Forecasting issues

Forecast Padawan 2

November 17, 2016

The goal of this experiment is to design the best model to forcaste the number of issue in the per day in the comming two weeks. We think that this could help Open Source organisation to manage there human ressources.

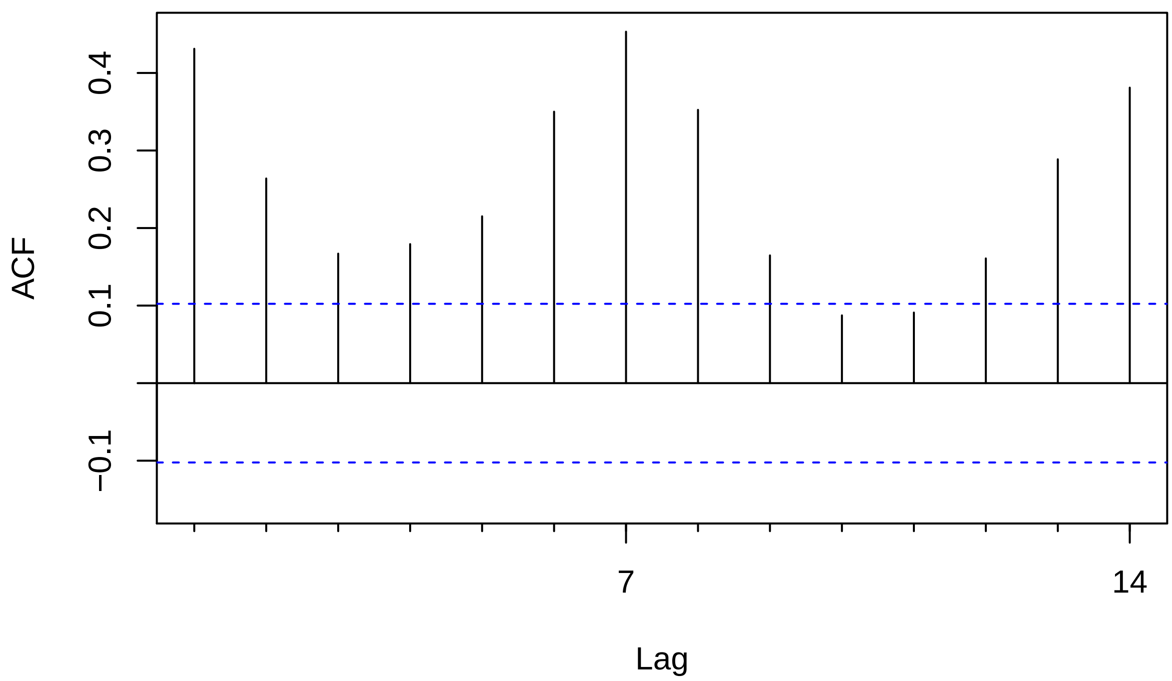
Load the data

```
#install.packages('forecast')

library('forecast')
library(knitr)
#load the data frame
repository.csv <- read.csv("time_series/julialang_julia_daily.csv")

repository.csv$date = as.POSIXlt(as.Date(repository.csv$date,format='%Y-%m-%d'))
```

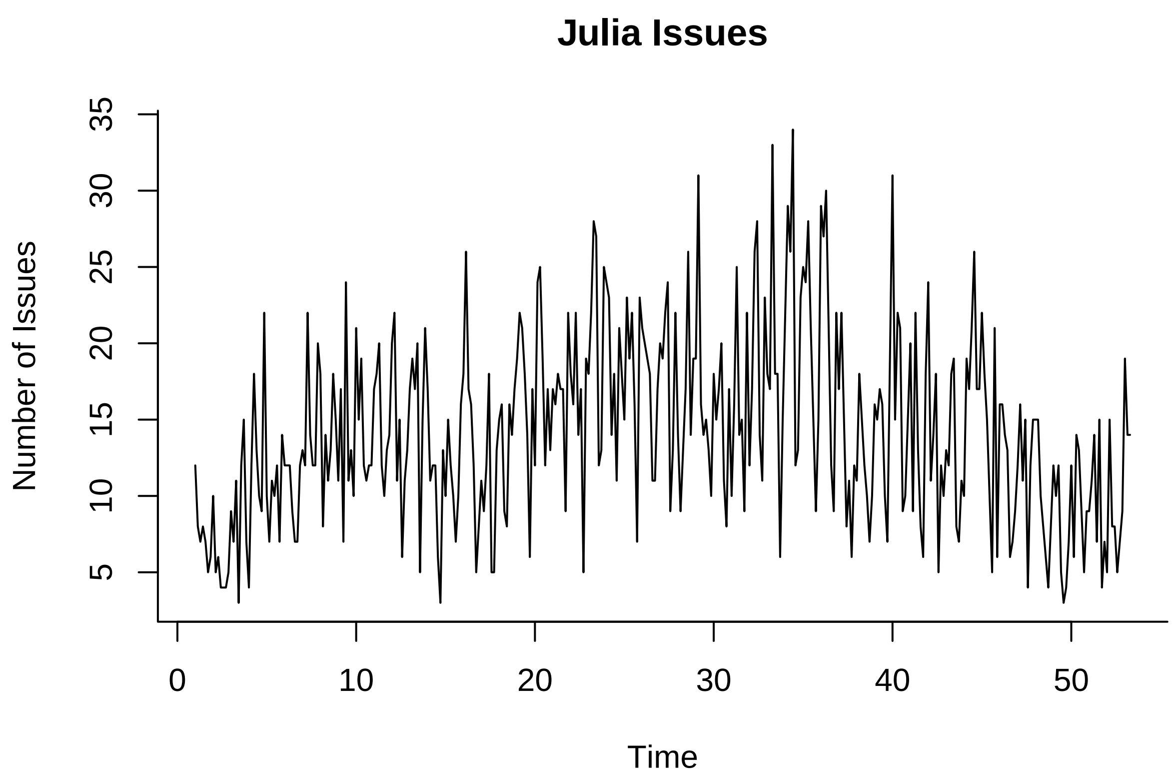
keep the last 12 months

```
to_date <- repository.csv$date[length(repository.csv$date)]
from_date <- to_date
from_date$year <- from_date$year - 1

repository.csv <- subset(repository.csv, date <= to_date & date >= from_date)
```

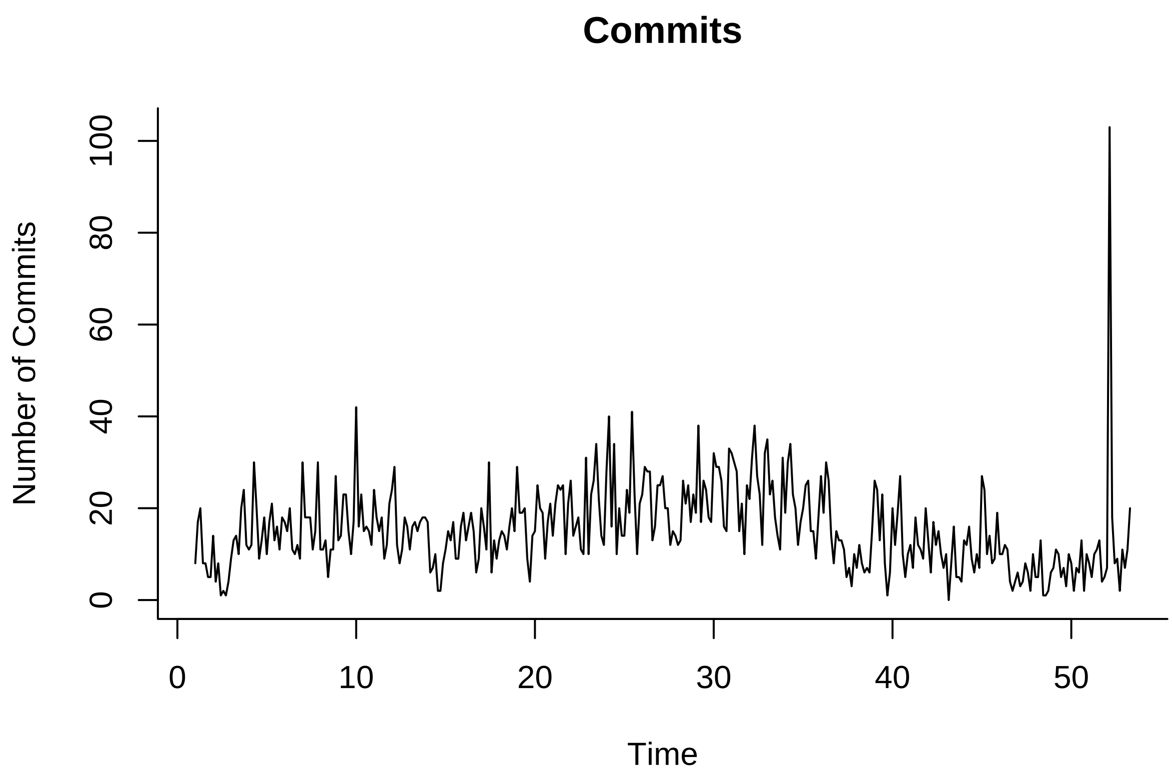
```
#loading issues and commits into a ts object
issues.ts <- ts(repository.csv$number_of_issues, frequency = 7)


Acf(issues.ts, lag.max = 14, main = "")
```




```r
commits.ts <- ts(repository.csv$number_of_commits, frequency = 7)
pull_requests.ts <- ts(repository.csv$number_of_pull_requests, frequency = 7)

plot(issues.ts, main = 'Julia Issues', bty = 'l', ylab = 'Number of Issues')
```



```
plot(commits.ts, main = 'Commits', bty = 'l', ylab = 'Number of Commits')
```




```
time <- time(issues.ts)

n.sample <- 28
n.valid <- 21


separate.train.test <- function(timeserie, n.valid) {
  time <- time(timeserie)
  n.train <- length(timeserie) - n.valid
  results <- list()
  results$train.ts <- window(timeserie, start=time[1], end=time[n.train])
  results$valid.ts <- window(timeserie, start=time[n.train+1], end=time[n.train+n.valid])
  return(results)
}
# create a matrix of 14 column, each column is a time series create by rolling forward
all.issues <- sapply(0:(n.sample - 1), function(i) return(separate.train.test(window(issues.ts,start=ti
all.commits <- sapply(0:(n.sample - 1), function(i) return(separate.train.test(window(commits.ts,start=

issues  <- separate.train.test(issues.ts, n.valid)
commits  <- separate.train.test(commits.ts, n.valid)
```

```
# utility functions
# all.forecast is a matirx of 21(length of validation period) * 14(14 rolling forward)
mean.all.accuracy <- function(all.forecast) {
  Reduce("+", all.forecast['summary',])/length(all.forecast['summary',])
}
```

```r
plot.all.residuals <- function(all.forecast) {
  plot(1, type="l", main="Residuals", xlim=c(35, 53.3), ylim=c(-40, 40), xlab = 'Week', ylab = 'Errors')
  sapply(1:n.sample, function(i) lines(all.forecast['train', i]$train - all.forecast['fitted', i]$fitted
  sapply(1:n.sample, function(i) lines(all.forecast['residual',i]$residual, col = 'blue'))
  return(NULL)
}

plot.all.pred <- function(all.forecast) {
  plot(issues.ts, main="Prediction", xlim=c(35, 53.3), xlab = 'Week', ylab = 'Number of Issues')
  if (class(all.forecast['pred',1]$pred) == "forecast") {
    sapply(1:n.sample, function(i) lines(all.forecast['pred',i]$pred$mean, col=rgb(0, 0, 1, 0.5)))
  } else {
    sapply(1:n.sample, function(i) lines(all.forecast['pred',i]$pred, col=rgb(0, 0, 1, 0.5)))
  }
  return(NULL)
}

plot.pred <- function(forecast.with.interval.ts) {
  plot(issues.ts, main="Prediction Interval", xlim=c(35, 53.3), xlab = 'Week', ylab = 'Number of Issues
  # how to plot shade, why is it not working here...~''
  apply(forecast.with.interval.ts, 2, function(x) lines(x))
  return(NULL)
}

hist.all.residuals <- function(all.forecast) {
  residuals <- sapply(1:n.sample, function(i) as.numeric(all.forecast['residual',i]$residual))
  hist(residuals)
  quantile(residuals,c(0.975,0.90,0.10,0.025))
}

# plot the boxplot of 21 validation period prediction residuals
boxplot.all.residuals <- function(all.forecast) {
  residuals <- sapply(1:n.sample, function(i) as.numeric(all.forecast['residual',i]$residual))
  boxplot(apply(residuals, 1, quantile.helper))
  return (quantile(residuals, c(0.975,0.90,0.10,0.025)))
}

# retrun the vector of qunatile of 0.975, 0.90, 0.10, 0.025
quantile.helper <- function(matrix) {
  return (quantile(matrix, c(0.975, 0.90, 0.10, 0.025)))
}

# get the quantile of each point prediction
get.quantile.of.residuals <- function(all.forecast) {
  residuals <- sapply(1:n.sample, function(i) as.numeric(all.forecast['residual',i]$residual))
  return (apply(residuals, 1, quantile.helper))
}

forecast.confidence <- function(ets.test.model.pred, quantile.of.residuals) {
  forecast.confidence.interval <- apply(quantile.of.residuals, 1, function(a.quantile) return(a.quantile
  return(forecast.confidence.interval)
}
```

```r
forecast.manual.interval <- function(x.train, f.train, f.pred, f.lower, f.upper) {
  mean <- f.pred
  x <- x.train
  residuals <- x.train - f.train
  fitted <- f.train
  level <- c(80, 95)
  lower <- f.lower
  upper <- f.upper

  # Construct output list
  output <- list(mean=mean, x=x, residuals=residuals, fitted=fitted, level=level, lower=lower, upper=upp
  # Return with forecasting class
  return(structure(output, class='forecast'))
}

# to build custom forecast object
forecast.manual <- function(x.train, f.train, f.pred) {
  mean <- f.pred
  x <- x.train
  residuals <- x.train - f.train
  fitted <- f.train

  # Construct output list
  output <- list(mean=mean, x=x, residuals=residuals, fitted=fitted)
  # Return with forecasting class
  return(structure(output, class='forecast'))
}
```

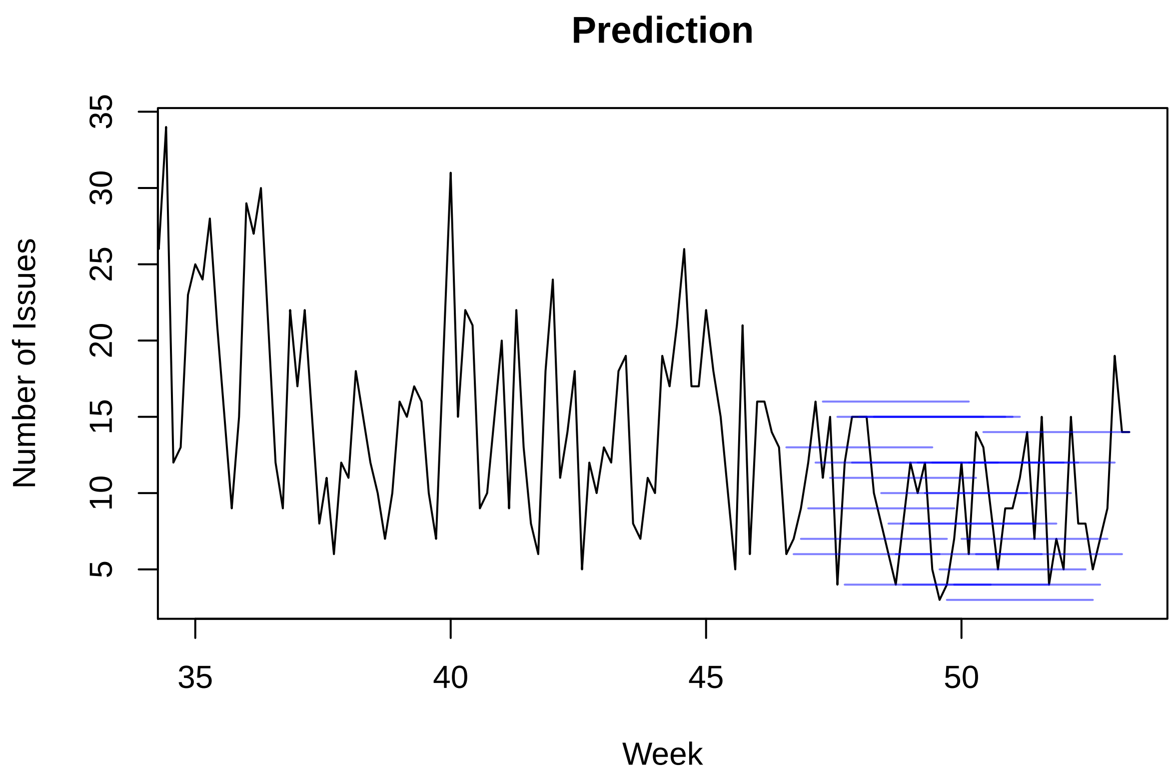
Naive Forecast

Naive

```r
naive.forecast <- function(sample) {
  results <- list()
  results$train <- sample$train.ts
  results$valid <- sample$valid.ts
  results$pred <- naive(sample$train.ts, h=n.valid)
  results$fitted <- results$pred$fitted
  results$residual <- sample$valid.ts - results$pred$mean
  results$summary <- accuracy(results$pred, sample$valid.ts)
  return(results)
}

all.naive.forecast <- sapply(1:n.sample, function(i) return(naive.forecast(all.issues[,i])))

kable(mean.all.accuracy(all.naive.forecast))
```

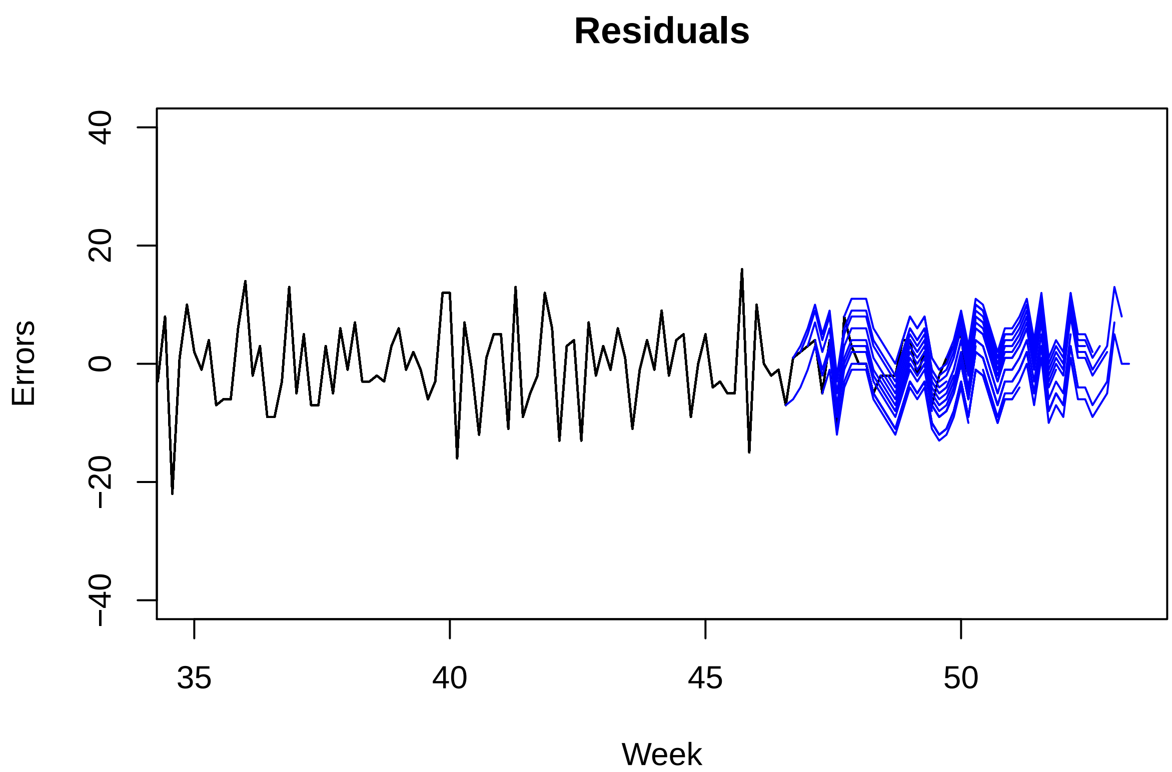
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.0069311	6.637465	5.202128	-13.75767	42.81377	1.0314978	-0.3466143	NA
Test set	-0.4914966	5.389969	4.552721	-30.65311	65.10251	0.9026147	0.1800717	1.340516

```r
plot.all.pred(all.naive.forecast)
```



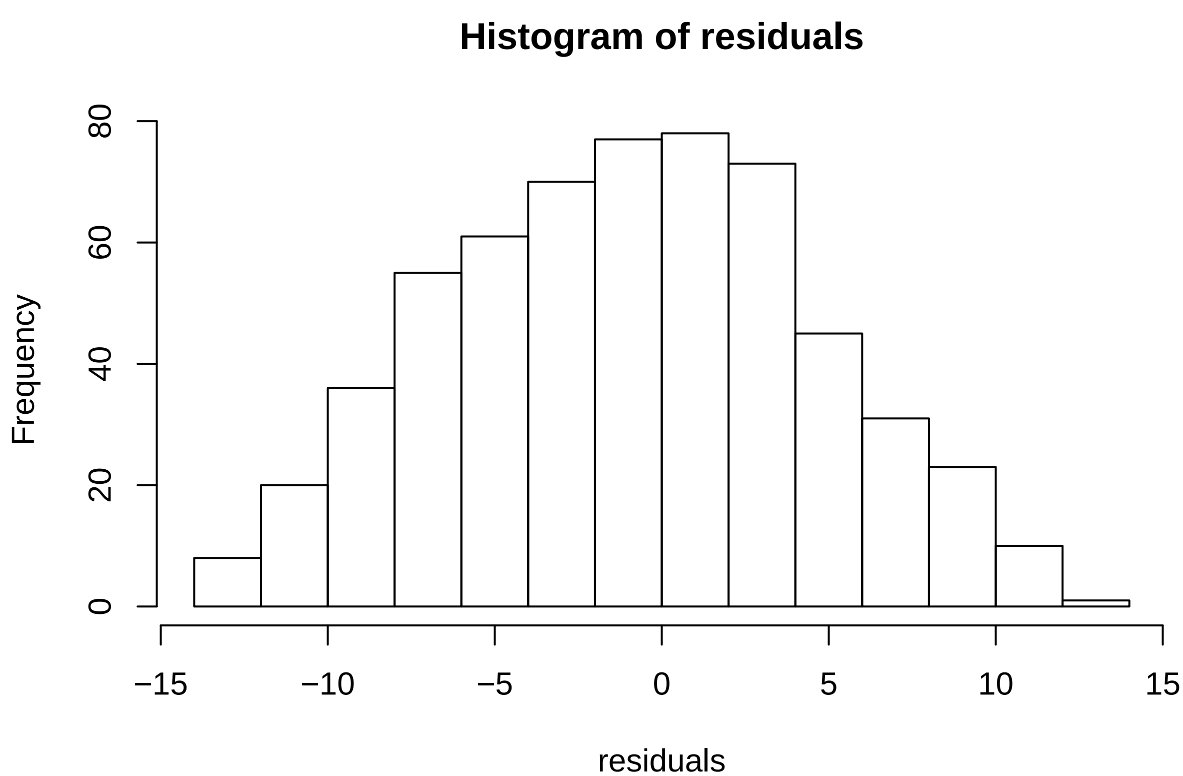

```
## NULL
```

```r
plot.all.residuals(all.naive.forecast)
```



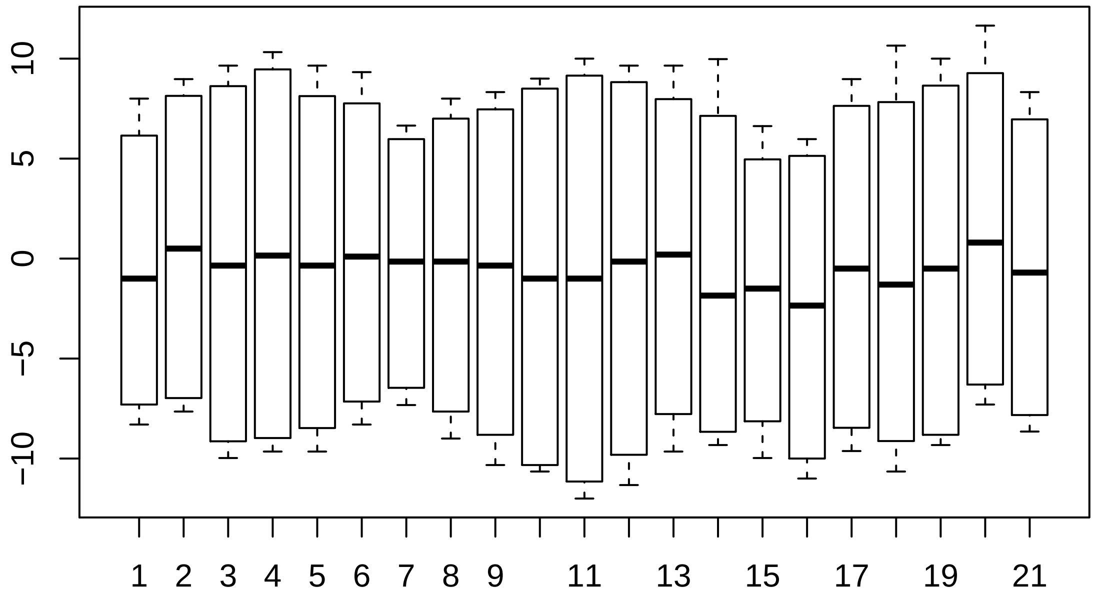

```
## NULL
```

```
hist.all.residuals(all.naive.forecast)
```



```
## 97.5%    90%    10%   2.5%
##    10      7     -8    -11
```

```
boxplot.all.residuals(all.naive.forecast)
```



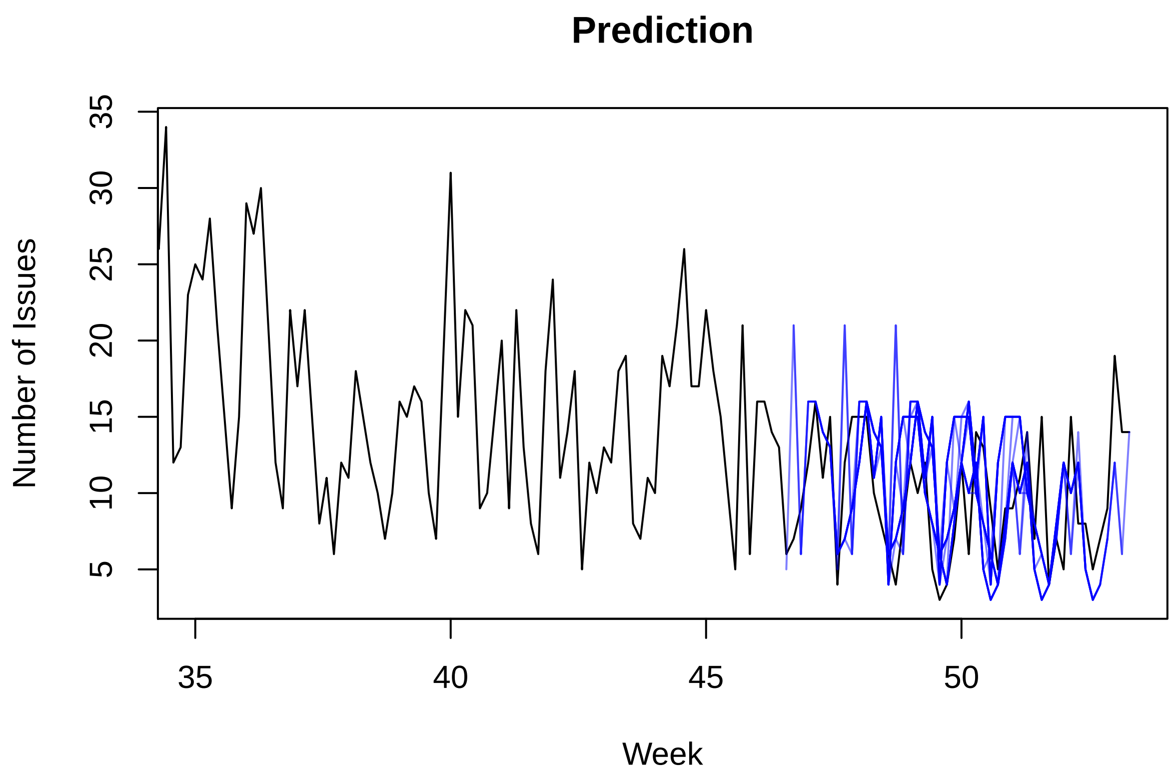

```
## 97.5%    90%    10%   2.5%
##    10      7     -8    -11
```

Seasonal Naive

```
snaive.forecast <- function(sample) {
  results <- list()
  results$train <- sample$train.ts
  results$valid <- sample$valid.ts
  results$pred <- snaive(sample$train.ts, h=n.valid)
  results$fitted <- results$pred$fitted
  results$residual <- sample$valid.ts - results$pred$mean
  results$summary <- accuracy(results$pred, sample$valid.ts)

  return(results)
}

all.snaive.forecast <- sapply(1:n.sample, function(i) return(snaive.forecast(all.issues[,i])))

kable(mean.all.accuracy(all.snaive.forecast))
```

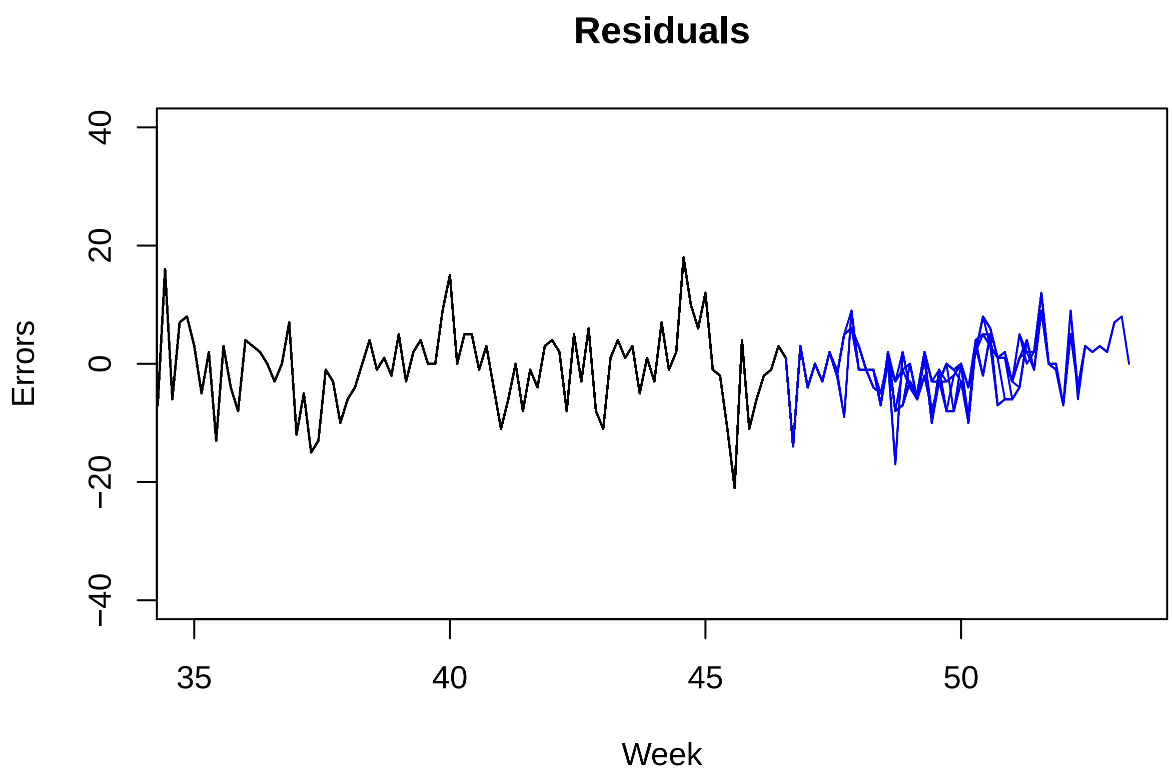
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0570770	6.440847	5.043303	-11.54706	39.95075	1.0000000	0.1271297	NA
Test set	-0.9965986	4.739375	3.778912	-26.55378	50.76779	0.7494087	-0.0616888	1.112054

```r
plot.all.pred(all.snaive.forecast)
```



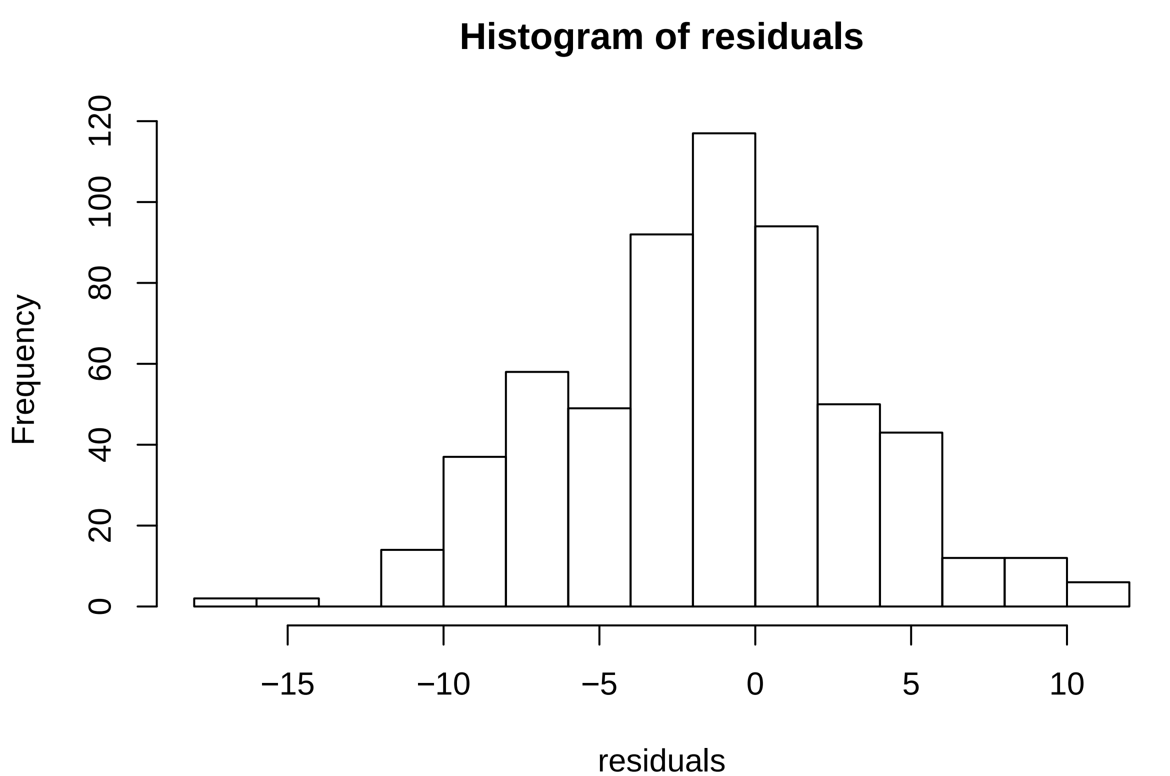

```
## NULL
```

```r
plot.all.residuals(all.snaive.forecast)
```



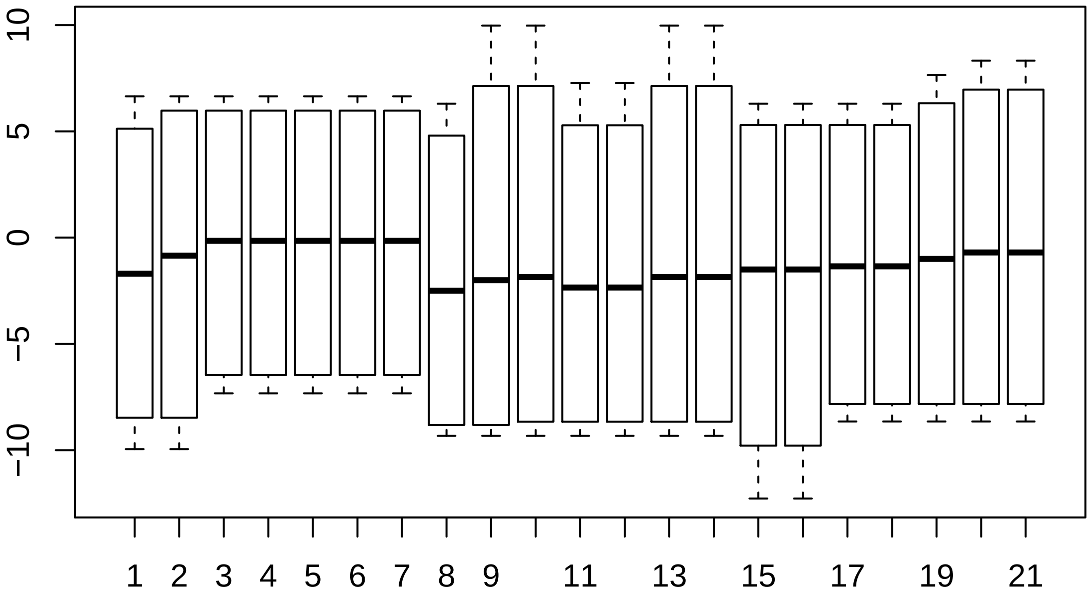

```
## NULL
```

```
hist.all.residuals(all.snaive.forecast)
```



```
## 97.5%   90%   10%  2.5%
##     9     5    -7   -10
```

```
boxplot.all.residuals(all.snaive.forecast)
```




```
## 97.5%   90%   10%  2.5%
##     9     5    -7   -10
```

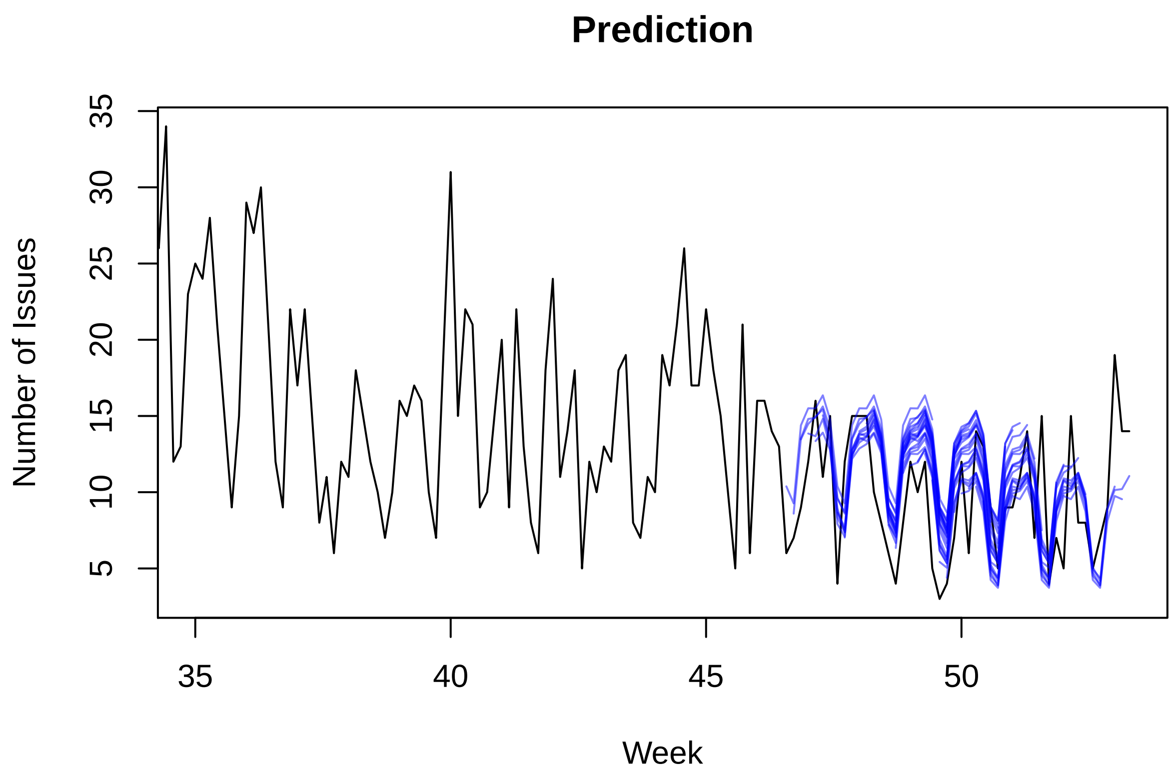
Smoothing

Exponential smoothing ZNA

```
hw.forecast <- function(sample) {
  results <- list()
  results$train <- sample$train.ts
  results$valid <- sample$valid.ts
  results$model <- ets(sample$train.ts, model = "ZNA")
  results$pred <- forecast(results$model, h=n.valid)
  results$fitted <- results$pred$fitted
  results$residual <- sample$valid.ts - results$pred$mean
  results$summary <- accuracy(results$pred, sample$valid.ts)
  return(results)
}


all.hw.forecast <- sapply(1:n.sample, function(i) return(hw.forecast(all.issues[,i])))

kable(mean.all.accuracy(all.hw.forecast))
```

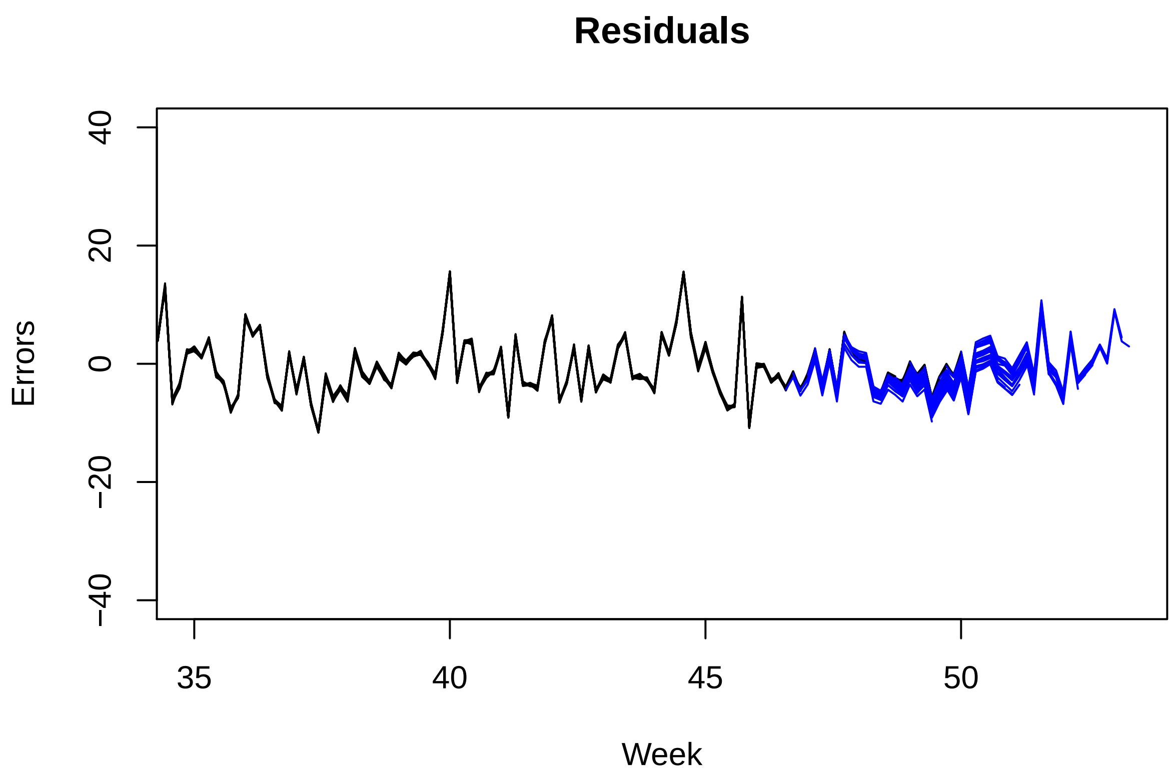
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0769785	4.752271	3.768151	-11.11154	31.66913	0.7471643	0.0645979	NA
Test set	-1.6468916	3.940982	3.255232	-34.62224	47.20882	0.6450005	0.0149976	0.9374334

```
plot.all.pred(all.hw.forecast)
```



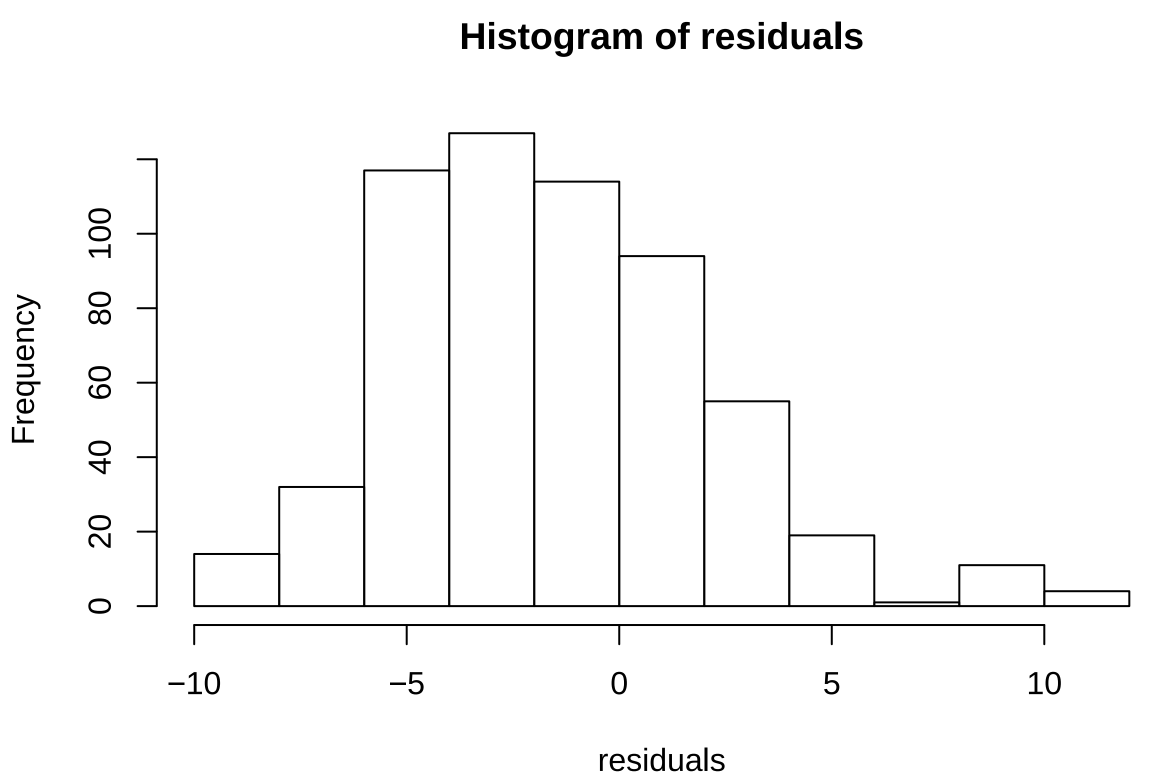

```
## NULL
```

```
plot.all.residuals(all.hw.forecast)
```



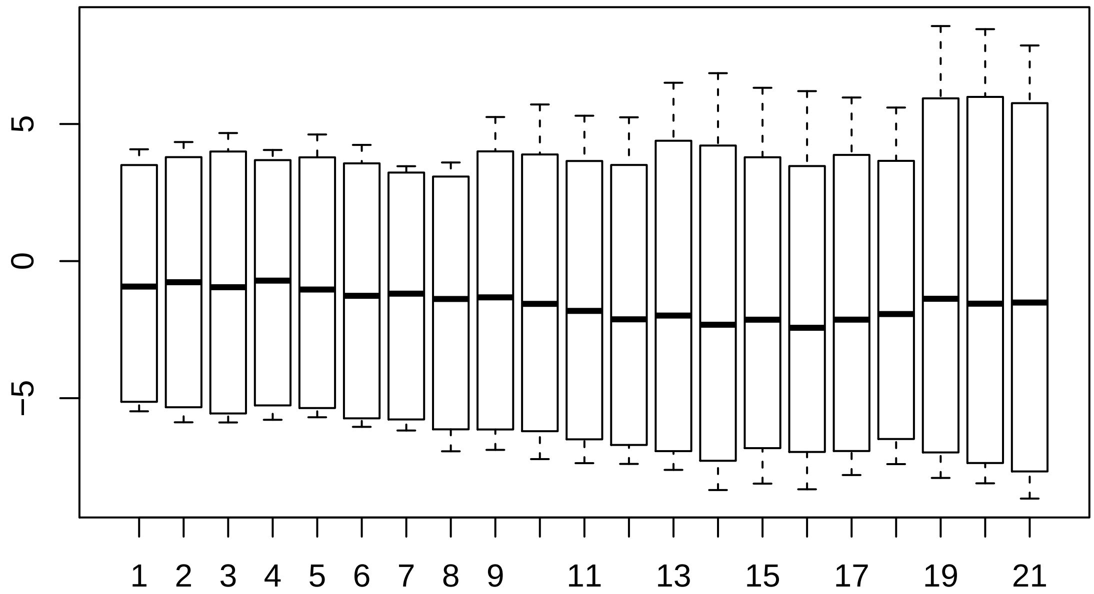

```
## NULL
```

```
hist.all.residuals(all.hw.forecast)
```



```
##     97.5%       90%       10%      2.5%
##  7.685960  2.936361 -5.733590 -7.919203
```

```
boxplot.all.residuals(all.hw.forecast)
```



```
##     97.5%       90%       10%      2.5%
##  7.685960  2.936361 -5.733590 -7.919203
```

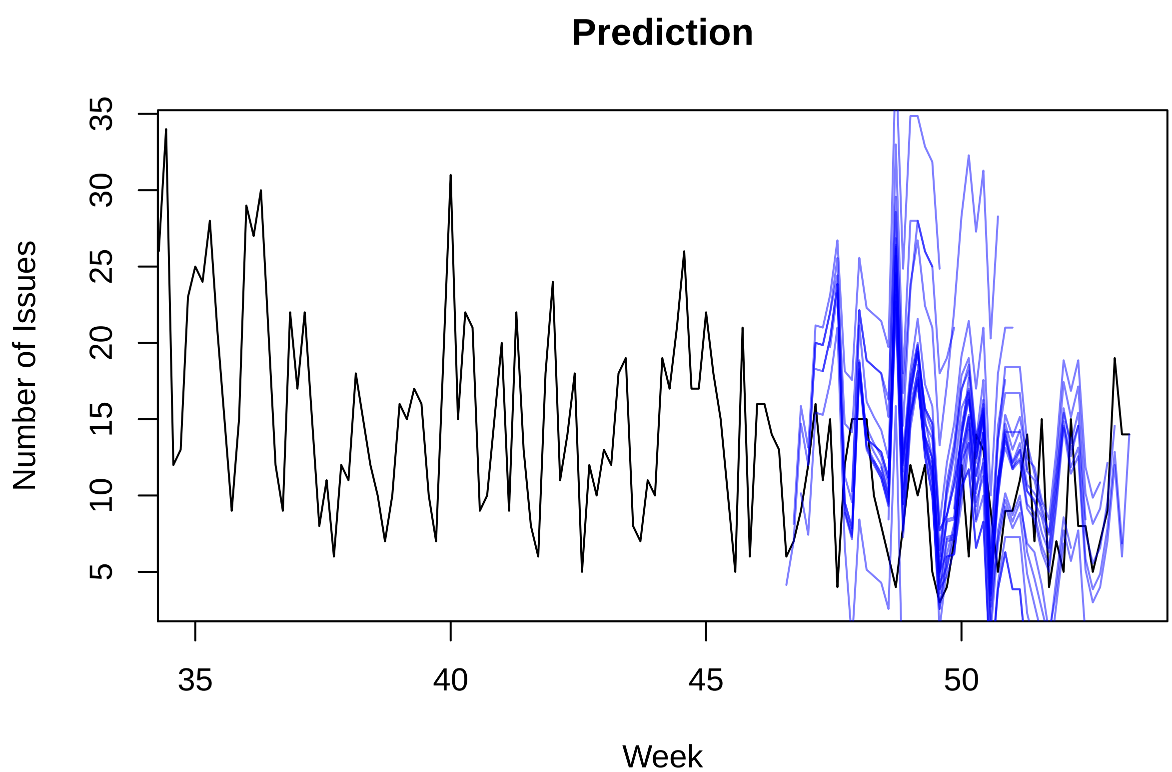
Double differencing

```
ma.dd.forecast <- function(sample) {
  train.issues.d1 <- diff(sample$train.ts, lag = 1)
  train.issues.d1.d7 <- diff(train.issues.d1, lag = 7)

  ma.trailing <- rollmean(train.issues.d1.d7, k = 7, align = "right")
  last.ma <- tail(ma.trailing, 1)
  ma.trailing.pred <- ts(c(ma.trailing, rep(last.ma, n.valid)), start=c(3, 1), frequency = 7)

  ma.dd.pred.d1 <- train.issues.d1
  ma.dd.pred <- sample$train.ts

  for(i in 1:(n.valid/7)) {
    ma.dd.pred.d1 <- ma.trailing.pred + lag(ma.dd.pred.d1,k = -7)
    ma.dd.pred <- ma.dd.pred.d1 + lag(ma.dd.pred,k = -8)
  }

  results <- list()
  results$train <- sample$train.ts
  results$valid <- sample$valid.ts

  valid.time <- time(results$valid)
  train.time <- time(results$train)
```

```
  dd.fitted <- window(ma.dd.pred, start=c(5,3), end=end(train.time), frequency=frequency(train.time))
  dd.pred <- window(ma.dd.pred, start=start(valid.time), end=end(valid.time), frequency=frequency(valid

  results$pred <- forecast.manual(window(results$train, start=c(5,3)), dd.fitted, dd.pred)
  results$fitted <- results$pred$fitted

  results$residual <- sample$valid.ts - results$pred$mean
  results$summary <- accuracy(results$pred, sample$valid.ts)

  return(results)
}

all.ma.dd.forecast <- sapply(1:n.sample, function(i) return(ma.dd.forecast(all.issues[,i])))

kable(mean.all.accuracy(all.ma.dd.forecast))
```

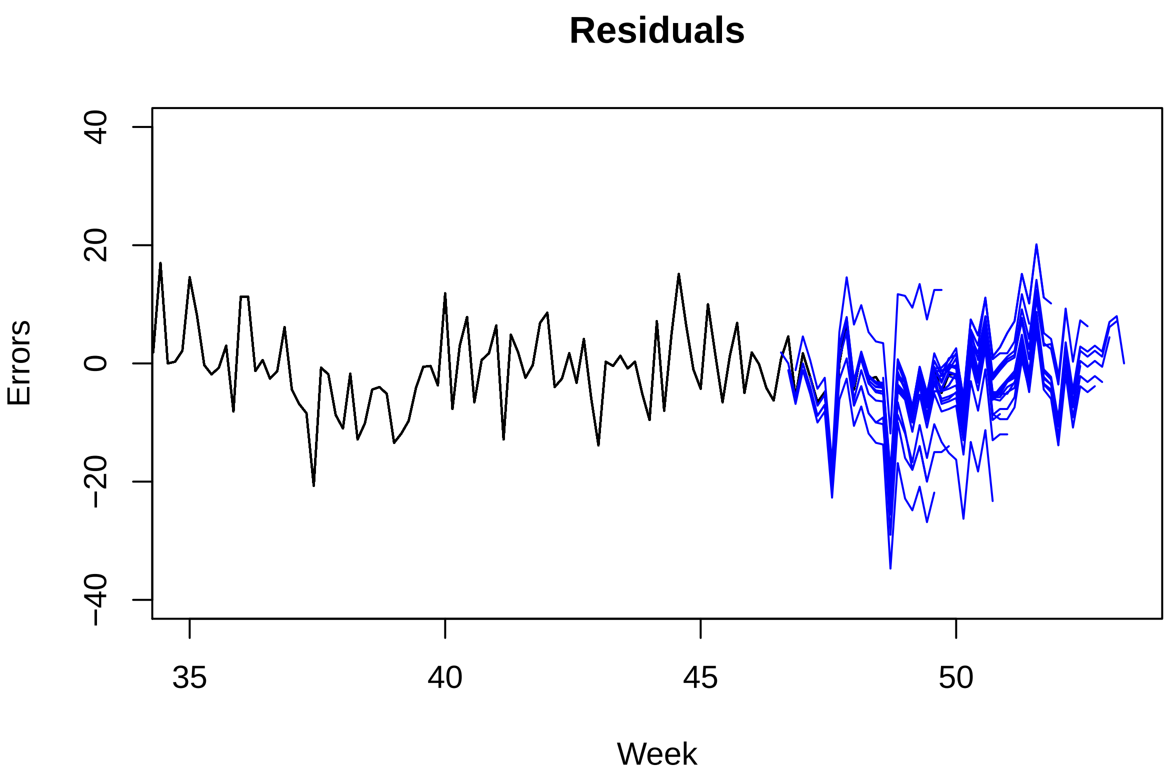
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.2352895	6.223104	4.877039	-8.188283	36.53273	0.9566381	0.1491360	NA
Test set	-3.2108844	7.660966	6.131195	-61.692651	91.64038	1.1998645	0.0352088	1.718854

```
plot.all.pred(all.ma.dd.forecast)
```



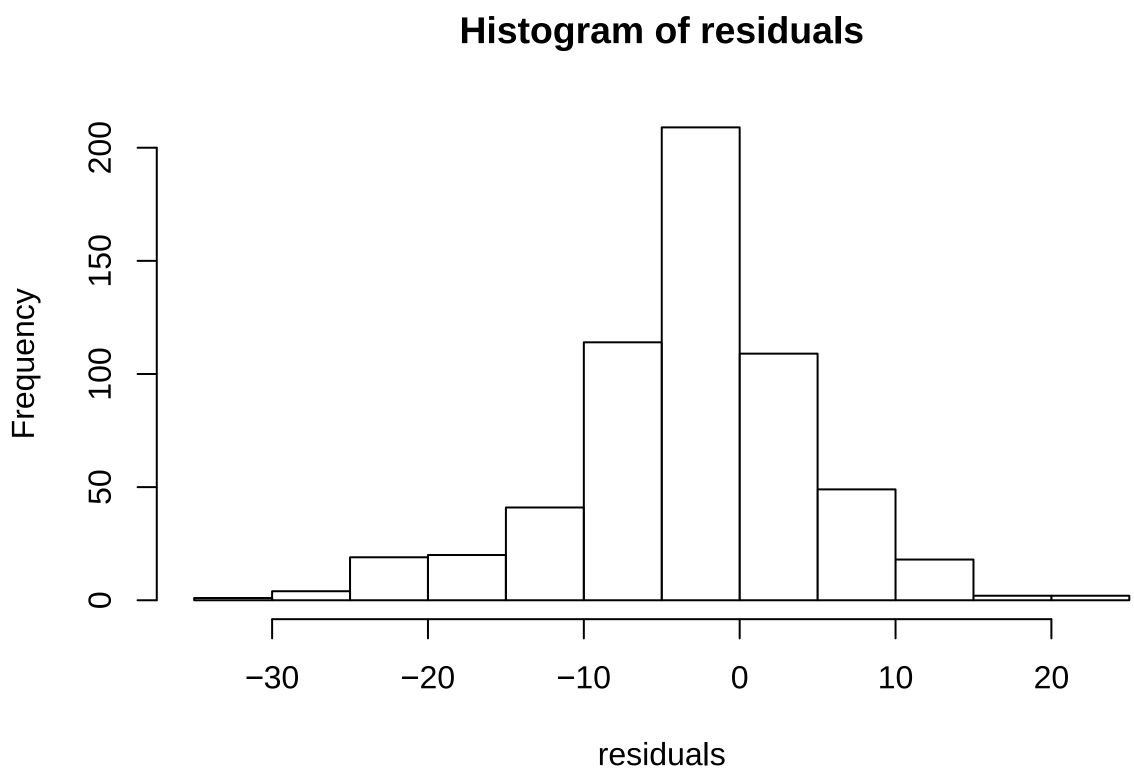

```
## NULL
```

```r
plot.all.residuals(all.ma.dd.forecast)
```



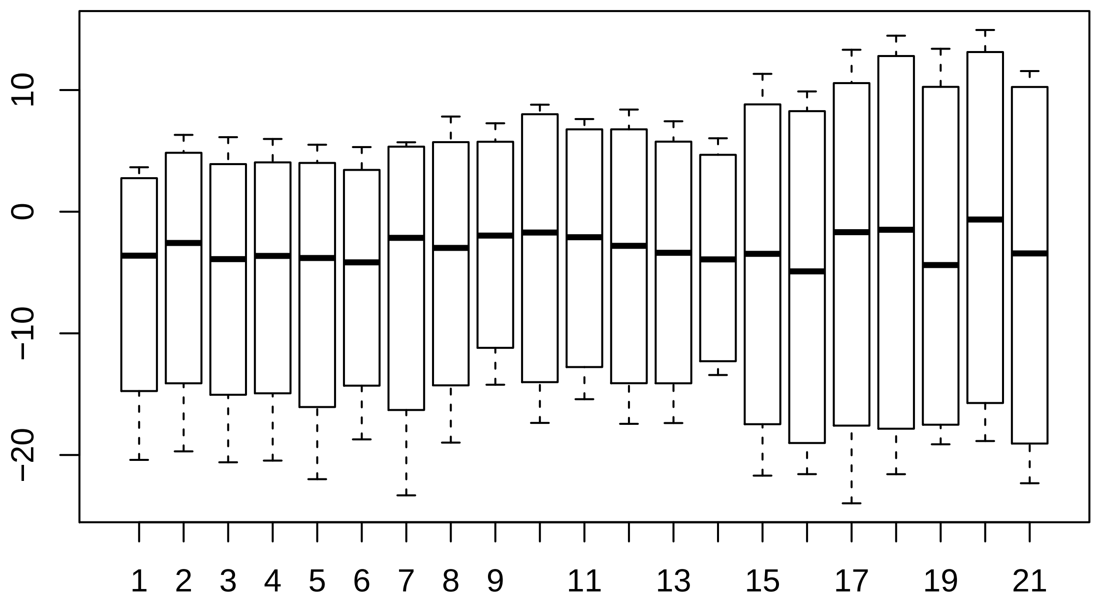

```
## NULL
```

```r
hist.all.residuals(all.ma.dd.forecast)
```




```
##       97.5%         90%         10%        2.5%
##   11.142857    5.614286 -11.900000 -21.760714
```

```r
boxplot.all.residuals(all.ma.dd.forecast)
```




```
##       97.5%         90%         10%        2.5%
##   11.142857    5.614286 -11.900000 -21.760714
```

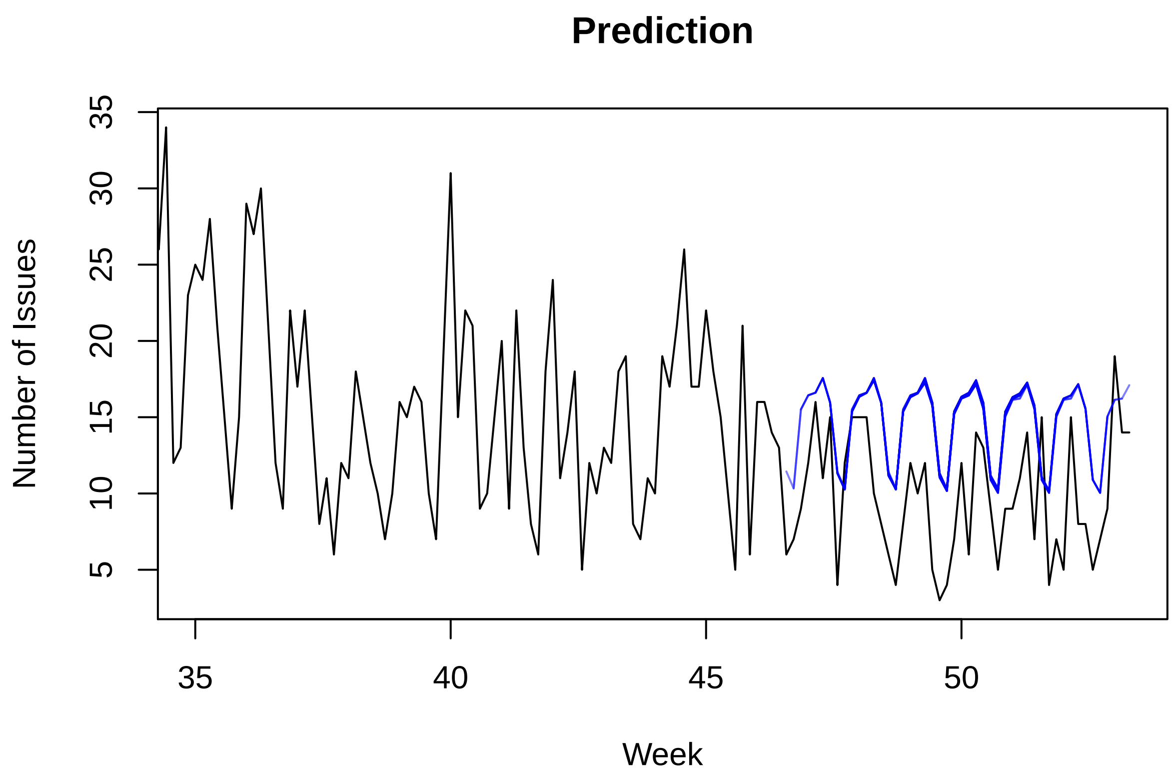
Regression

Linear additive regression season

```
regr.add.forecast <- function(sample) {
  results <- list()
  results$train <- sample$train.ts
  results$valid <- sample$valid.ts
  results$model <- tslm(sample$train.ts ~ season)
  results$pred <- forecast(results$model, h=n.valid)
  results$fitted <- results$pred$fitted
  results$residual <- sample$valid.ts - results$pred$mean
  results$summary <- accuracy(results$pred, sample$valid.ts)

  return(results)
}

all.regr.add.forecast <- sapply(1:n.sample, function(i) return(regr.add.forecast(all.issues[,i])))

kable(mean.all.accuracy(all.regr.add.forecast))
```

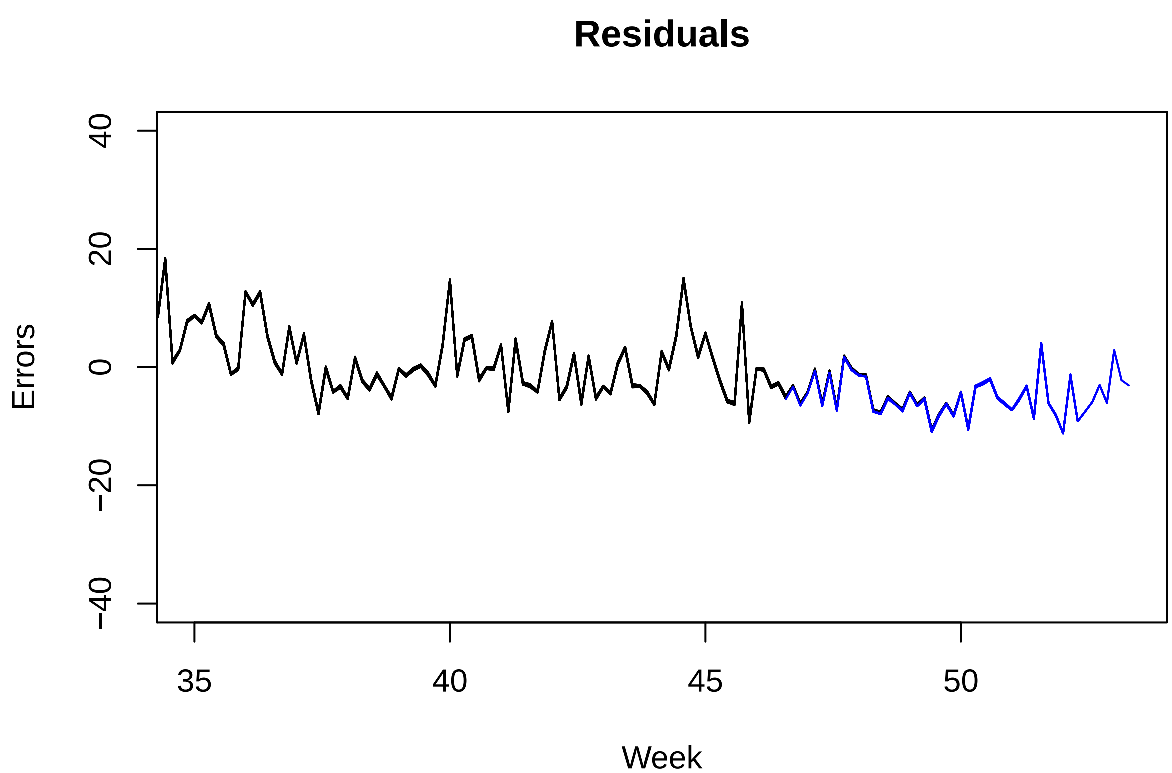
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.00000	5.482428	4.350410	-18.97454	39.40074	0.8626839	0.3915251	NA
Test set	-5.47042	6.339038	5.729814	-85.65481	87.43106	1.1364039	0.0145259	1.538883

```
plot.all.pred(all.regr.add.forecast)
```



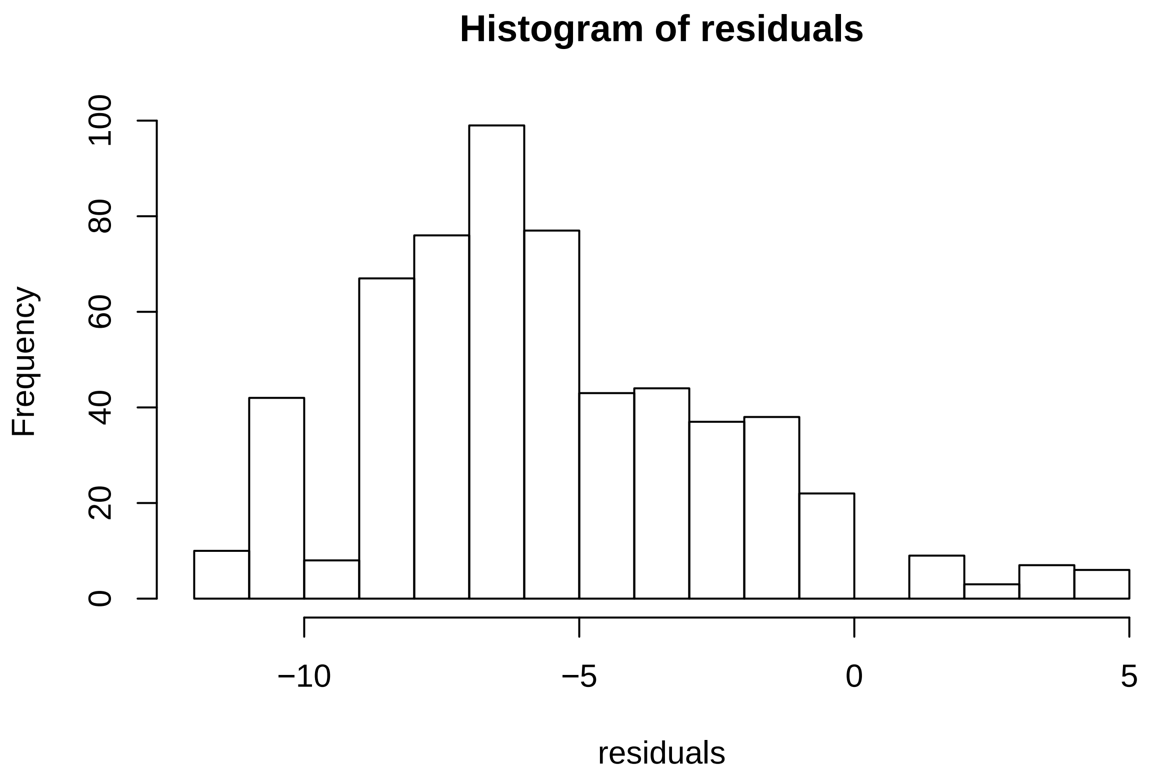

```
## NULL
```

```
plot.all.residuals(all.regr.add.forecast)
```



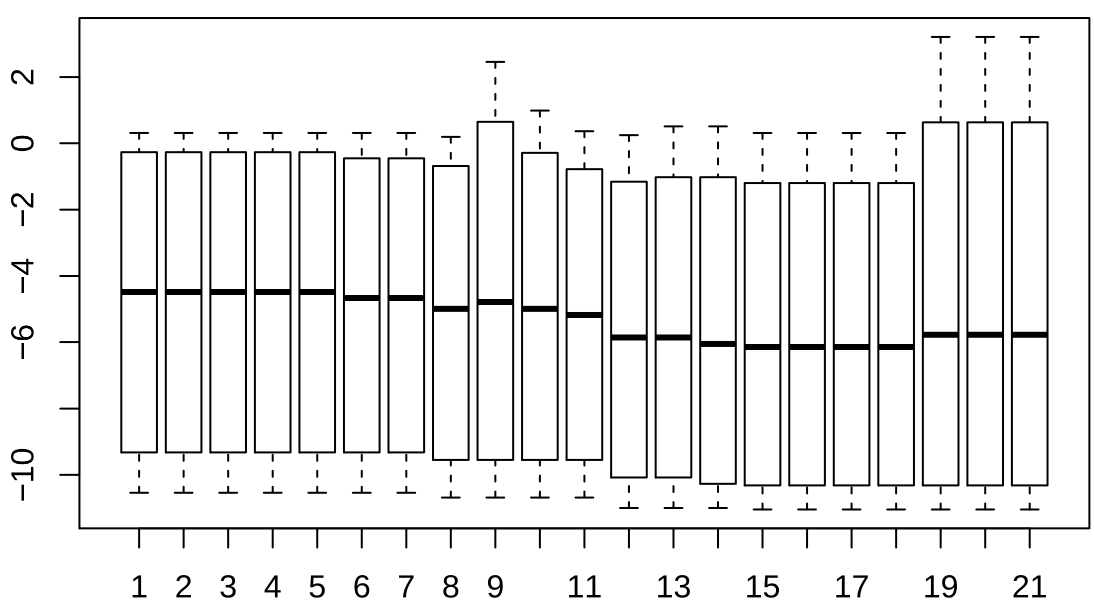

```
## NULL
```

```r
hist.all.residuals(all.regr.add.forecast)
```




```
##      97.5%        90%        10%       2.5%
##   2.860000  -1.428571  -9.118980 -10.956522
```

```r
boxplot.all.residuals(all.regr.add.forecast)
```



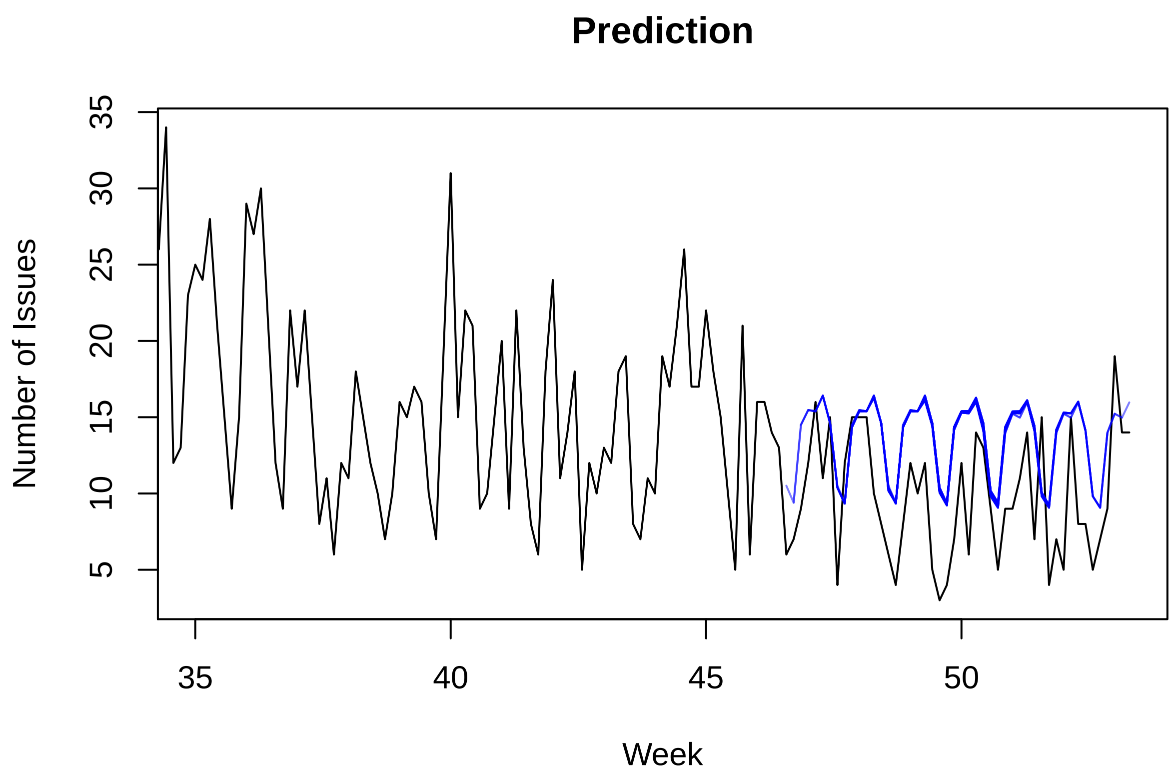

```
##      97.5%        90%        10%       2.5%
##   2.860000  -1.428571  -9.118980 -10.956522
```

linear multiplicative regression

```
regr.mult.forecast <- function(sample.issues) {
  train.ts <- sample.issues$train.ts
  valid.ts <- sample.issues$valid.ts
  train.lm <- tslm(train.ts ~ season, lambda = 0)
  train.lm.pred <- forecast(train.lm, h=n.valid)
  lm.summary <- accuracy(train.lm.pred, valid.ts)

  results <- list()
  results$train <- train.ts
  results$valid <- valid.ts
  results$model <- train.lm
  results$pred <- train.lm.pred
  results$fitted <- train.lm.pred$fitted
  results$residual <- valid.ts - train.lm.pred$mean
  results$summary <- lm.summary

  return(results)
}

all.regr.mult.forecast <- sapply(1:n.sample, function(i) return(regr.mult.forecast(all.issues[,i])))

kable(mean.all.accuracy(all.regr.mult.forecast))
```

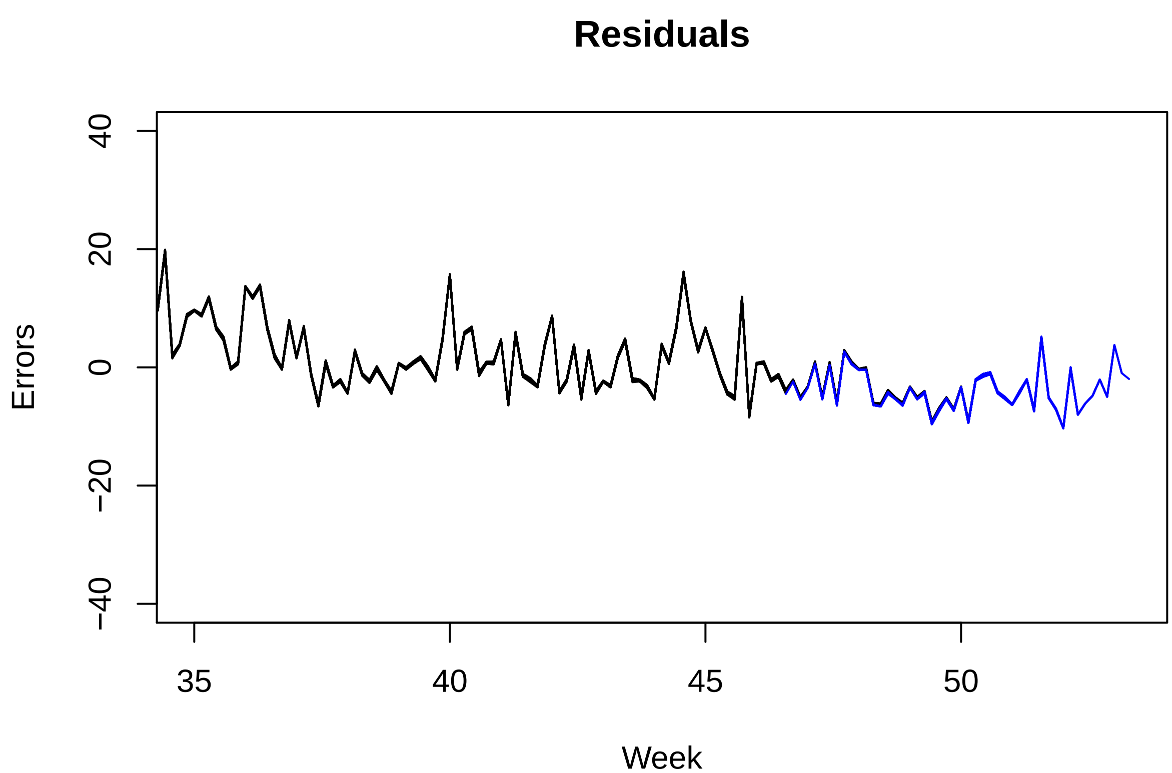
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	1.086185	5.590949	4.411127	-9.914468	36.78695	0.8747236	0.3912911	NA
Test set	-4.384427	5.422056	4.768093	-71.363892	73.99850	0.9456474	0.0262037	1.328968

```
plot.all.pred(all.regr.mult.forecast)
```



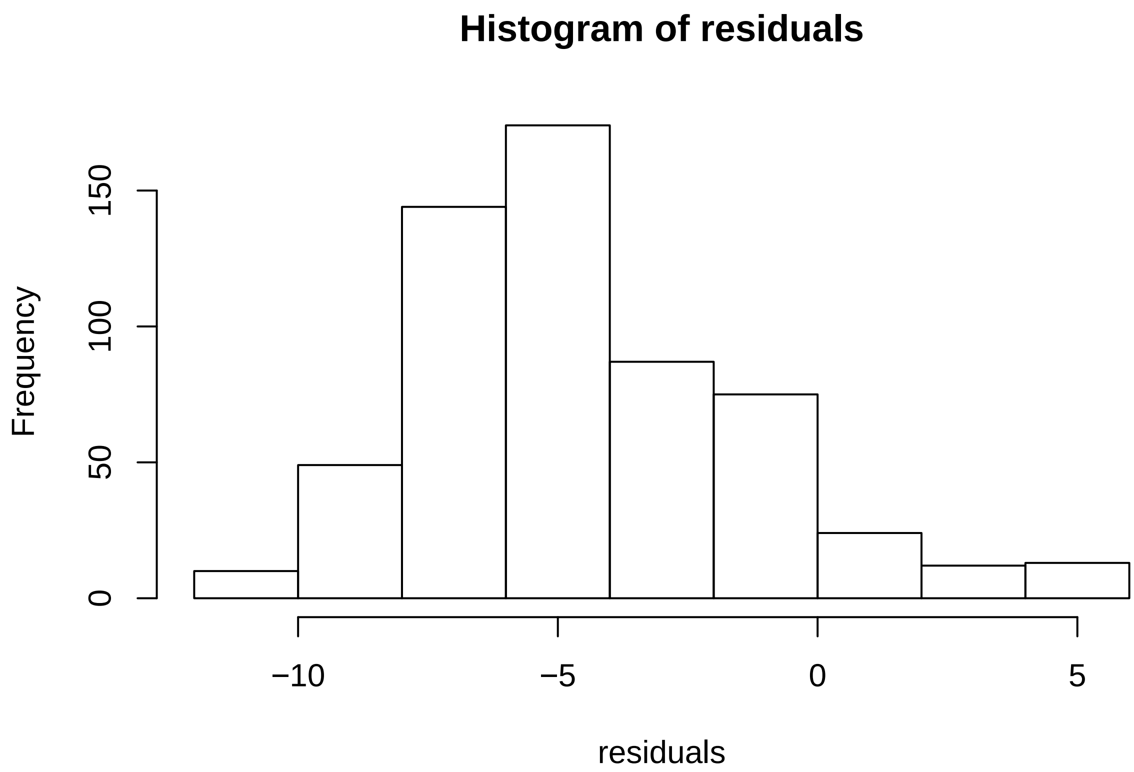
NULL

```
plot.all.residuals(all.regr.mult.forecast)
```



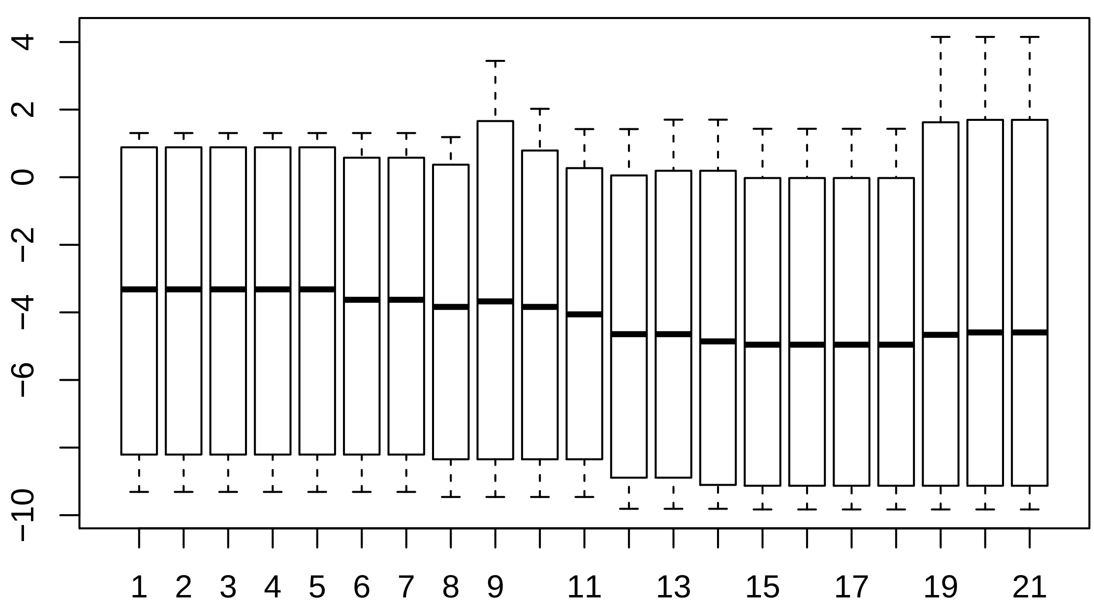
NULL

```r
hist.all.residuals(all.regr.mult.forecast)
```




```
##       97.5%        90%        10%       2.5%
##   3.7726891 -0.3852125 -7.9838244 -9.6120322
```

```r
boxplot.all.residuals(all.regr.mult.forecast)
```



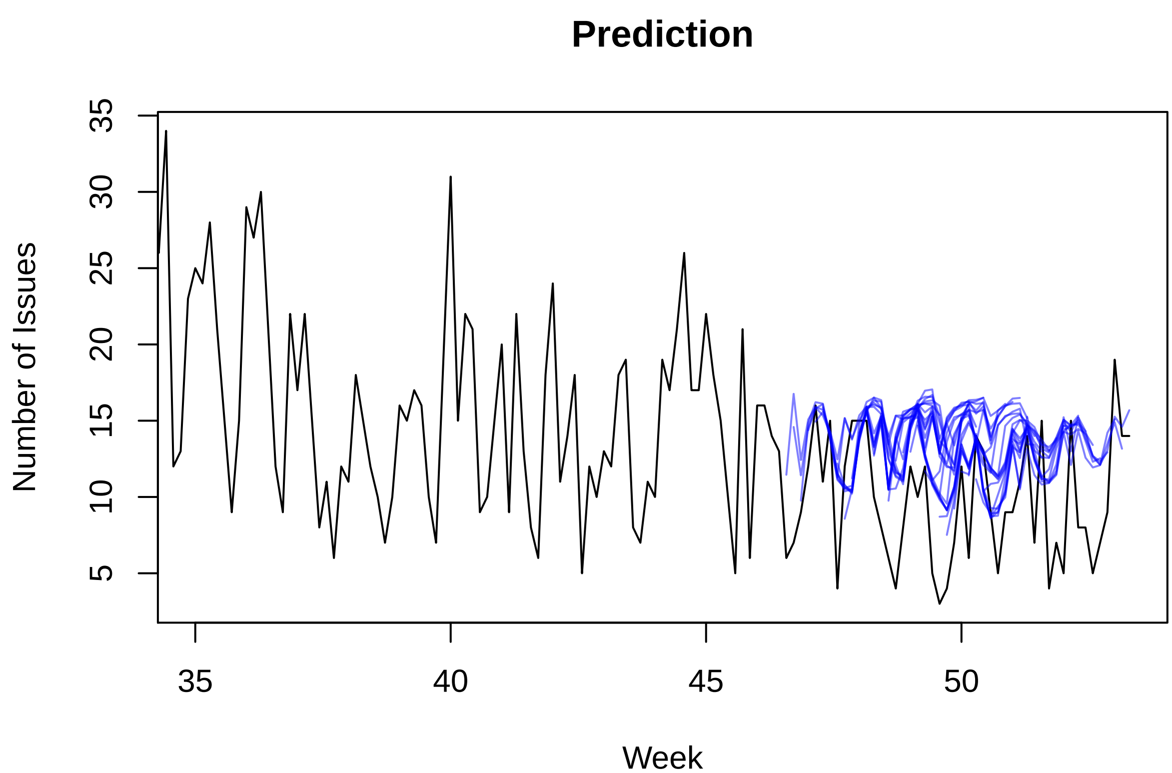

```
##       97.5%        90%        10%       2.5%
##   3.7726891 -0.3852125 -7.9838244 -9.6120322
```

Neural Network (repeats = 20, p=1, P=1, size=7)

```
nnetar.forecast <- function(sample) {
  results <- list()
  results$train <- sample$train.ts
  results$valid <- sample$valid.ts
  results$model <- nnetar(sample$train.ts, repeats = 20, p=1, P=1, size=7 )
  results$pred <- forecast(results$model, h=n.valid)
  results$fitted <- results$pred$fitted
  results$residual <- sample$valid.ts - results$pred$mean
  results$summary <- accuracy(results$pred, sample$valid.ts)

  return(results)
}

all.nnetar.forecast <- sapply(1:n.sample, function(i) return(nnetar.forecast(all.issues[,i])))

kable(mean.all.accuracy(all.nnetar.forecast))
```

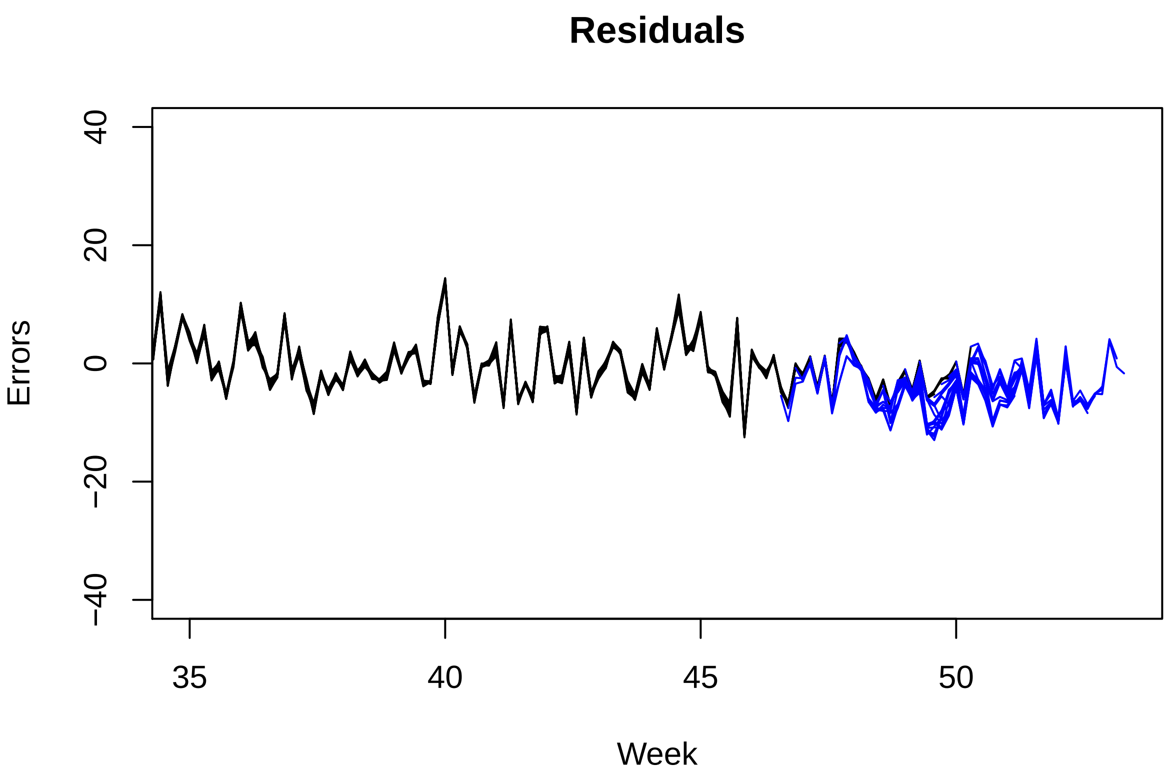
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.0004805	4.814845	3.894361	-15.43442	34.29355	0.7722168	-0.0444574	NA
Test set	-4.3922232	5.782742	4.918525	-78.76061	82.39677	0.9746752	0.1072383	1.516532

```
plot.all.pred(all.nnetar.forecast)
```



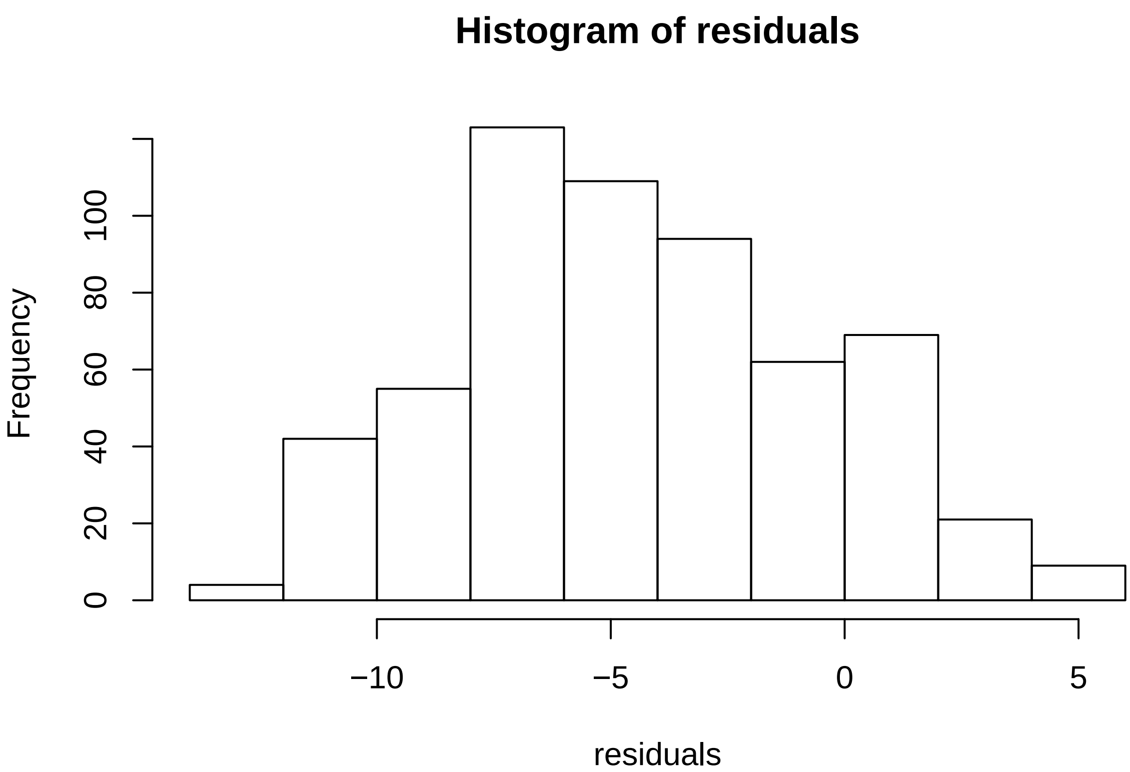

```
## NULL
```

```r
plot.all.residuals(all.nnetar.forecast)
```



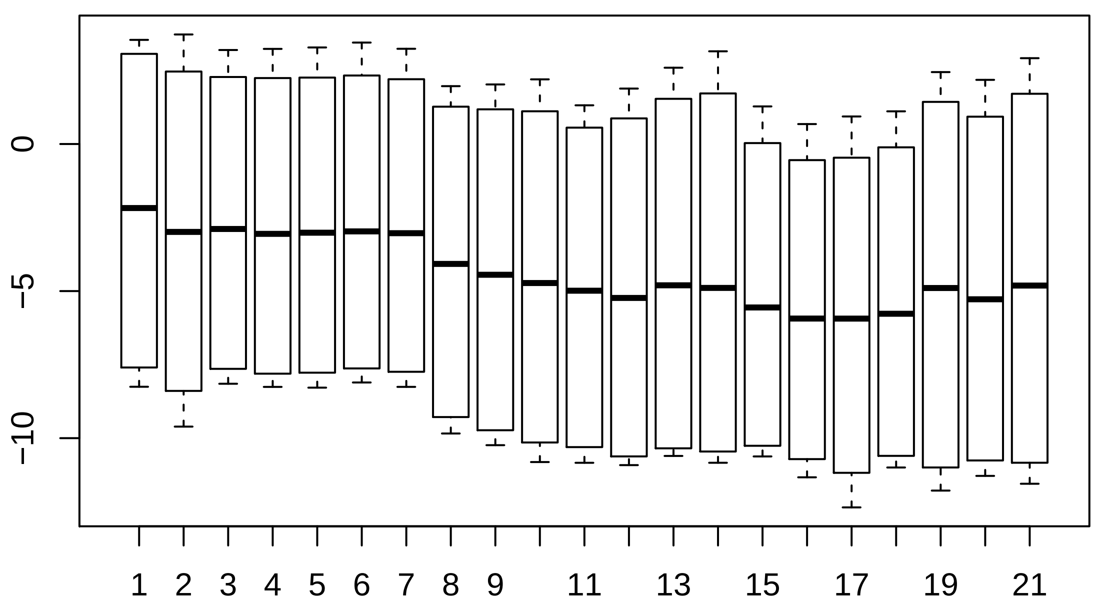

```
## NULL
```

```r
hist.all.residuals(all.nnetar.forecast)
```




```
##        97.5%          90%          10%         2.5%
##    3.6980990    0.8369044   -9.6722141 -10.9315731
```

```
boxplot.all.residuals(all.nnetar.forecast)
```



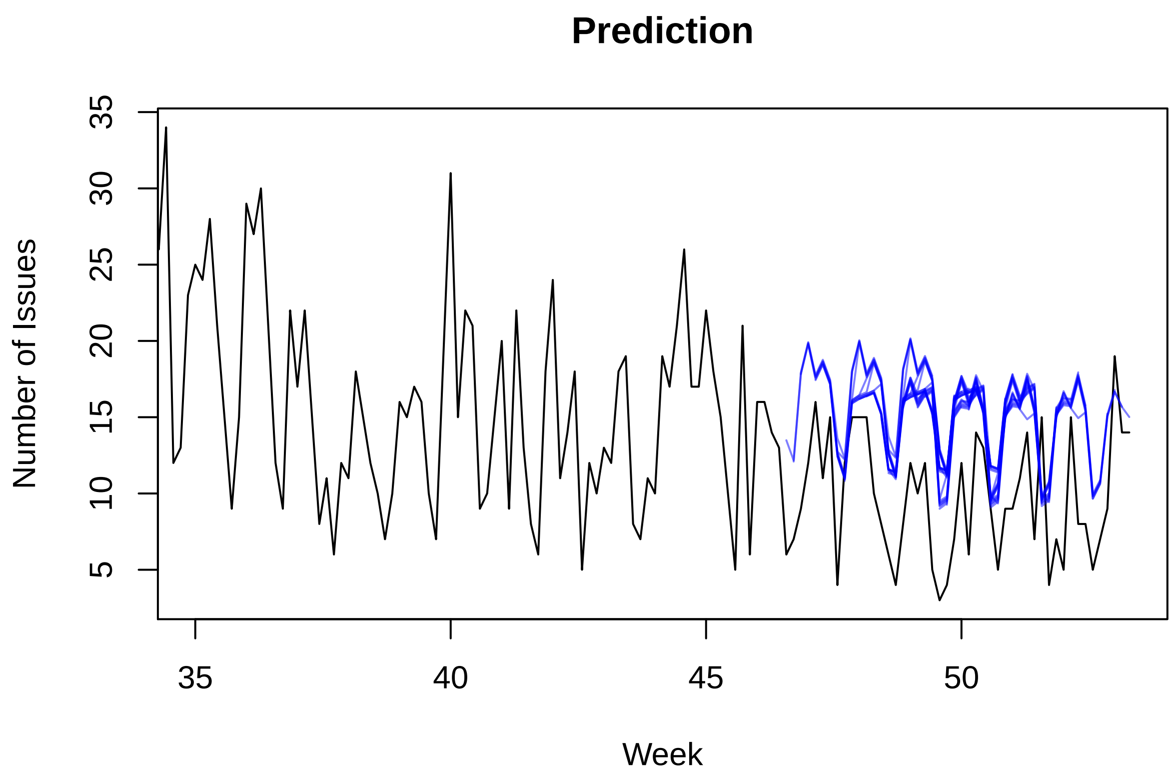

```
##        97.5%          90%          10%         2.5%
##    3.6980990    0.8369044   -9.6722141 -10.9315731
```

External info Numerical using regression model

```
regr.ext.forecast <- function(issues, commits.sample) {
  commits_x <- ts(c(commits.sample$train.ts[1:(length(commits.sample$train.ts) - 1)]), frequency = 7, st
  issues$train.ts <- window(issues$train.ts, start=c(1,2))

  newdata <- data.frame(as.numeric(snaive(commits_x, h=n.valid)$mean))
  colnames(newdata) <- c('commits_x')

  results <- list()
  results$train <- issues$train.ts
  results$valid <- issues$valid.ts
  results$model <- tslm(issues$train.ts ~ season + trend + commits_x)
  results$pred <- forecast(results$model, h=n.valid, newdata=newdata)
  results$fitted <- results$pred$fitted
  results$residual <- issues$valid.ts - results$pred$mean
  results$summary <- accuracy(results$pred, issues$valid.ts)

  return(results)
}

all.regr.ext.forecast <- sapply(1:n.sample, function(i) return(regr.ext.forecast(all.issues[,i], all.co

kable(mean.all.accuracy(all.regr.ext.forecast))
```

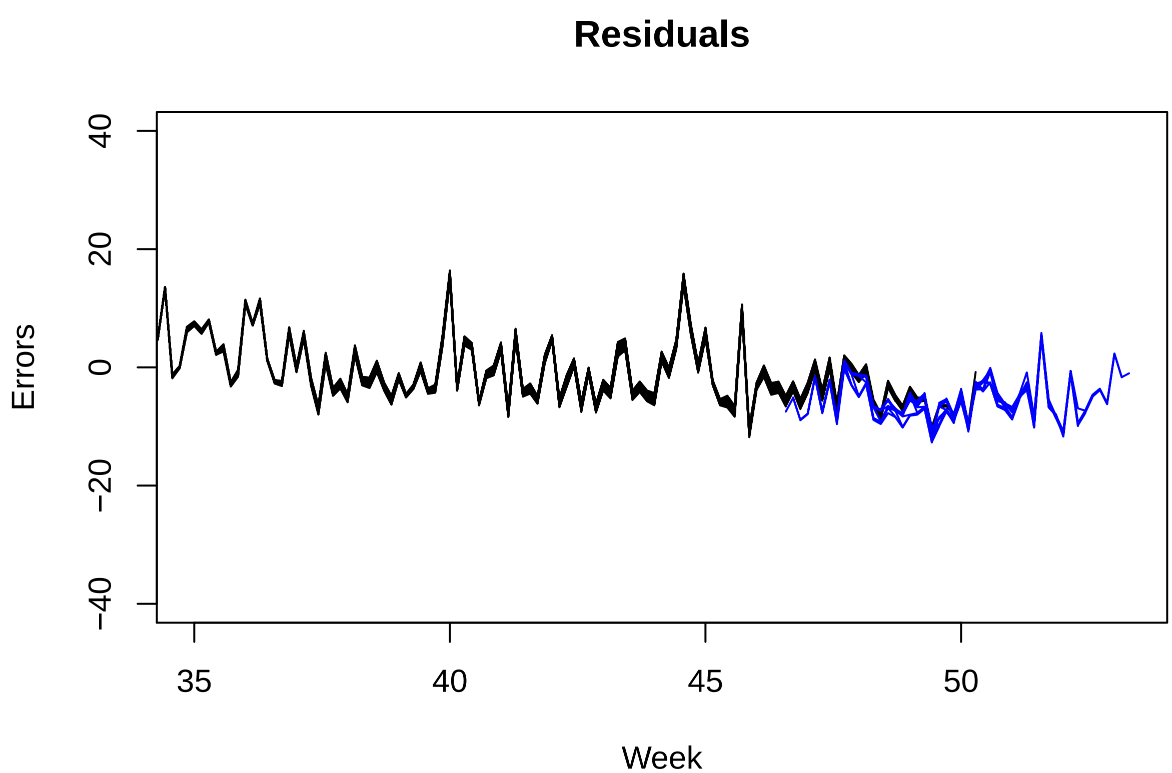
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.000000	5.000597	4.025116	-14.95993	34.52879	0.7966817	0.0980625	NA
Test set	-5.784193	6.718202	6.069042	-90.24174	92.14728	1.2006837	0.0036496	1.657235

```
plot.all.pred(all.regr.ext.forecast)
```



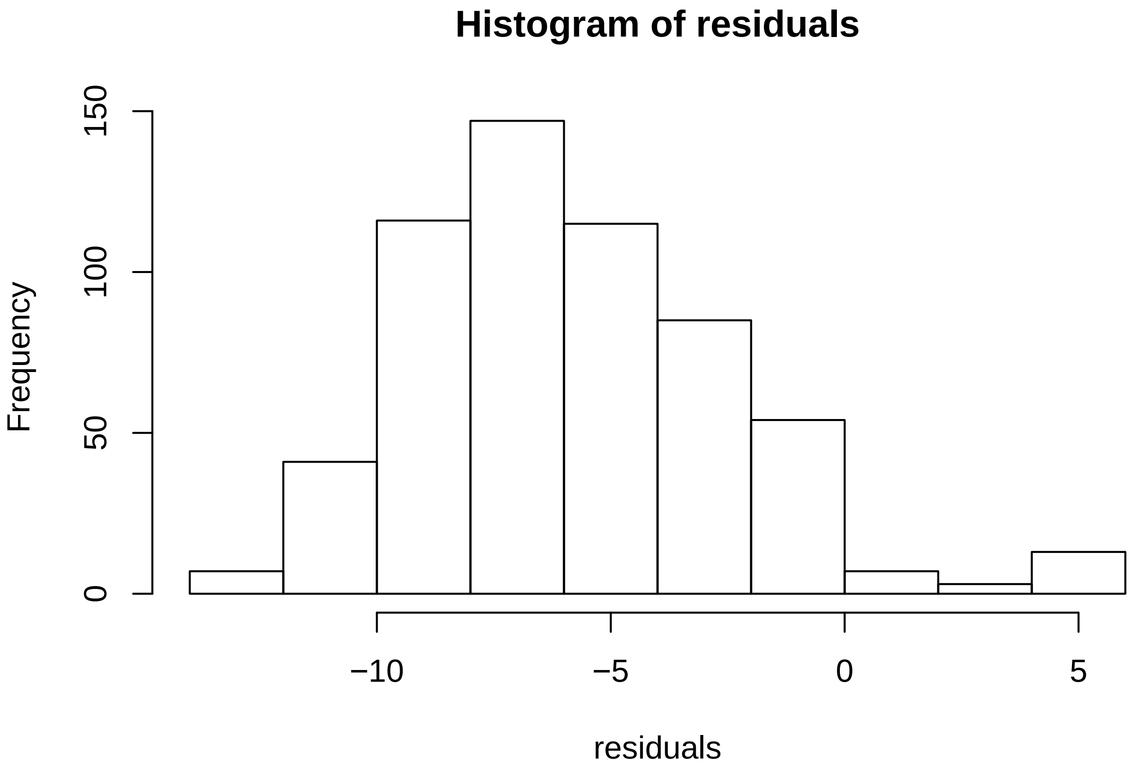

```
## NULL
```

```
plot.all.residuals(all.regr.ext.forecast)
```



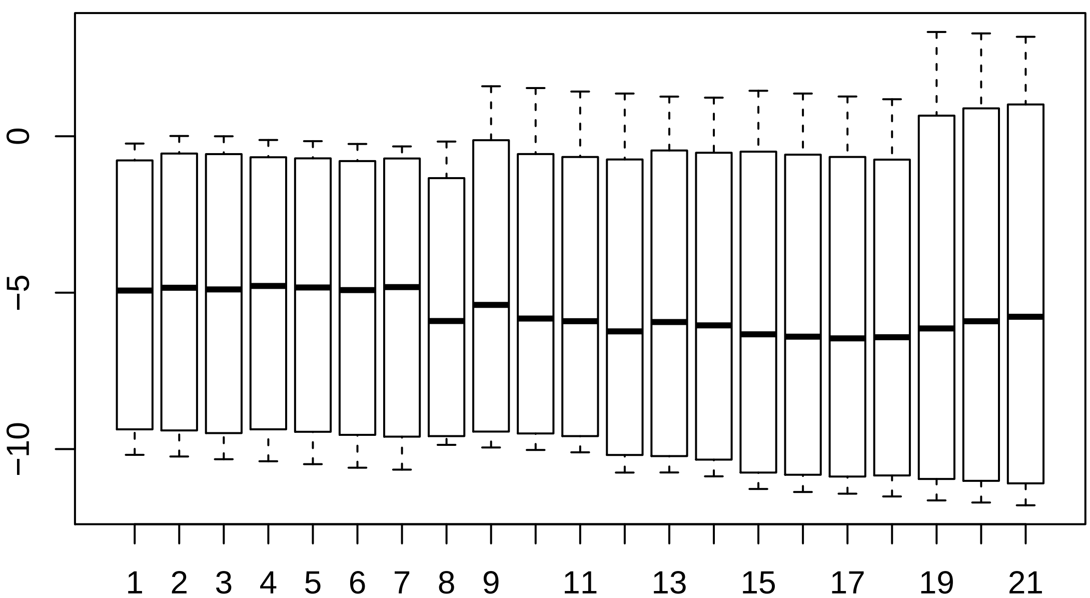

```
## NULL
```

```r
hist.all.residuals(all.regr.ext.forecast)
```




```
##      97.5%        90%        10%       2.5%
##   2.248575  -1.281229  -9.813937 -11.605701
```

```r
boxplot.all.residuals(all.regr.ext.forecast)
```



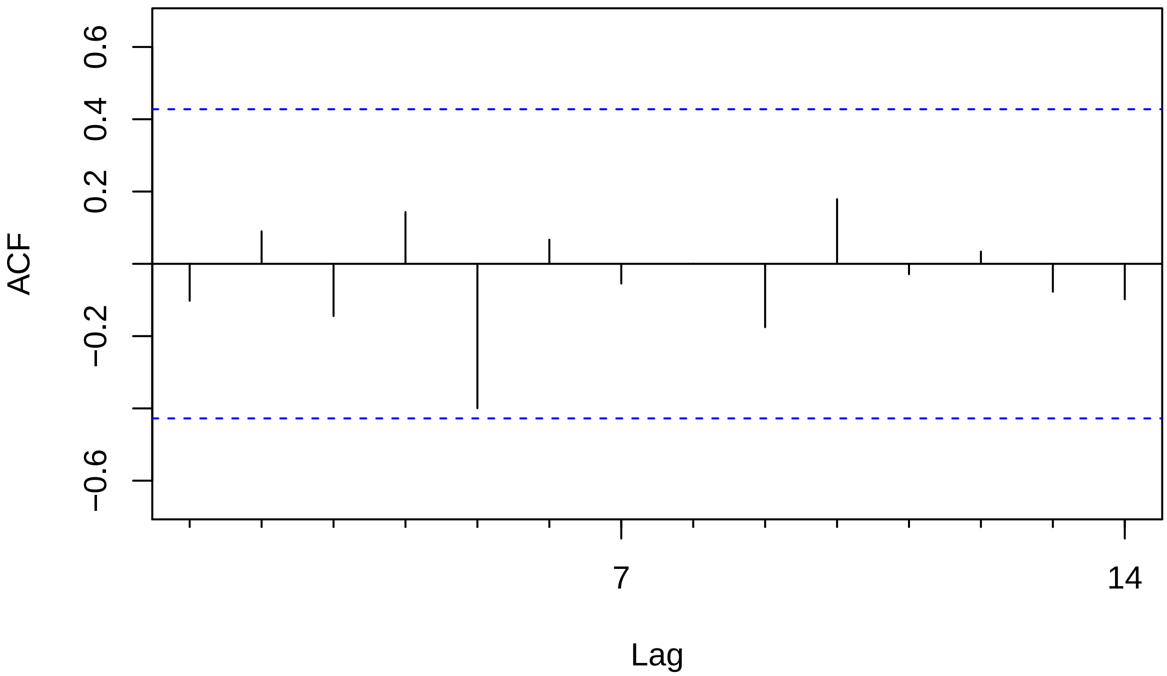

```
##      97.5%        90%        10%       2.5%
##   2.248575  -1.281229  -9.813937 -11.605701
```

Ensemble (all.regr.mult.forecast[,i], all.hw.forecast[,i])

```
ensemble.forecast <- function(list.of.forecast) {
  results <- list()
  results$train <- list.of.forecast[[1]]$train
  results$valid <- list.of.forecast[[1]]$valid

  valid.time <- time(results$valid)
  train.time <- time(results$train)

  mean.pred <- ts(
    rowMeans(sapply(list.of.forecast, function(forecast) forecast$pred$mean)),
    start=start(valid.time),
    end=end(valid.time),
    frequency=frequency(valid.time))

  mean.fitted <- ts(
    rowMeans(sapply(list.of.forecast, function(forecast) window(forecast$fitted, start=c(5,3)))),
    start=start(train.time),
    end=end(train.time),
    frequency=frequency(train.time))

  results$pred <- forecast.manual(window(results$train, start=c(5,3)), mean.fitted, mean.pred)

  results$fitted <- results$pred$fitted

  results$residual <- results$valid - results$pred$mean
  results$summary <- accuracy(results$pred, results$valid)

  return(results)
}

all.ensemble.forecast <- sapply(
  1:n.sample,
  function(i) return(ensemble.forecast(list(all.regr.mult.forecast[,i], all.hw.forecast[,i])))
)

kable(mean.all.accuracy(all.ensemble.forecast))
```

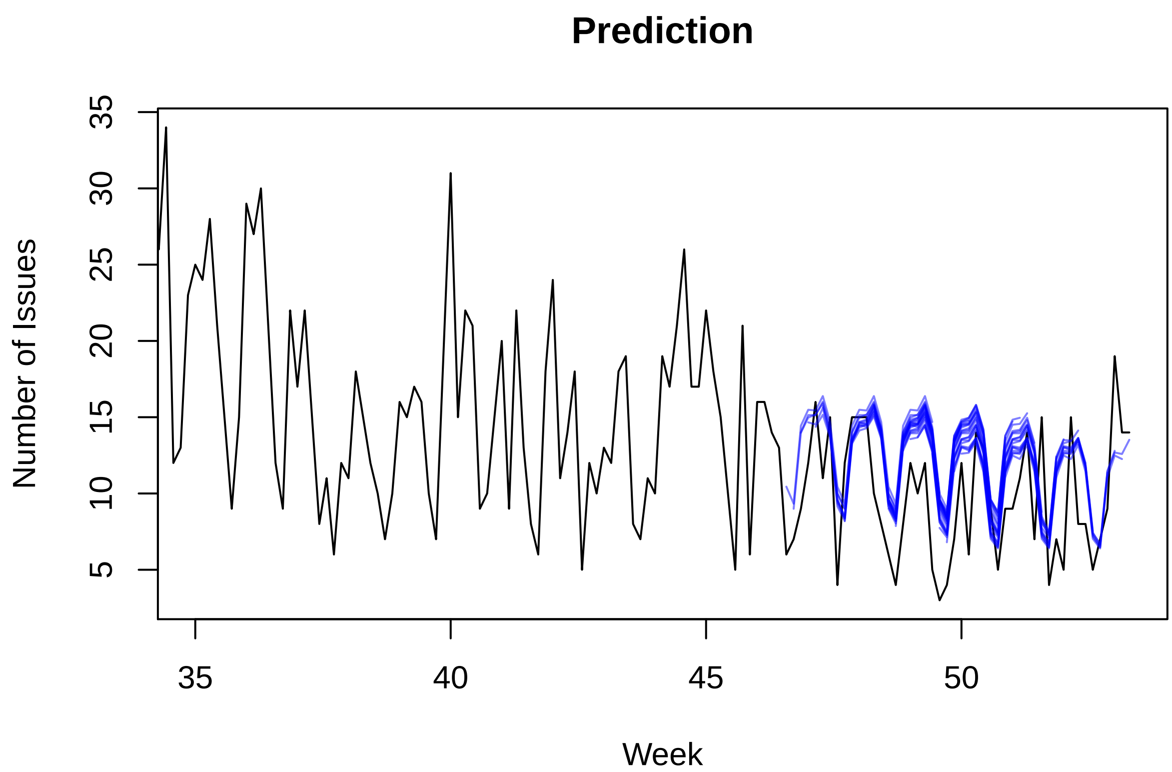
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.8490921	6.790917	5.410272	-13.91752	43.72367	1.0612569	0.3342306	NA
Test set	-3.0156594	4.467467	3.807256	-52.99306	58.68572	0.7462721	0.0206131	1.091487

```
Acf(all.ensemble.forecast[,1]$residual, lag.max = 14, main = "")
```



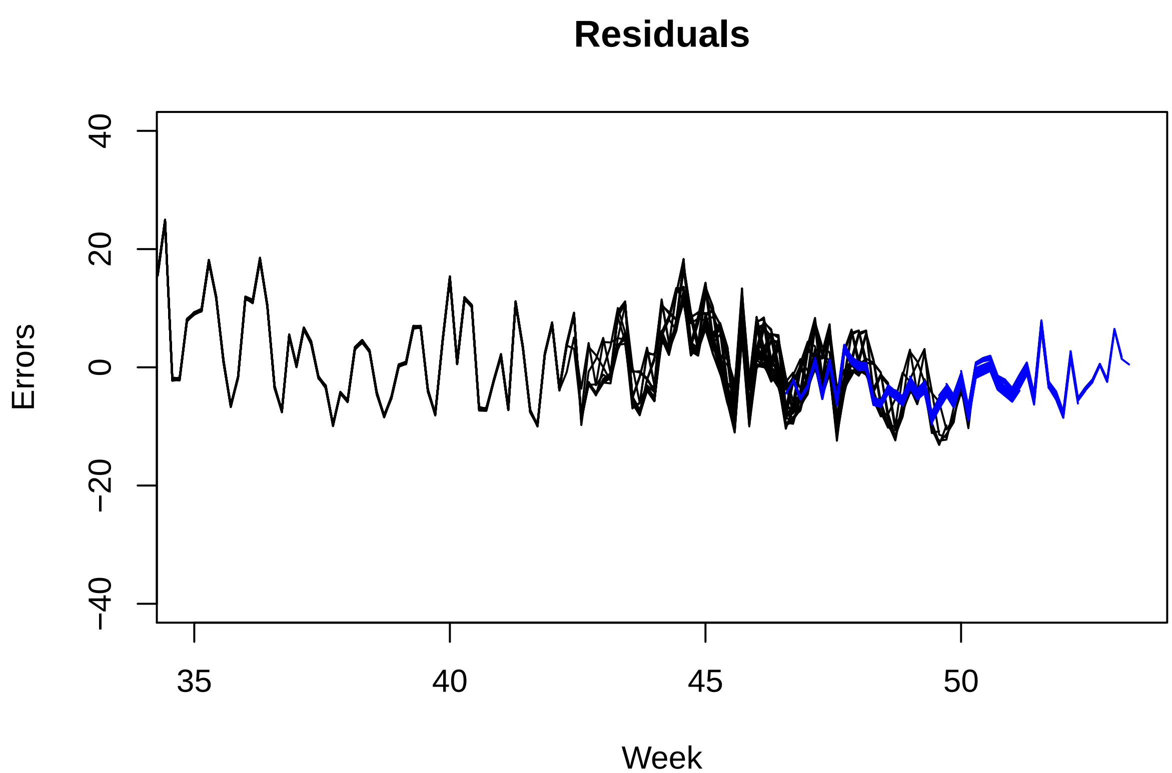

```
test <- list()
test$train.ts <- issues.ts
test$valid.ts <- naive(issues.ts, h=n.valid)$mean

test.forecast <- ensemble.forecast(list(regr.mult.forecast(test), hw.forecast(test)))
quantile.of.residuals <- get.quantile.of.residuals(all.ensemble.forecast)
# the prediction intrerval of each time stamp
test.forecast.confidence.interval <- forecast.confidence(test.forecast$pred$mean, quantile.of.residuals)
# convert the prediction interval into time series object
test.forecast.confidence.interval.ts <- ts(test.forecast.confidence.interval, start = c(53, 4), end = c

forecast.object <- forecast.manual.interval(
  x.train=issues.ts,
  f.train=test.forecast$fitted,
  f.pred=test.forecast$pred$mean,
  f.lower=test.forecast.confidence.interval.ts[,1:2],
  f.upper= test.forecast.confidence.interval.ts[,3:4])
```

```
plot.all.pred(all.ensemble.forecast)
```



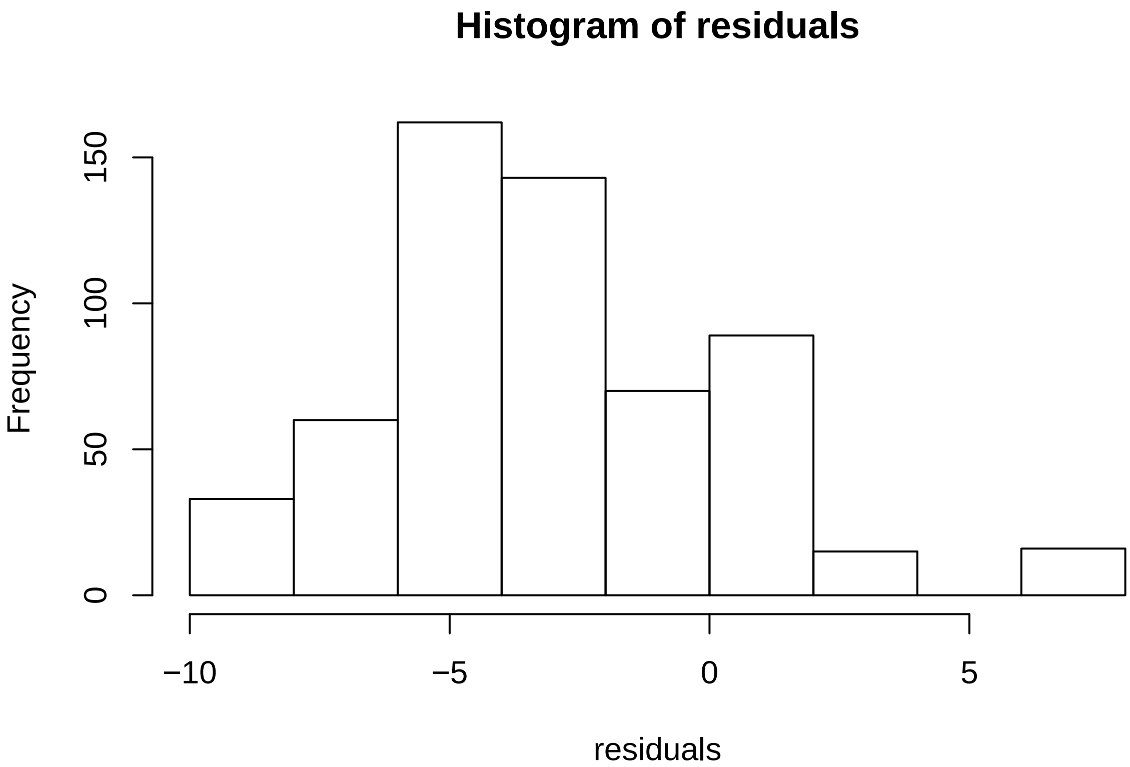

```
## NULL
```

```r
plot.all.residuals(all.ensemble.forecast)
```



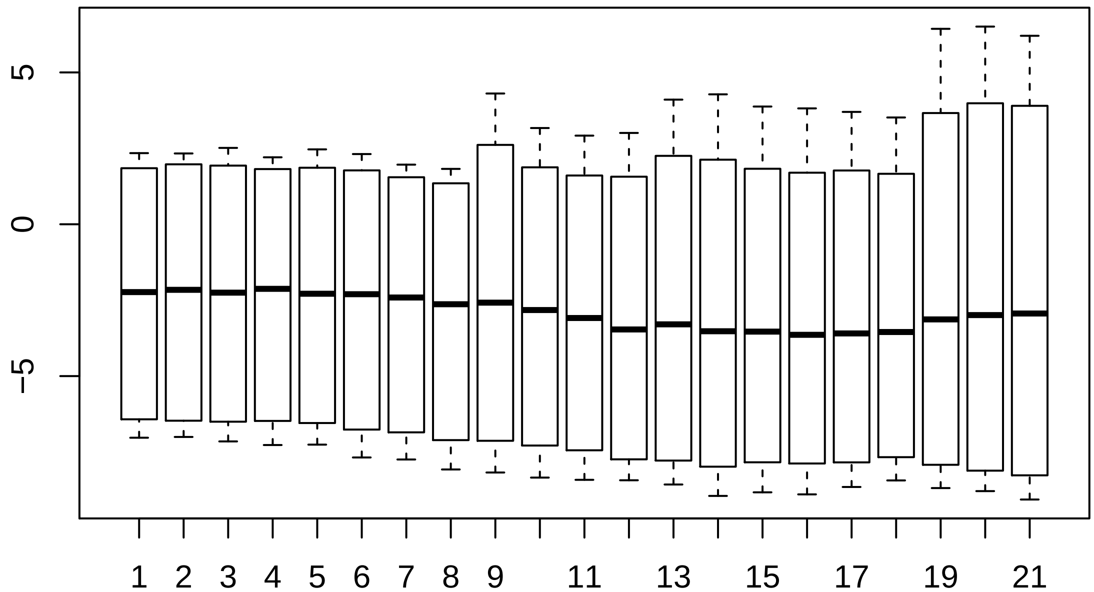

```
## NULL
```

```r
hist.all.residuals(all.ensemble.forecast)
```




```
##      97.5%       90%       10%      2.5%
##   6.205877  1.155084 -6.634113 -8.760837
```

```r
boxplot.all.residuals(all.ensemble.forecast)
```




```
##      97.5%       90%       10%      2.5%
##   6.205877  1.155084 -6.634113 -8.760837
```

```r
# plot the prediction on test period with the prediction interval
plot(forecast.object, xlim=c(35, 56), main = 'Julia Forecasted No. of issues')
```