chain. This is thought to be reasonable as it can be seen in the plot that the chain converges very fast. The orange line specifies the burn-in period that is chosen to be 200.

1.3

A comparison between the results obtained in step 1 and step 2 shows that the chi-square distribution seem to be a better choice than the log-normal distribution for the proposal distribution. That since the chain received for the case with the log-normal as proposal distribution never converged. The markov chain obtained in step 2 on the other hand did converge and therefore is the chi-square distribution thought to be the better choice.

1.4

10 MCMC sequences are generated with the generator from step 2. The respective starting points are 1,2,...,10. The Gelman-Rubin method is used for analysing the convergence of the sequences.
The output is given below and the most interesting number to interpret is the *Upper C.I.*. If this number is very close to 1, approximately around 1-1.2, convergence is concluded to have been reached. As the output shows is the *Upper C.I.* for these sequences equal to 1, hence convergence is achieved.

```
## Potential scale reduction factors:
##
##      Point est. Upper C.I.
## [1,]          1          1
```

1.5

Using the sample from step 1 the estimation of the integral is given by:

$$\frac{1}{n}\sum_{t=1}^{n}(X_t)$$

So the estimate of the mean given by the sample from step 1 is 3.679.
For the sample from step 2 is the first 200 observations in the sample removed due to the burn-in period. So, the estimation of the integral is calculated in the following way:

$$\frac{1}{n-200}\sum_{t=200+1}^{n}(X_t)$$

With this sample is the integral estimated to be 6.032.

1.6

The probability density function for a gamma distribution is:

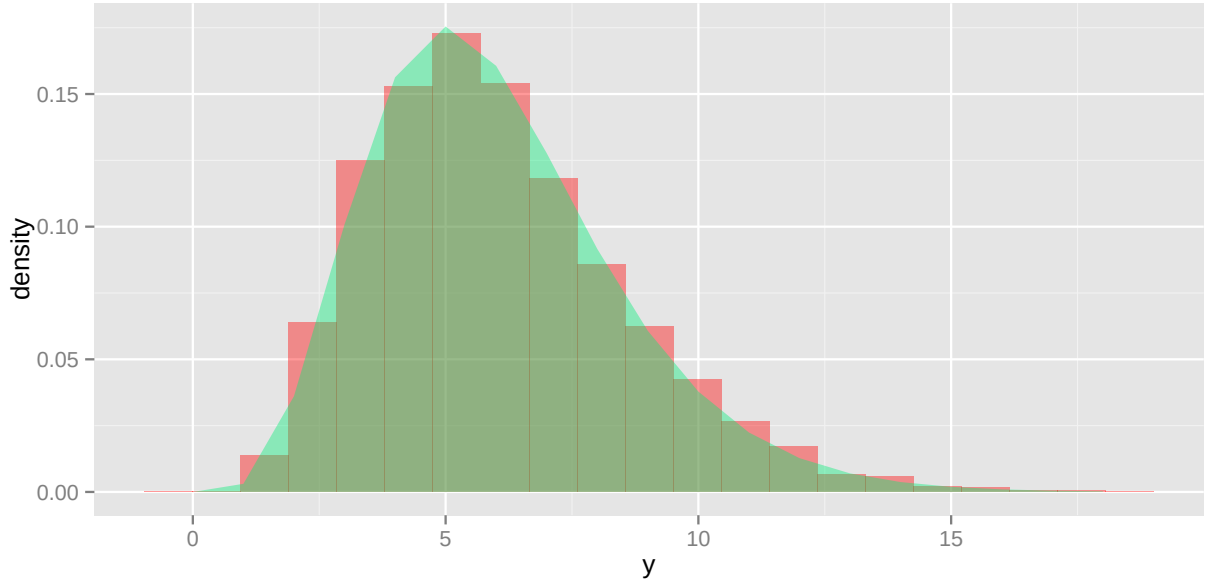
$$\frac{1}{\Gamma(\kappa)\theta^{\kappa}}x^{\kappa-1}e^{-\frac{x}{\theta}}$$

An comparsion of the target distribution and the gamma distribution gives that κ is equal to 6 and that θ is equal to 1.
The expected mean for a gamma distribution is given by the follwoing equation:

$$E[X] = \kappa\theta$$

Hence, the actual value of the integral is 6. The mean for step 1 is far away from this value and the mean for step 2 is very close to the actual value. This result is expected since the chain did not converge in step 1 but did so in step 2.

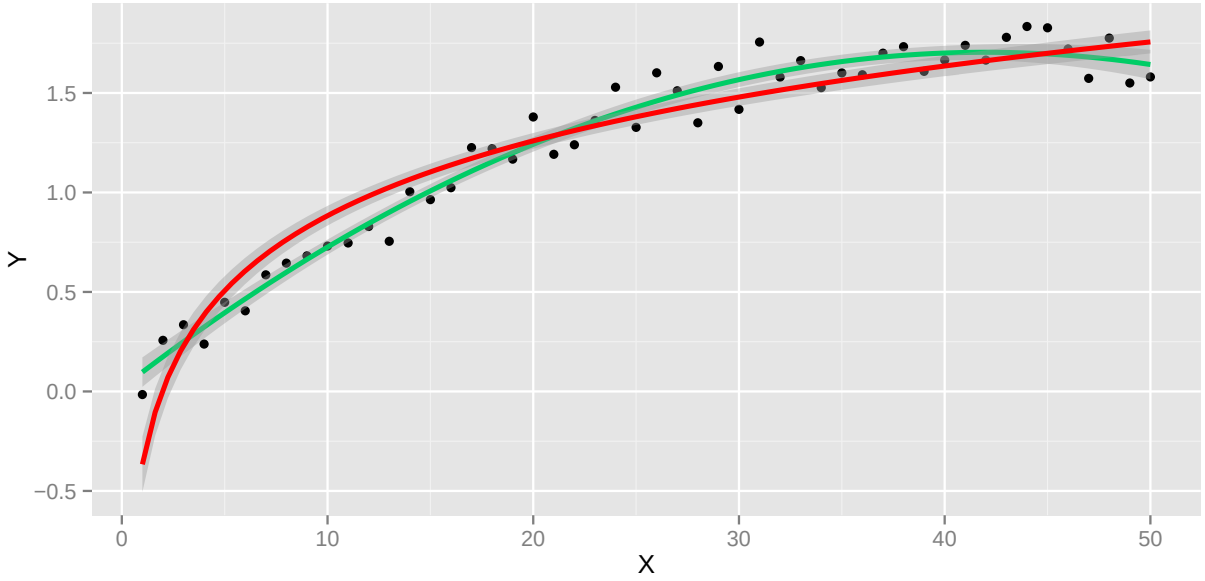
The next plot is used for comparing the theoretical density for *Gamma(6,1)*, the green area, and the histogram for the values generated in step 2, the red area. The conclusion is that the generated values really well follows the target distribution.



Assignment 2

2.1

The dependence of the variable y on the variable x is illustrated with the plot below. The fits of two different models are also visualised. The green line illustrates an quadratic model and the red line an log-transformed model.



Both the quadratic model and the log-transformed model seem to be relatively good fits to data. Perhaps is the quadratic model a little bit better, but that is hard to say definitely just from the plot.

2.2

The formula for the likelihood p($\mathbf{Y}|\mu$) is:

$$\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n exp\left(-\frac{\sum_{i=1}^{n}(Y_i-\mu_i)^2}{2\sigma^2}\right)$$

The formula for the prior $p(\mu)$ is:

$$\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{n-1} exp\left(-\frac{\sum_{i=2}^{n}(\mu_i-\mu_{i-1})^2}{2\sigma^2}\right)$$

2.3

The posterior is proportional to the product of the *likelihood* and the *prior* presented in *2.2*. So, the posterior is:

$$\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{n+n-1} exp\left(-\frac{\sum_{i=1}^{n}(Y_i-\mu_i)^2-\sum_{i=2}^{n}(\mu_i-\mu_{i-1})^2}{2\sigma^2}\right)$$

The derived posterior is then used for finding the distributions for $\mu_i|\mu_{-1}Y$. This results in three different distributions. One for the first μ, one for the $\mu : s$ in the interval 2-49 and one for the last μ.
The distribution for the first μ:

$$\propto exp\left(-\frac{(\mu_1-\frac{Y_1+\mu_2}{2})^2}{\sigma^2}\right)$$

The distribution in the span 2-49:

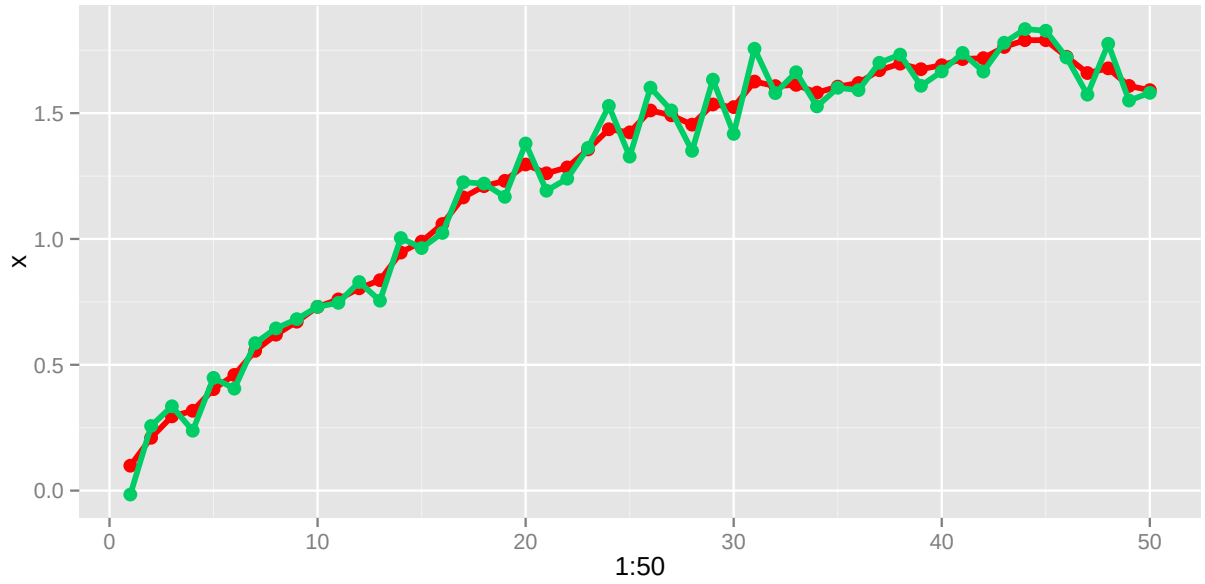
$$\propto exp\left(-\frac{(\mu_i - \frac{\mu_{i-1}+Y_i+\mu_{i+1}}{3})^2}{\frac{2}{3}\sigma^2}\right)$$

The distribution for the last μ:

$$\propto exp\left(-\frac{(\mu_{50} - \frac{Y_{50}+\mu_{49}}{2})^2}{\sigma^2}\right)$$

2.4

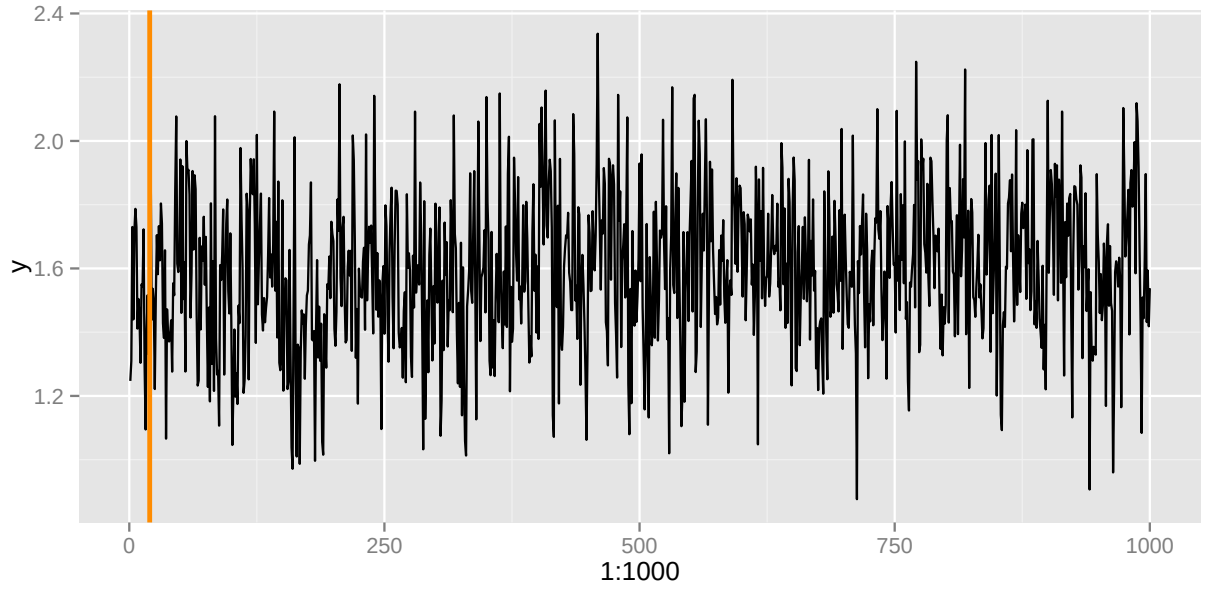
The expected value of μ obtained with the Gibbs sampler and the Monte Carlo approach is visualised with the red line in the plot below. They are compared to the observed values that are plotted with the green line.



Most of the noice seem to have been removed. The line for the generated values are more smooth compared to the line for the observed values. For the generated values is the variance lower and the data points are less noisy. The expected value of μ does really well to capture the the true underlying dependence between Y and X.

2.5

The trace plot for μ_{50} is plotted below.



The trace for μ_{50} seems to converge rapidly. Hence, 2 % of the observations are assigned in to the the burn-in period. This period of values is visualized with the orange line.

Appendix

```
library(coda)
library(ggplot2)
library(gridExtra)
f_x <- function(x){
  thaVal <- x^5*exp(-x)
  return(thaVal)
}
x_ta <- 2
set.seed(311015)
for(i in  1:9999){
  Y_point <- rlnorm(1, x_ta[i], 1)
  U_point <- runif(1, 0, 1)
  q_x <- dlnorm(x = x_ta[i], meanlog = Y_point, 1)
  q_y <- dlnorm(x = Y_point, meanlog = x_ta[i], 1)
  alpha <- min(c(1, ((f_x(Y_point) * q_x )  /
                       (f_x(x_ta[i]) * q_y))))
  if (U_point <= alpha) {
    x_ta[i+1] <- Y_point
  }else{
    x_ta[i+1] <- x_ta[i]
  }
}
x_ta <- data.frame(y=x_ta, x=1:10000)
time_ln <- ggplot(x_ta, aes(y=y, x=x)) + geom_line(size=1.15)
```

```
time_ln
set.seed(311015)
x_tb <- 2
for(i in  1:9999){
  Y_point <- rchisq(1, floor(x_tb[i] +1))
  U_point <- runif(1, 0, 1)
  q_x <- dchisq(x_tb[i], floor(Y_point+1))
  q_y <- dchisq(Y_point, floor(x_tb[i]+1))
  alpha <- min(c(1, ((f_x(Y_point) * q_x )  /
                       (f_x(x_tb[i]) * q_y))))
  if (U_point <= alpha) {
    x_tb[i+1] <- Y_point
  }else{
    x_tb[i+1] <- x_tb[i]
  }
}
x_tb <- data.frame(y=x_tb, x=1:10000)
time_chi <- ggplot(x_tb, aes(y=y, x=x)) + geom_line(size=1) +geom_vline(xintercept=200, col="darkorange
time_chi
## 1.4 ##
x_xt <- as.data.frame(matrix(seq(10001),nrow=10001,ncol=10))
for(j in 1:10){
  x_t <- j
  t <- 0
  for(i in  1:10000){
    Y_point <- rchisq(1, floor(x_t[i] +1))
    U_point <- runif(1, 0, 1)
    q_x <- rchisq(1, floor(x_t[i]+1))
    q_y <- rchisq(1, floor(Y_point+1))
    alpha <- min(c(1, ((f_x(Y_point) * q_x )  /
                         (f_x(x_t[i]) * q_y))))
    if (U_point <= alpha) {
      x_t[i+1] <- Y_point
    }else{
      x_t[i+1] <- x_t[i]
    }
  }
  x_xt[, j] <- x_t
}

f=mcmc.list()
for (i in 1:10) f[[i]]=as.mcmc(x_xt[,i])
gelman.diag(f)
Gamm <- data.frame(y=dgamma(0:18, 6, 1))
ggplot(x_tb[210:10000,], aes(y,..density..)) + geom_histogram(binwidth=0.95, fill="red", alpha=0.4) +
  geom_area(data=Gamm,aes(x=0:18, y=y), fill="springgreen2", alpha=0.4)
chemic <- data.frame(load("C:/Users/Gustav/Documents/Computational-Statistics/Lab4/chemical.RData"))
chemic <- data.frame(x=X, y=Y)
ggplot(chemic, aes(x=X, y=Y)) + geom_point() + geom_smooth(method="lm",formula = y ~ x + I(x^2), size=1
sigma2 <- 0.2
y <- Y
mu_update <- as.data.frame(matrix(seq(1000),nrow=1000,ncol=50))
mu_update[,1:50] <- 0
```

```
  for(j in 1:1000){
    if(j == 1){
      mu_update[j,1] <- rnorm(1, mean=(y[1]+mu_update[j,2])/2, sd = sqrt(sigma2)/2)
      for(h in 2:49){
        mu_update[j,h] <- rnorm(1, mean=(mu_update[j,h-1]+y[h]+mu_update[j,h+1])/3, sd = sqrt(sigma2)/3)
      }
      mu_update[j,50] <- rnorm(1, mean=(y[50]+mu_update[j,49])/2, sd = sqrt(sigma2)/2)
    }else{
      mu_update[j,1] <- rnorm(1, mean=(y[1]+mu_update[j-1,2])/2, sd = sqrt(sigma2)/2)
      for(h in 2:49){
        mu_update[j,h] <- rnorm(1, mean=(mu_update[j,h-1]+y[h]+mu_update[j-1,h+1])/3, sd = sqrt(sigma2),
      }
      mu_update[j,50] <- rnorm(1, mean=(y[50]+mu_update[j,49])/2, sd = sqrt(sigma2)/2)
    }
  }
mu_updateSum <- data.frame(x=colMeans(mu_update))
ggplot(mu_updateSum, aes(y=x, x = 1:50)) + geom_point(size=3, col="red") + geom_line(size=1.25, col="red
  geom_point(data=chemic, aes(x=x, y=y), col="springgreen3", size=3) + geom_line(data=chemic, aes(x=x, y
mu_50 <- data.frame(y=mu_update[, 50])
ggplot(mu_50, aes(y=y, x = 1:1000))+geom_line() + geom_vline(xintercept = 20, col="darkorange", size=1.0
## NA
```