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.

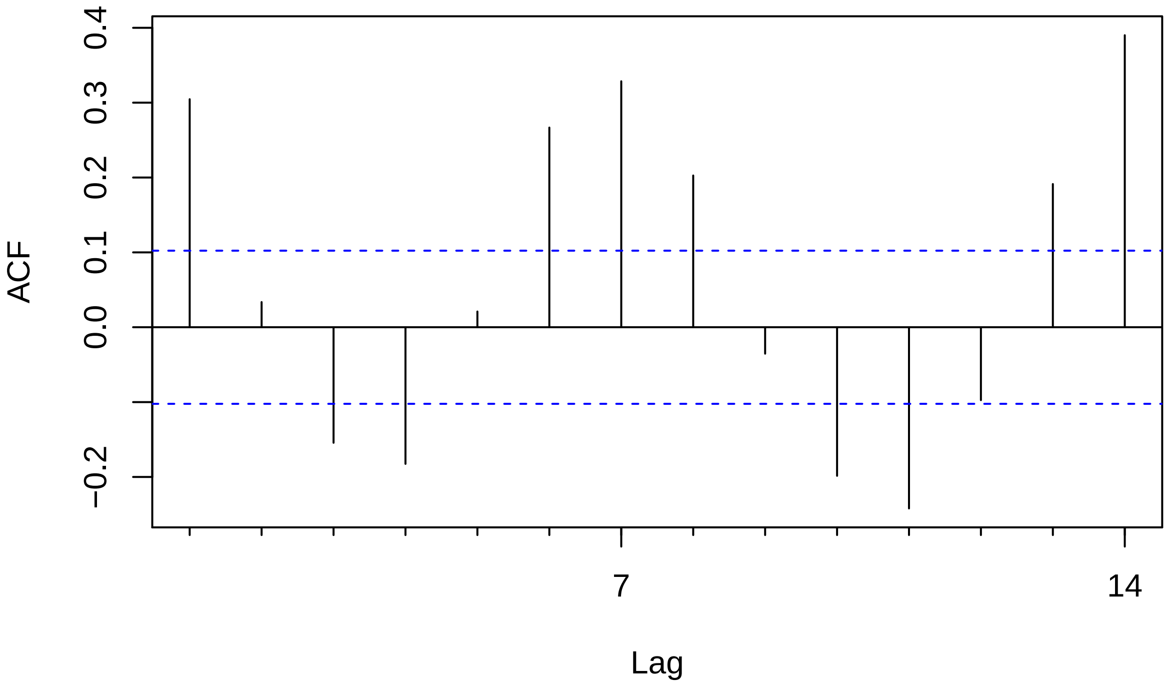
Load the data

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

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

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

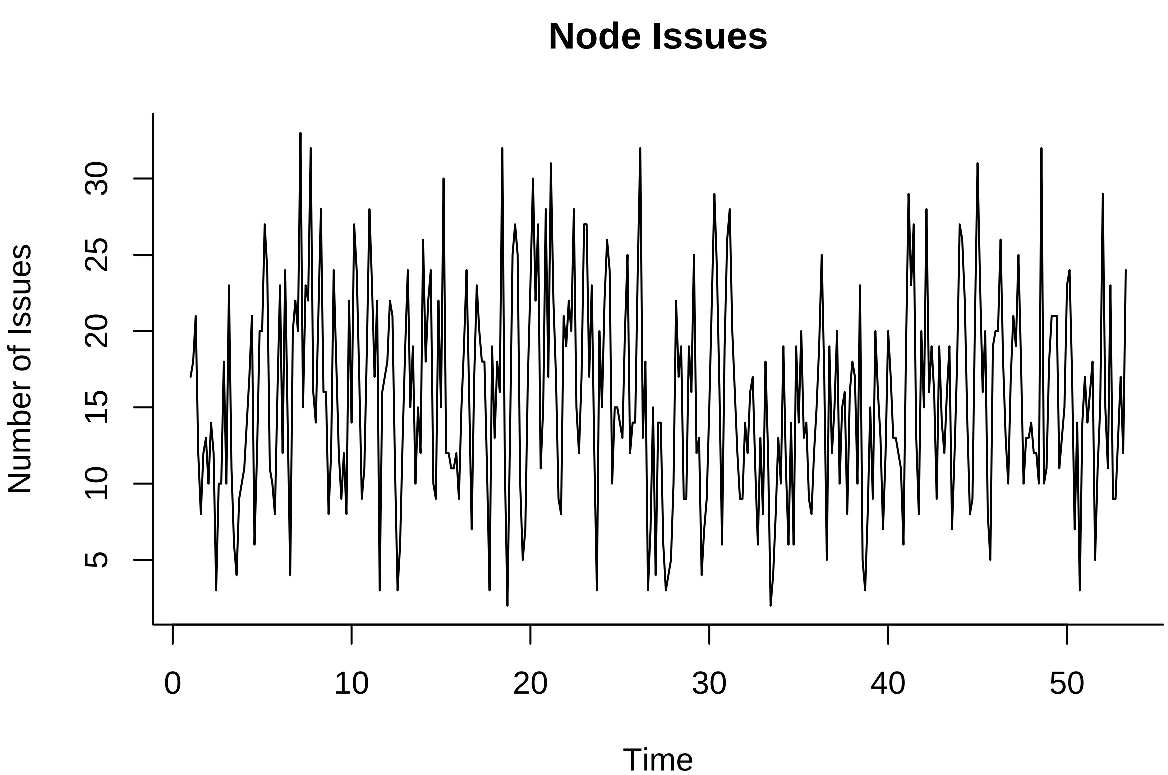
keep the last 12 months

```
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)
```

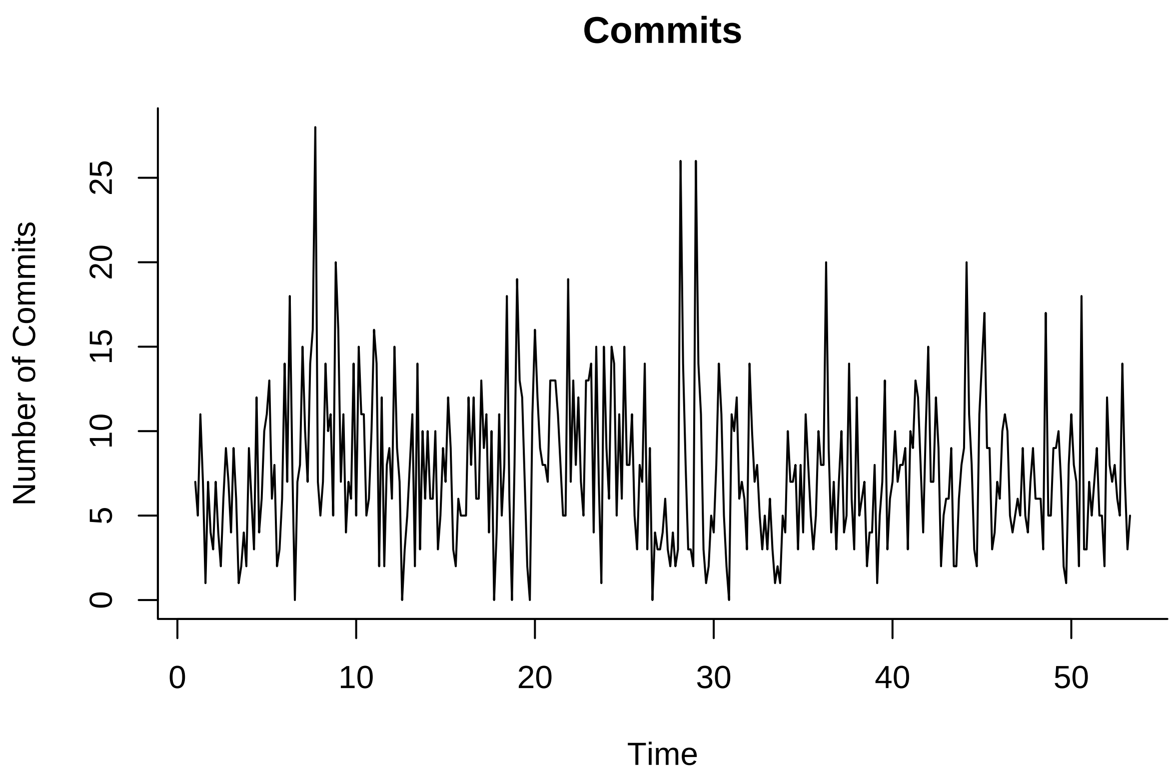
```
#loading issues and commits into a ts object
issues.ts <- ts(repository.csv$number_of_issues, frequency = 7)


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




```
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 = 'Node Issues', bty = 'l', ylab = 'Number of Issues')
```



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




```r
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)
```

```r
# 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',])
}
```

```
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]$train - all.forecast['fitted', i]$fitte
  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.quantil
  return(forecast.confidence.interval)
}
```

```
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'))
}
```

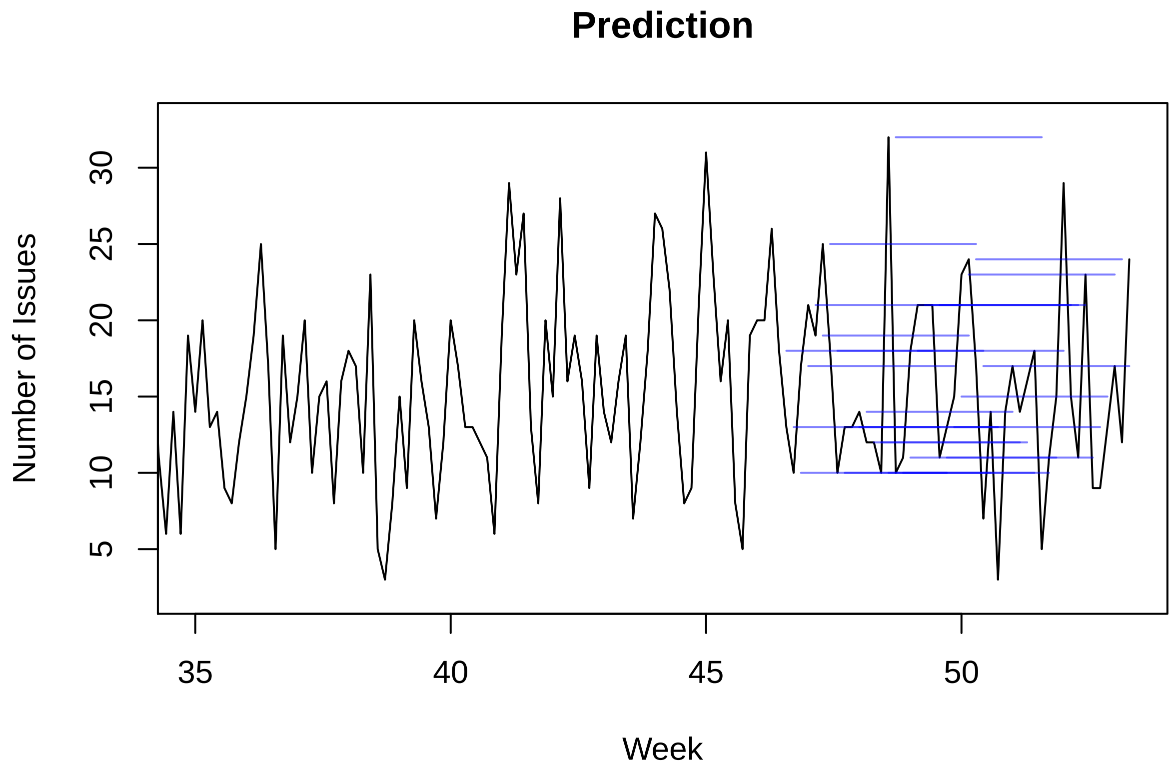
Naive Forecast

Naive

```
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))
```

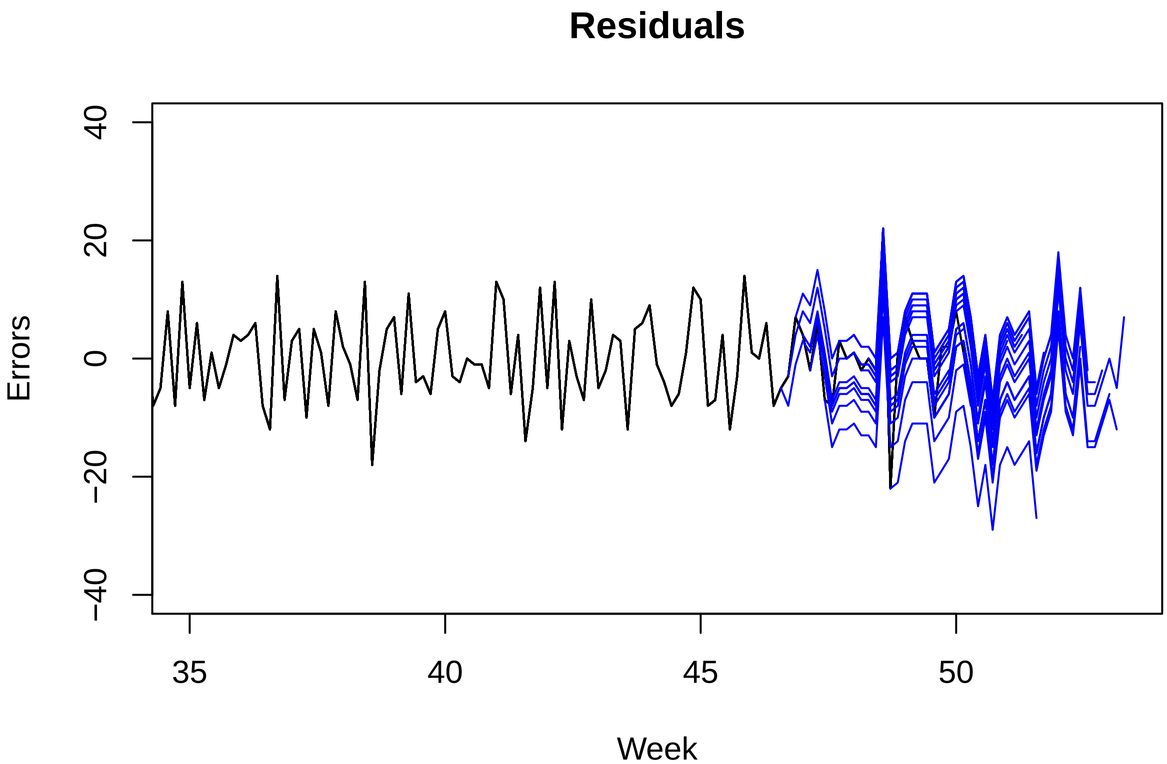
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.0015705	7.853183	6.443738	-21.90097	56.17743	1.041970	-0.2923773	NA
Test set	-1.0901361	7.969542	6.617347	-34.34137	60.43484	1.069698	0.0669863	0.8256278

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



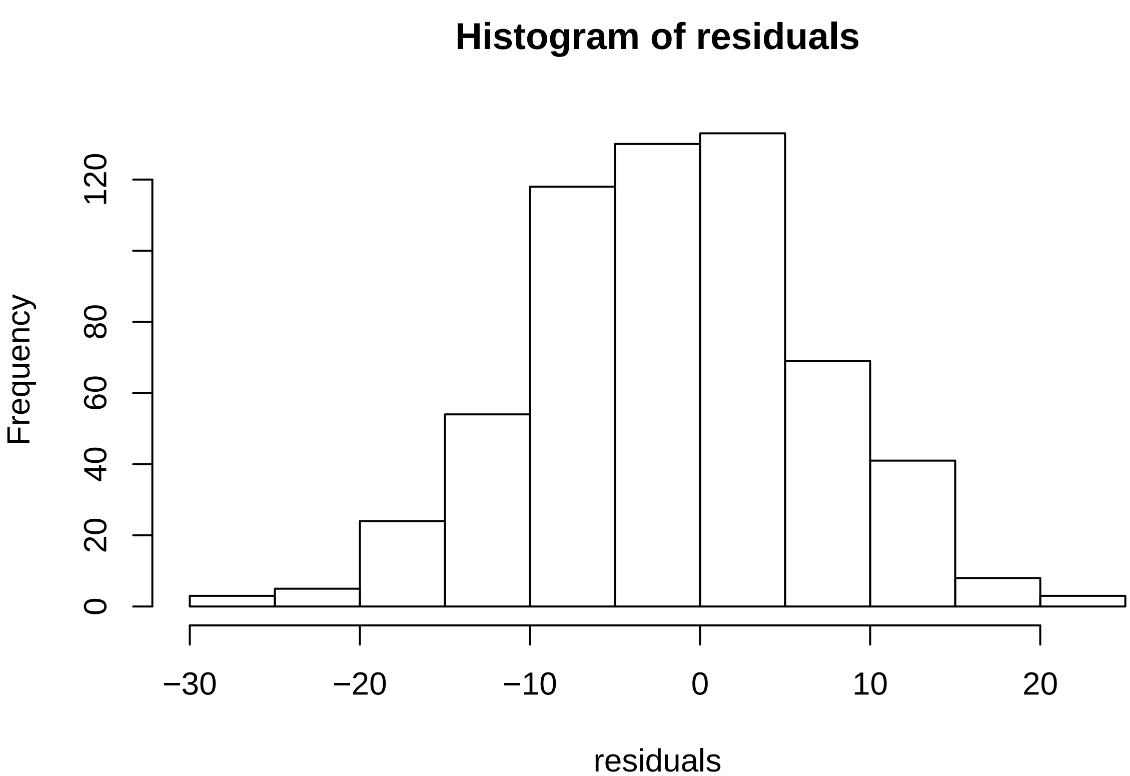

```
## NULL
```

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



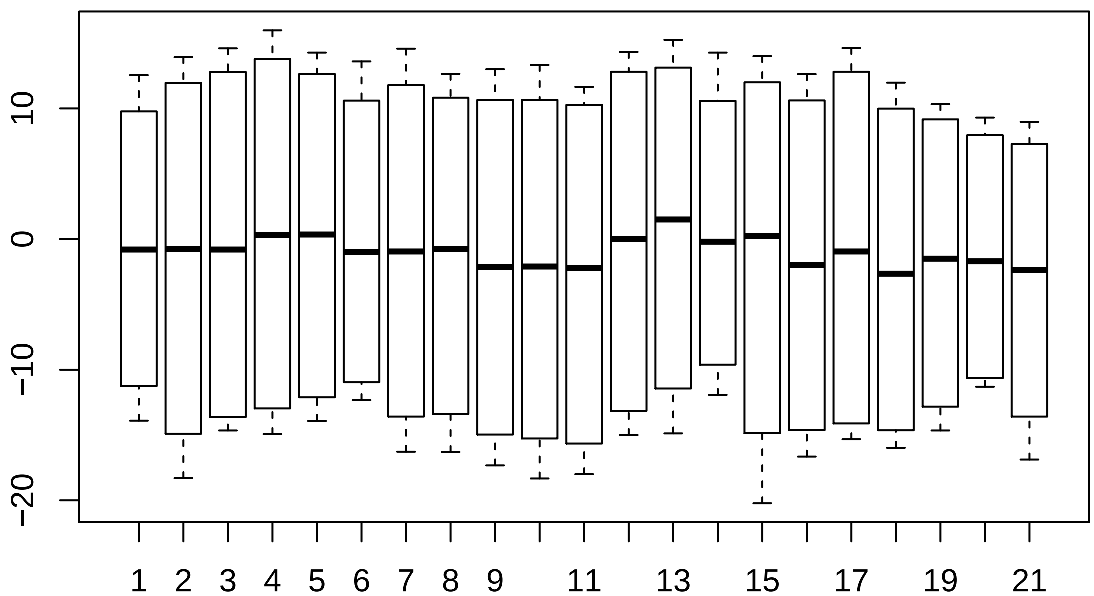

```
## NULL
```

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



```
## 97.5%   90%   10%  2.5%
##    14    10   -11   -18
```

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



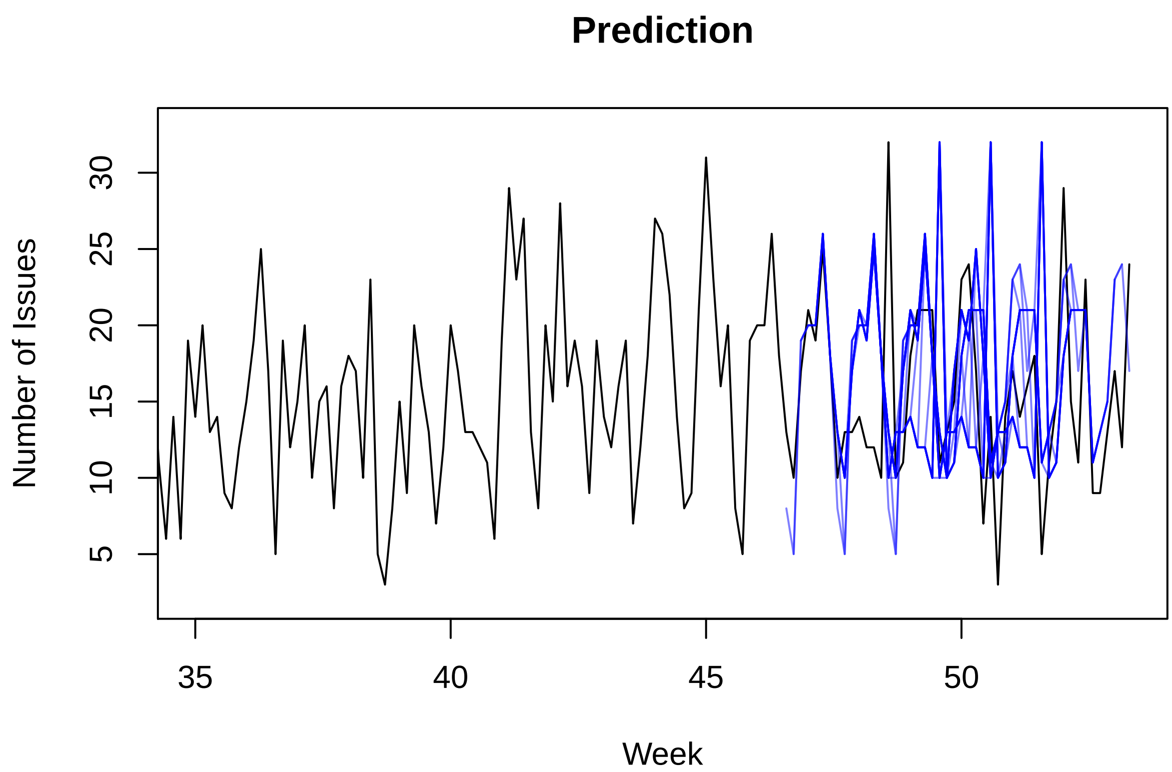

```
## 97.5%   90%   10%  2.5%
##    14    10   -11   -18
```

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))
```

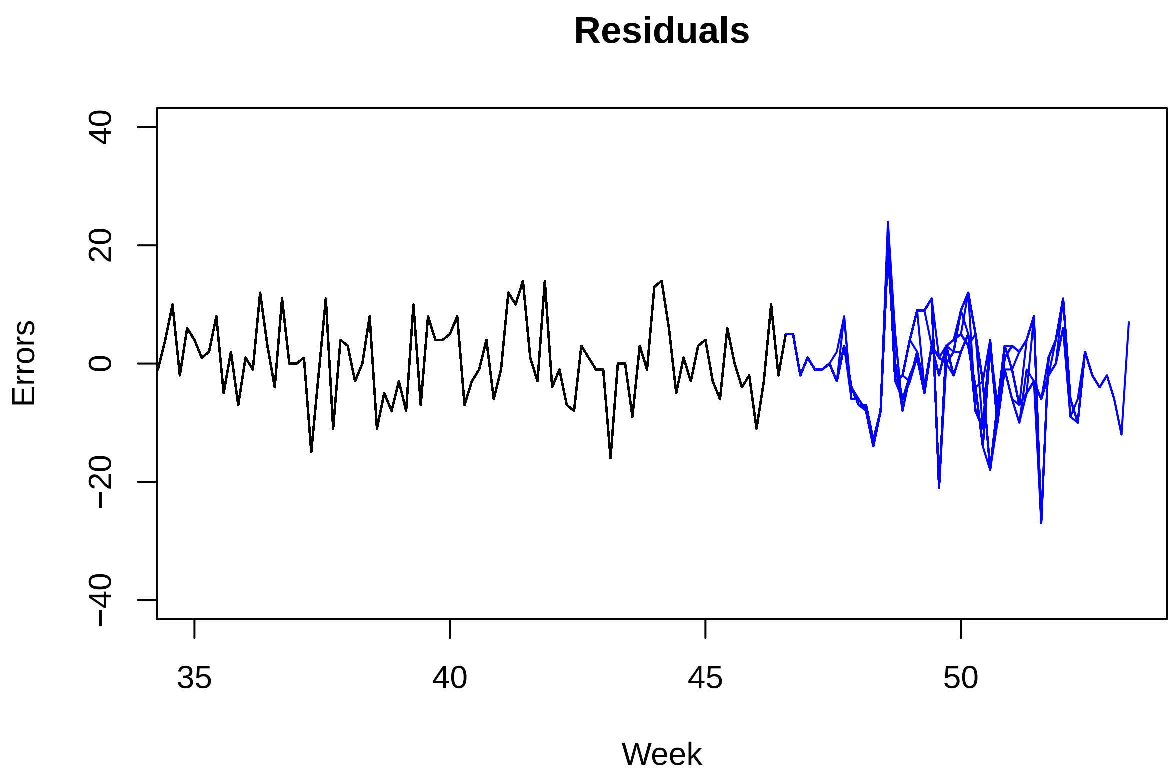
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.046285	7.731857	6.184432	-18.41707	50.93641	1.0000000	0.1275944	NA
Test set	-1.166667	7.760235	5.904762	-30.38964	53.50434	0.9551085	-0.0261017	0.772494

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



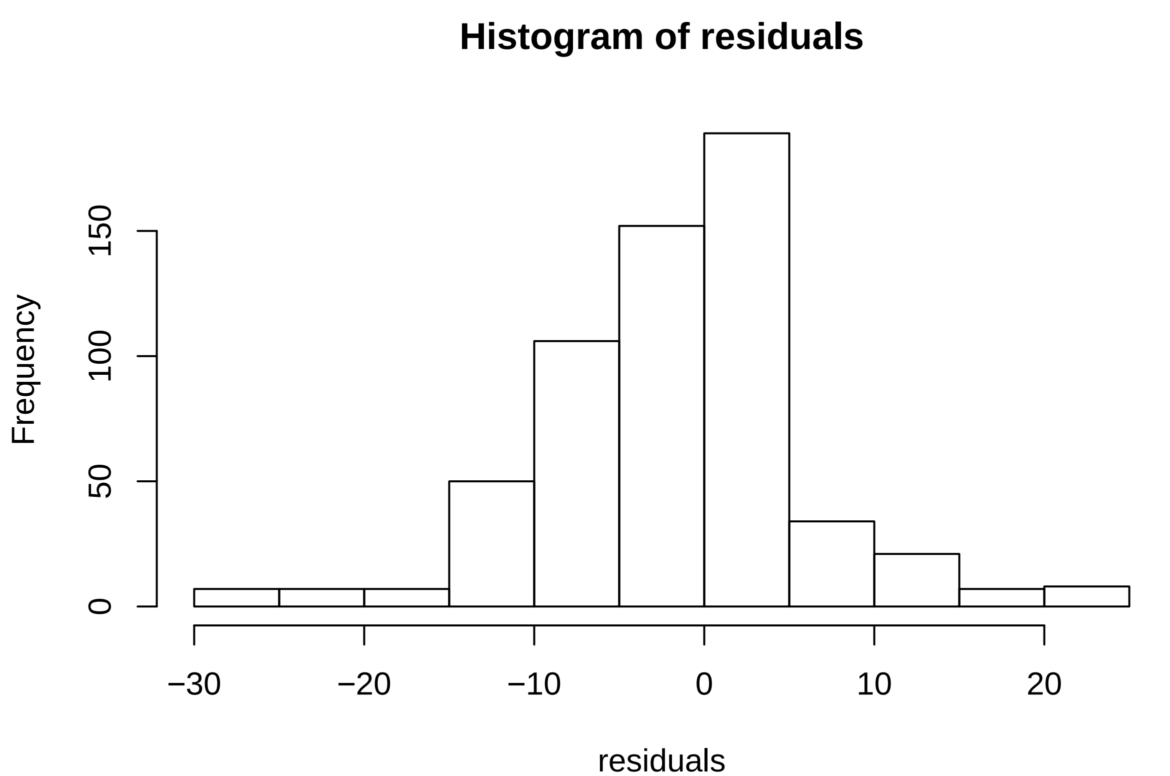

```
## NULL
```

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



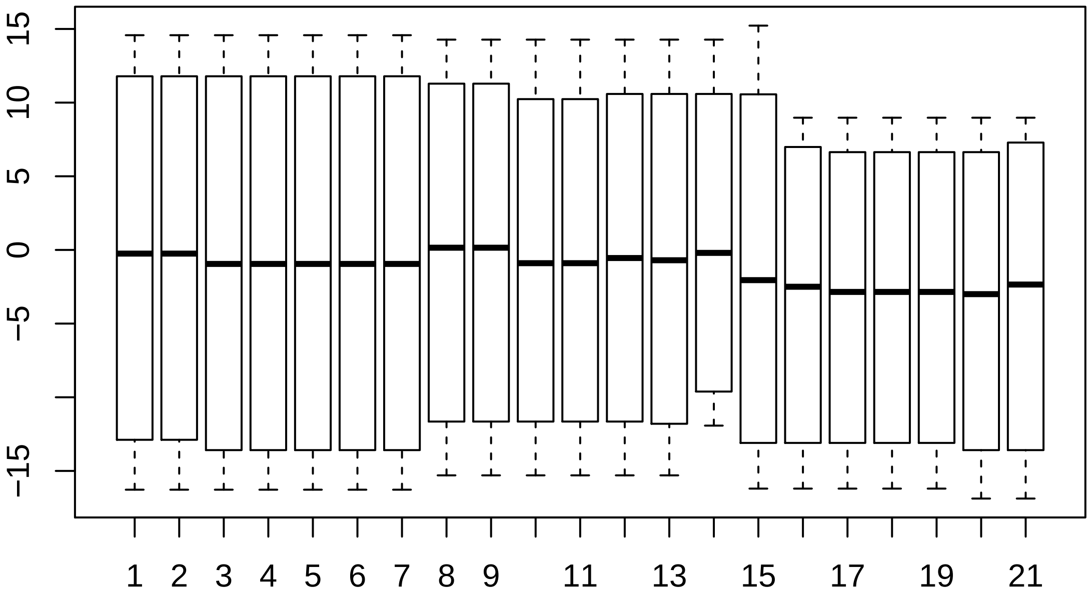

```
## NULL
```

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



```
##   97.5%     90%     10%    2.5%
##  14.275   8.000 -10.000 -18.000
```

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



```
##   97.5%     90%     10%    2.5%
##  14.275   8.000 -10.000 -18.000
```

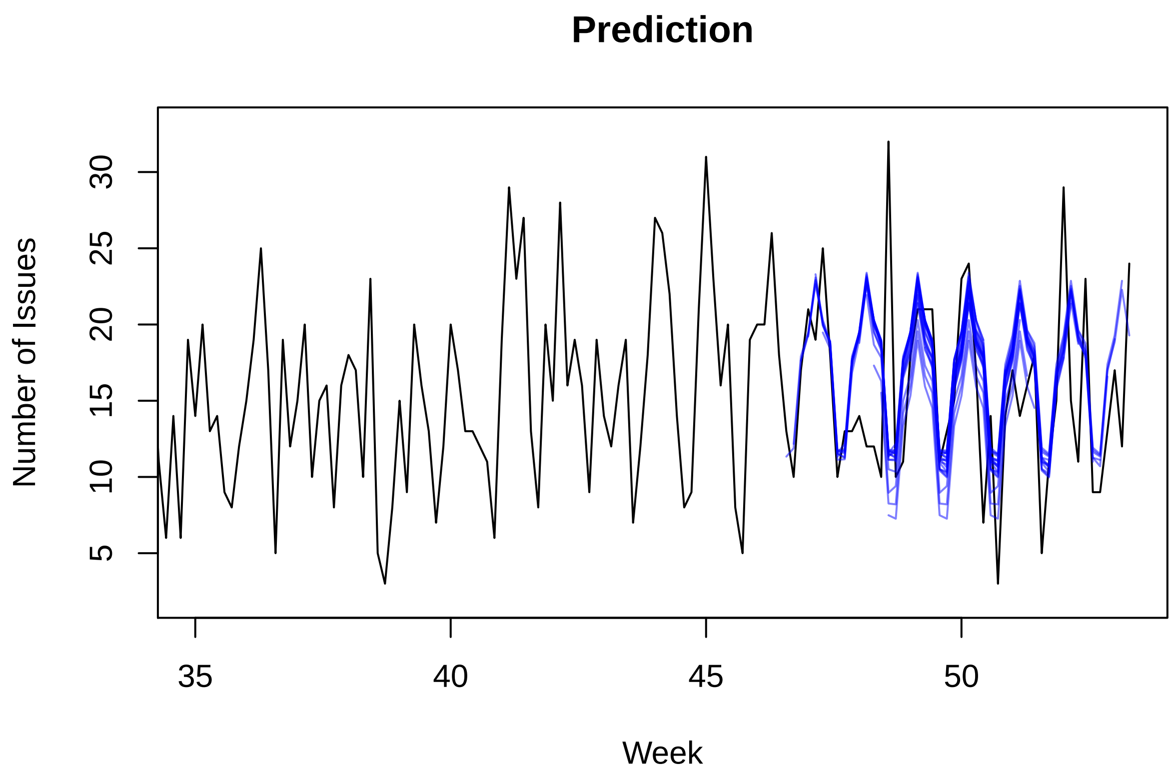
Smoothing

Exponential smoothing ZNA

```r
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))
```

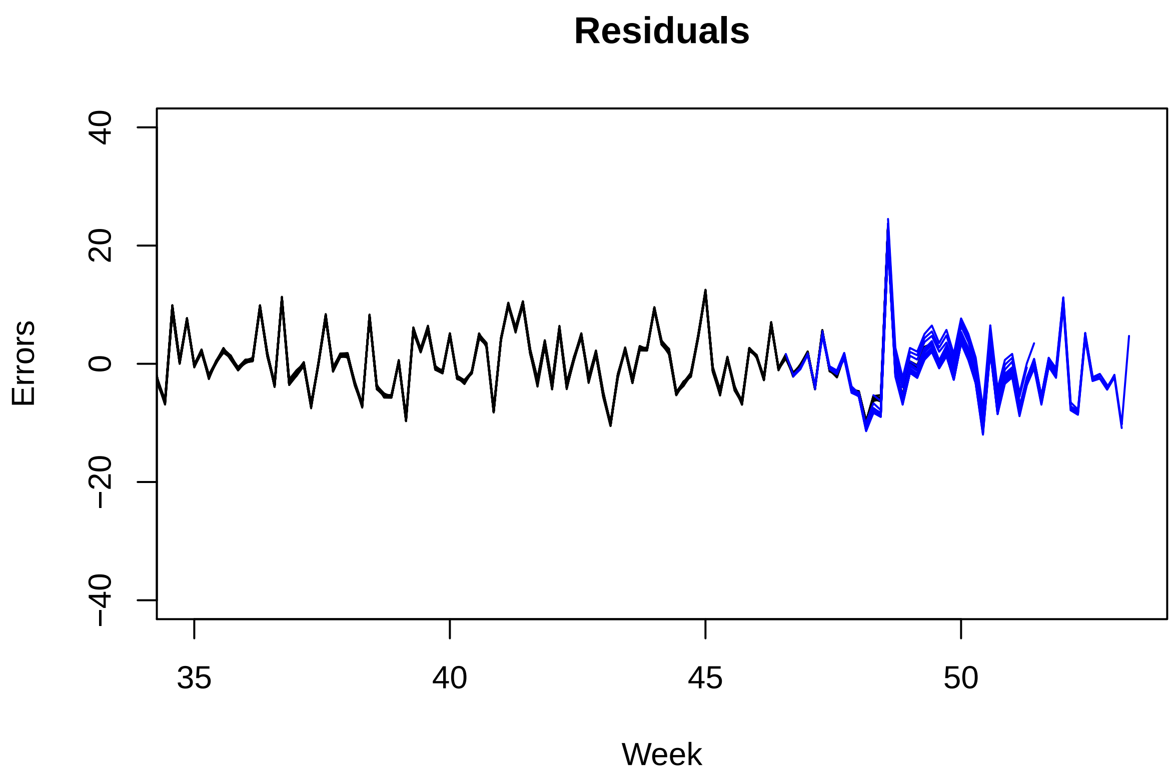
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0932442	5.322839	4.292383	-18.00811	39.20567	0.6940845	0.0895703	NA
Test set	-1.1748063	5.905106	4.315817	-24.89249	38.81290	0.6979953	-0.1096244	0.6326221

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



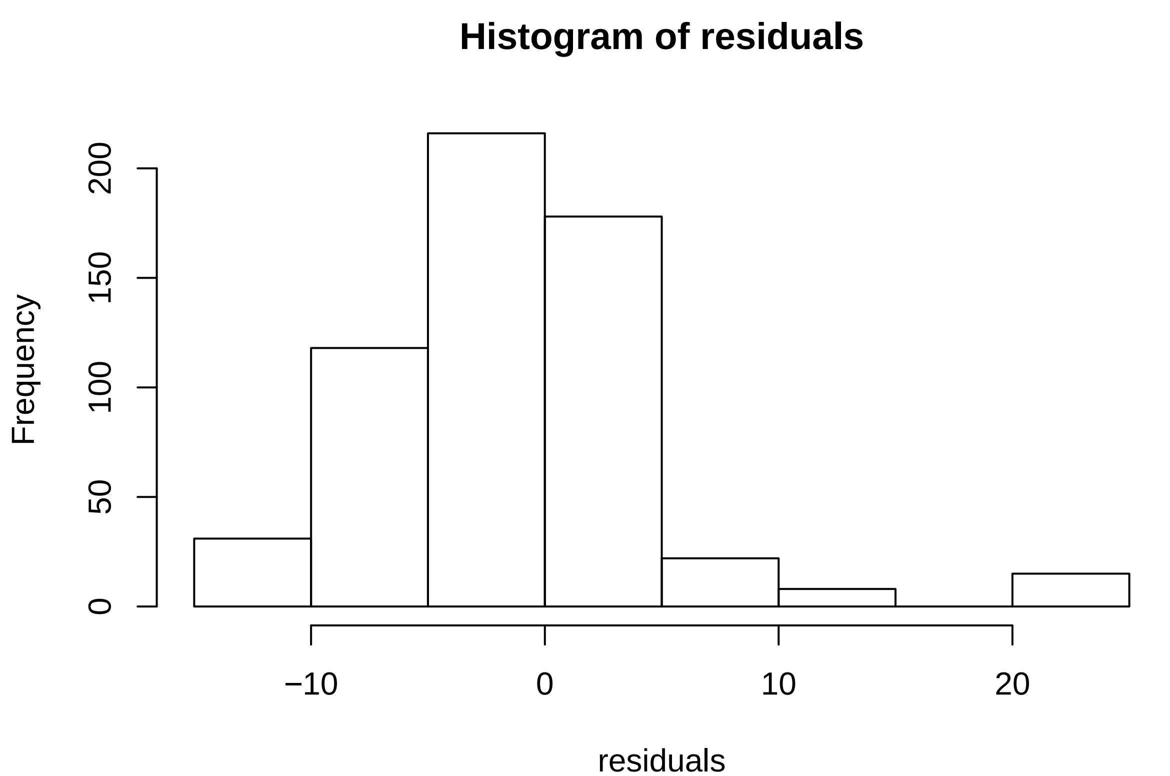

```
## NULL
```

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



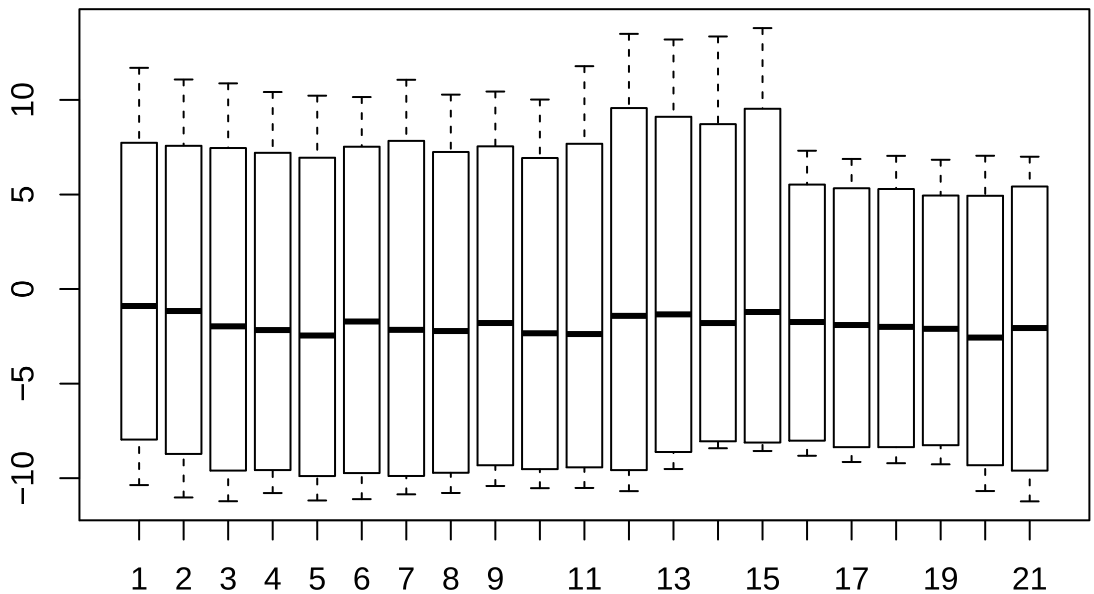

```
## NULL
```

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



```
##      97.5%        90%        10%       2.5%
##  14.164569   4.596827  -8.274644 -10.945837
```

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




```
##      97.5%        90%        10%       2.5%
##  14.164569   4.596827  -8.274644 -10.945837
```

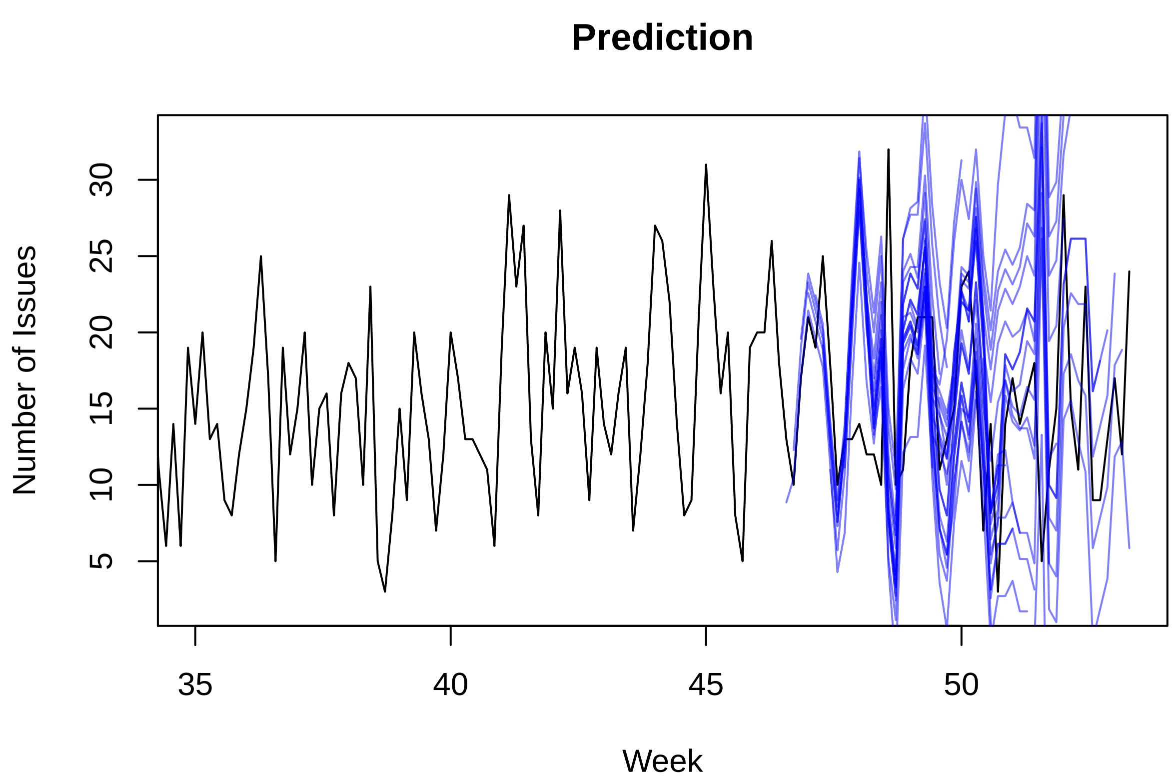
Double differencing

```r
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))
```

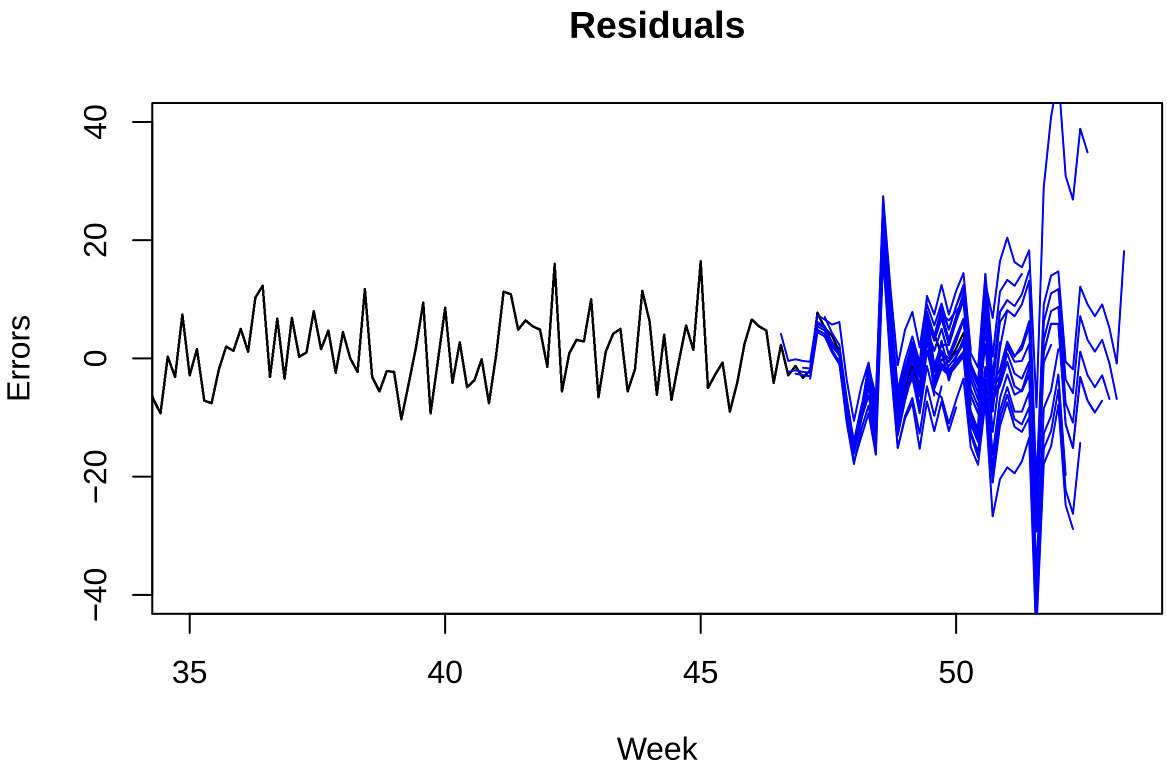
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.254614	6.885617	5.480876	-17.69630	48.98434	0.8840747	0.2055918	NA
Test set	-1.478134	10.286756	7.954810	-34.96466	73.11205	1.2822404	0.0797860	1.113564

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



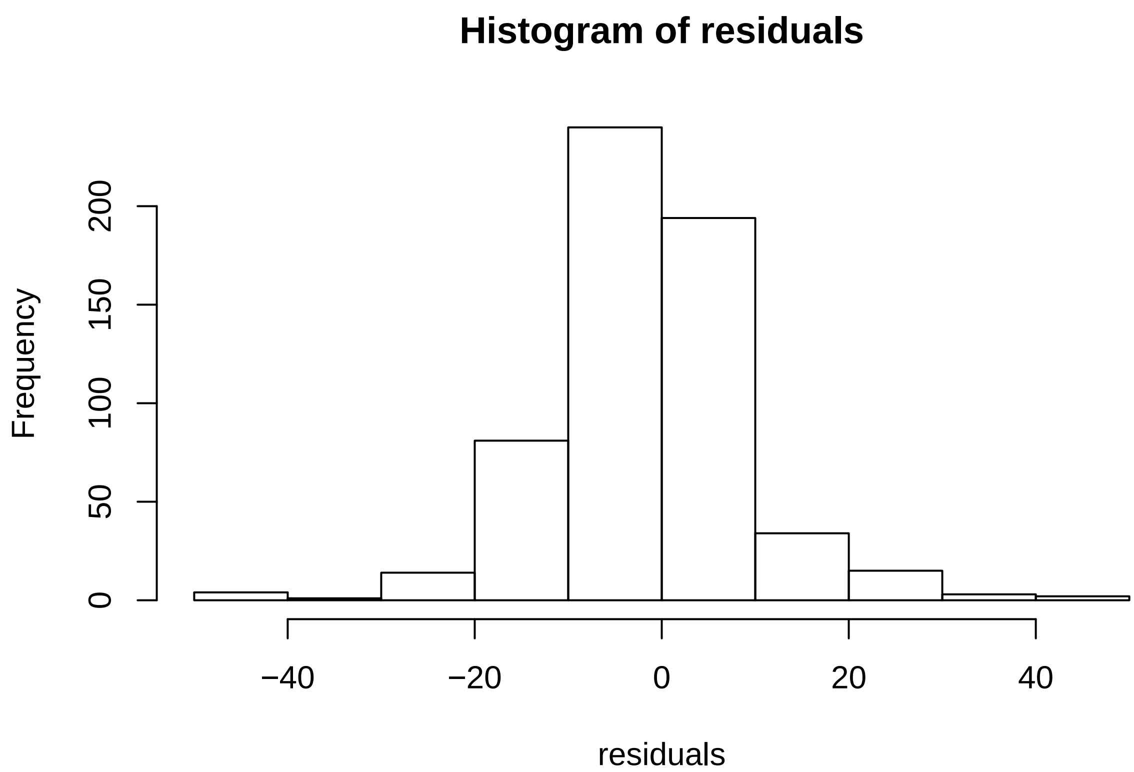

```
## NULL
```

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



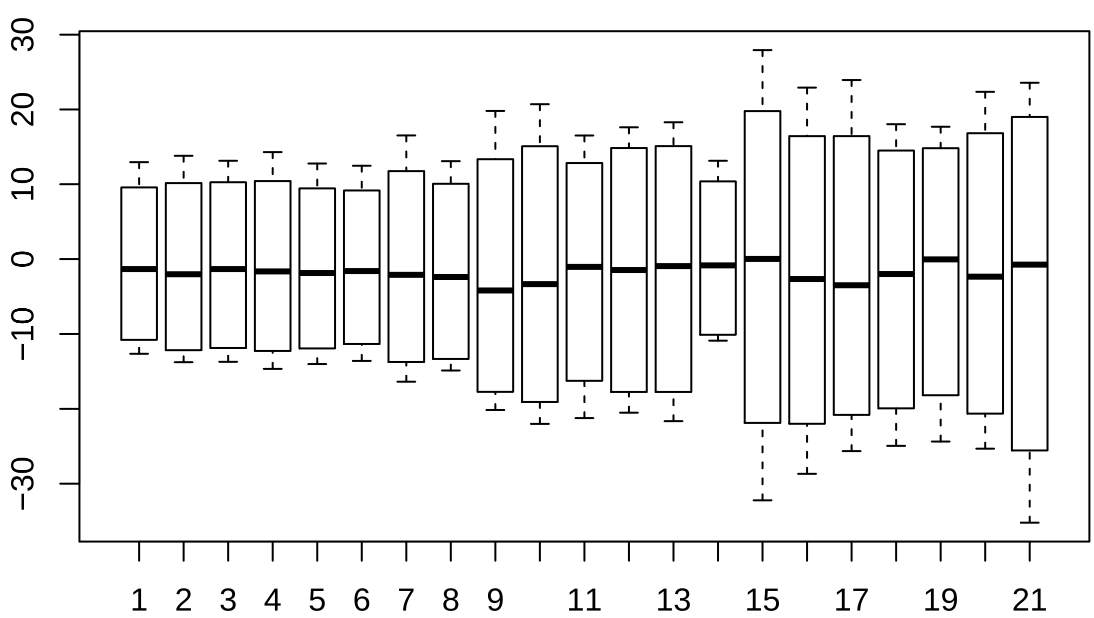

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##  23.285714   9.657143 -12.900000 -21.996429
```

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




```
##      97.5%        90%        10%       2.5%
##  23.285714   9.657143 -12.900000 -21.996429
```

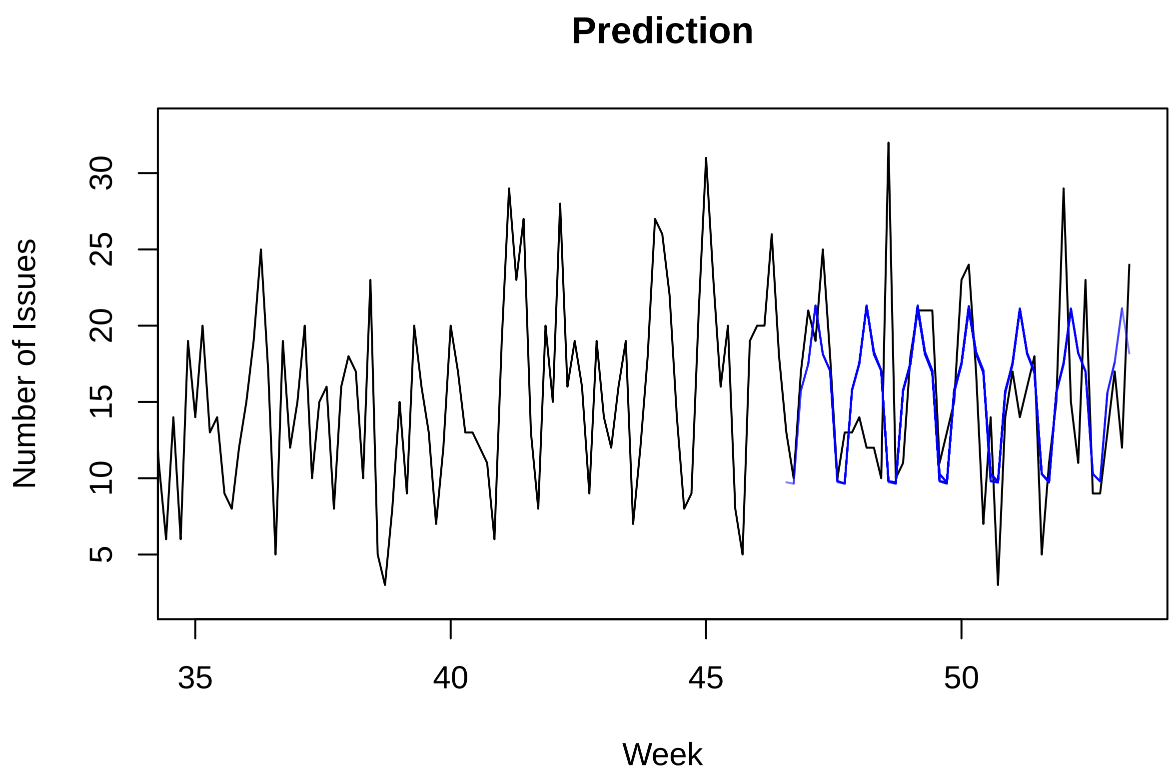
Regression

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))
```

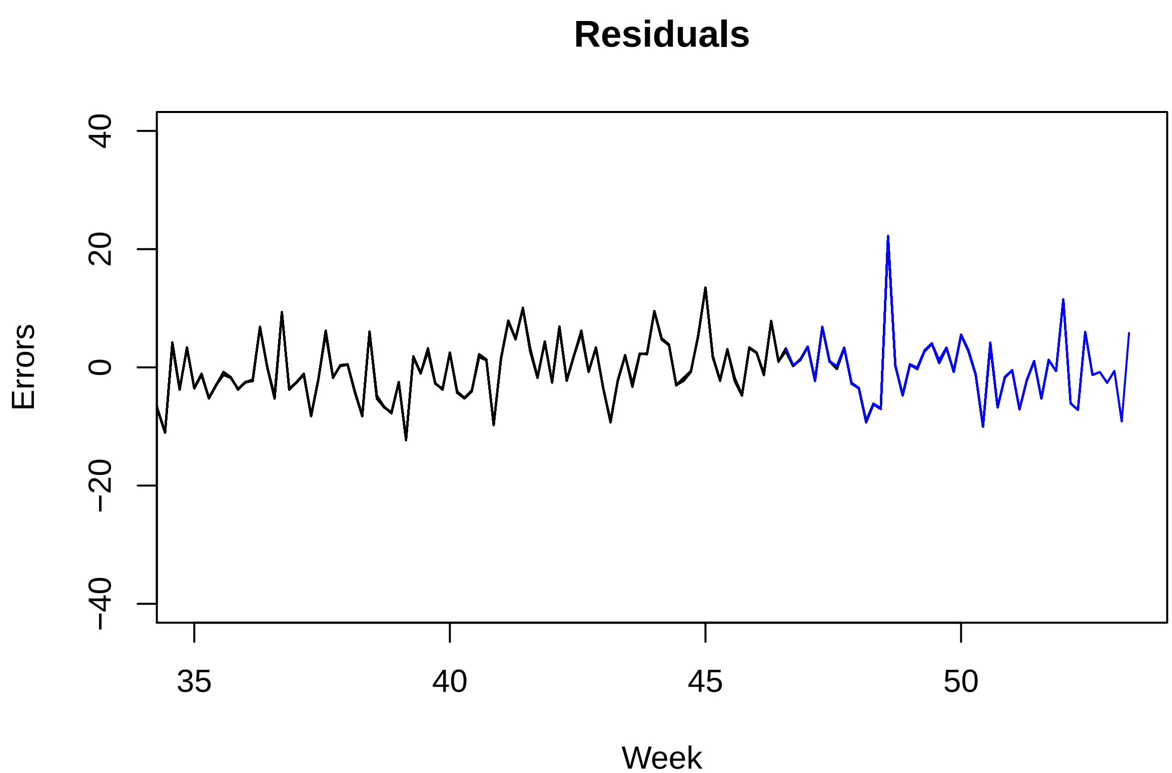
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0000000	5.491848	4.351689	-21.20929	40.99356	0.7036720	0.2437525	NA
Test set	-0.2082656	5.687318	4.033644	-17.23947	35.04906	0.6524141	-0.1093520	0.614897

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



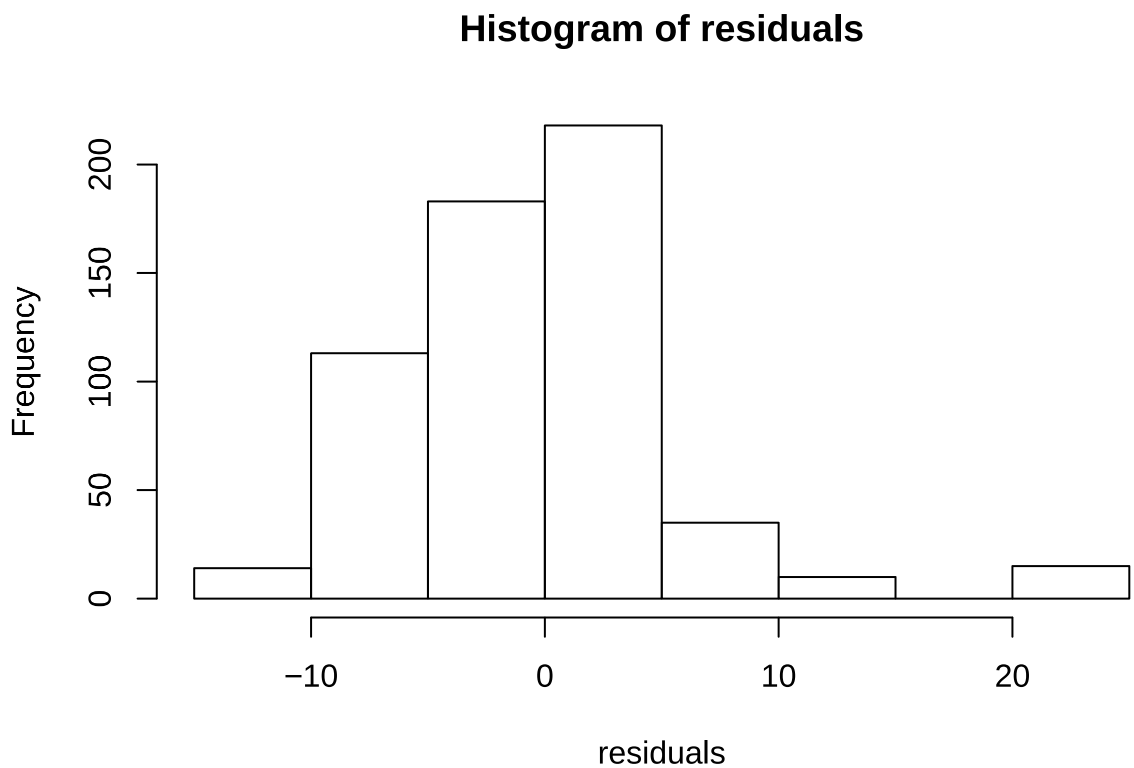

```
## NULL
```

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



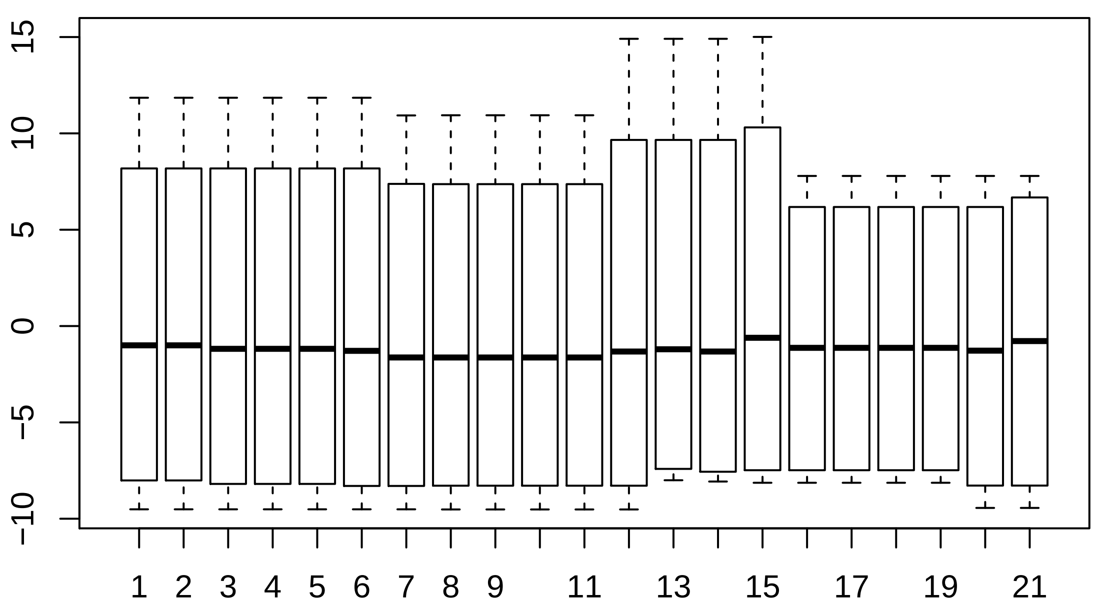

```
## NULL
```

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




```
##      97.5%       90%       10%      2.5%
## 14.981622  5.446809 -7.069171 -9.916667
```

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



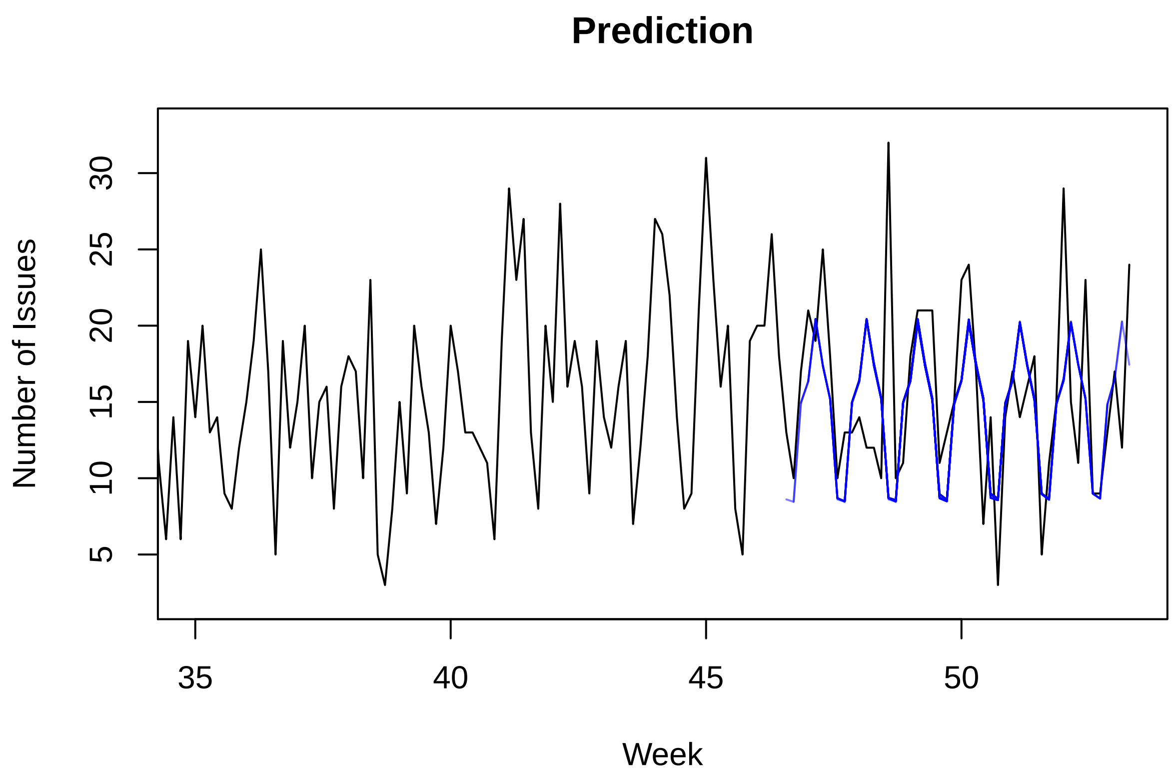

```
##      97.5%       90%       10%      2.5%
## 14.981622  5.446809 -7.069171 -9.916667
```

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))
```

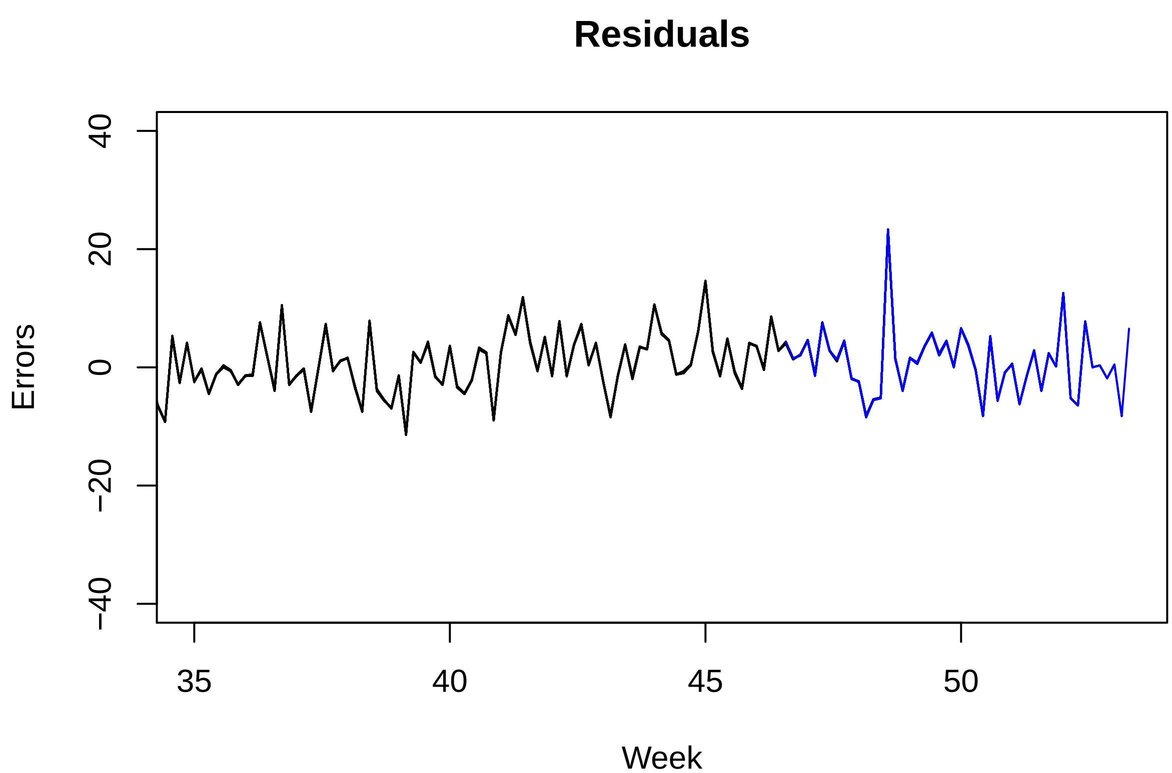
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	1.098677	5.610920	4.411284	-11.487727	37.80905	0.7133096	0.2422292	NA
Test set	0.890428	5.758203	4.170011	-8.028692	33.39988	0.6744938	-0.1006868	0.6309672

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



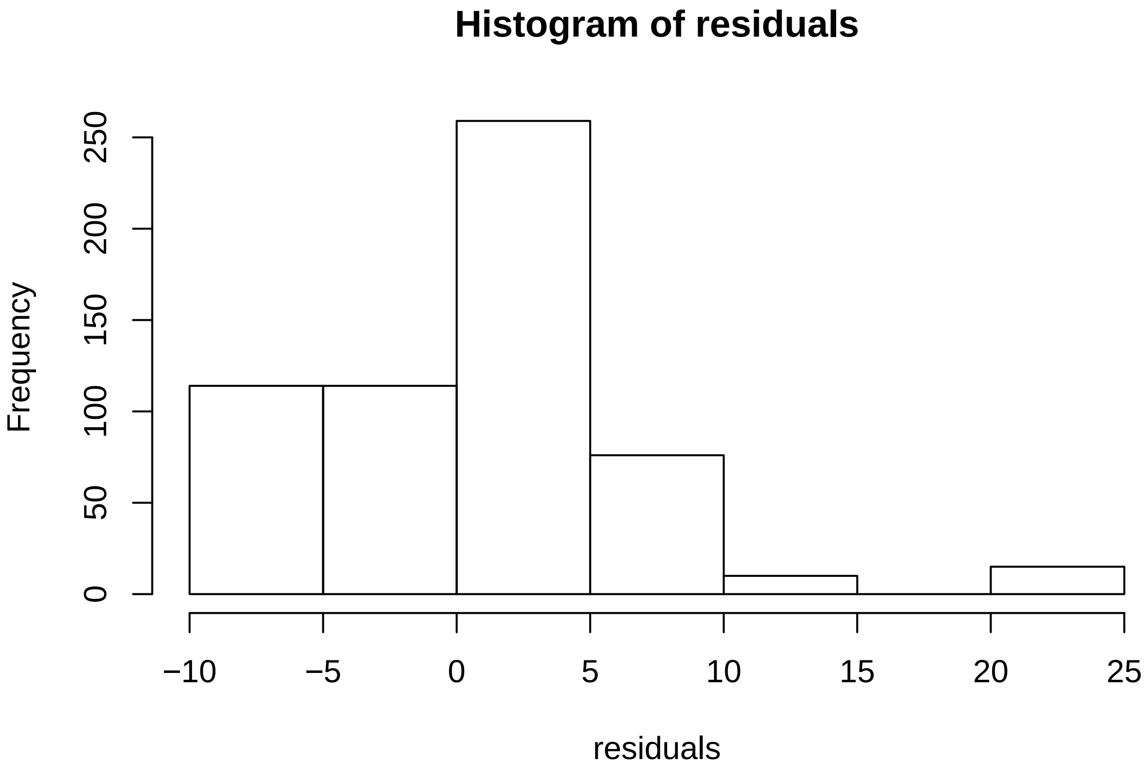

```
## NULL
```

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



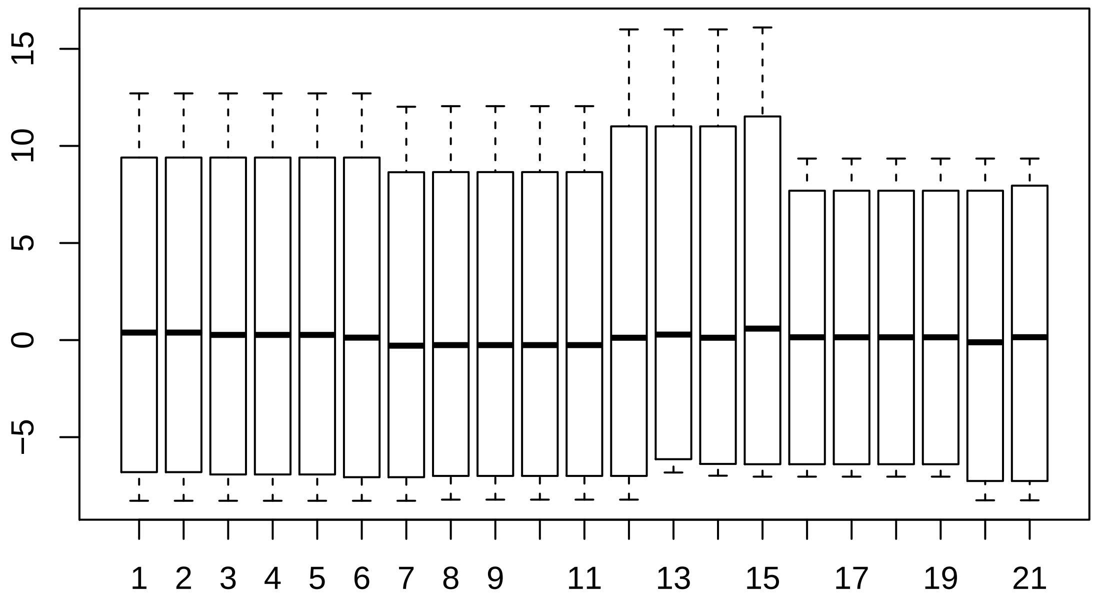

```
## NULL
```

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




```
##      97.5%       90%       10%      2.5%
## 16.066785  6.560412 -5.821855 -8.243734
```

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



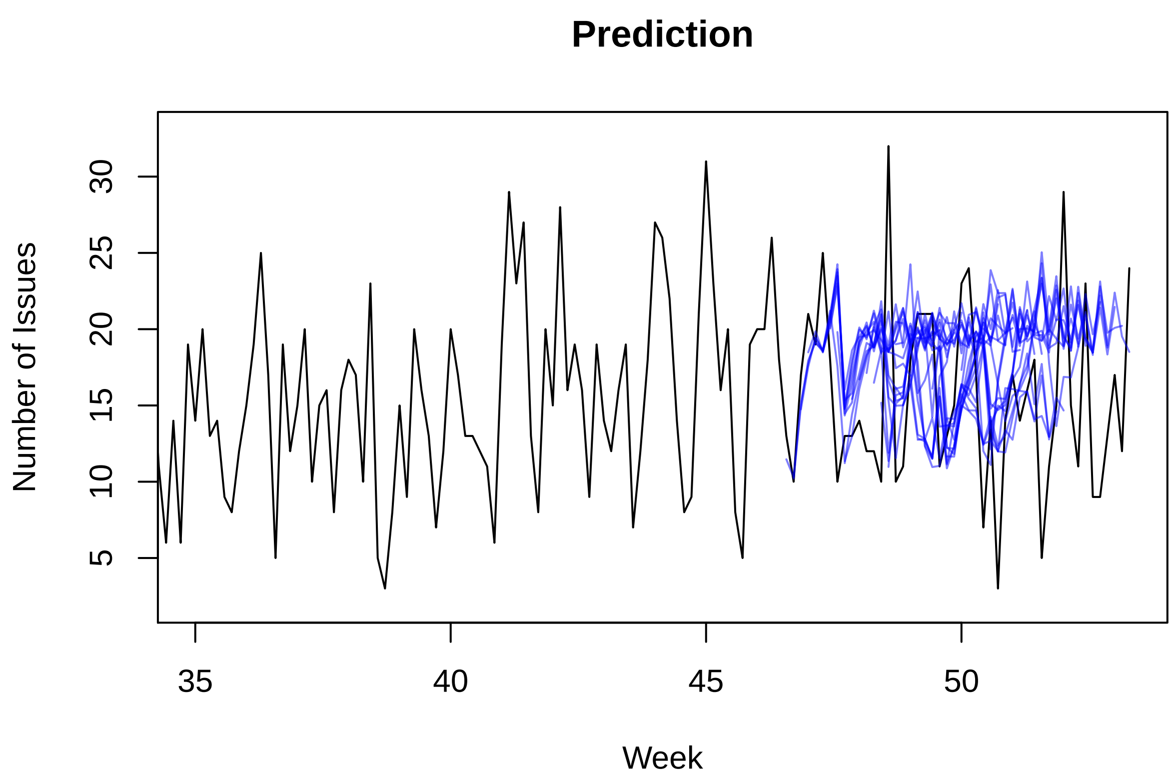

```
##      97.5%       90%       10%      2.5%
## 16.066785  6.560412 -5.821855 -8.243734
```

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))
```

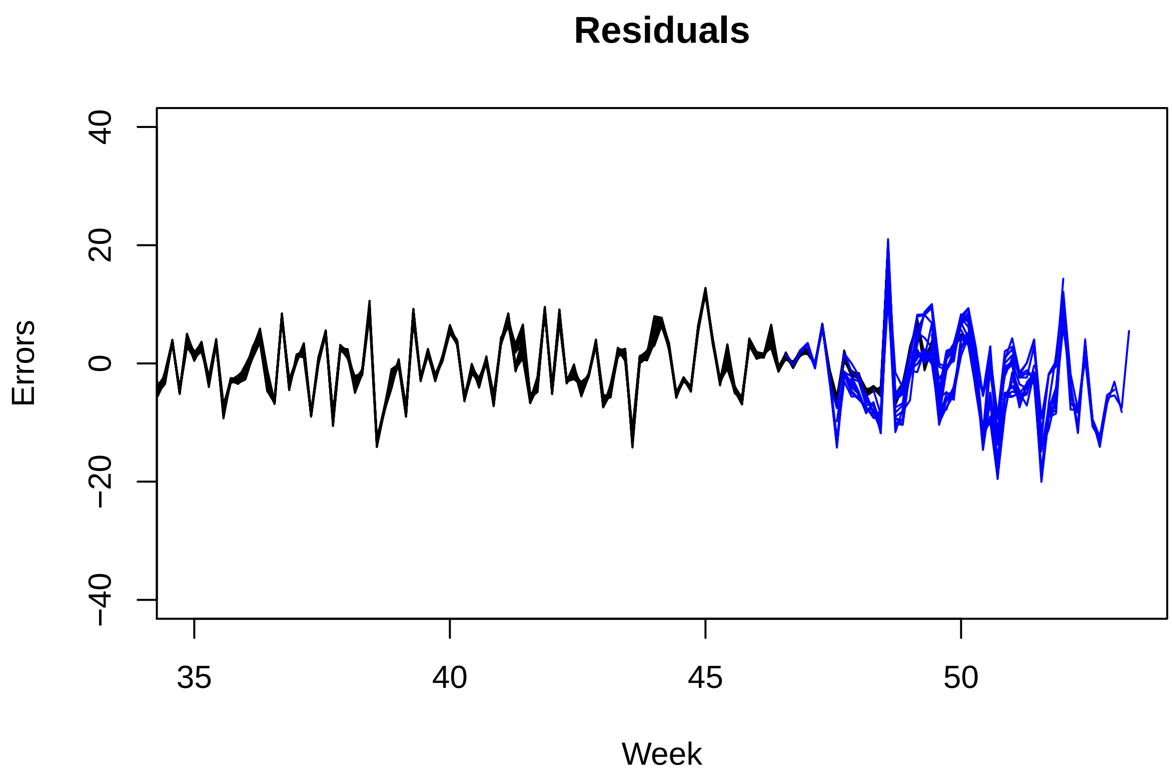
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0043712	5.459721	4.450876	-22.97322	42.84582	0.7196801	0.0062077	NA
Test set	-2.6458606	7.326478	5.920191	-46.83562	60.69053	0.9572721	0.0575069	0.7479885

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



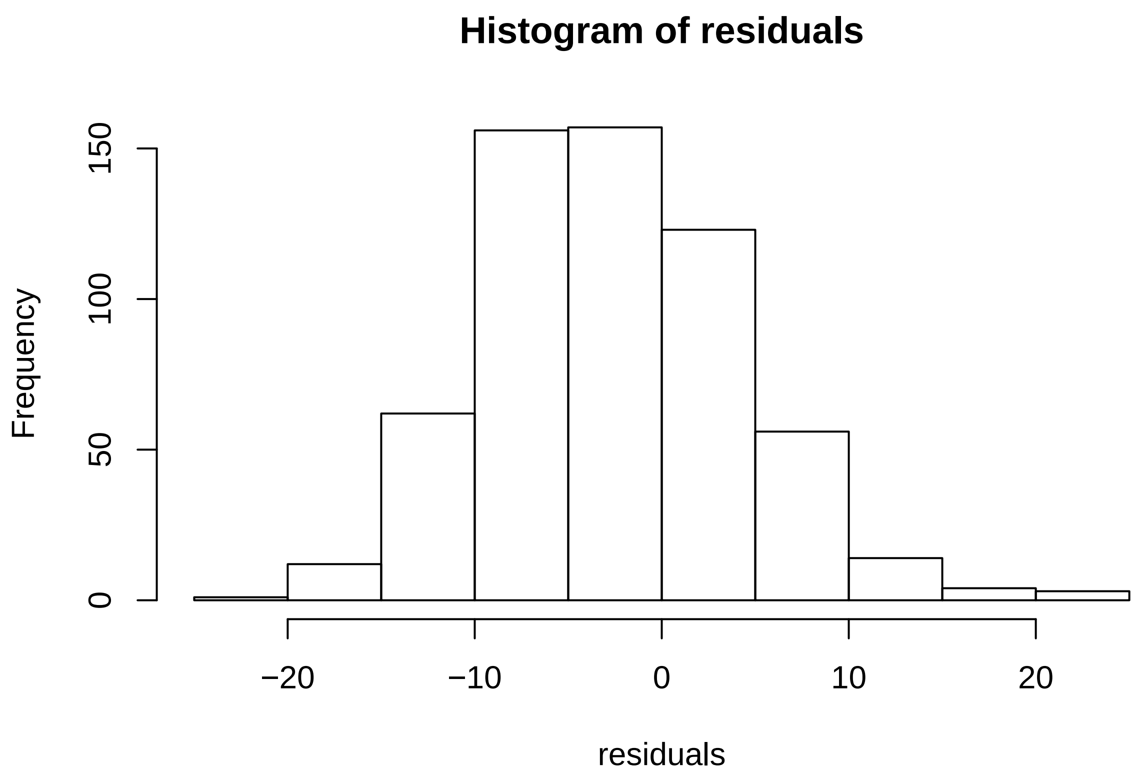

```
## NULL
```

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



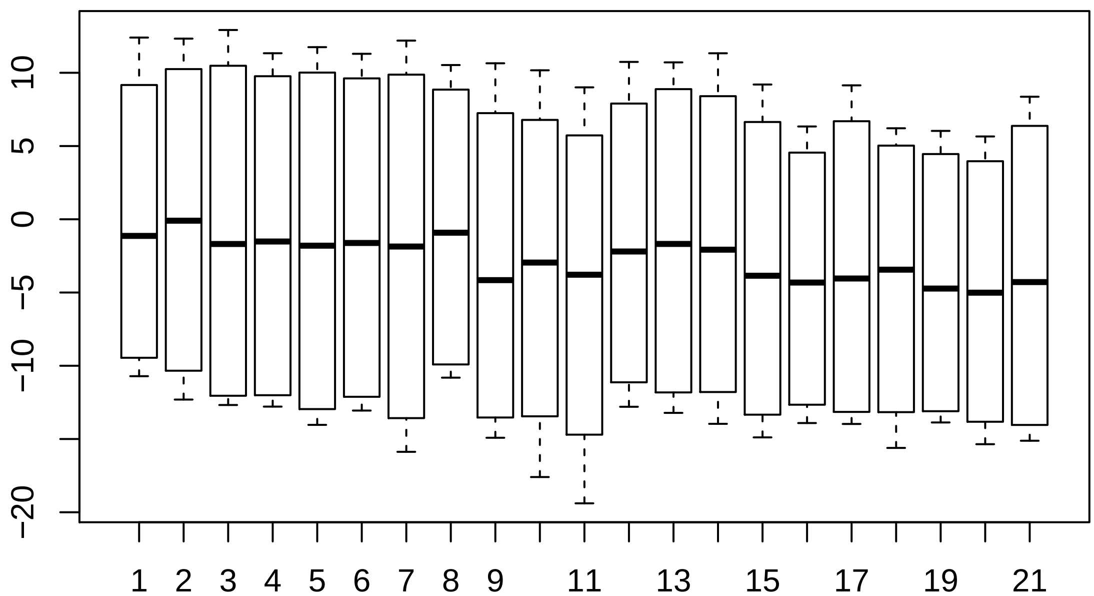

```
## NULL
```

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




```
##       97.5%         90%         10%        2.5%
##   12.424069   6.719952 -11.318358 -14.628073
```

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



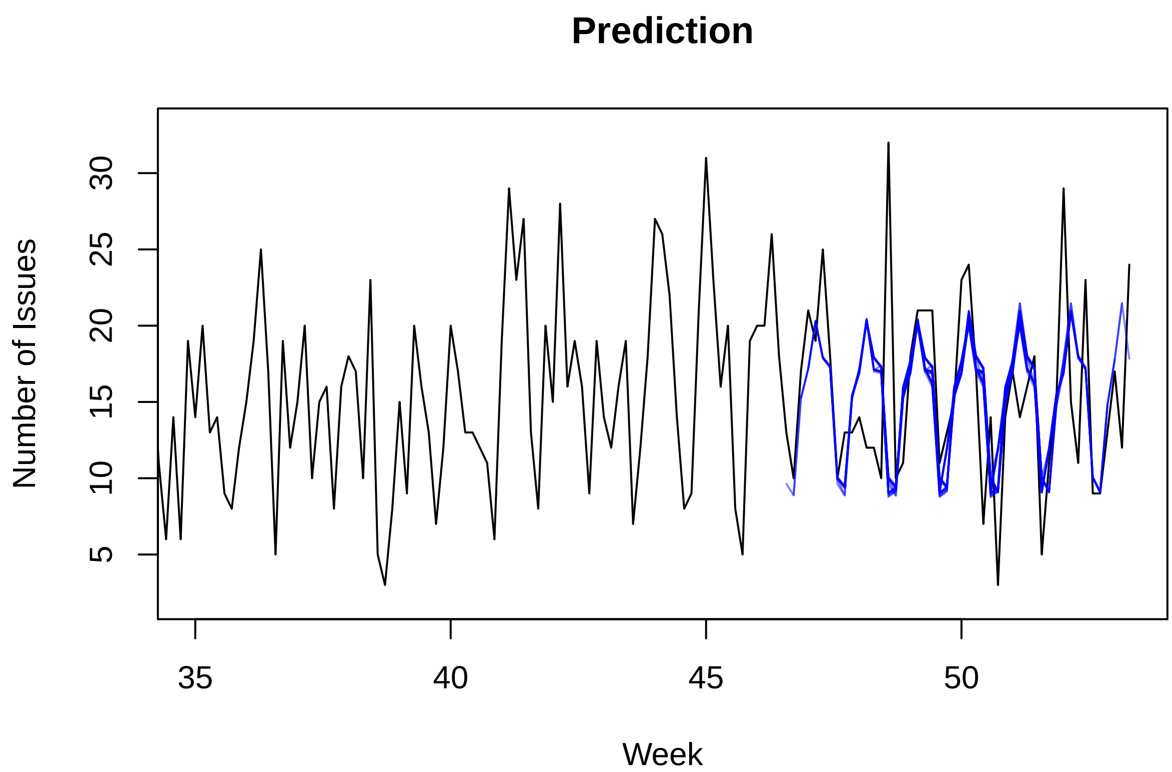

```
##       97.5%         90%         10%        2.5%
##   12.424069   6.719952 -11.318358 -14.628073
```

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))
```

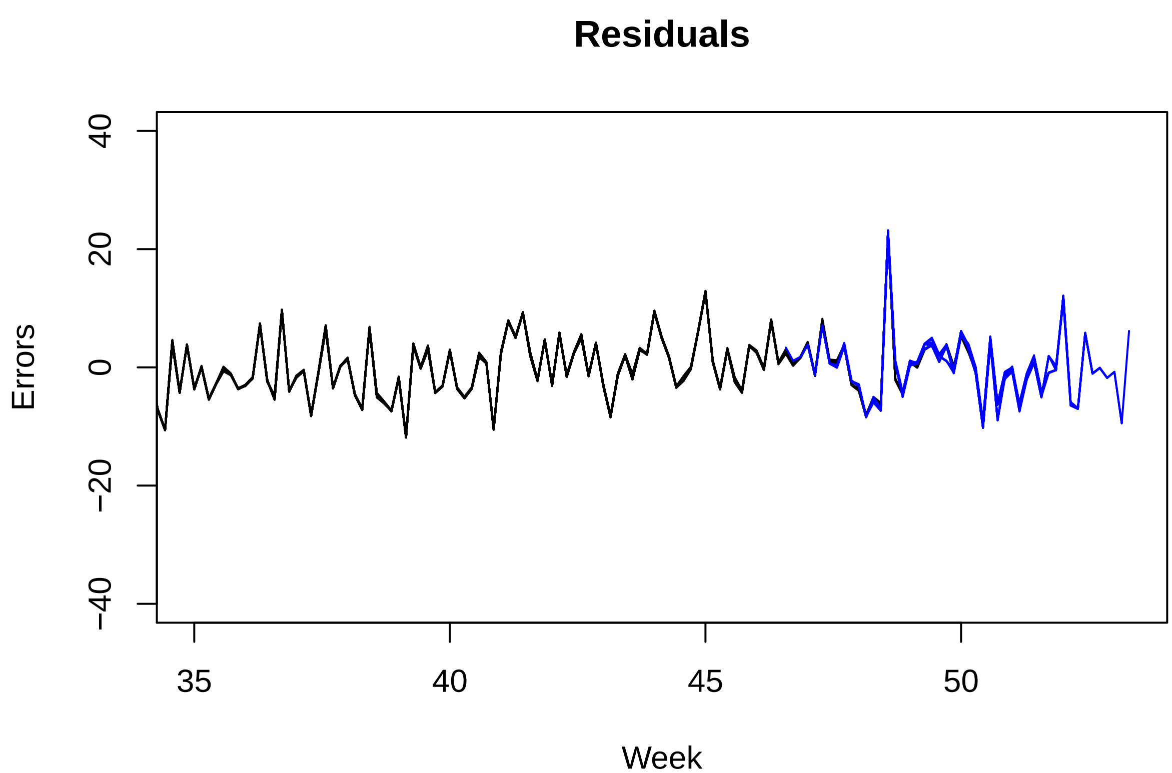
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.000000	5.424103	4.346803	-20.57193	40.52153	0.7031672	0.1321365	NA
Test set	0.119937	5.723262	4.037382	-15.31422	34.93454	0.6533576	-0.1197018	0.6190436

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



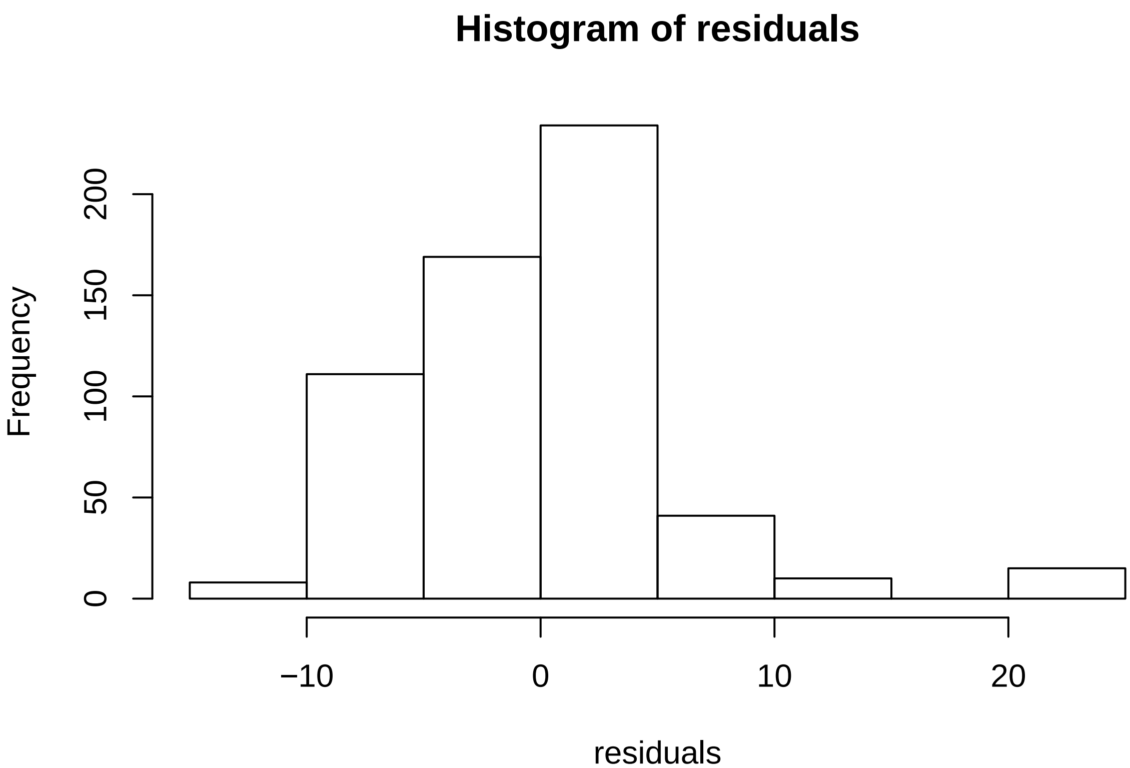

```
## NULL
```

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



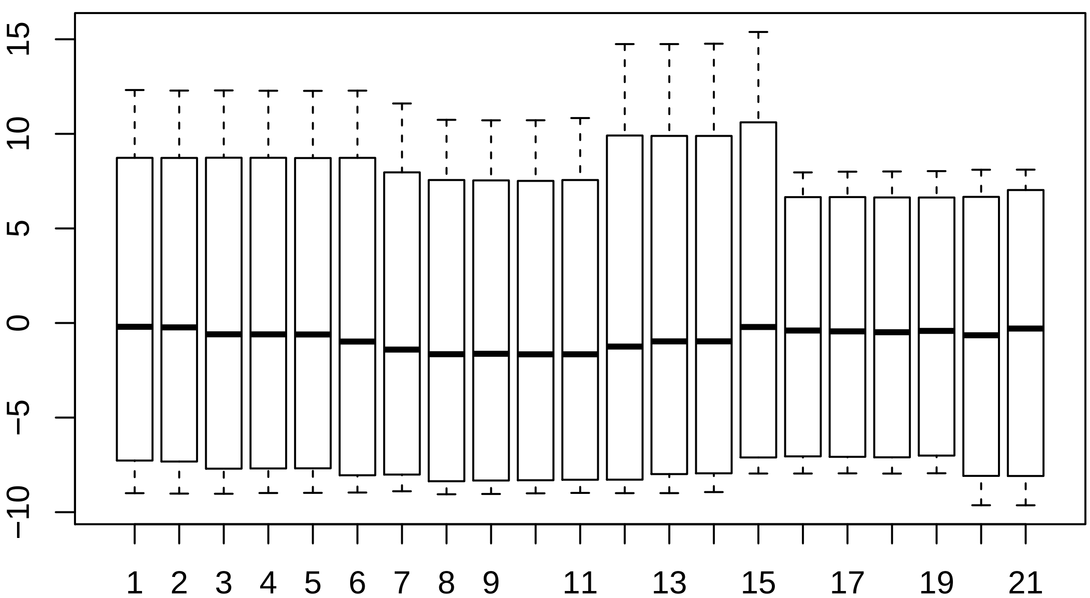

```
## NULL
```

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




```
##      97.5%       90%       10%      2.5%
## 15.321147  5.283710 -6.982441 -9.467192
```

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



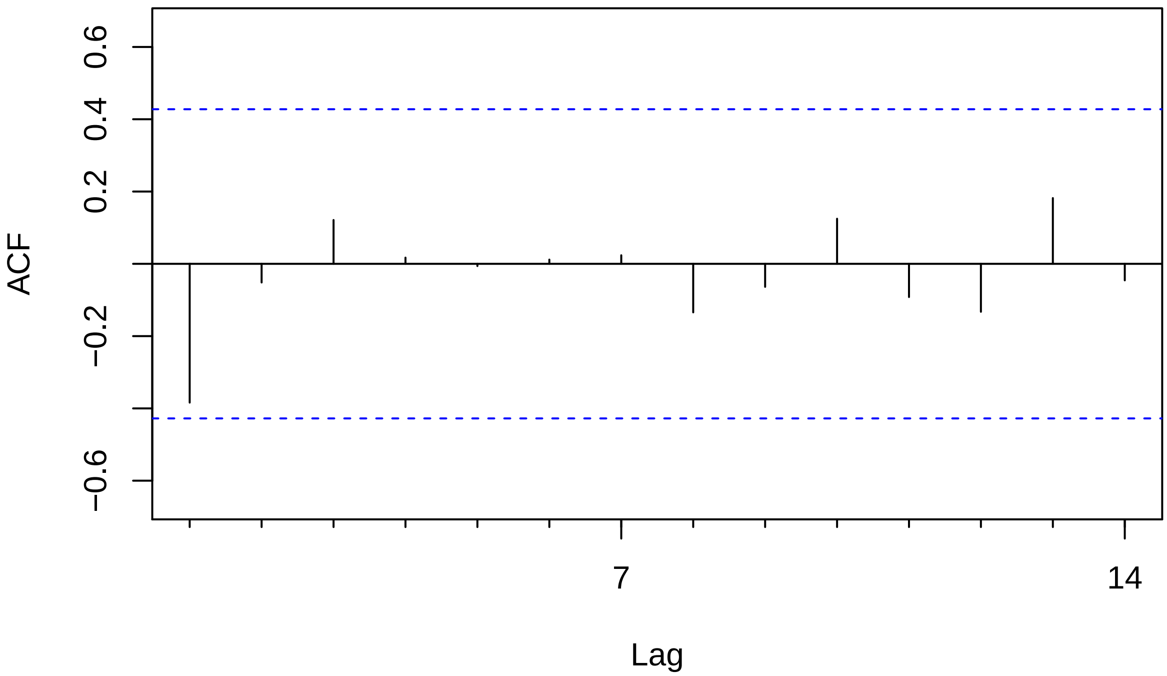

```
##      97.5%       90%       10%      2.5%
## 15.321147  5.283710 -6.982441 -9.467192
```

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))
```

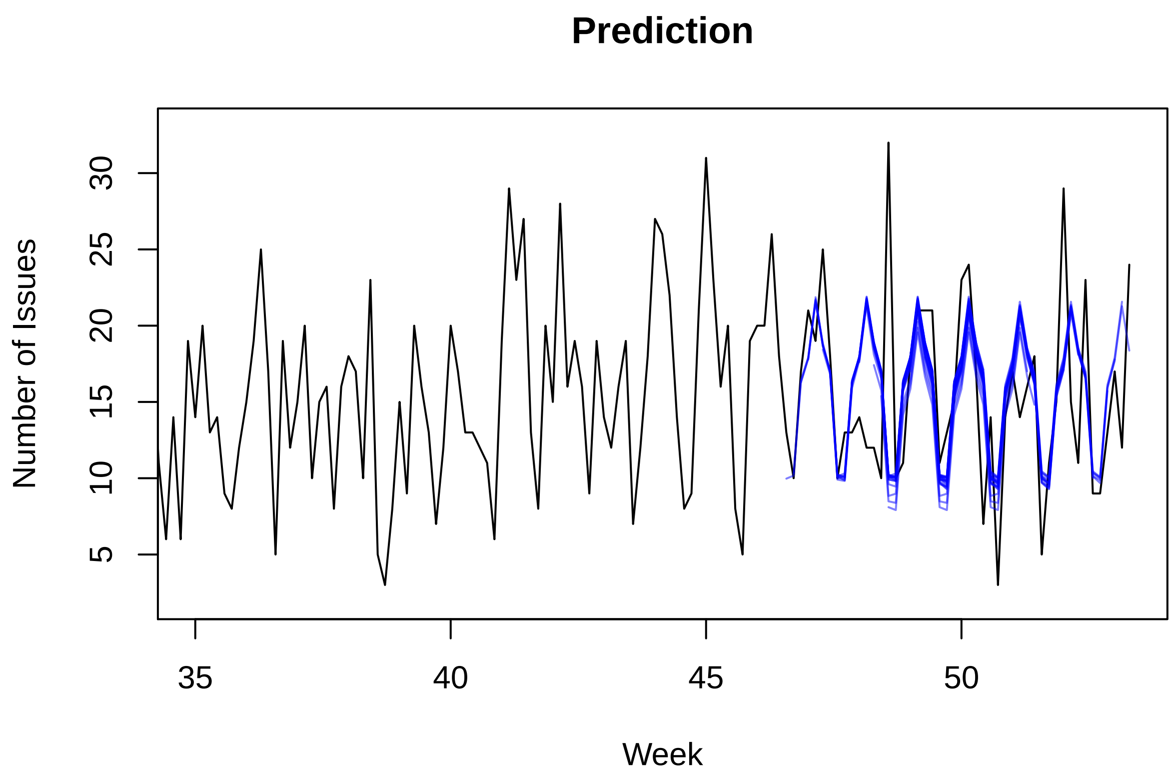
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.6972798	8.376390	6.908317	-29.71593	64.45100	1.1143712	0.3993600	NA
Test set	-0.1421892	5.716864	4.094753	-16.46059	34.99502	0.6606975	-0.1050965	0.6180334

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



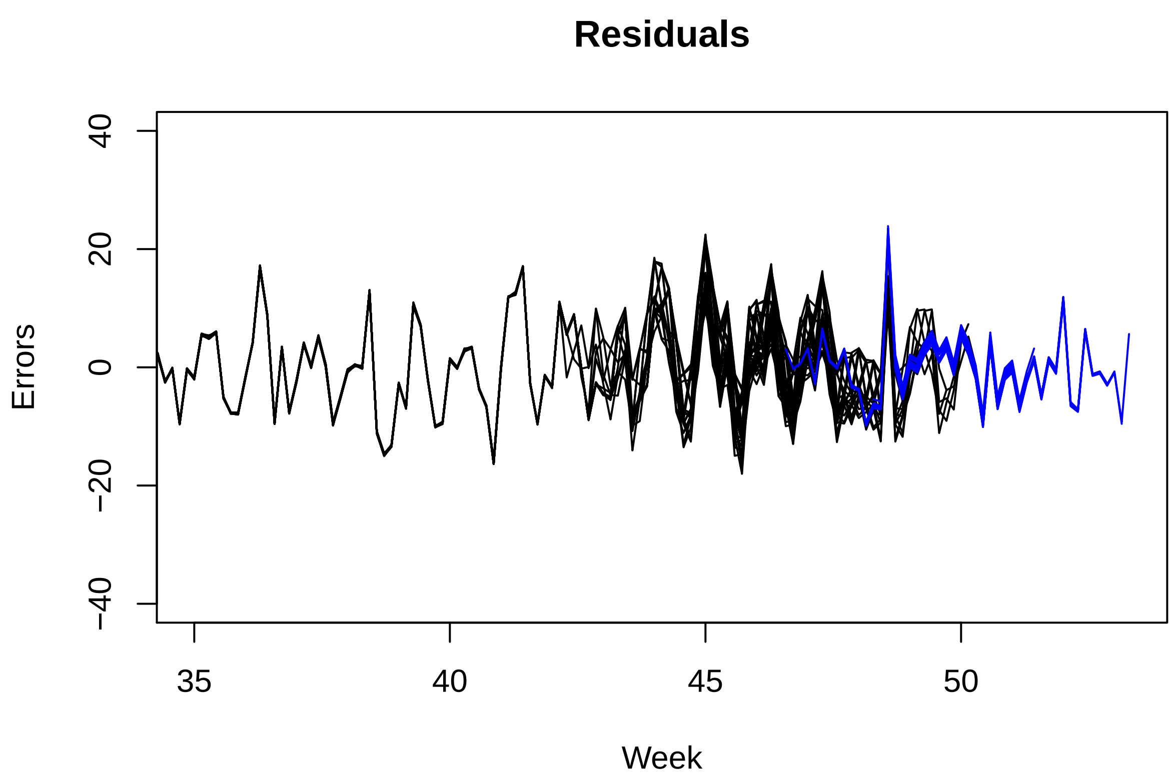

```
test <- list()
test$train.ts <- issues.ts
test$valid.ts <- naive(issues.ts, h=n.valid)$mean

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)
```



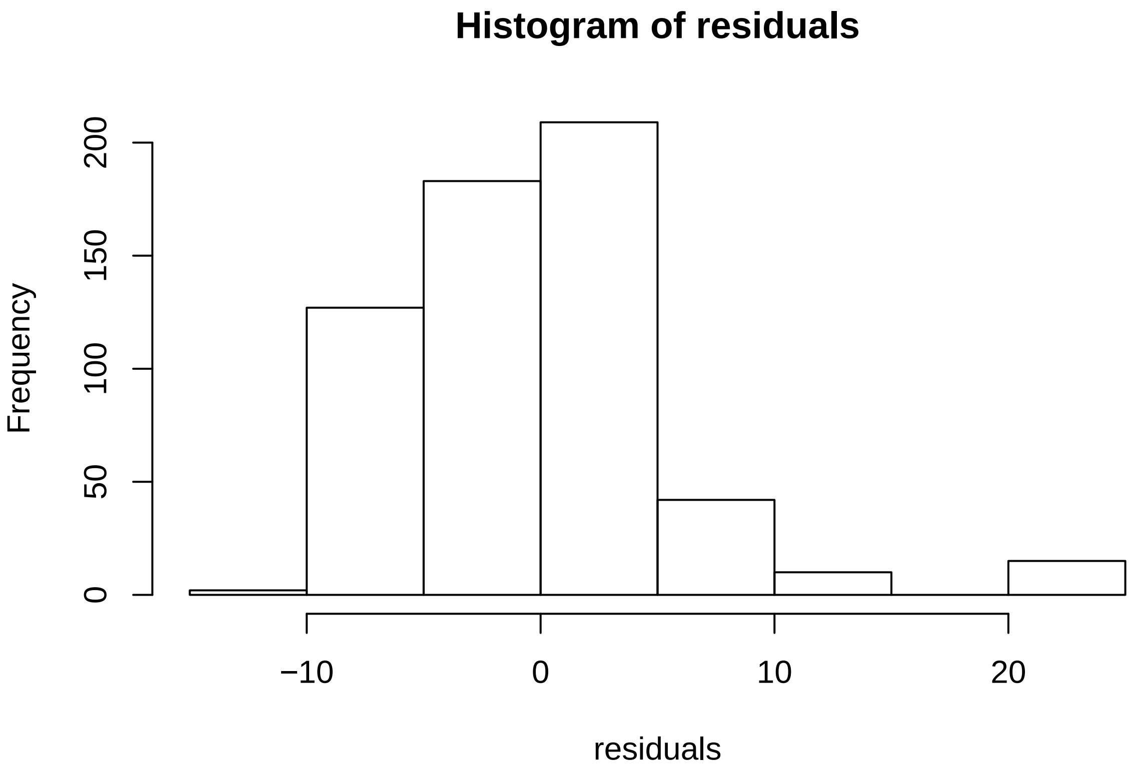

```
## NULL
```

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



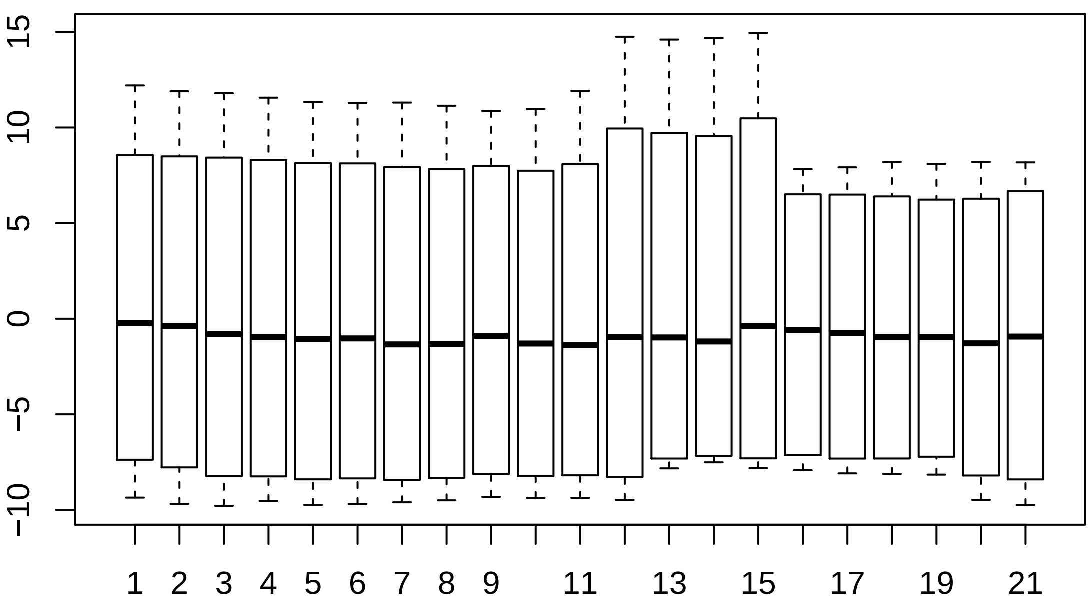

```
## NULL
```

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




```
##      97.5%       90%       10%      2.5%
## 15.119905  5.247453 -7.032985 -9.630716
```

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




```
##      97.5%       90%       10%      2.5%
## 15.119905  5.247453 -7.032985 -9.630716
```

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