Lab 2

Nadzeya_Boyeva

2024-10-25

```
library(ggplot2)
library(tidyr)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Task 1

nls() – determines the nonlinear (weighted) least-squares estimates of the parameters of a nonlinear model.

predict() – predicts results of various model fitting functions.

coef() – extracts model coefficients from objects returned by modeling functions.

confint() – computes confidence intervals for one or more parameters in a fitted model.

vcov() – returns the variance-covariance matrix of the main parameters (those returned by coef()) of a fitted model object.

diag() – extracts matrix diagonal; creates identity matrix; creates matrix with given diagonal and zero off-diagonal entries.

qt() – calculates quantiles for the t distribution.

abline() – adds one or more straight lines through the current plot.

Task 2

Load file with data. Plot experimental data.

V1 – "x variable", based on which we will predict V2 (let's say it's length of organism. for example);

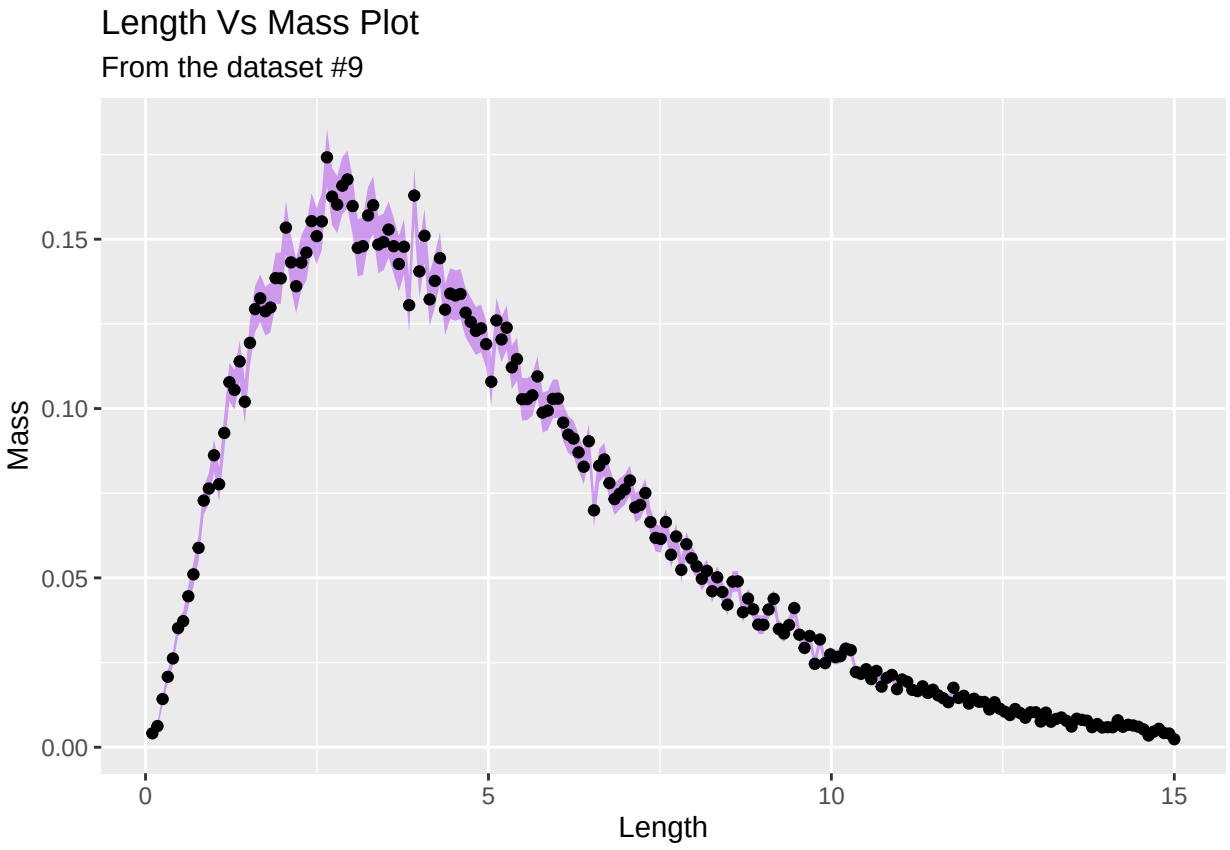
V2 – "y variable", which will be predicted (let's say it's mass);

V3 – standard deviation of V2.

```
df = read.table("data9.txt")
head(df)
```

```
##          V1         V2         V3
## 1 0.1000000 0.00411224 0.00097088
## 2 0.1748744 0.00625279 0.00121640
## 3 0.2497487 0.01423044 0.00150333
## 4 0.3246231 0.02079266 0.00181641
## 5 0.3994975 0.02621625 0.00214580
## 6 0.4743719 0.03516479 0.00248455
```

```
ggplot(data=df,
       aes(y=V2, x=V1)) +
  geom_ribbon(aes(ymin = V2 - V3,
                  ymax = V2 + V3),
              alpha=0.4, fill='purple') +
  geom_point() +
  labs(title="Length Vs Mass Plot",
       subtitle="From the dataset #9",
       y="Mass",
       x="Length")
```



Task 3

Implement three functions(x, **theta**) according to the variant given.

1) normal distribution:

$$f(x) = \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{-\frac{(x-m)^2}{2\sigma^2}}$$

```
# theta - vector of m (mean) and sigma (standart deviation)
my_norm <- function(x, theta) {
  return (dnorm(x, mean=theta[1], sd=theta[2], log=FALSE))
}
```

2) exponential distribution:

$$f(x) = \lambda \cdot e^{-\lambda x}$$

```
# theta = lambda
my_exp <- function(x, theta) {
  return (dexp(x, rate=theta, log=FALSE))
}
```

3) chi-squared distribution:

$$f(x) = \frac{1}{2^{\frac{n}{2}} \cdot \Gamma(\frac{n}{2})} \cdot x^{\frac{n}{2}-1} \cdot e^{-\frac{x}{2}}$$

```
# theta = n - degrees of freedom
my_chisq <- function(x, theta) {
  return (dchisq(x, df=theta, ncp=0, log=FALSE))
}
```

Task 4

Run nonlinear least squares method using functions defines above. Select initial conditions arbitrary. For optimization use nls() function.

Normal distribution

```
model_norm <- nls(V2 ~ my_norm(V1, norm_theta),
                  data=df,
                  start=list(norm_theta = c(mean = 2.5, sd = 3)))

print(coef(model_norm))
```

```
## norm_theta1 norm_theta2
##    3.823572    2.729641
```

Exponential distribution

```
model_exp1 <- nls(V2 ~ my_exp(V1, exp_theta),
                  data=df,
                  start=list(exp_theta = 0))

print(coef(model_exp1))
```

```
## exp_theta
## 0.1379485
```

```
model_exp2 <- nls(V2 ~ my_exp(V1, exp_theta),
                  data=df,
                  start=list(exp_theta = 129))

print(coef(model_exp2))
```

```
## exp_theta
##  101.0805
```

Depending on initial condition selected, theta returns one of two values. Most probably second one is incorrect since it's too big, but further we will examine the second fitted model as well (exponential model with second exp_theta).

Chi-squared distribution

```
model_chisq1 <- nls(V2 ~ my_chisq(V1, chisq_theta),
                  data=df,
                  start=list(chisq_theta = 1),
                  trace=TRUE)
```

```
## 4.706857    (3.59e-01): par = (1)
## 2.801697    (1.10e+00): par = (1.748248)
## 0.7685303   (1.86e+00): par = (2.926118)
## 0.09324713  (3.70e+00): par = (4.143597)
## 0.005142089 (4.74e-01): par = (4.868488)
## 0.004208919 (1.10e-03): par = (4.959567)
## 0.004208914 (1.43e-05): par = (4.959351)
## 0.004208914 (1.85e-07): par = (4.959354)
```

```
print(coef(model_chisq1))
```

```
## chisq_theta
##    4.959354
```

```
model_chisq2 <- nls(V2 ~ my_chisq(V1, chisq_theta),
                  data=df,
                  start=list(chisq_theta = 30),
                  trace=TRUE)
```

```
## 1.435689    (3.94e-02): par = (30)
## 1.431412    (1.29e-02): par = (23.81684)
## 1.431070    (6.35e-03): par = (24.22738)
## 1.430986    (3.03e-03): par = (24.44629)
## 1.430967    (1.42e-03): par = (24.55554)
## 1.430963    (6.62e-04): par = (24.608)
## 1.430962    (3.07e-04): par = (24.63269)
## 1.430962    (1.42e-04): par = (24.64419)
## 1.430962    (6.56e-05): par = (24.64953)
## 1.430962    (3.03e-05): par = (24.65199)
## 1.430962    (1.40e-05): par = (24.65314)
## 1.430962    (6.47e-06): par = (24.65366)
```

```
print(coef(model_chisq2))
```

```
## chisq_theta
##    24.65366
```

Like with exponential distribution, we have 2 alternative theta values deriving from different initial conditions.

Task 5

Let's figure out, which model is better for our data.

Normalized criterion

Normalized criterion for a model, which provides a good description of data should be approximately 1. In our case chi-squared model 1 (with theta 4.95...) seems to be the best.

```
model_names <- c('Normal',
                 'Exponential1',
                 'Exponential2',
                 'Chi-squared1',
                 'Chi-squared2')
models <- list(model_norm, model_exp1, model_exp2, model_chisq1, model_chisq2)
df_values <- c(197, 198, 198, 198, 198)

for (i in seq_along(models)) {
  chi_sq <- sum(((predict(models[[i]]) - df$V2)/df$V3)^2)
  n_chi_sq <- chi_sq/df_values[i]
  print(paste0('Normalized Chi-squared criterion for model ',
               model_names[i], ': ',
               round(n_chi_sq, 2)))
}
```

```
## [1] "Normalized Chi-squared criterion for model Normal: 61.78"
## [1] "Normalized Chi-squared criterion for model Exponential1: 286.57"
## [1] "Normalized Chi-squared criterion for model Exponential2: 210.38"
## [1] "Normalized Chi-squared criterion for model Chi-squared1: 1.01"
## [1] "Normalized Chi-squared criterion for model Chi-squared2: 214.76"
```

Residuals plot

Let's define a function to write fitting steps and corresponding residuals of the model.

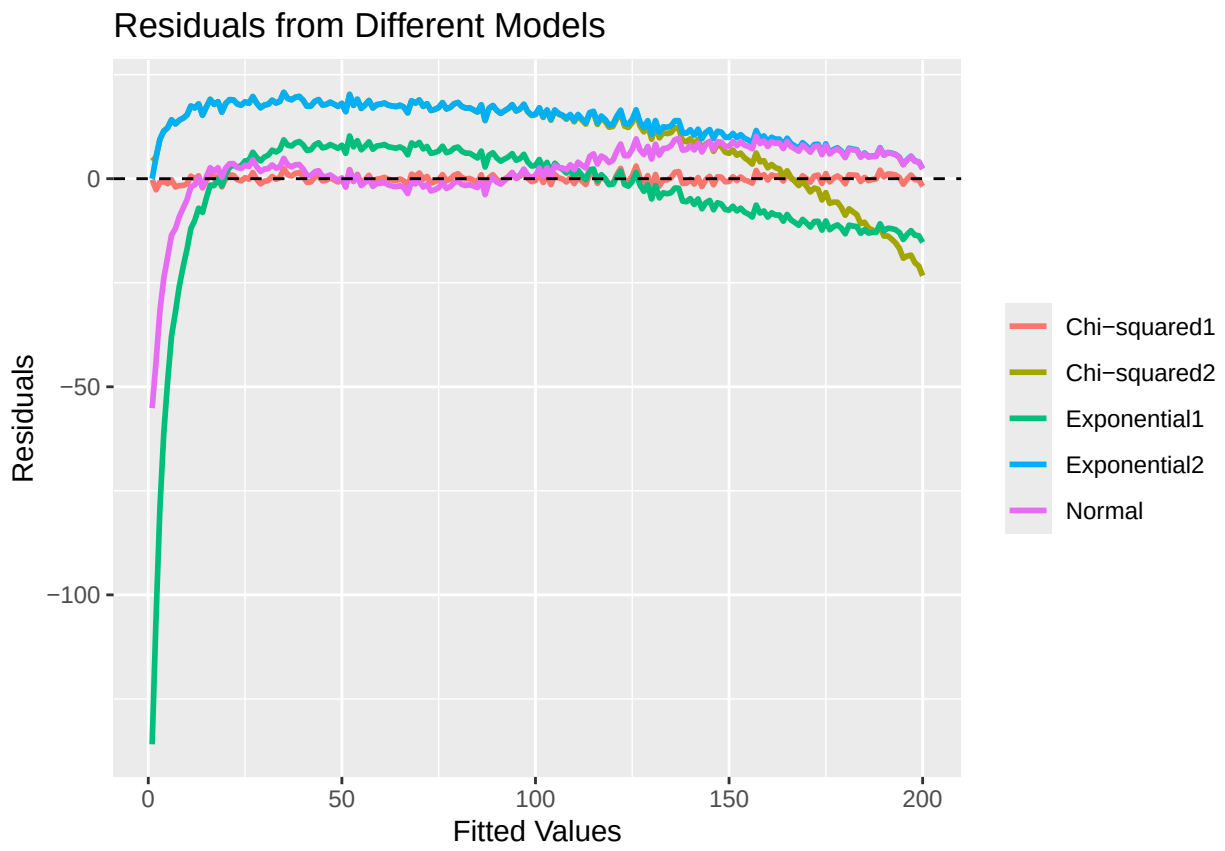
```
get_df_res <- function(models,
                       model_names,
                       data=df,
                       n_observations = 200,
                       yname='V2',
                       errname='V3') {
  df_res <- data.frame(c(1:n_observations))
  for (model in models) {
    df_res <- cbind(df_res,
                    (data[yname] - predict(model))/data[errname])
  }
  colnames(df_res) <- c('fitted', model_names)
  return(df_res)
}

df_res <- get_df_res(models = models,
           model_names = c('Normal',
                           'Exponential1',
                           'Exponential2',
                           'Chi-squared1',
                           'Chi-squared2'))
```

Due to specific of input data for ggplot, converting data frame with separate columns for different models residuals into data frame with 3 columns: fitting steps, residuals and name of the model, to which corresponds each residuals value.

```
df_res <- df_res %>%
  pivot_longer(cols = c('Normal',
                        'Exponential1',
                        'Exponential2',
                        'Chi-squared1',
                        'Chi-squared2'),
               names_to = "model",
               values_to = "residuals")
```

Make a residuals plot for all models for convenient comparison:

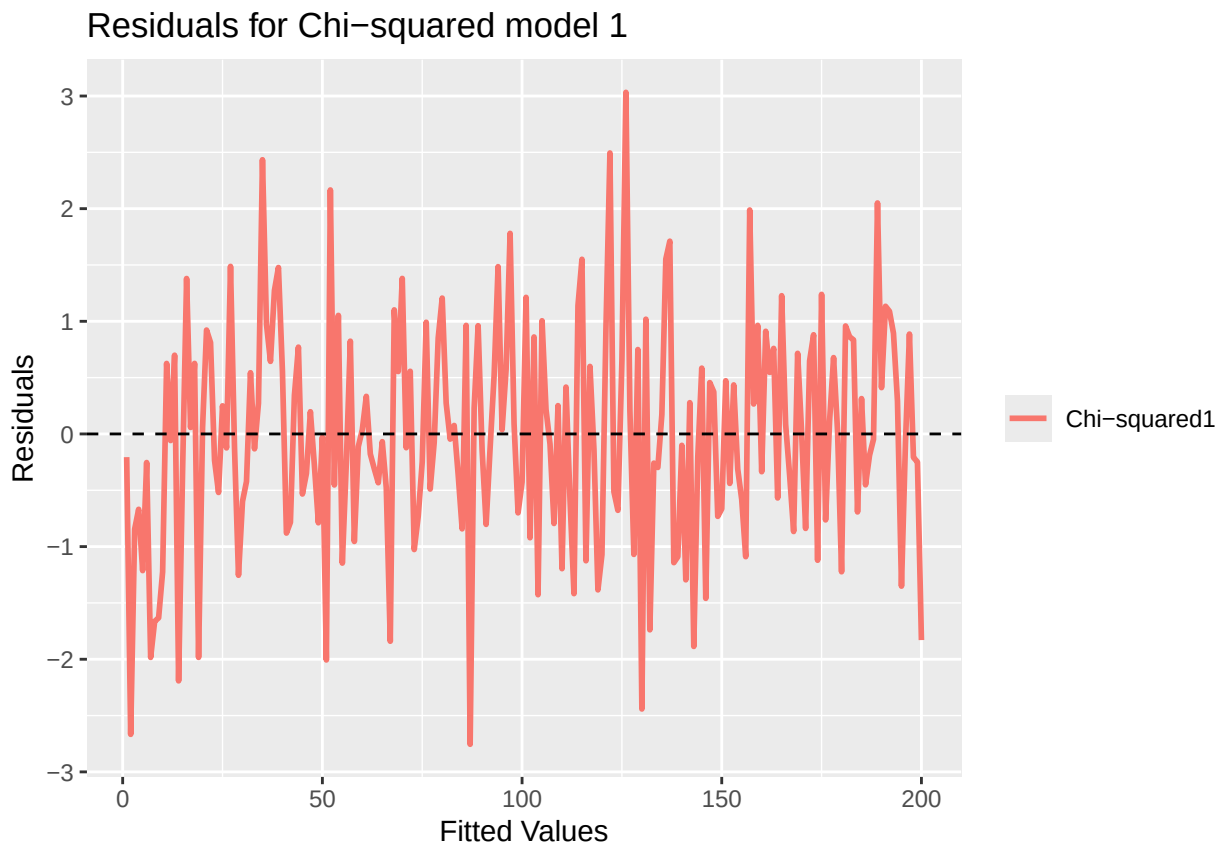
```
ggplot(df_res, aes(x = fitted, y = residuals, color = model)) +
  geom_line(linewidth = 1) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "black") +
  labs(title = "Residuals from Different Models",
       x = "Fitted Values",
       y = "Residuals") +
  theme(legend.title = element_blank())
```



As we can see, chi-squared model 1 has residuals about zero over all fitting steps. This supports the fact this model is the best for our data.

Residual plot for chi-squared 1 model only:

```
df_res_chisq1 <- get_df_res(models = list(model_chisq1),
           model_names = c('Chi-squared1'))

df_res_chisq1 <- df_res_chisq1 %>%
  pivot_longer(cols = c('Chi-squared1'),
               names_to = "model",
               values_to = "residuals")

ggplot(df_res_chisq1, aes(x = fitted, y = residuals, color = model)) +
  geom_line(linewidth = 1) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "black") +
  labs(title = "Residuals for Chi-squared model 1",
       x = "Fitted Values",
       y = "Residuals") +
  theme(legend.title = element_blank())
```

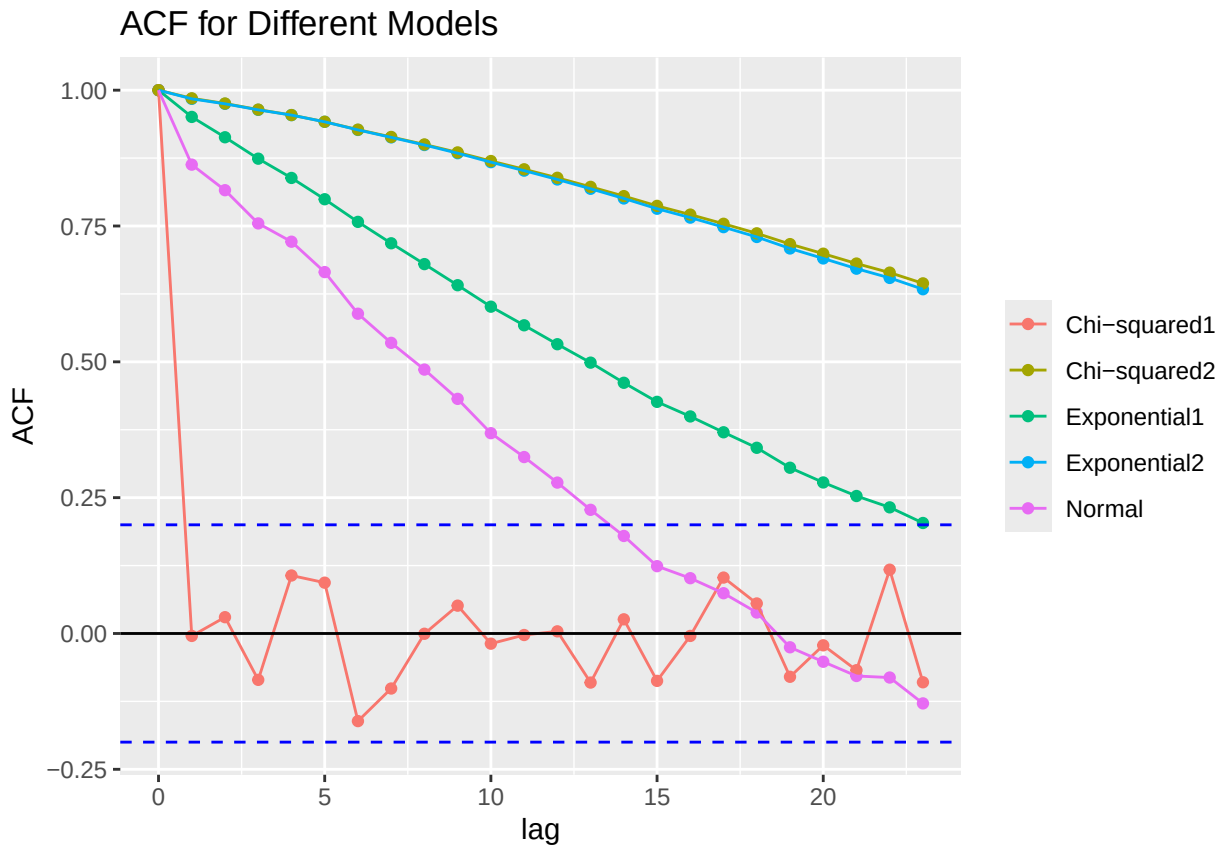


Autocorrelation function

```
get_df_acf <- function(models, model_names) {
  acf_vals <- c()
  lag_vals <- c()
  model_vals <- c()

  for (i in seq_along(models)) {
    model_residuals <- residuals(models[[i]])

    acf_result <- acf(model_residuals, plot = FALSE)

    acf_vals <- c(acf_vals, acf_result$acf)
    lag_vals <- c(lag_vals, acf_result$lag)
    model_vals <- c(model_vals, rep(model_names[i], length(acf_result$acf)))
  }

  acf_df <- data.frame(acf = acf_vals, lag = lag_vals, model = model_vals)
  return(acf_df)
}

df_acf <- get_df_acf(
  models = models,
```

```
  model_names = model_names
)
```

```
ggplot(df_acf, aes(x = lag, y = acf, color = model)) +
  geom_point() +
  geom_line() +
  geom_hline(yintercept = 0, color = "black") +
  geom_hline(yintercept = 0.2, linetype = "dashed", color = "blue") +
  geom_hline(yintercept = -0.2, linetype = "dashed", color = "blue") +
  labs(title = "ACF for Different Models",
       x = "lag",
       y = "ACF") +
  theme(legend.title = element_blank())
```



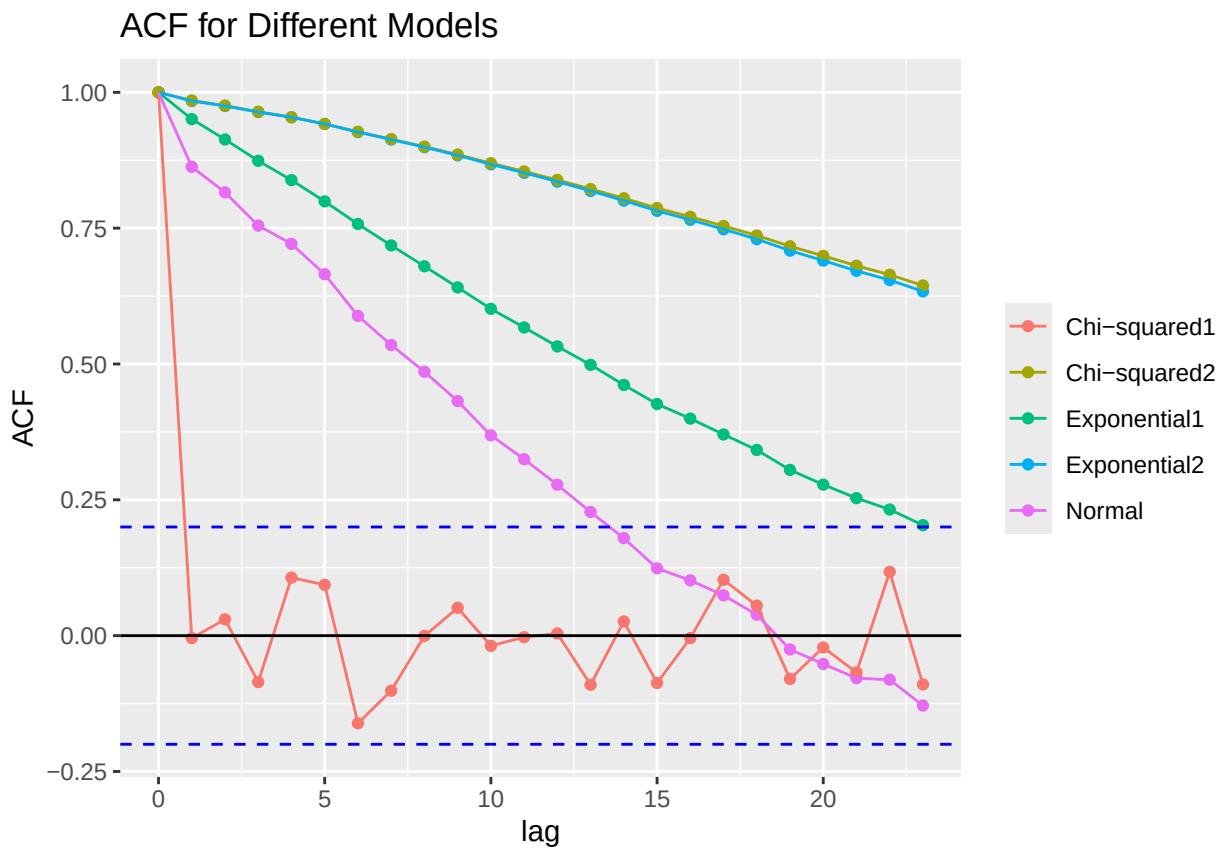
Let's check if we can create the same plot, but calculating ACF value via its formula, without using acf() function. Formula for the autocorrelation function A_k for the given lag k is (R – residuals):

$$A_k = \frac{\frac{1}{N-k+1}\sum_{i=1}^{N-k+1} R_i R_{i+k-1}}{\frac{1}{N}\sum_{i=1}^{N} R_i^2}$$

```
get_df_custom_acf <- function(models, model_names) {
  acf_vals <- c()
  lag_vals <- c()
  model_vals <- c()
```

```
  for (i in seq_along(models)) {
    model_residuals <- residuals(models[[i]])
    n <- length(model_residuals)

    denominator <- mean(model_residuals^2)

    for (lag in 0:(n - 1)) {
      if (lag == 0) {
        numerator <- mean(model_residuals^2)
      } else {
        numerator <- mean(model_residuals[1:(n - lag)] *
                          model_residuals[(lag + 1):n])
      }

      acf_value <- numerator / denominator

      acf_vals <- c(acf_vals, acf_value)
      lag_vals <- c(lag_vals, lag)
      model_vals <- c(model_vals, model_names[i])
    }
  }

  acf_df <- data.frame(acf = acf_vals, lag = lag_vals, model = model_vals)
  return(acf_df)
}

df_custom_acf <- get_df_acf(
  models = models,
  model_names = model_names
)
```

```
ggplot(df_custom_acf, aes(x = lag, y = acf, color = model)) +
  geom_point() +
  geom_line() +
  geom_hline(yintercept = 0, color = "black") +
  geom_hline(yintercept = 0.2, linetype = "dashed", color = "blue") +
  geom_hline(yintercept = -0.2, linetype = "dashed", color = "blue") +
  labs(title = "ACF for Different Models",
       x = "lag",
       y = "ACF") +
  theme(legend.title = element_blank())
```



As we can see, the result is similar, and also supports chi-squared model 1, since for this model ACF mostly oscillates around zero. That means that difference between data and model predictions is mot systemic.

Task 6

Plot 68% confidence intervals for parameters estimations.

Normal distribution parameters

For this we can use confint() function:

```
confint(model_norm, level=0.68)
```

```
## Waiting for profiling to be done...
```

```
##                    16%      84%
## norm_theta1 3.767479 3.879606
## norm_theta2 2.681783 2.778841
```

Let's calculate confidence intervals with custom function and see if results differ.

$$\overline{\theta} + t_{\alpha/2,v}\sqrt{\chi^2_v C_{jj}} \leq \theta \leq \overline{\theta} + t_{1-\alpha/2}\sqrt{\chi^2_v C_{jj}}$$

t_{\alpha/2, v}, t_{\alpha/2, v} – Student distribution quantiles;

Chi_v^2 C_jj – chi-squared scaled diagonal of covariance matrix.

```
custom_confint <- function(fitted_model, conf_level = 0.68) {
  alpha <- 1 - conf_level

  theta_hat <- coef(fitted_model)

  # degrees of freedom
  v <- df.residual(fitted_model)

  # variance-covariance matrix of the parameter estimates
  cov_matrix <- vcov(fitted_model)

  lower_bound <- numeric(length(theta_hat))
  upper_bound <- numeric(length(theta_hat))

  for (j in seq_along(theta_hat)) {
    # variance (C_jj) for the j-th parameter
    C_jj <- cov_matrix[j, j]

    # t-distribution critical values
    t_lower <- qt(alpha / 2, df = v)
    t_upper <- qt(1 - alpha / 2, df = v)

    # chi-squared value with v degrees of freedom
    chi_v <- qchisq(1 - alpha, df = v)

    margin_error <- sqrt(chi_v * C_jj)

    lower_bound[j] <- theta_hat[j] + t_lower * margin_error
    upper_bound[j] <- theta_hat[j] + t_upper * margin_error
  }

  ci_df <- data.frame(
    estimate = theta_hat,
    lower = lower_bound,
    upper = upper_bound
  )

  return(ci_df)
}

custom_confint(model_norm)
```

```
##             estimate    lower    upper
## norm_theta1 3.823572 3.008639 4.638505
## norm_theta2 2.729641 2.067279 3.392004
```

The result is different, since confint() function assumes normality of errors and does not adjust for small sample sizes with a chi-squared scaling, which can produce narrower intervals.

Exponential distribution parameters

Since we have 2 fitted exponential models, let's check confidence intervals for parameters of both.

First model

```
confint(model_exp1, level=0.68)
```

```
## Waiting for profiling to be done...

##       16%       84%
## 0.1308885 0.1451476
```

```
custom_confint(model_exp1)
```

```
##            estimate      lower     upper
## exp_theta 0.1379485 0.01330551 0.2625916
```

CI, calculated with confint() are way more narrow, than chi-squared scaled CI once again.

Second model

```
confint(model_exp2, level=0.68)
```

```
## Waiting for profiling to be done...

##      16%      84%
## 98.54966       NA
```

```
custom_confint(model_exp2)
```

```
##           estimate     lower    upper
## exp_theta 101.0805 -3190.001 3392.162
```

CI for this model parameters could not be calculated correctly. That supports our idea about wrong fitting due to wrong initial condition.

Chi-squared distribution parameters

Like with exponential models, let's check confidence intervals for parameters of both chi-squared models.

First model

```
confint(model_chisq1, level=0.68)
```

```
## Waiting for profiling to be done...

##      16%      84%
## 4.945597 4.973153
```

```
custom_confint(model_chisq1)
```

```
##             estimate    lower    upper
## chisq_theta 4.959354 4.759466 5.159241
```

Second model

```
confint(model_chisq2, level=0.68)
```

```
## Waiting for profiling to be done...
```

```
##    16%    84%
## 21.845     NA
```

```
custom_confint(model_chisq2)
```

```
##             estimate     lower    upper
## chisq_theta 24.65366 -13.69066 62.99798
```

CI for this model parameters could not be calculated correctly. That supports our idea about wrong fitting due to wrong initial condition.

Optional: check approximation

The best model for our data id chi-squared with theta 4.95... Let's visualize it:

```
df_pred <- data.frame(data_x = df$V1,
                      data_y = df$V2,
                      chisq1 = predict(model_chisq1),
                      chisq2 = predict(model_chisq2),
                      exp1 = predict(model_exp1),
                      exp2 = predict(model_exp2),
                      normal = predict(model_norm))
```

```
ggplot(df_pred, aes(x = data_x, y = data_y)) +
  geom_point() +
  geom_line(aes(x = data_x, y = chisq1, color = 'Chi-squared 1'),
            lwd = 1.2, alpha = 0.7) +
  geom_line(aes(x = data_x, y = chisq2, color = 'Chi-squared 2'),
            lwd = 1, alpha = 0.7) +
  geom_line(aes(x = data_x, y = exp1, color = 'Exponential 1'),
            lwd = 1, alpha = 0.5) +
  geom_line(aes(x = data_x, y = exp2, color = 'Exponential 2'),
            lwd = 1, alpha = 0.7) +
  geom_line(aes(x = data_x, y = normal, color = 'Normal'),
            lwd = 1, alpha = 0.7) +
  labs(title = "Data and Model Approximations", y = "Mass", x = "Length") +
  scale_color_manual(
```

```
    values = c('Chi-squared 1' = 'purple',
               'Chi-squared 2' = 'blue',
               'Exponential 1' = 'red',
               'Exponential 2' = 'orange',
               'Normal' = 'yellow')
) +
theme(legend.title = element_blank())
```