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.

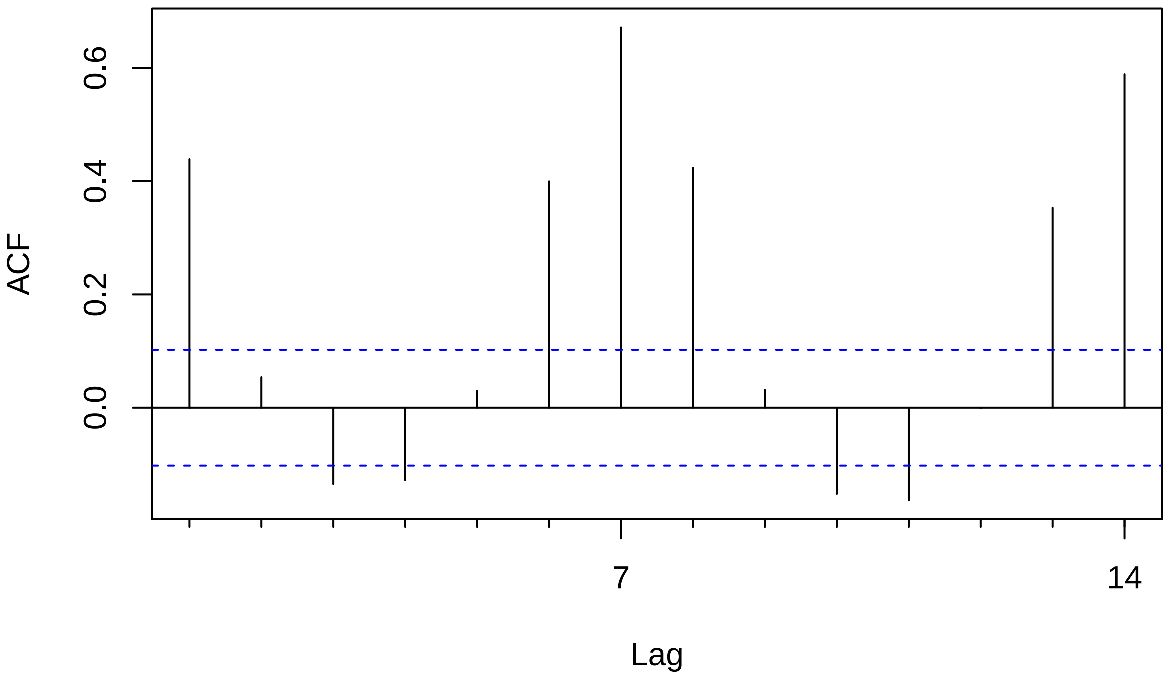
Load the data

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

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

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

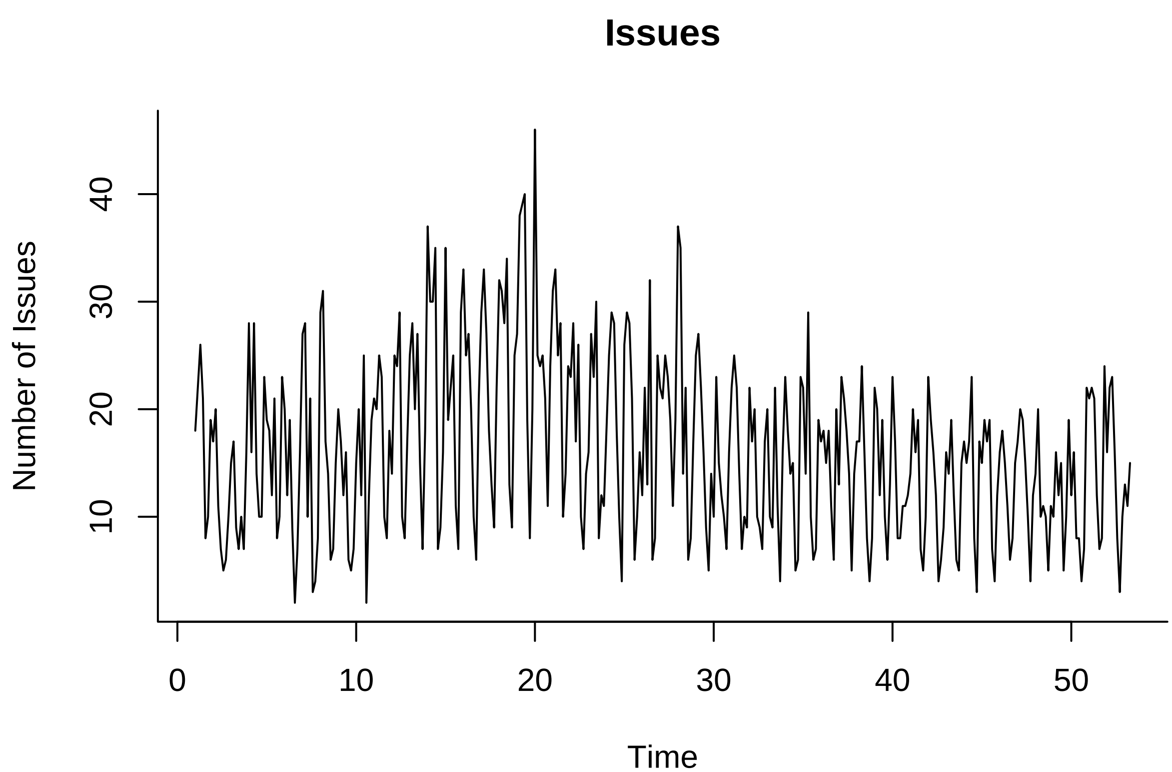
keep the last 12 months

```
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)
```

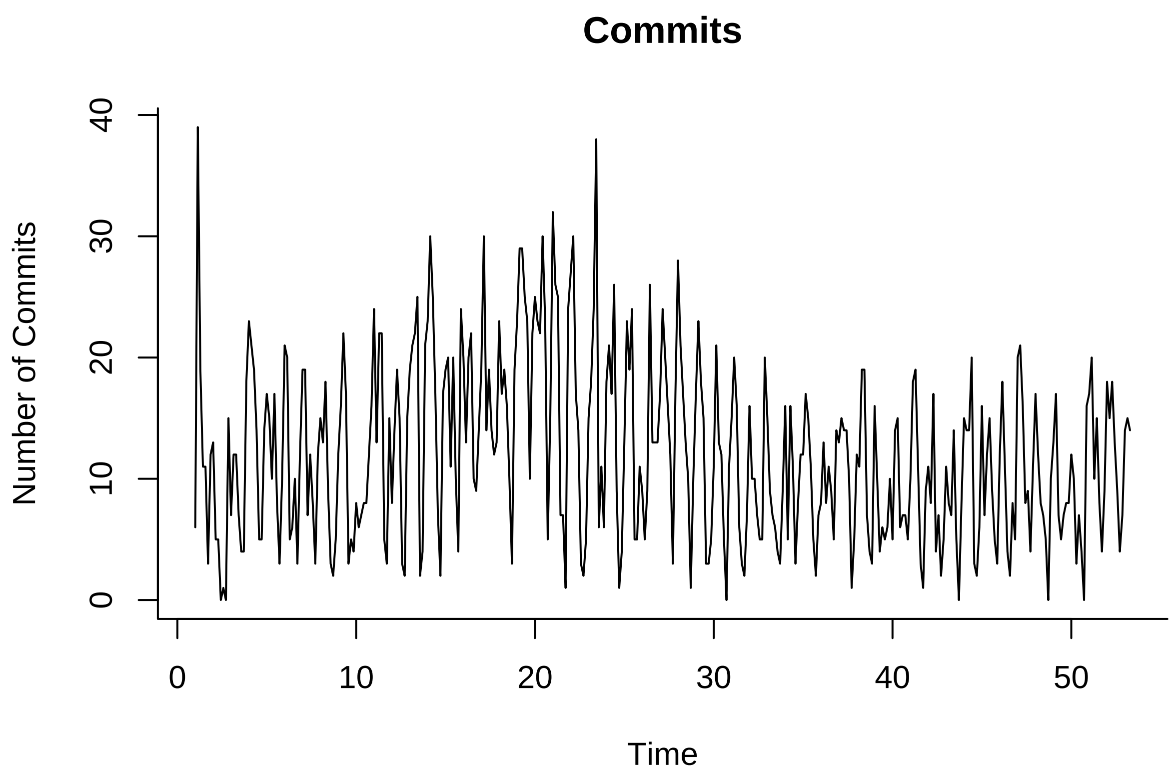
```
#loading issues and commits into a ts object
issues.ts <- ts(repository.csv$number_of_issues, frequency = 7)


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 = '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)
}
```

```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'))
}
```

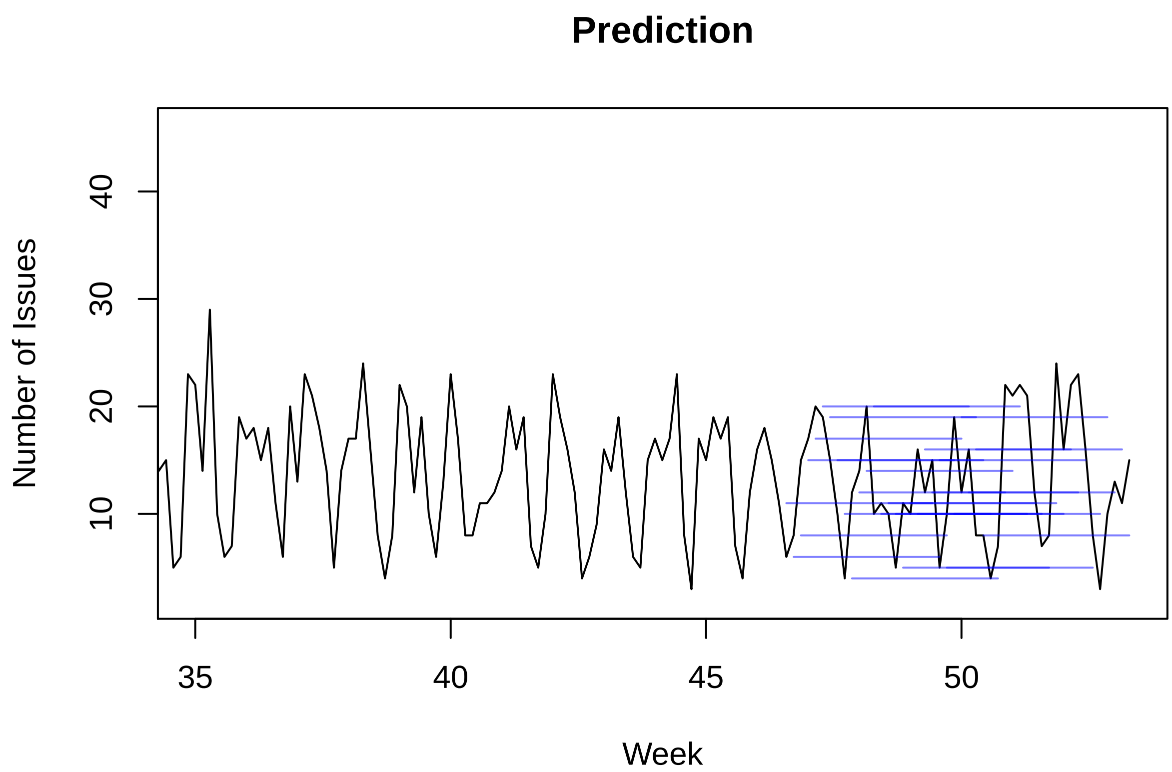
Naive Forecast

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))
```

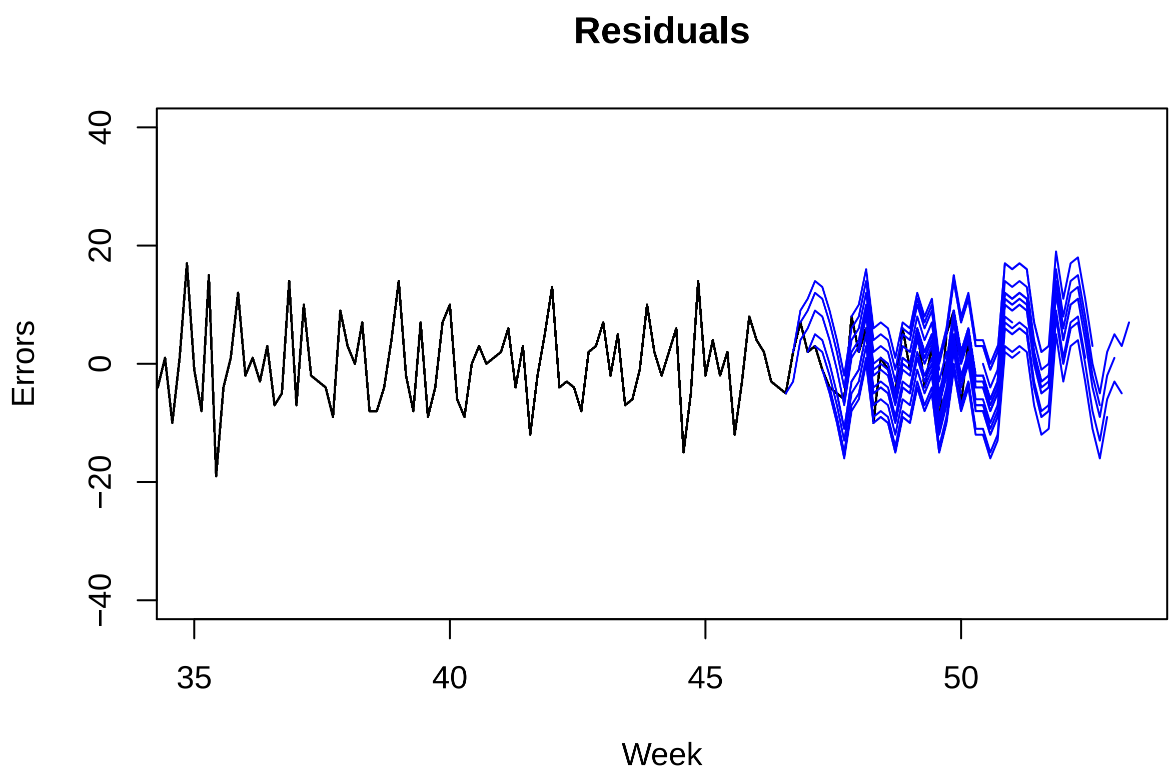
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.0175547	8.779646	6.915230	-23.39438	57.86929	1.359189	-0.1568400	NA
Test set	0.6564626	7.130379	6.017007	-21.58600	60.09941	1.183764	0.2981948	1.012527

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



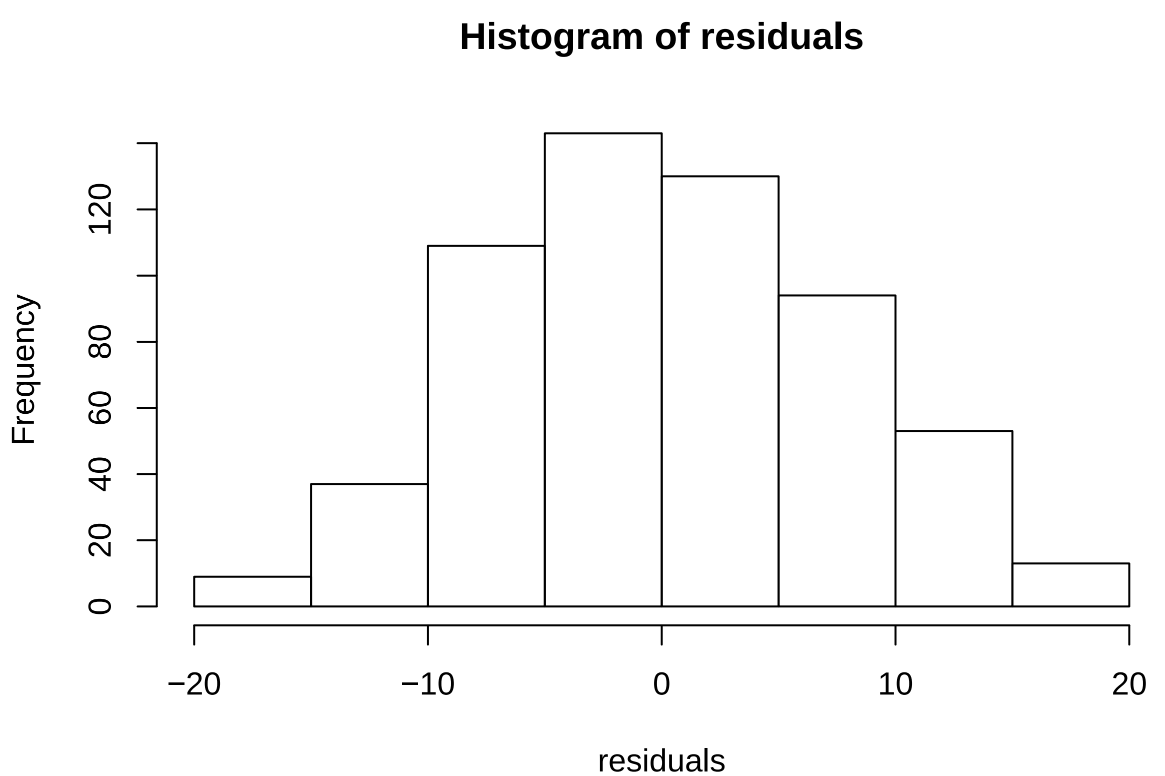

```
## NULL
```

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



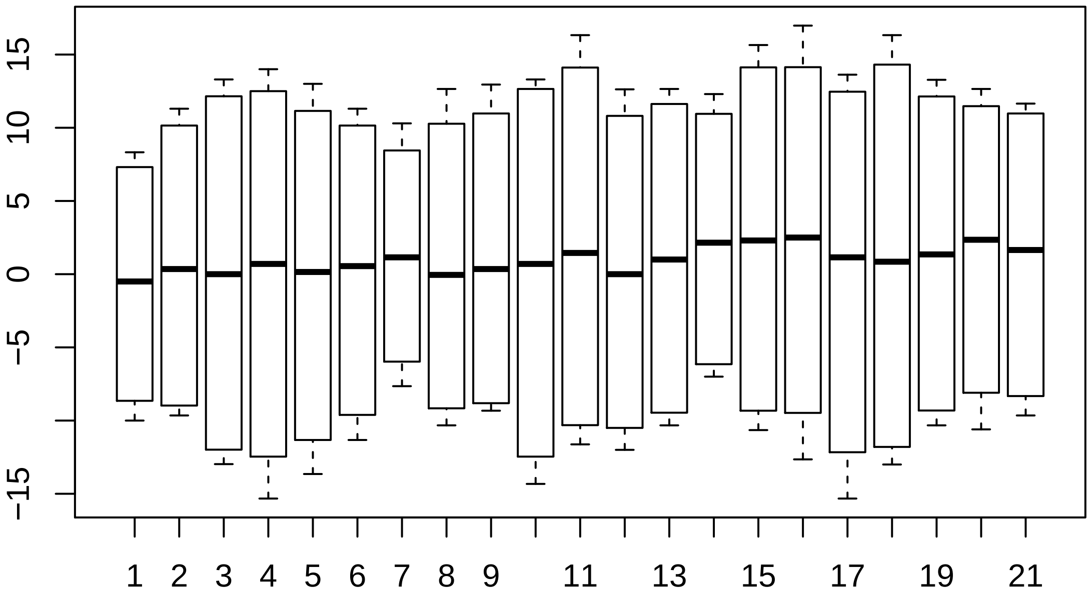

```
## NULL
```

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



```
##   97.5%     90%     10%    2.5% 
##  14.325  11.000  -9.000 -12.000
```

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



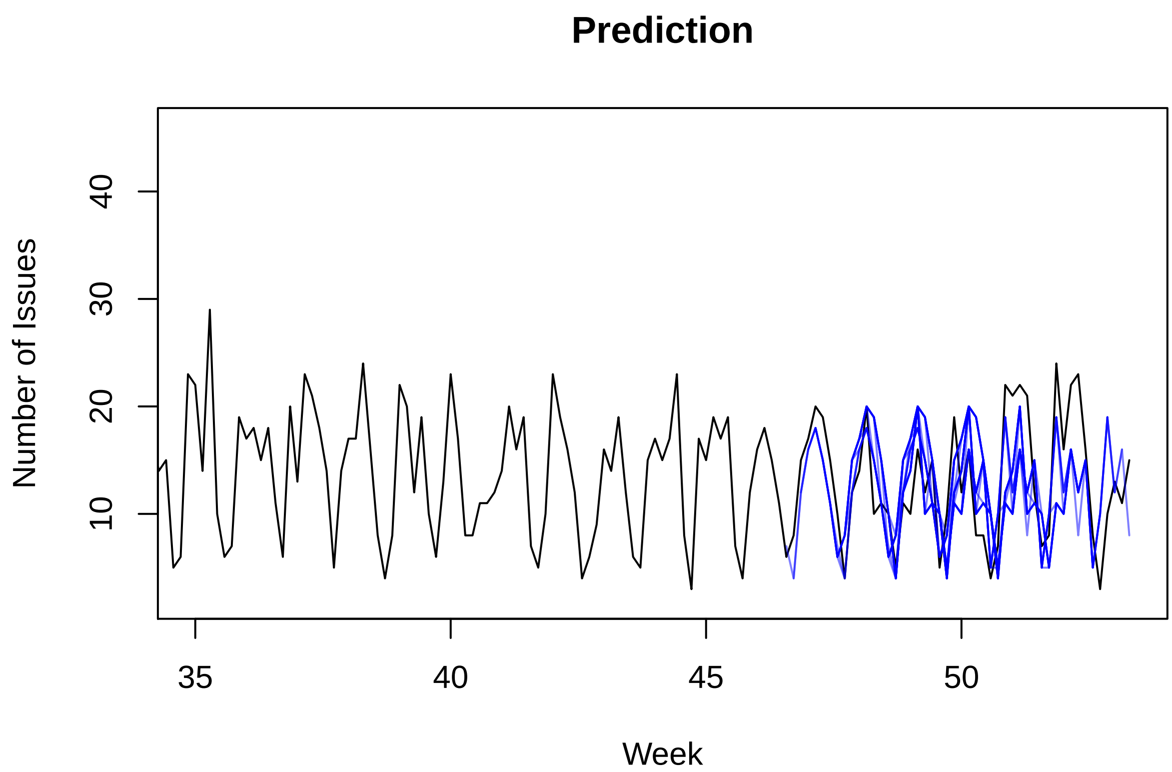

```
##   97.5%     90%     10%    2.5% 
##  14.325  11.000  -9.000 -12.000
```

Seasonal Naive

```r
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))
```

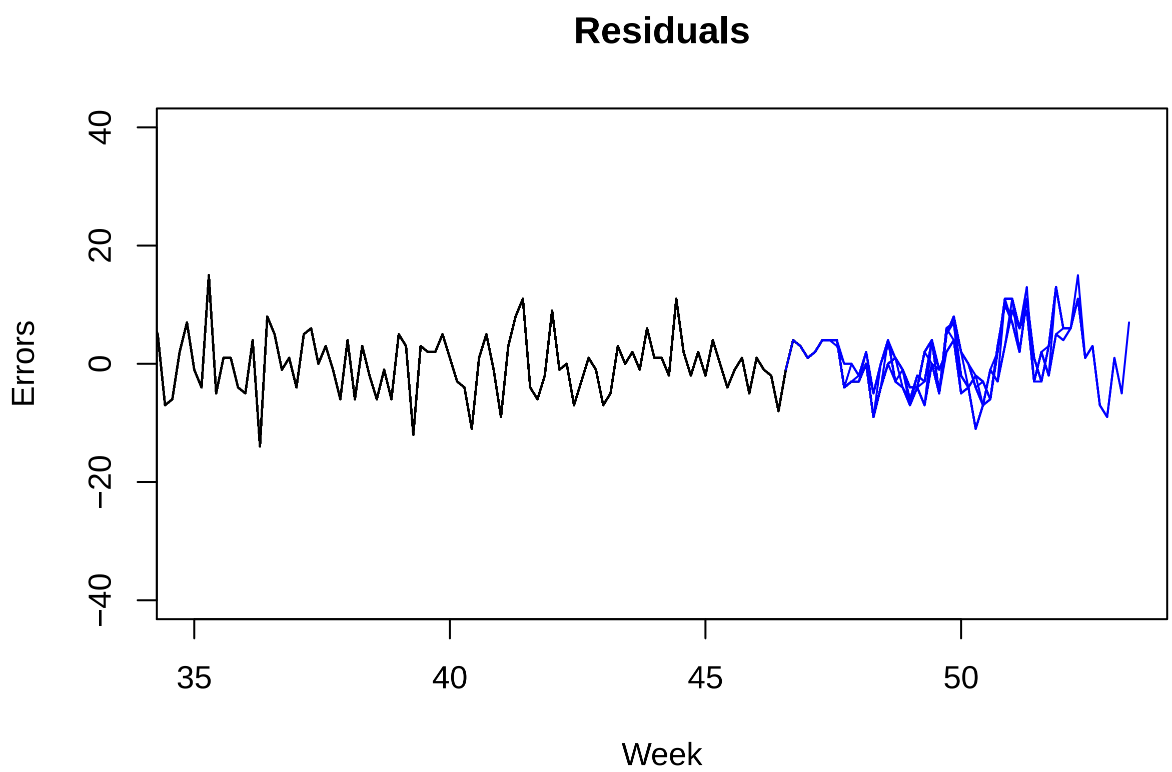
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.1147899	6.587703	5.087824	-11.10498	36.94525	1.000000	0.0298144	NA
Test set	0.4676871	5.216745	4.382653	-10.38518	40.58365	0.862931	0.2827566	0.7485362

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



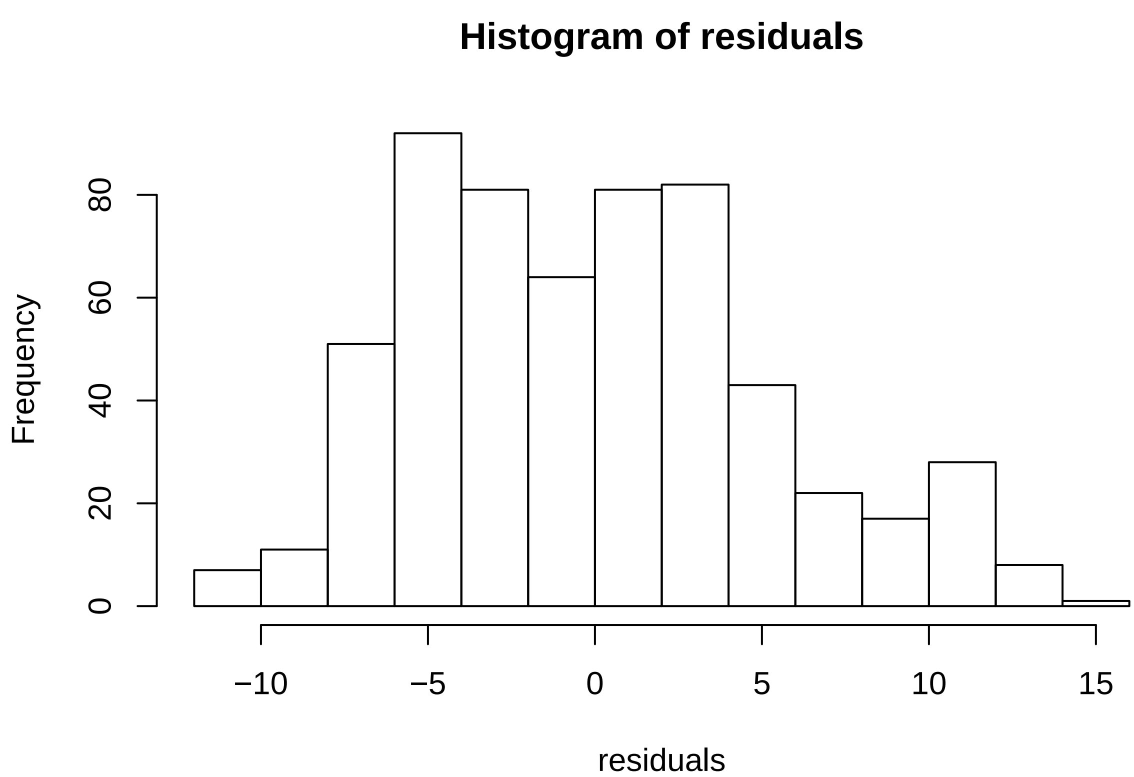

```
## NULL
```

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



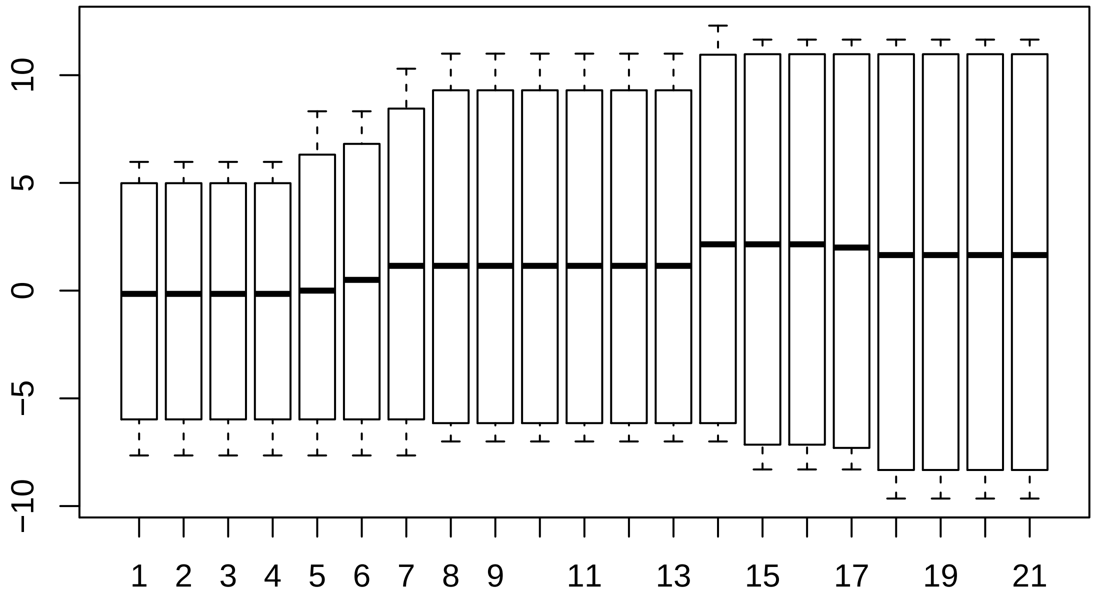

```
## NULL
```

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



```
## 97.5%    90%    10%   2.5%
##    11      8     -6     -9
```

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




```
## 97.5%    90%    10%   2.5%
##    11      8     -6     -9
```

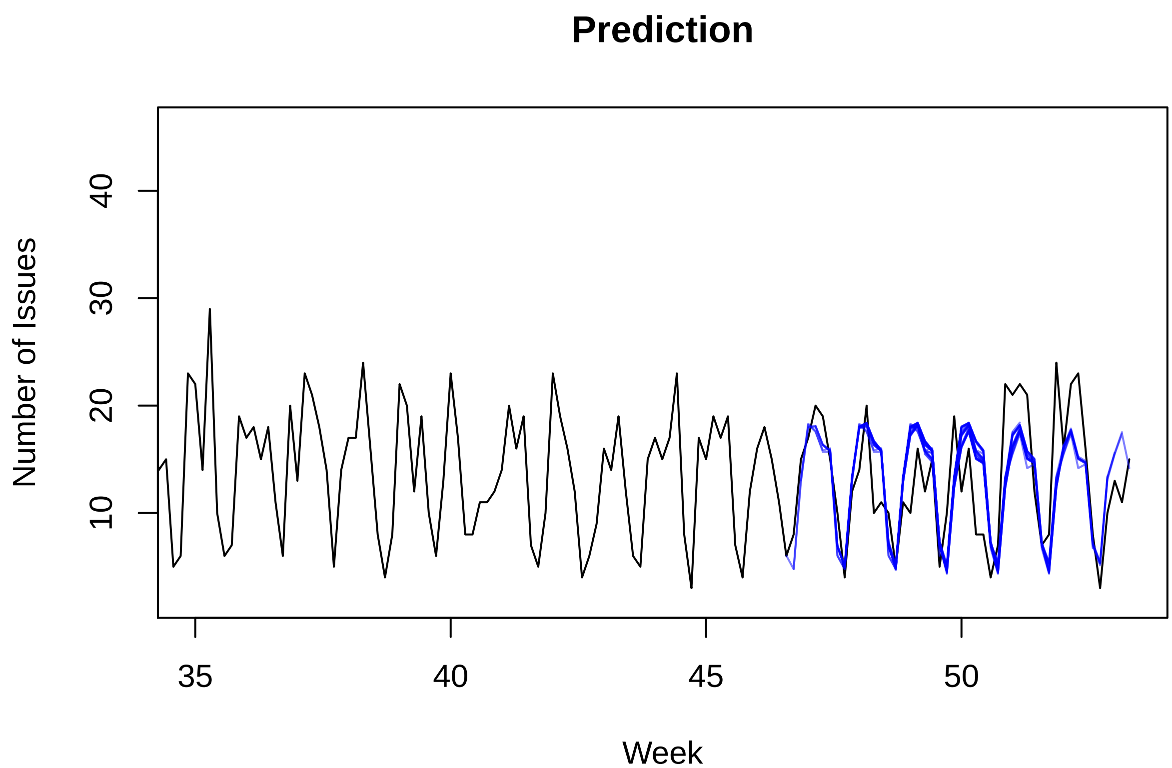
Smoothing

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))
```

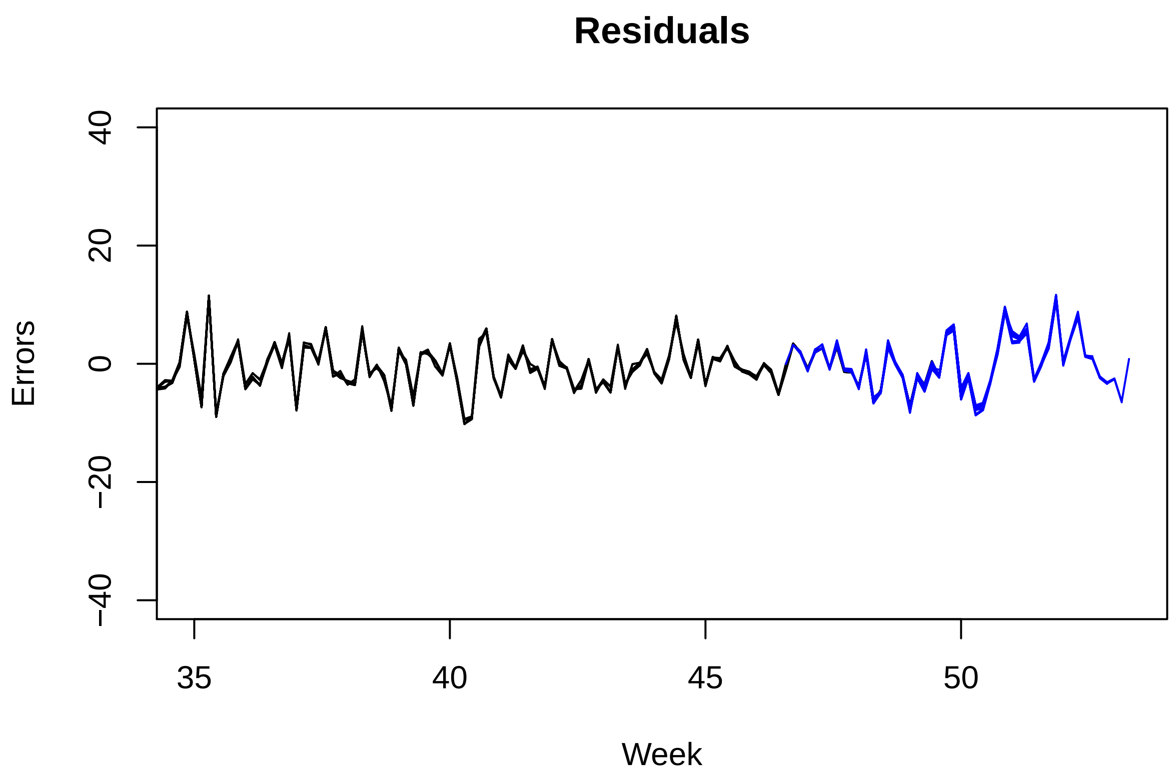
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.1901384	5.387917	4.055445	-13.97911	31.23014	0.7971078	0.1820010	NA
Test set	-0.1853370	4.641267	3.883833	-11.21427	34.61841	0.7642539	0.2835769	0.6782908

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



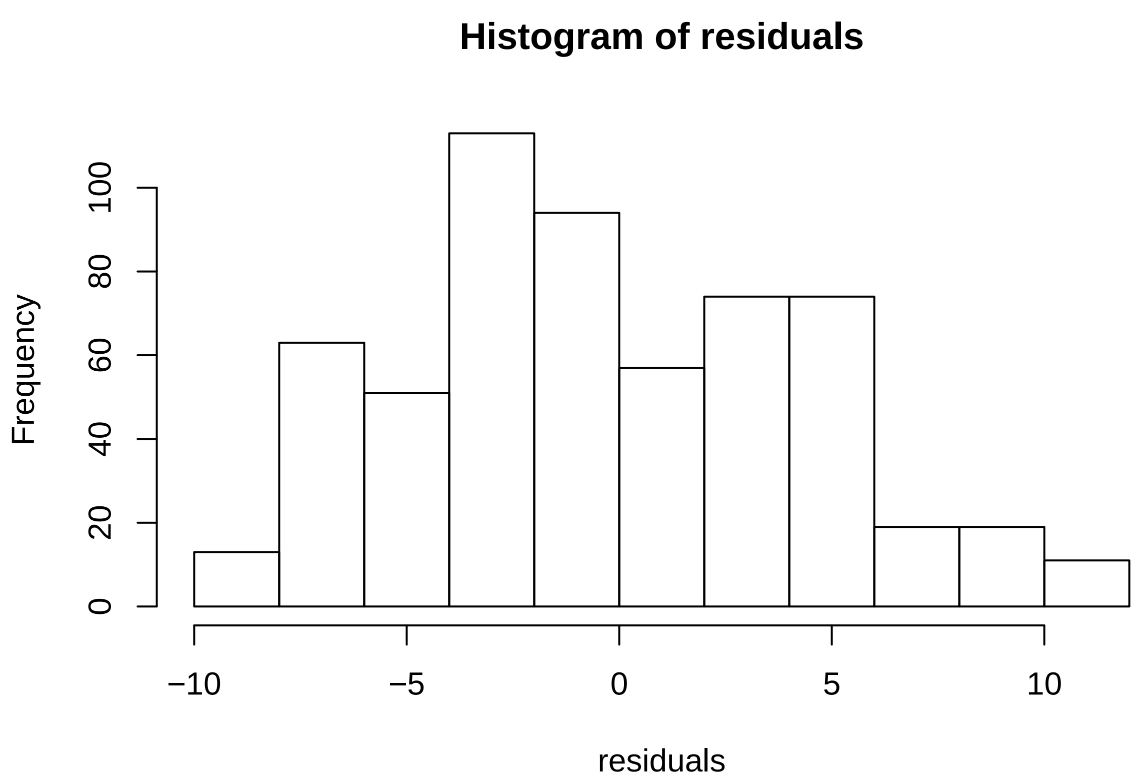

```
## NULL
```

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



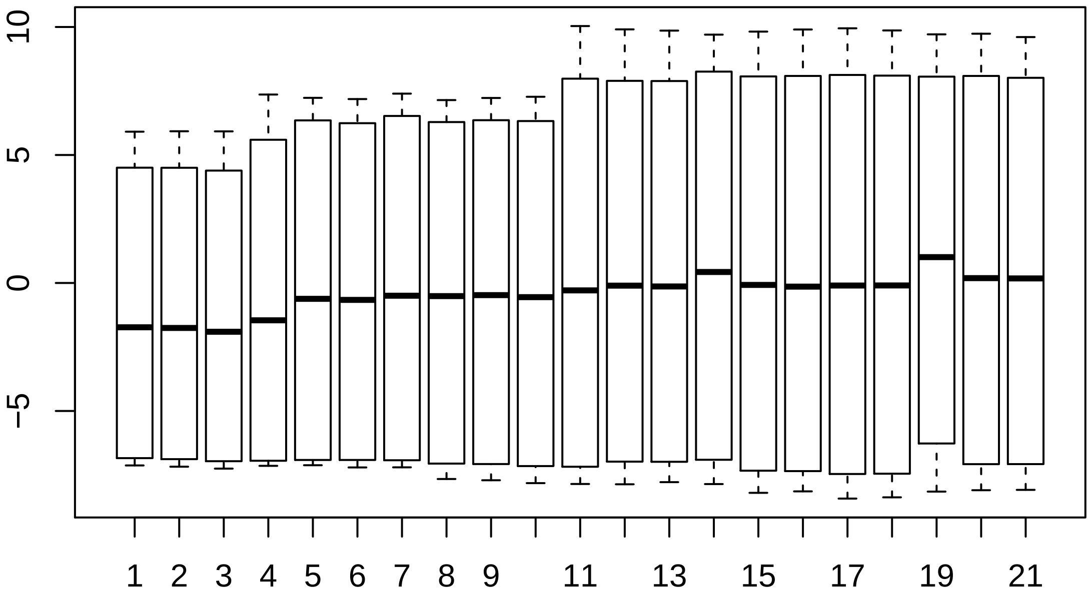

```
## NULL
```

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



```
##     97.5%       90%       10%      2.5%
##  9.489564  5.775199 -6.655933 -7.954244
```

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




```
##     97.5%       90%       10%      2.5%
##  9.489564  5.775199 -6.655933 -7.954244
```

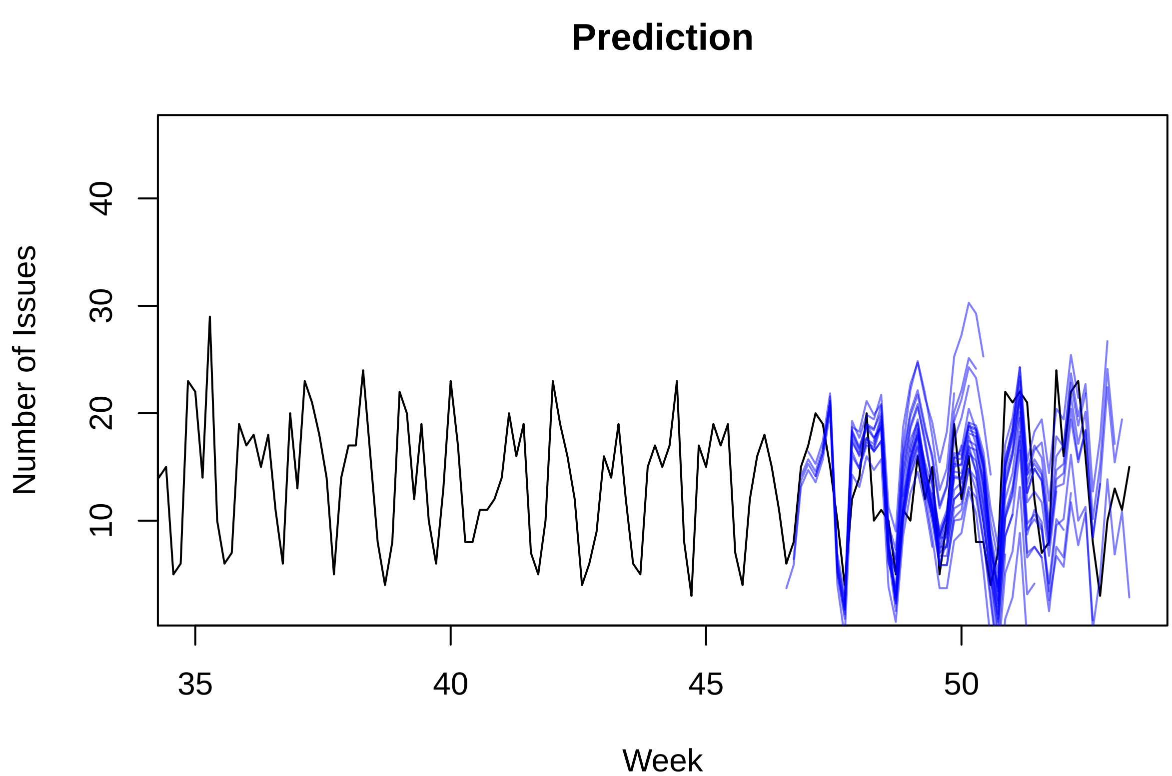
Double differencing

```
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)
```

```r
  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))
```

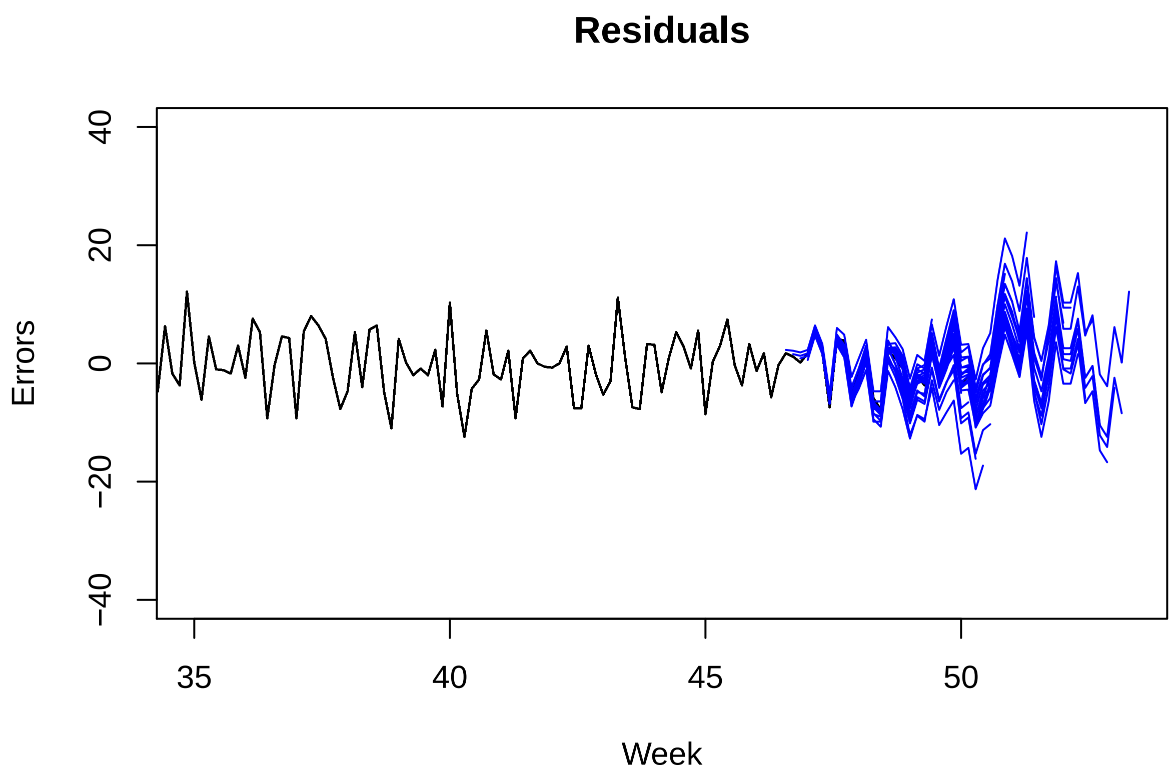
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.0841494	6.872223	5.247276	-11.90508	39.75925	1.0363981	0.1339662	NA
Test set	-0.2964043	6.094453	4.942663	-14.05875	46.89294	0.9769633	0.2970542	0.9561096

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



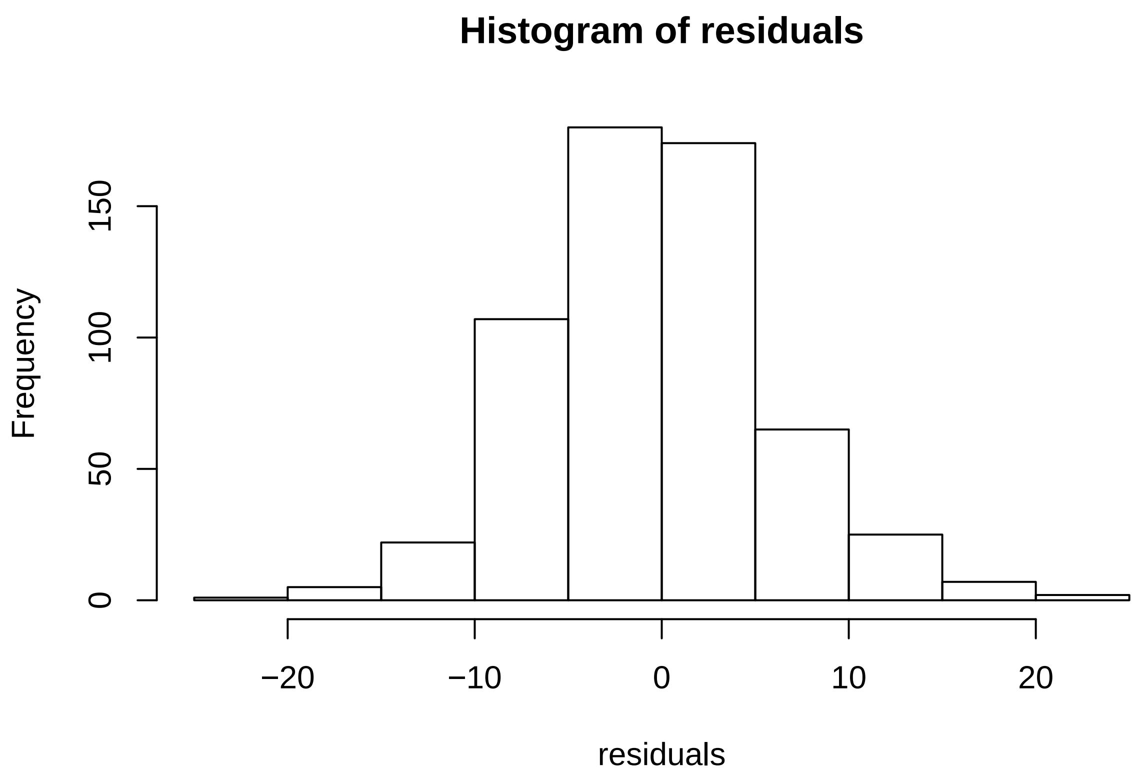

```
## NULL
```

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



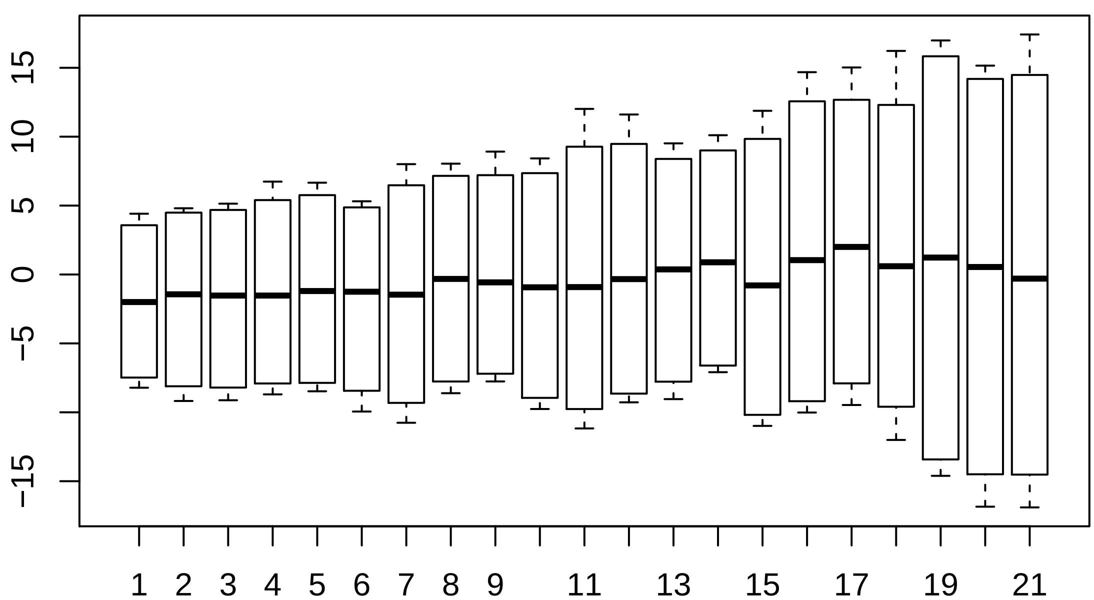

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##  13.567857   7.471429  -7.857143 -10.996429
```

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




```
##      97.5%        90%        10%       2.5%
##  13.567857   7.471429  -7.857143 -10.996429
```

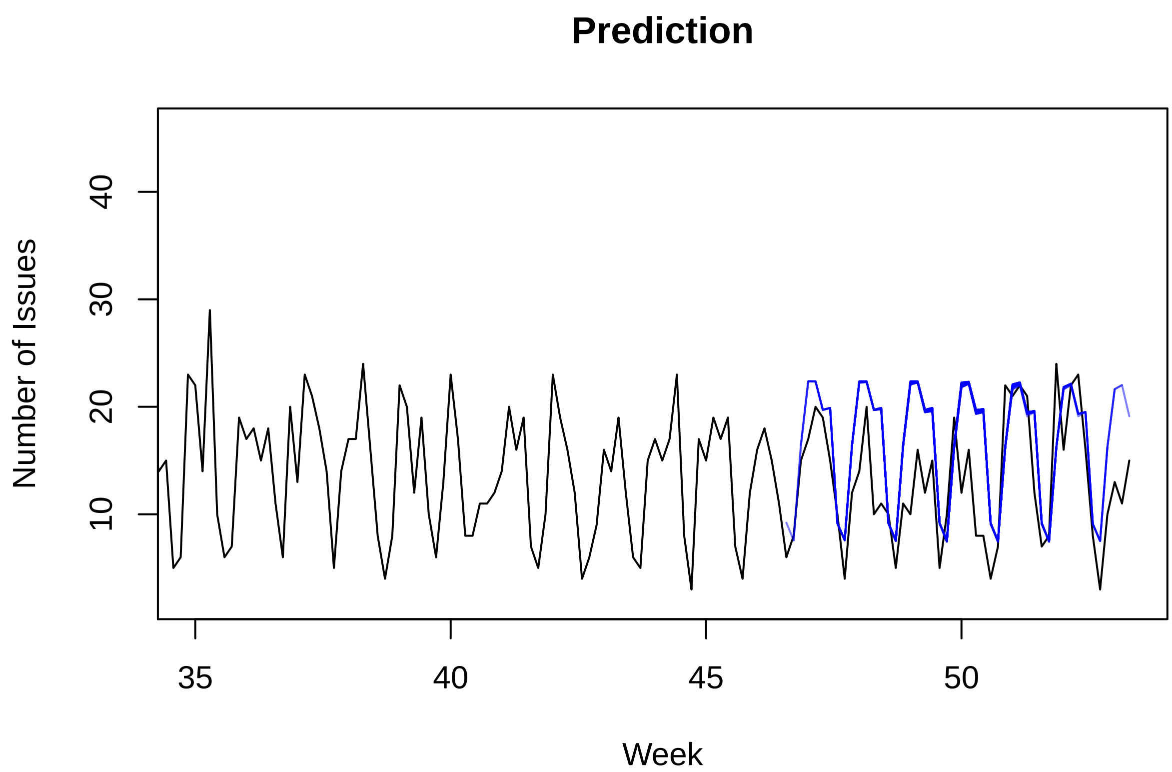
Regression

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))
```

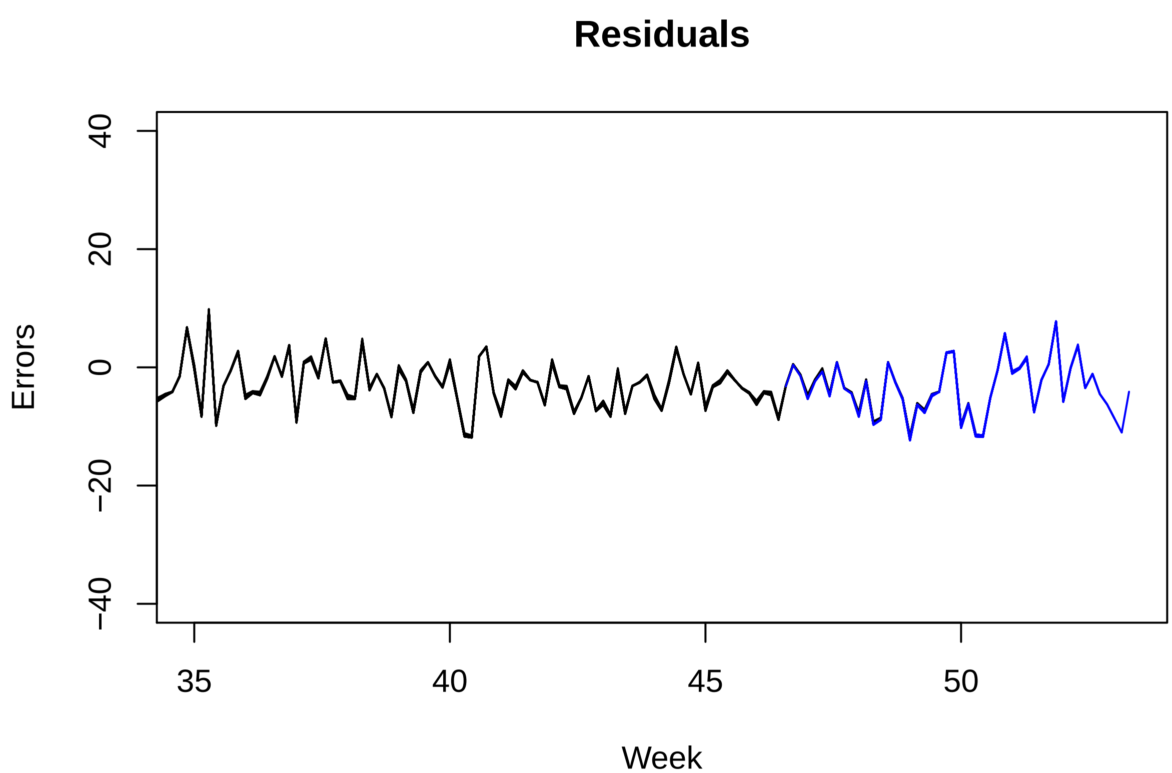
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.000000	6.033309	4.656781	-15.95276	35.38327	0.915350	0.3965010	NA
Test set	-3.815179	6.189310	5.097401	-43.81127	51.18027	1.001526	0.2687972	0.8385695

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



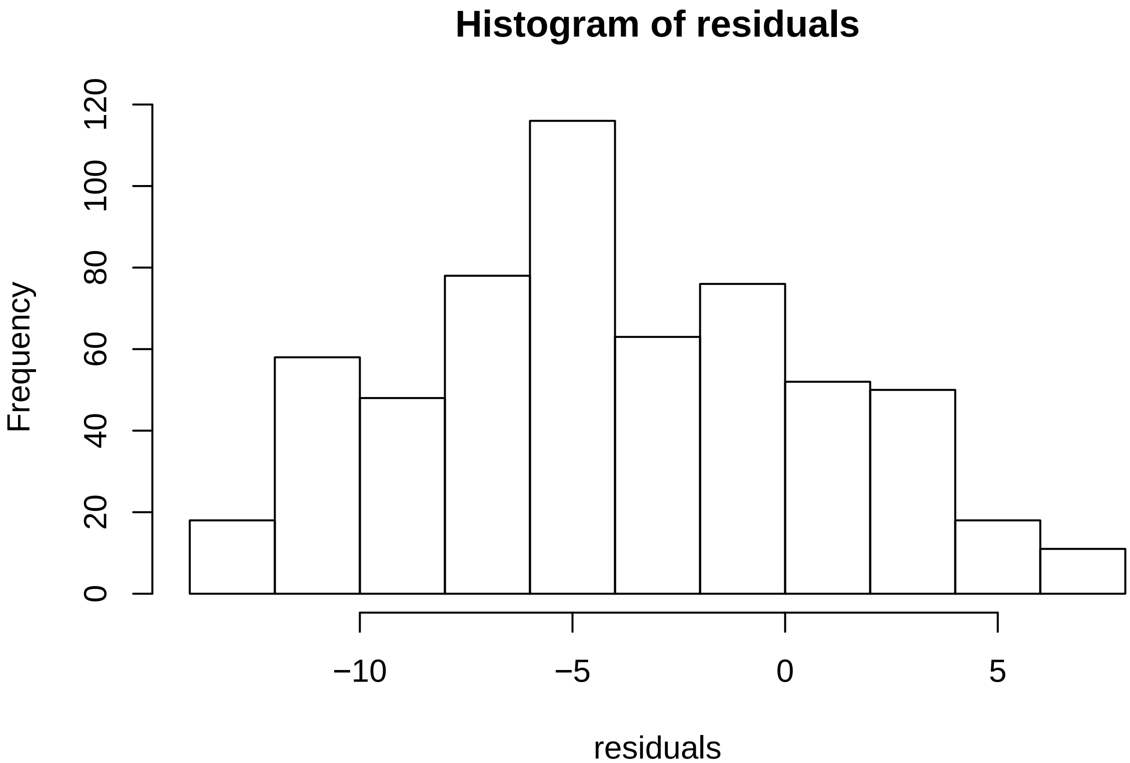

```
## NULL
```

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



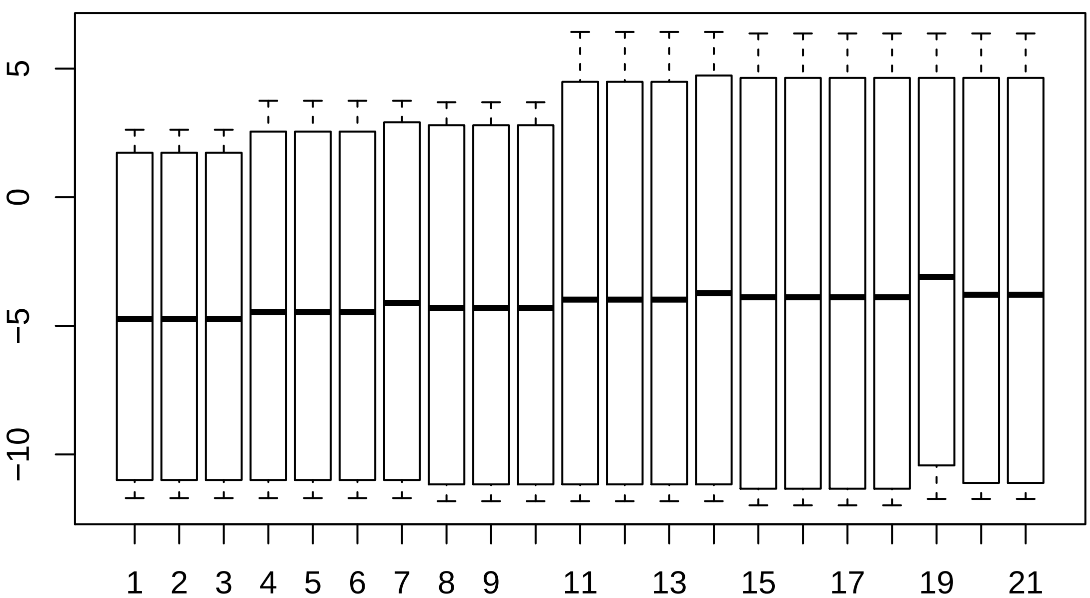

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##   5.791667   2.541667 -11.346939 -12.083333
```

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



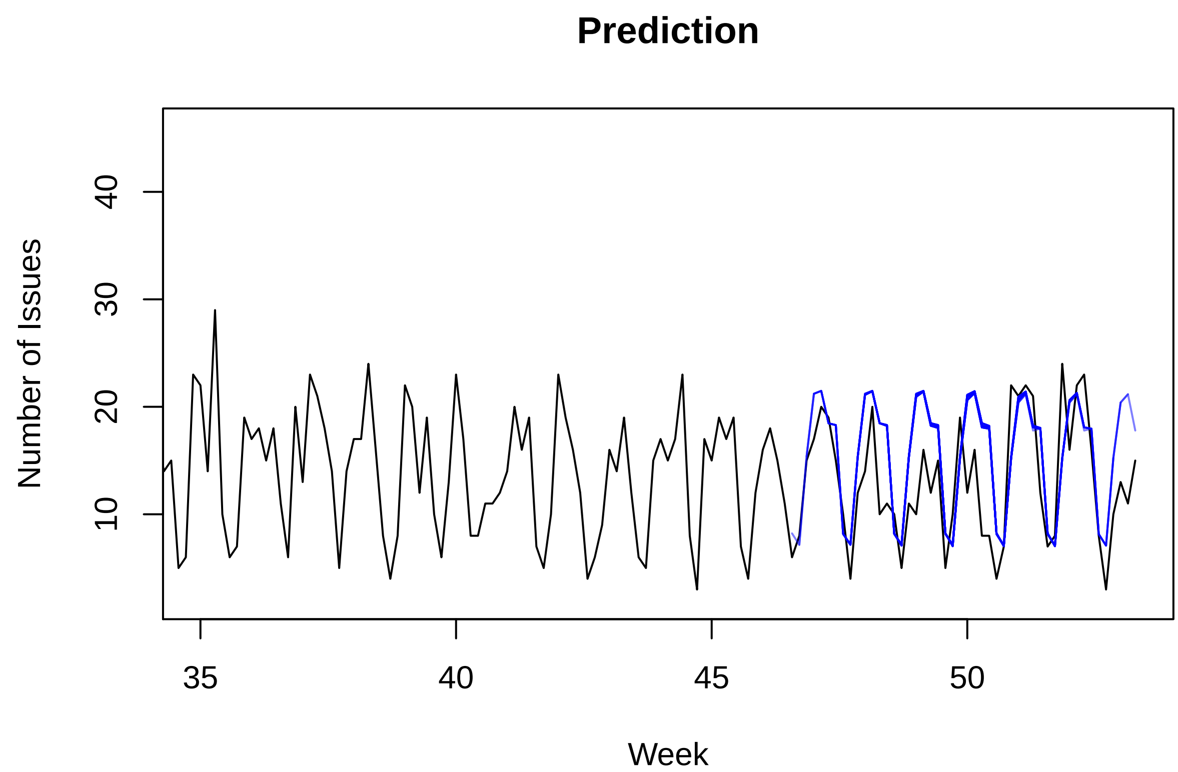

```
##      97.5%        90%        10%       2.5%
##   5.791667   2.541667 -11.346939 -12.083333
```

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))
```

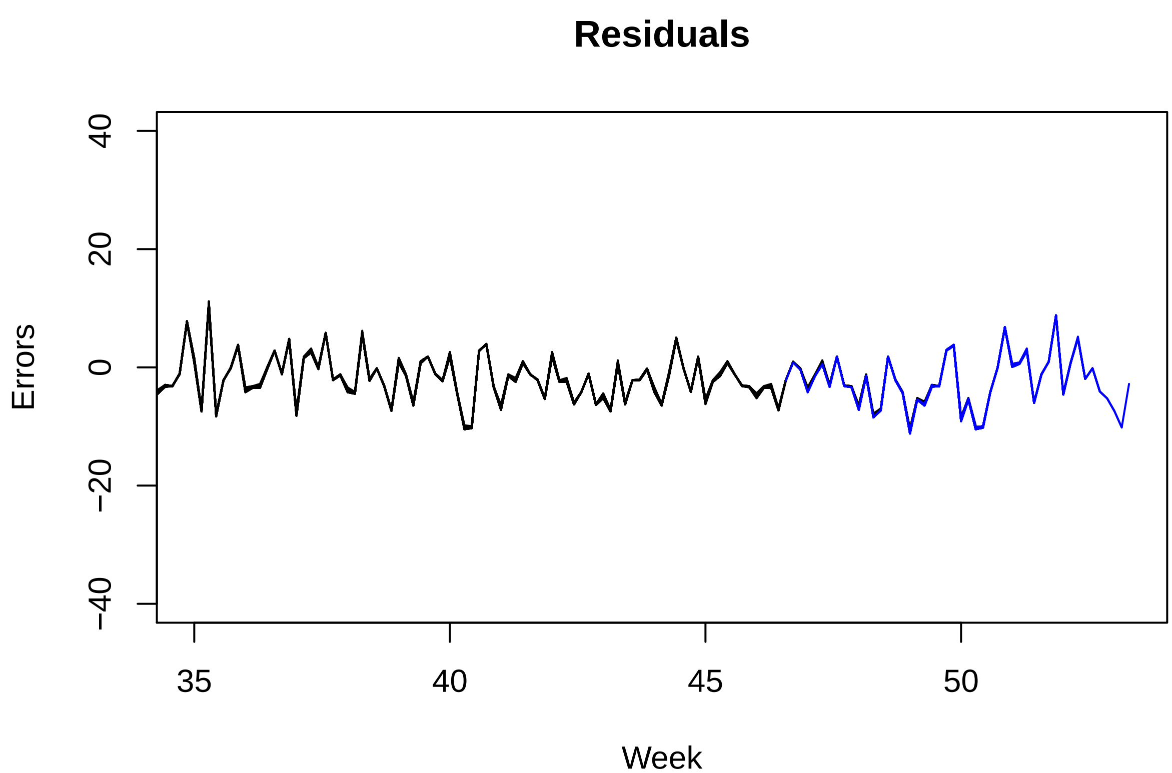
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	1.040575	6.131278	4.667556	-8.209624	32.95797	0.9174620	0.3955711	NA
Test set	-2.775293	5.498341	4.544867	-34.164788	44.54870	0.8932144	0.2655048	0.7520222

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



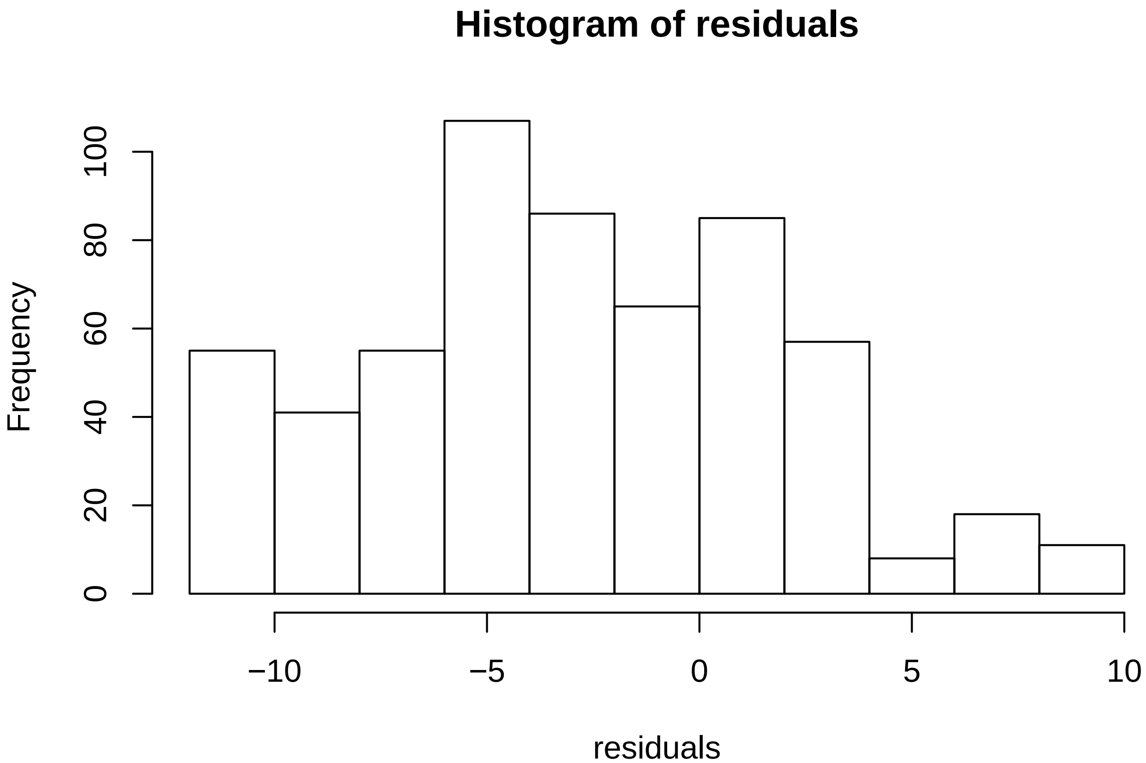

```
## NULL
```

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



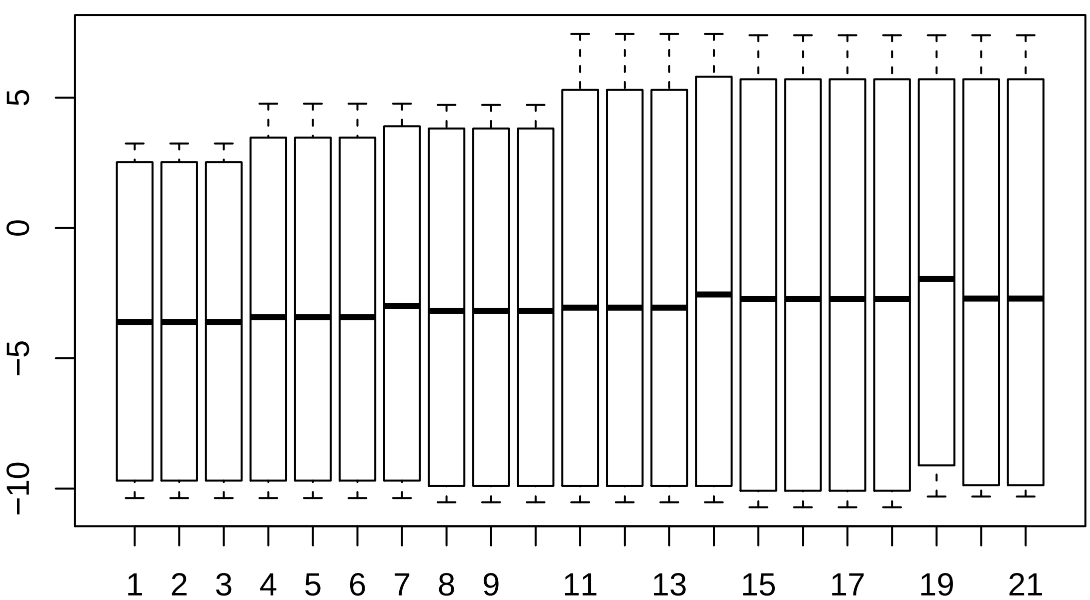

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##   6.816818   3.038201  -9.963907 -10.934907
```

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



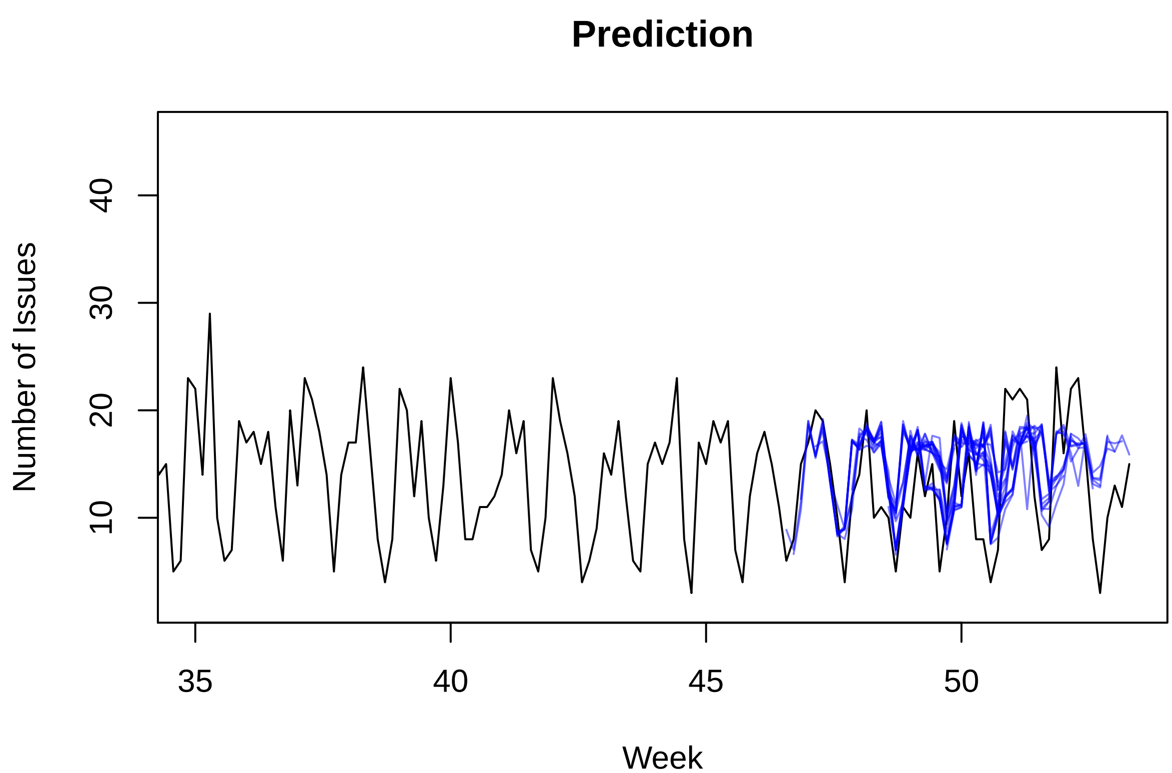

```
##      97.5%        90%        10%       2.5%
##   6.816818   3.038201  -9.963907 -10.934907
```

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))
```

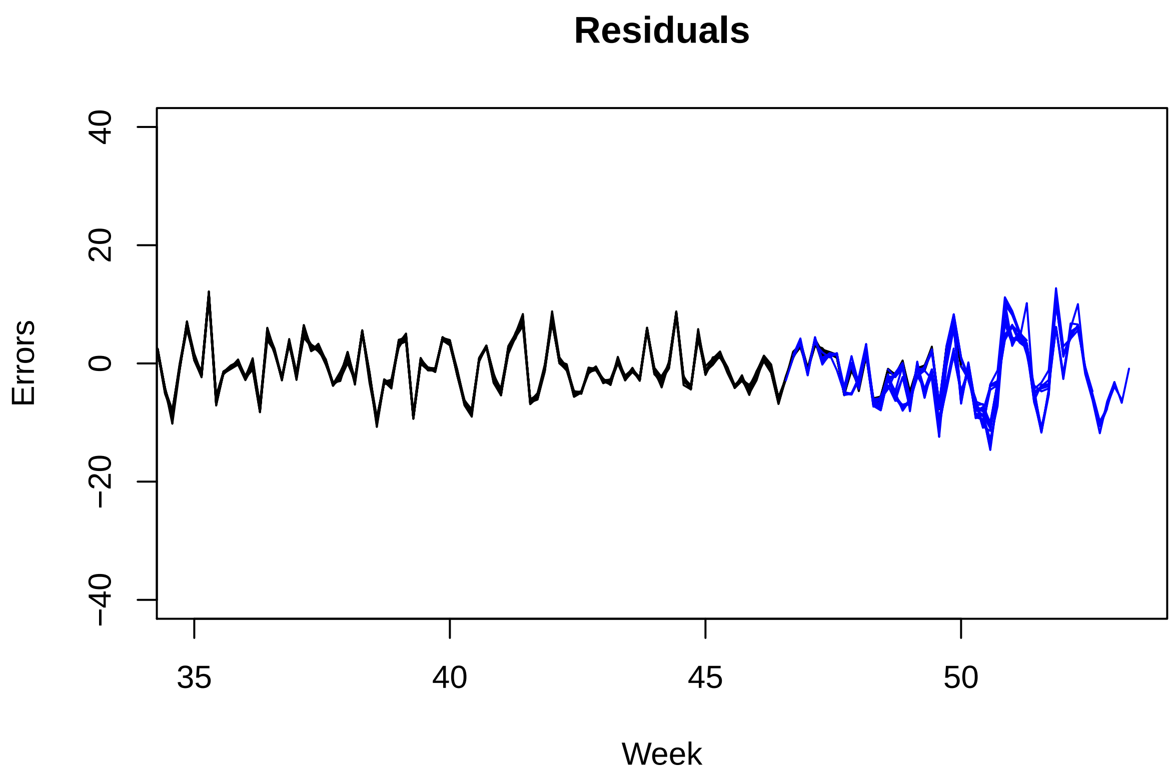
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.0016729	5.429519	4.224765	-16.08231	34.42618	0.8303827	-0.0394184	NA
Test set	-2.0080622	5.736196	4.846760	-41.02209	55.29003	0.9537827	0.2286840	0.8816352

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



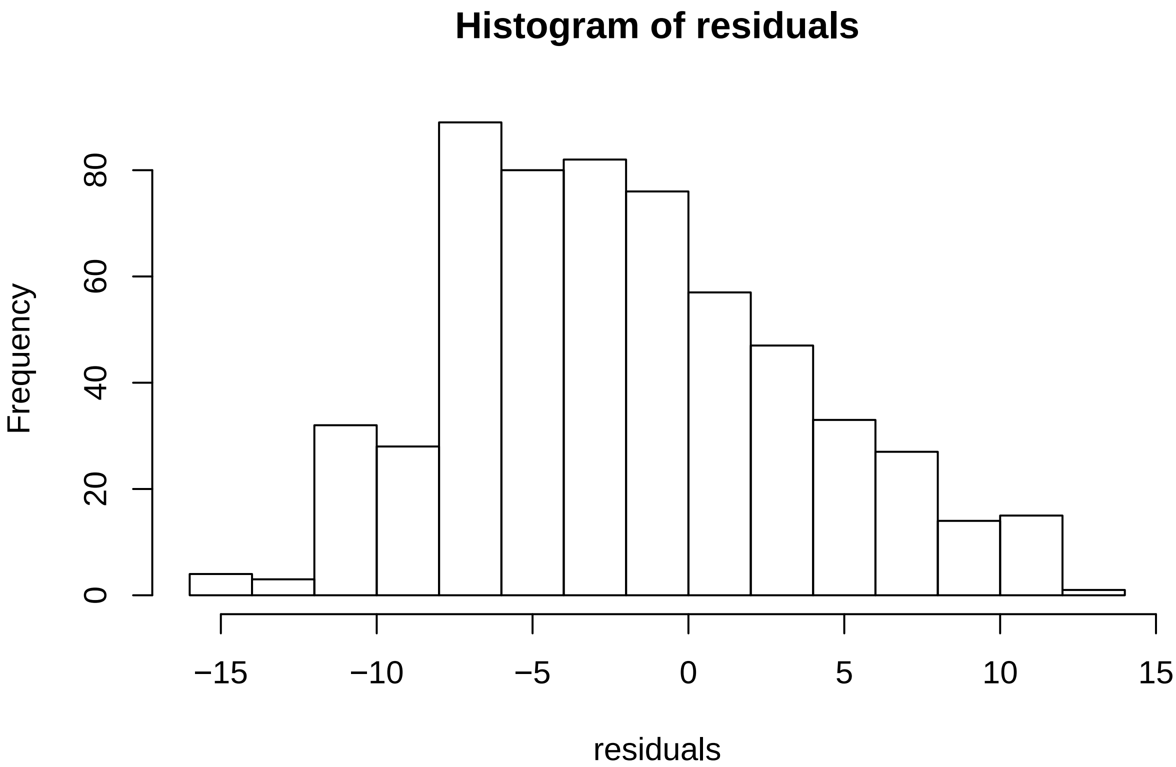

```
## NULL
```

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



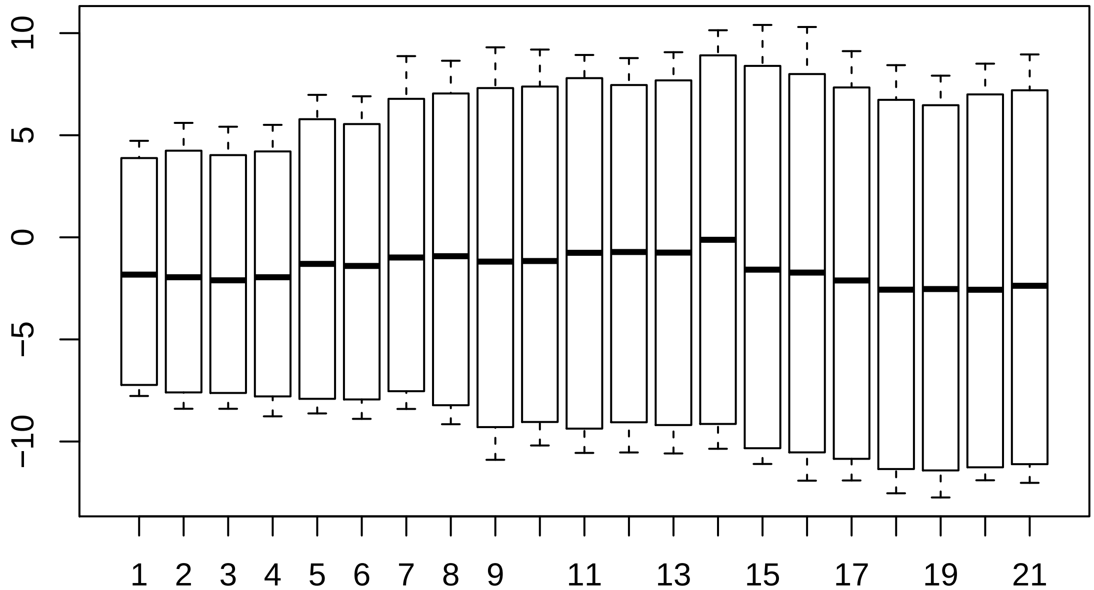

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##  10.027342   5.942721  -8.600657 -11.017805
```

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



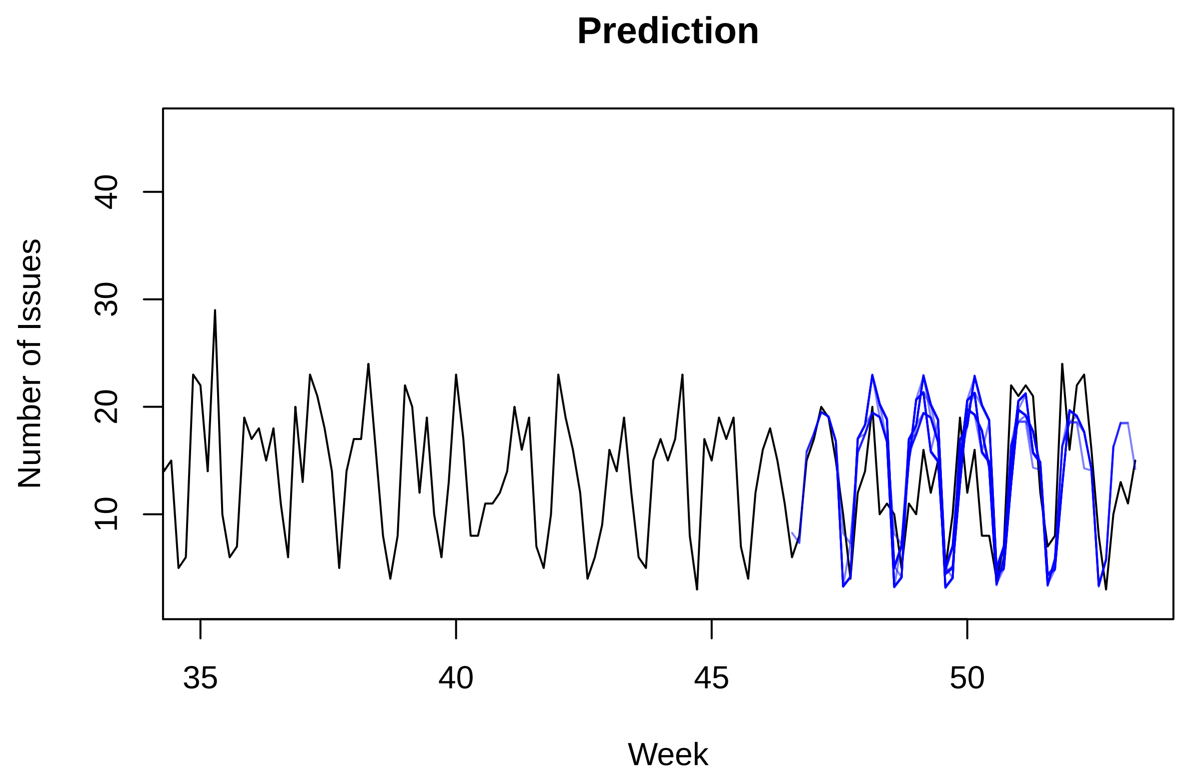

```
##      97.5%        90%        10%       2.5%
##  10.027342   5.942721  -8.600657 -11.017805
```

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, s
  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))
```

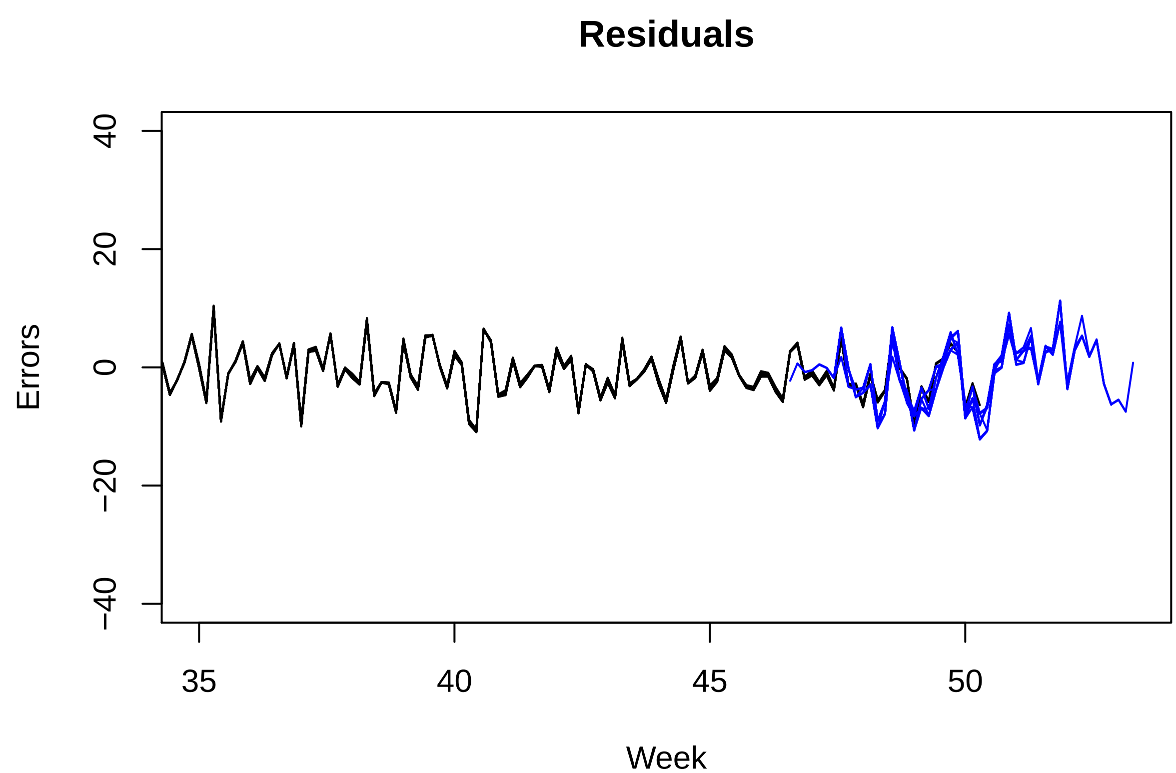
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.000000	5.273168	4.122599	-12.00874	31.68432	0.8083283	0.0840608	NA
Test set	-1.297881	5.327189	4.422941	-16.04401	39.73398	0.8673139	0.2983768	0.7872386

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



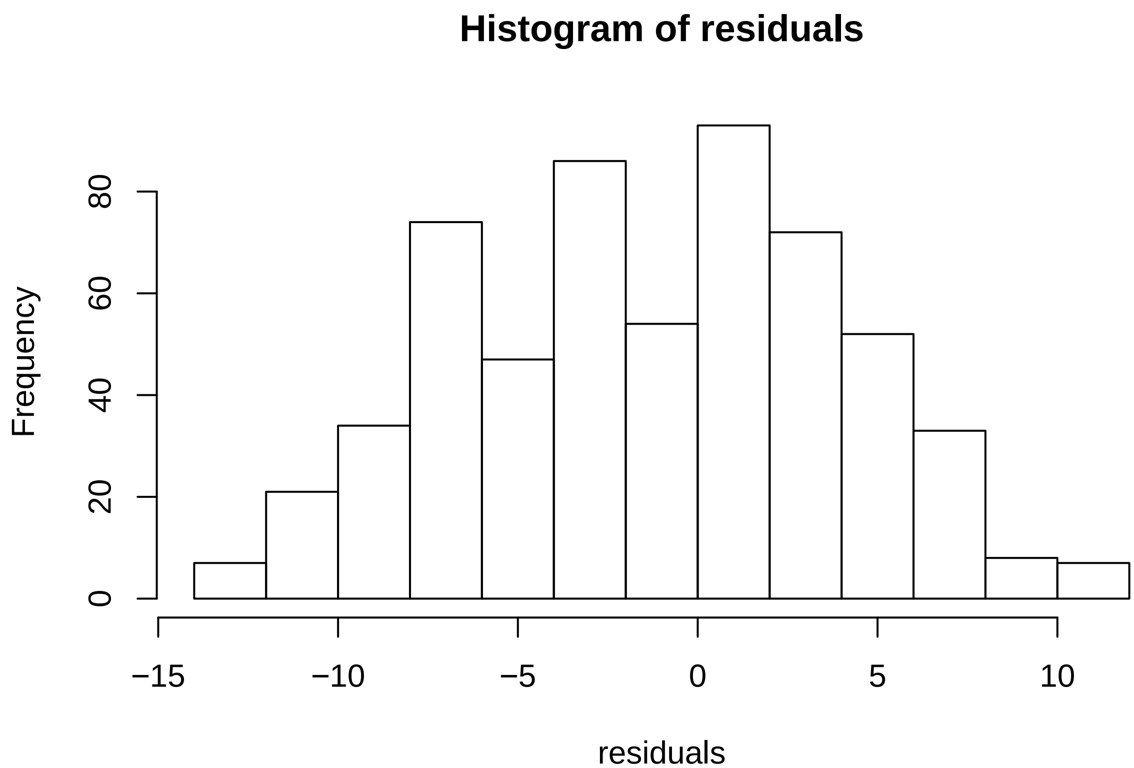

```
## NULL
```

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



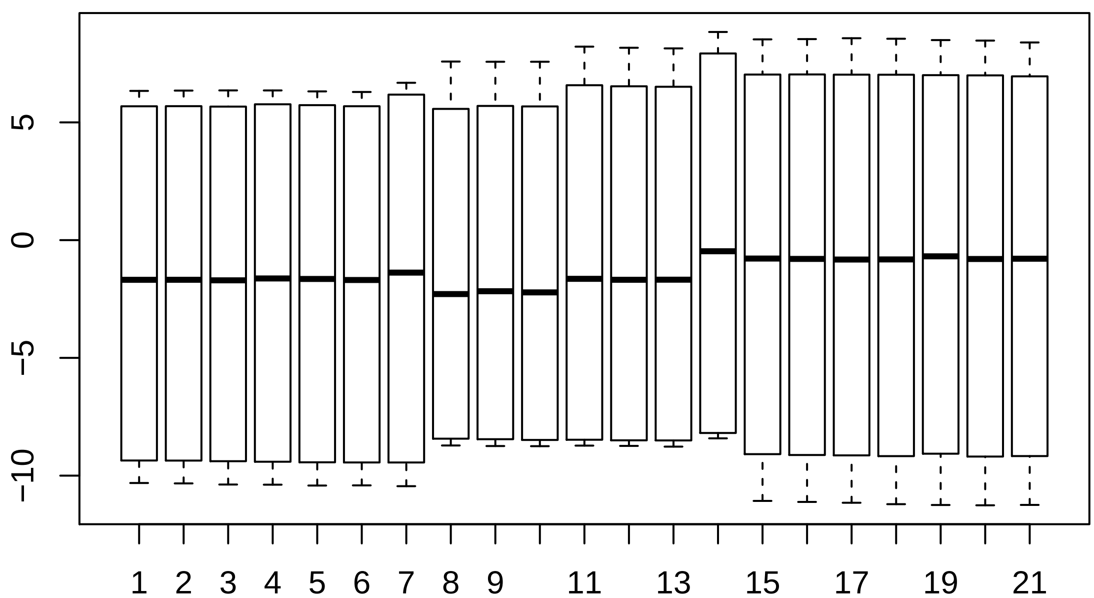

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##   8.043981   5.418443  -8.201970 -10.657138
```

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



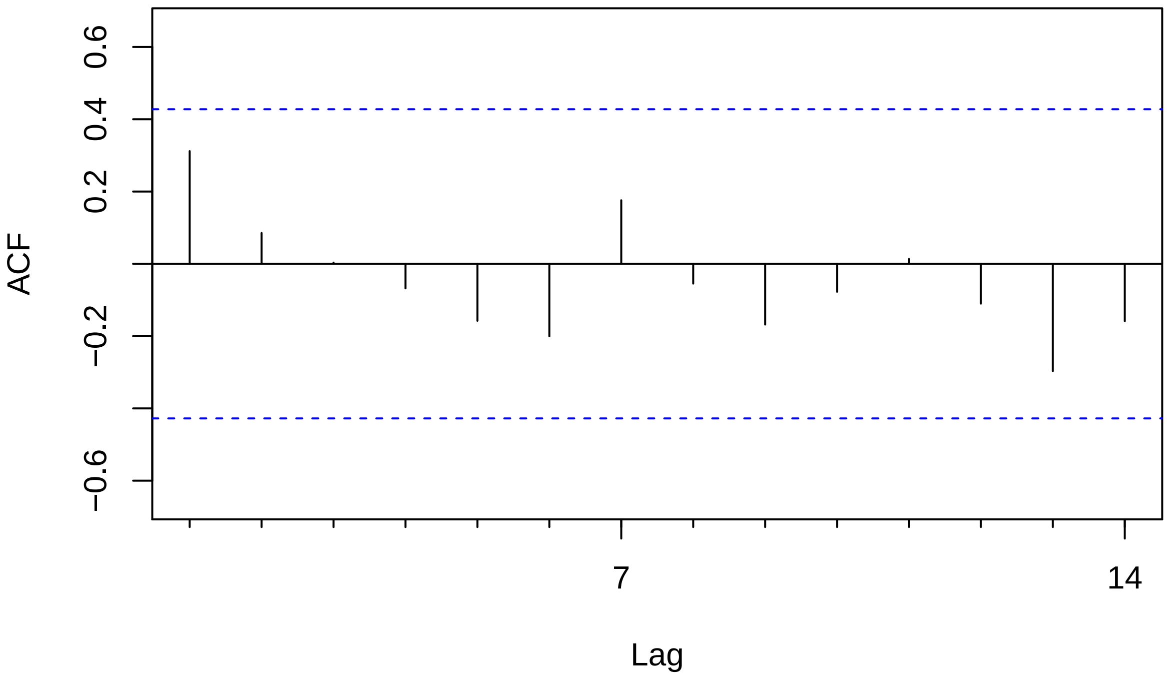

```
##      97.5%        90%        10%       2.5%
##   8.043981   5.418443  -8.201970 -10.657138
```

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

```r
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))
```

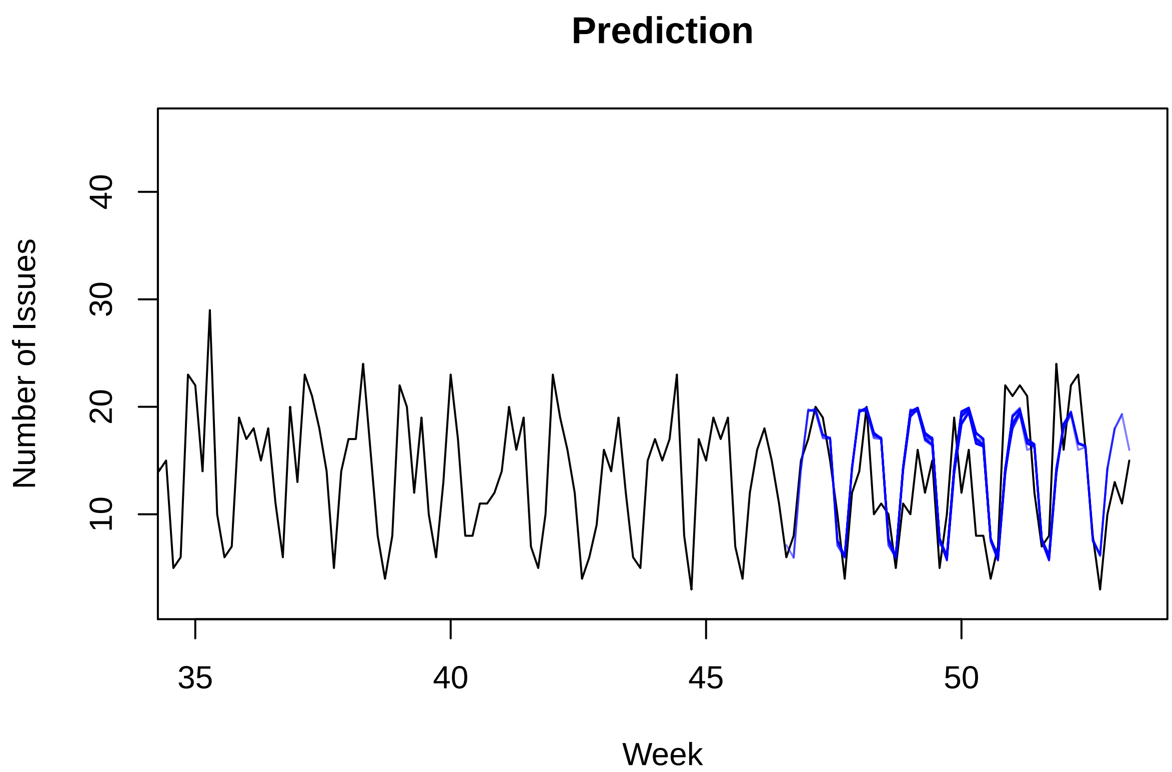
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.5424684	10.584560	8.780288	-38.99177	78.96350	1.7343539	0.3894676	NA
Test set	-1.4803149	4.909447	4.142006	-22.68953	39.11404	0.8186012	0.2756987	0.6937114

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



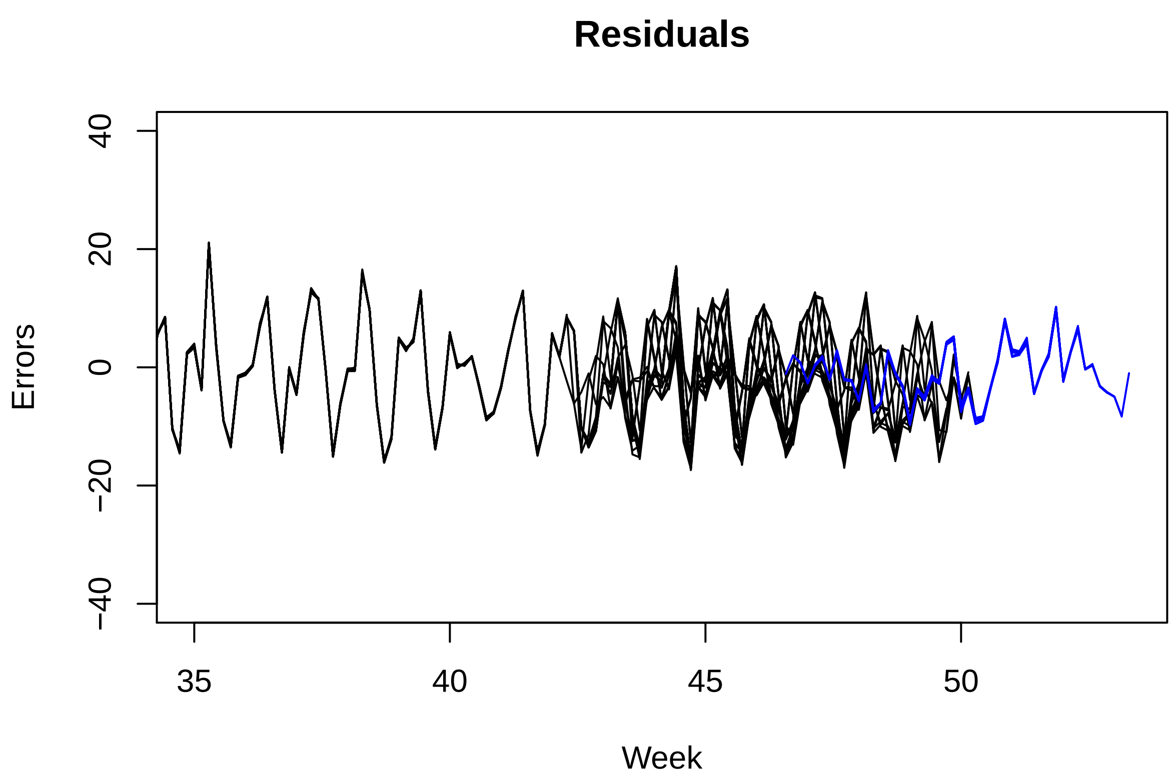

```
test <- list()
test$train.ts <- issues.ts
test$valid.ts <- naive(issues.ts, h=n.valid)$mean

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)
```



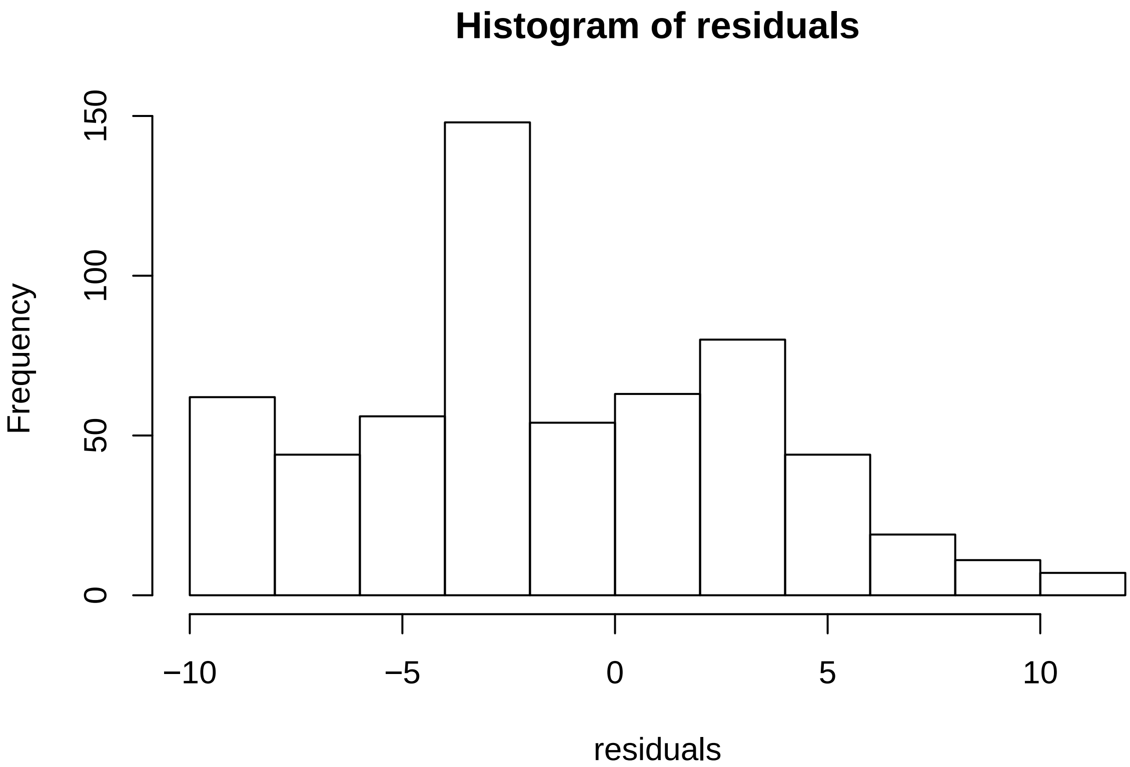

```
## NULL
```

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



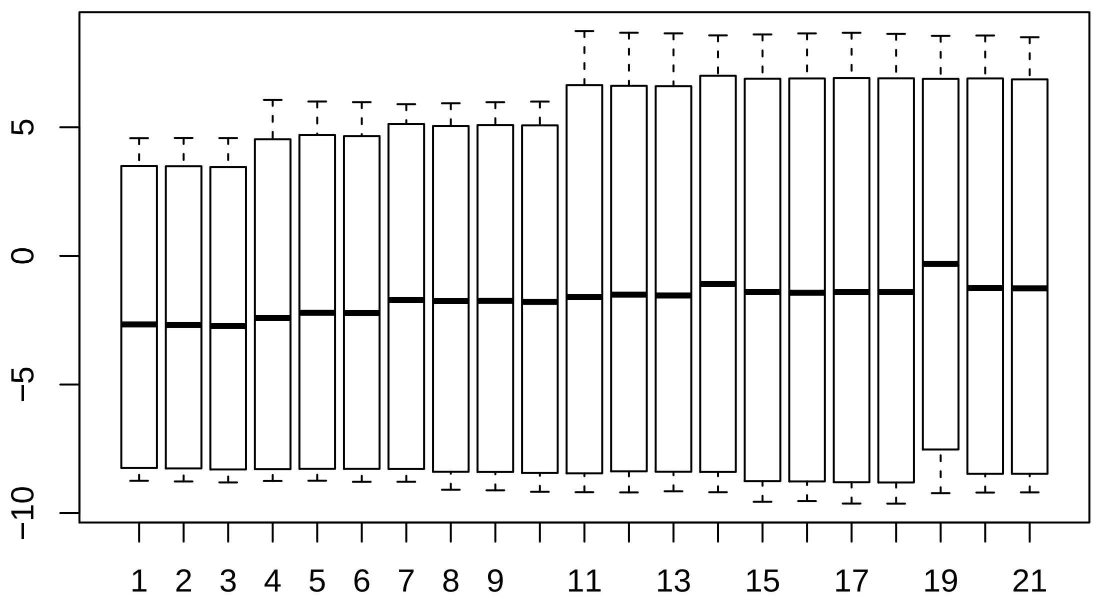

```
## NULL
```

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




```
##      97.5%       90%       10%      2.5%
##   8.153191  4.499105 -8.292231 -9.525869
```

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




```
##      97.5%       90%       10%      2.5%
##   8.153191  4.499105 -8.292231 -9.525869
```

```r
# plot the prediction on test period with the prediction interval
plot(forecast.object, xlim=c(35, 56))
```