# 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.

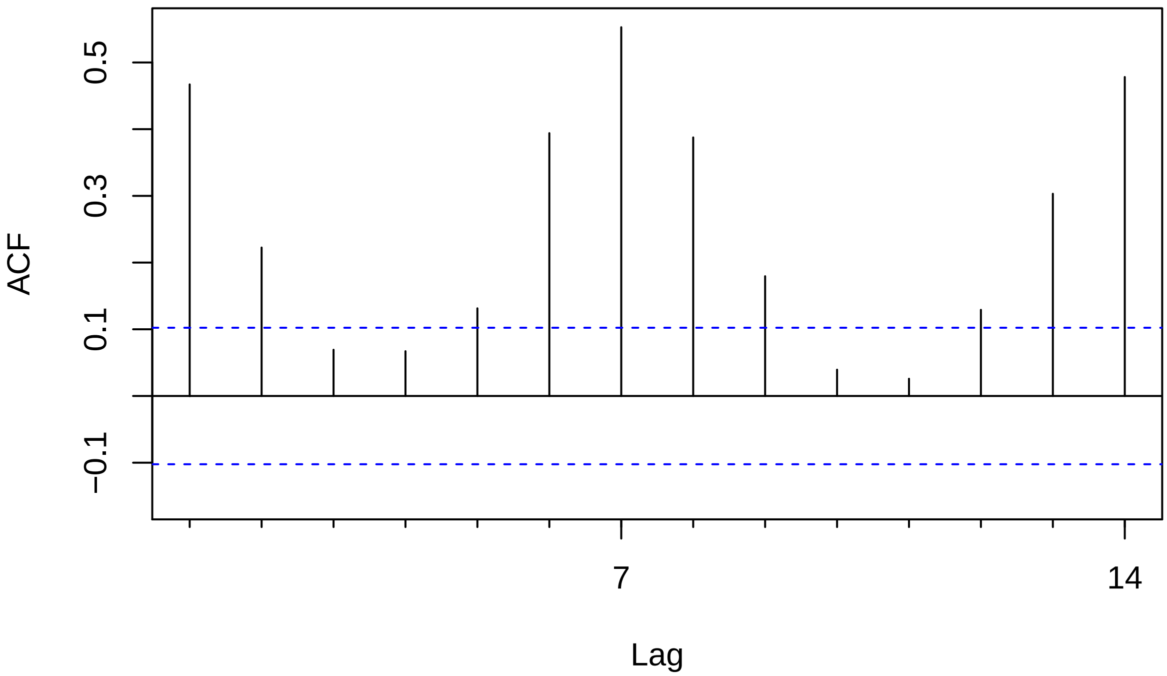
## Load the data

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

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

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

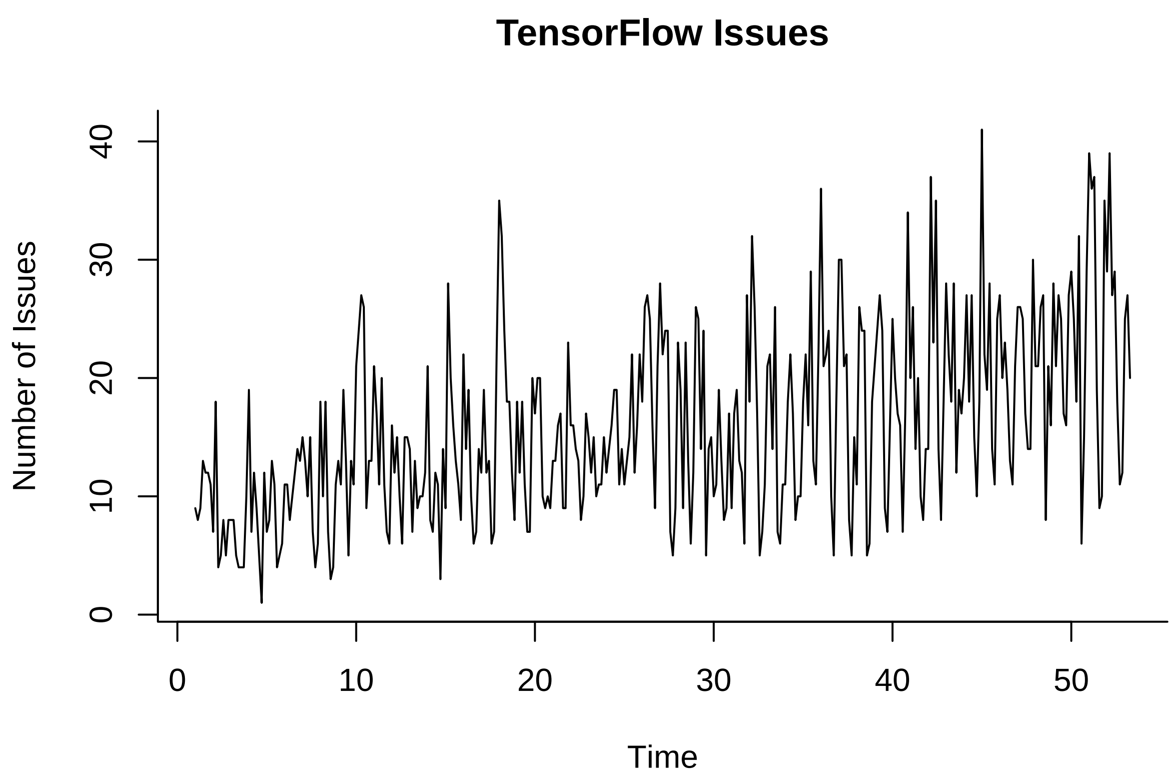
## keep the last 12 months

```
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)
```

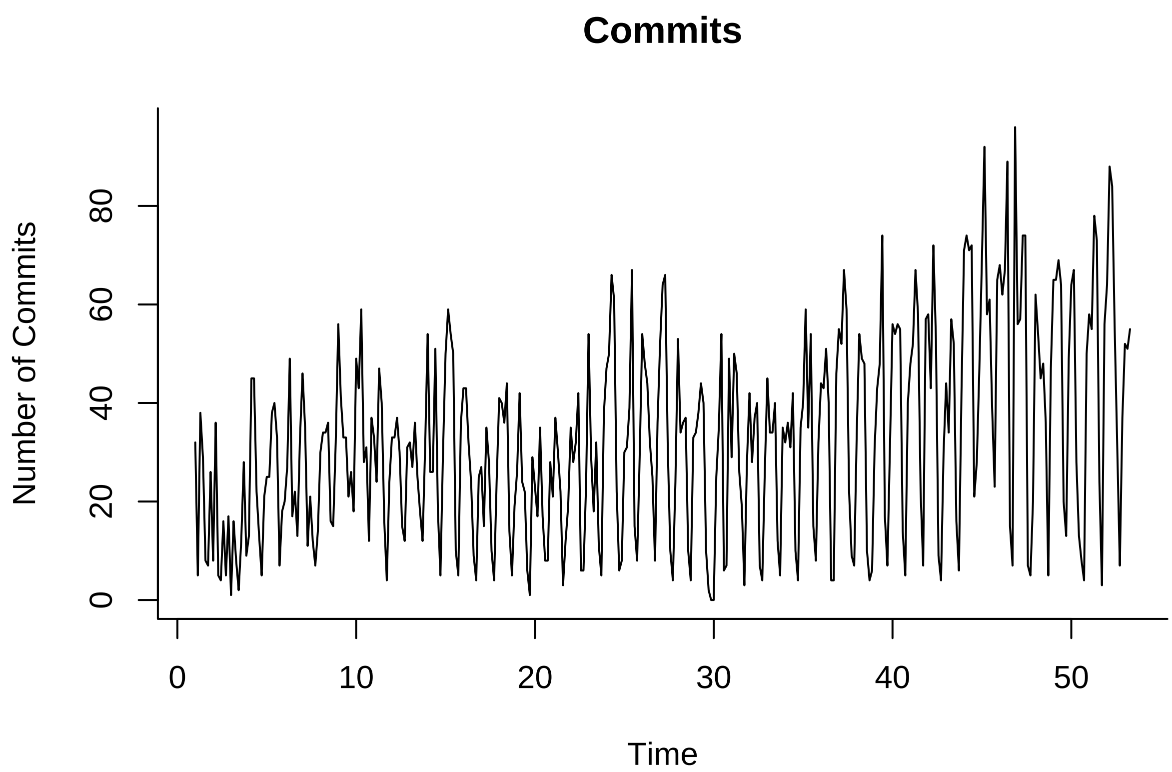
```
#loading issues and commits into a ts object
issues.ts <- ts(repository.csv$number_of_issues, frequency = 7)


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 = 'TensorFlow 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',])
}
```

```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'))
}
```

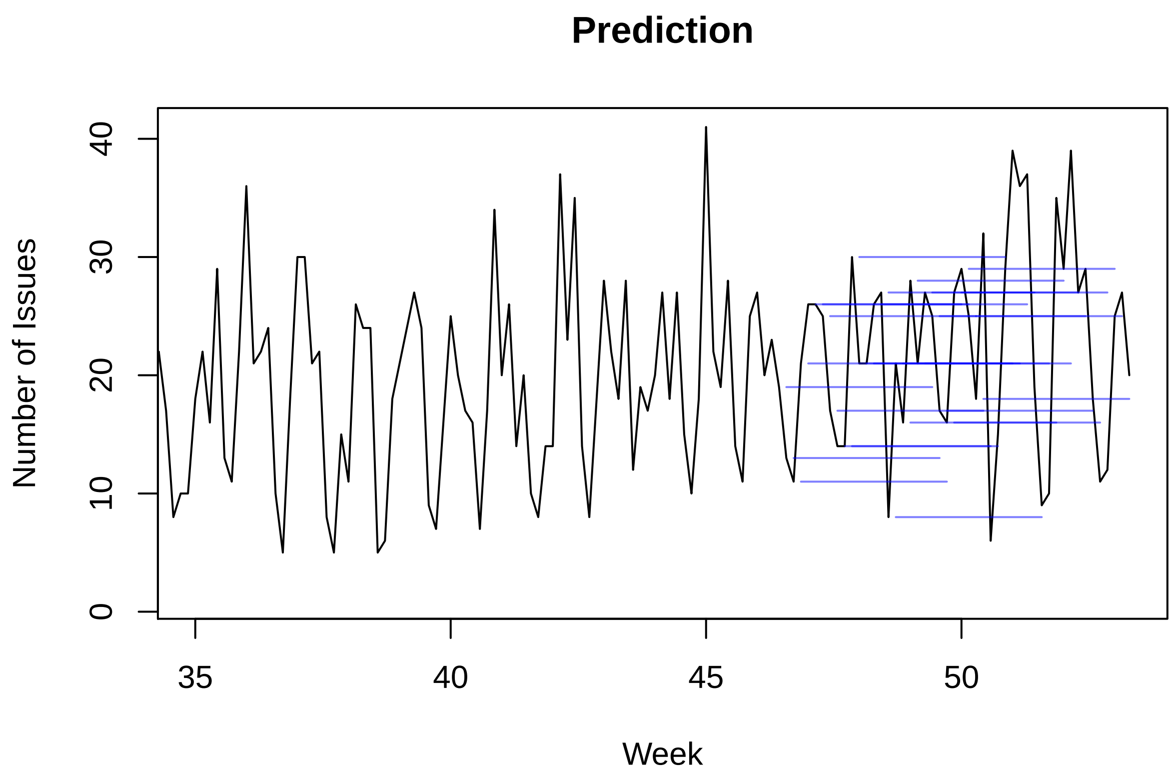
## Naive Forecast

### 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))
```

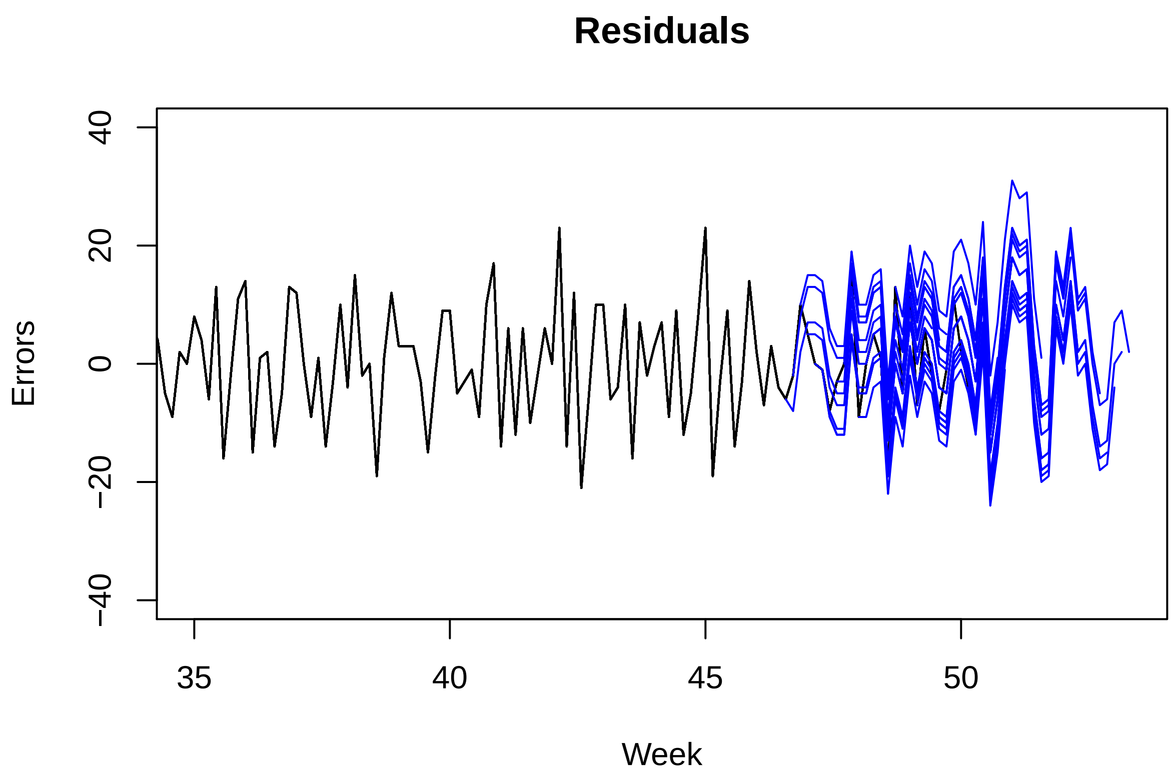
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 0.0362191 | 7.740198 | 5.984758 | -16.92127 | 48.38444 | 1.089436 | -0.2809288 | NA |
| Test set | 1.9353741 | 10.001513 | 8.401360 | -11.99704 | 47.87292 | 1.529827 | 0.1266963 | 0.8643042 |

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



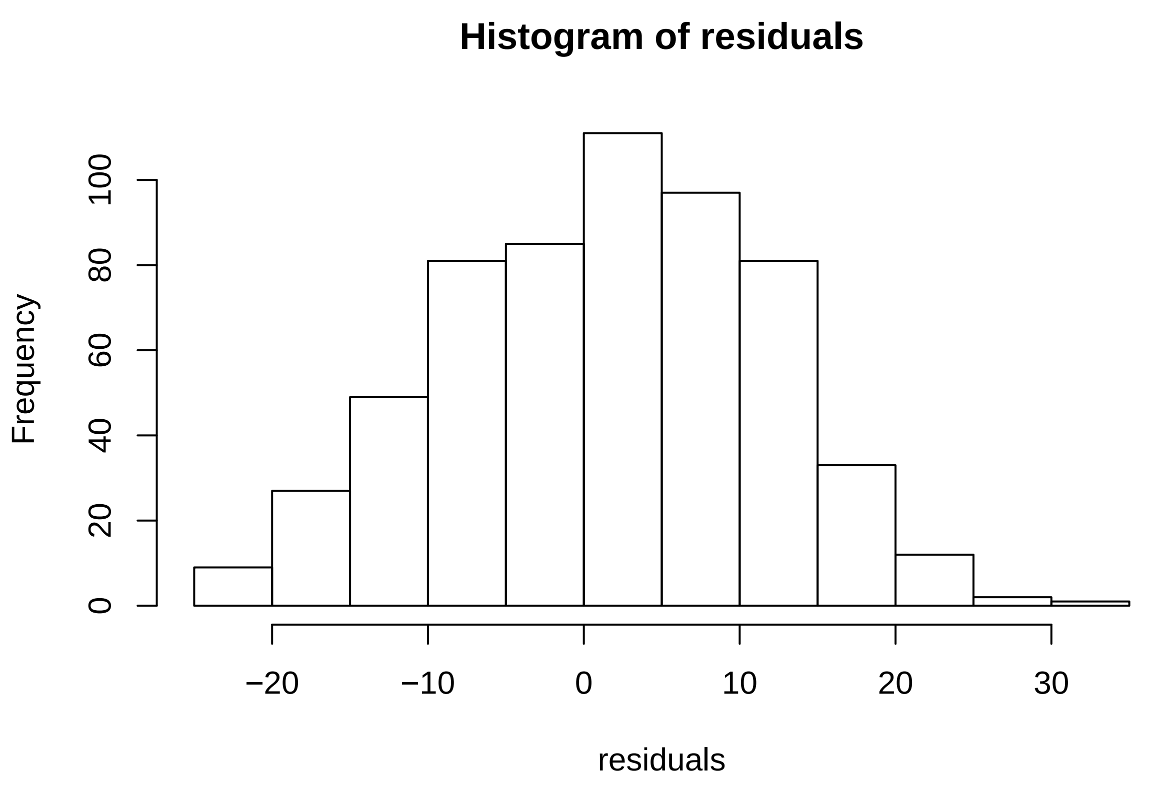

```
## NULL
```

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



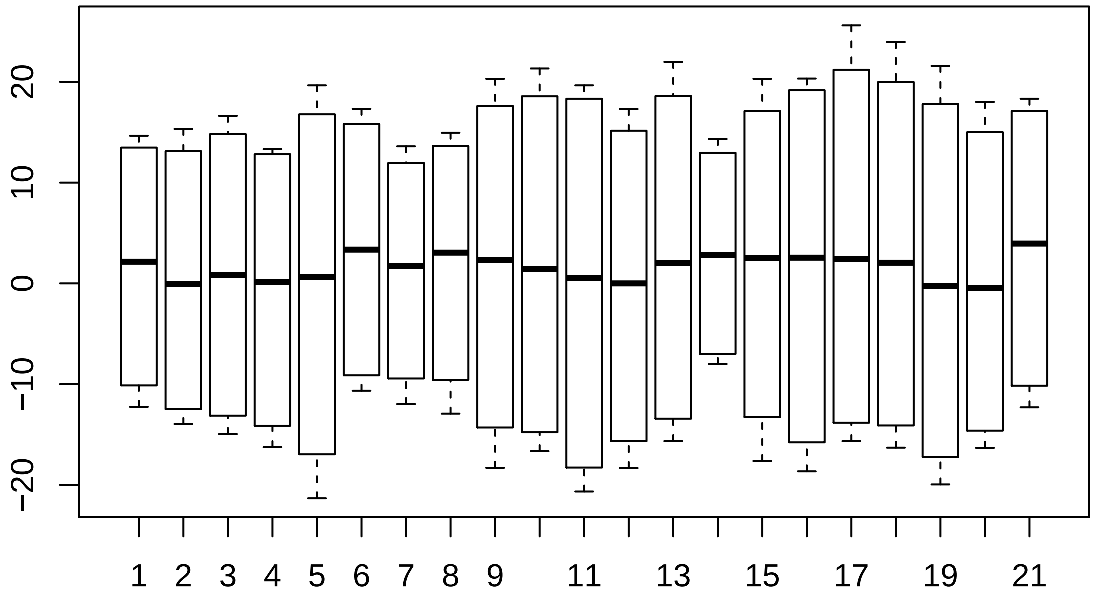

```
## NULL
```

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



```
##   97.5%     90%     10%    2.5%
##  20.325  14.300 -11.000 -18.000
```

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



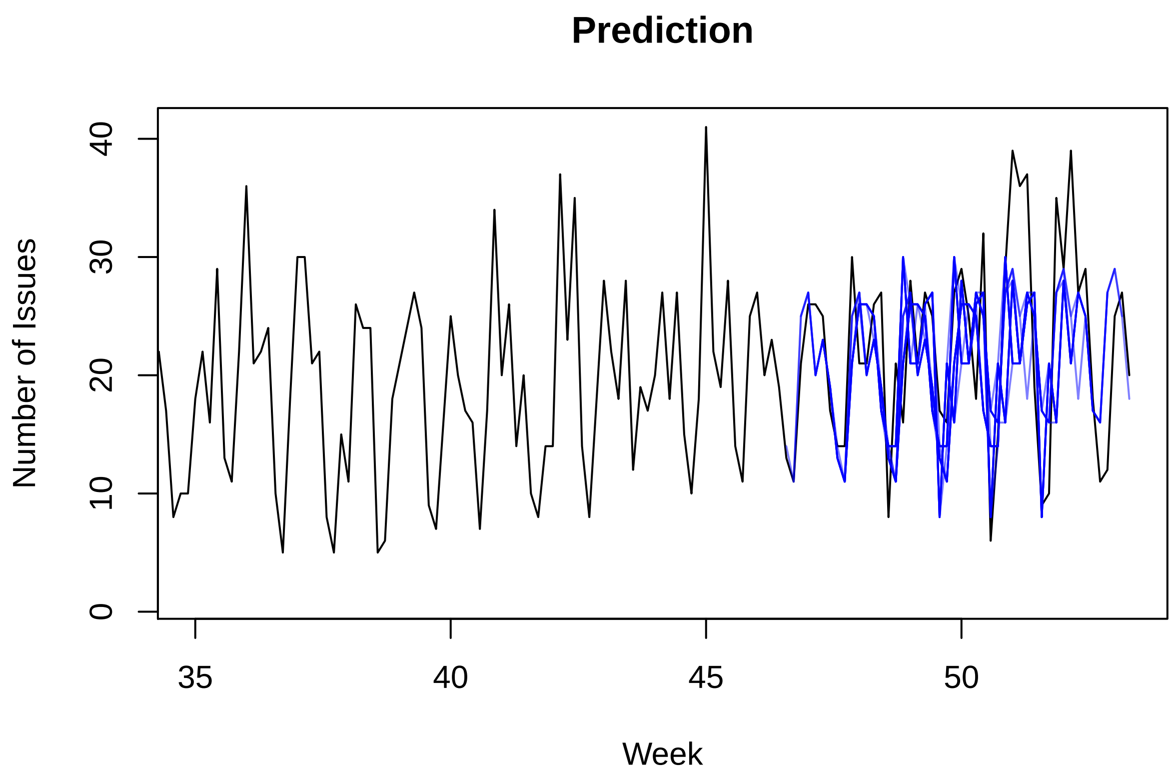
```
##   97.5%     90%     10%    2.5%
##  20.325  14.300 -11.000 -18.000
```

### 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))
```

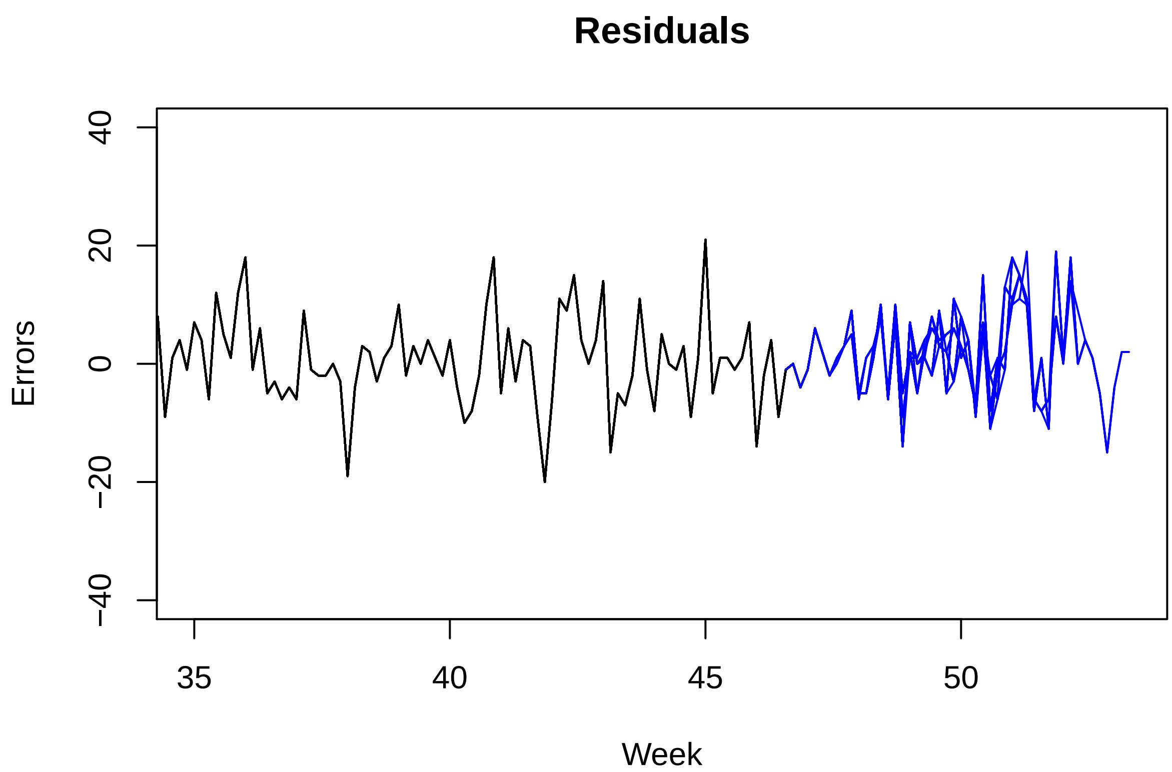
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 0.2222269 | 7.133274 | 5.493520 | -12.165041 | 43.18270 | 1.000000 | 0.1995944 | NA |
| Test set | 2.0476190 | 7.450688 | 6.088435 | -1.947782 | 30.79628 | 1.108953 | -0.1822976 | 0.6699061 |

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



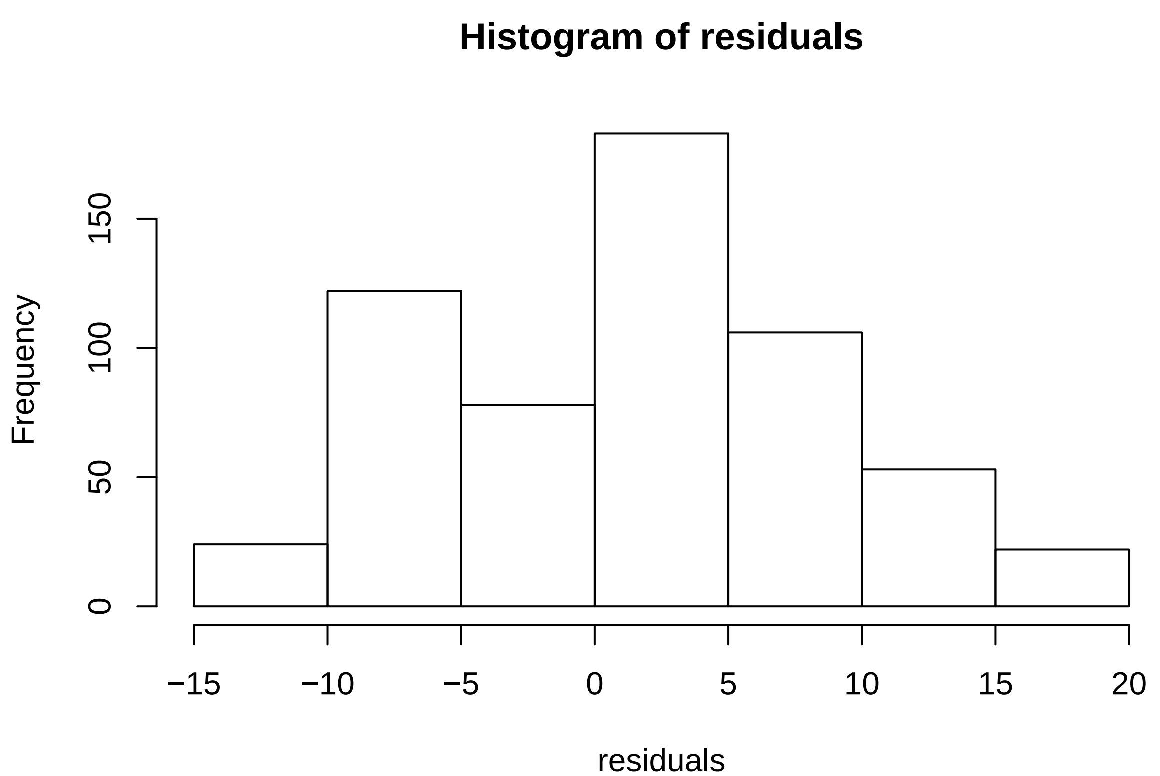

```
## NULL
```

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



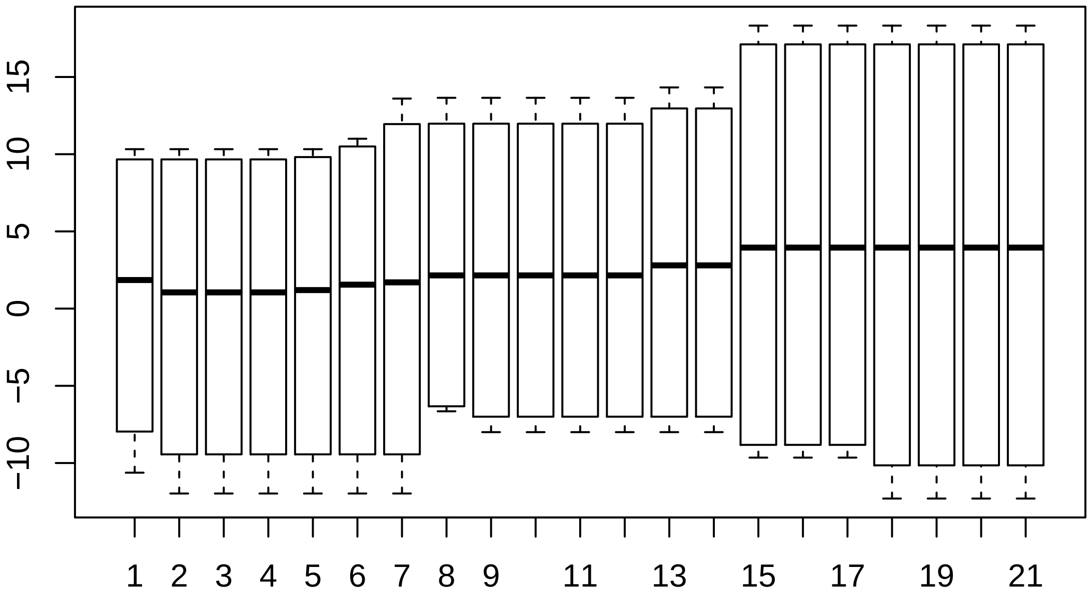

```
## NULL
```

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



```
## 97.5%    90%    10%   2.5%
##    18     11     -8    -11
```

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




```
## 97.5%    90%    10%   2.5%
##    18     11     -8    -11
```

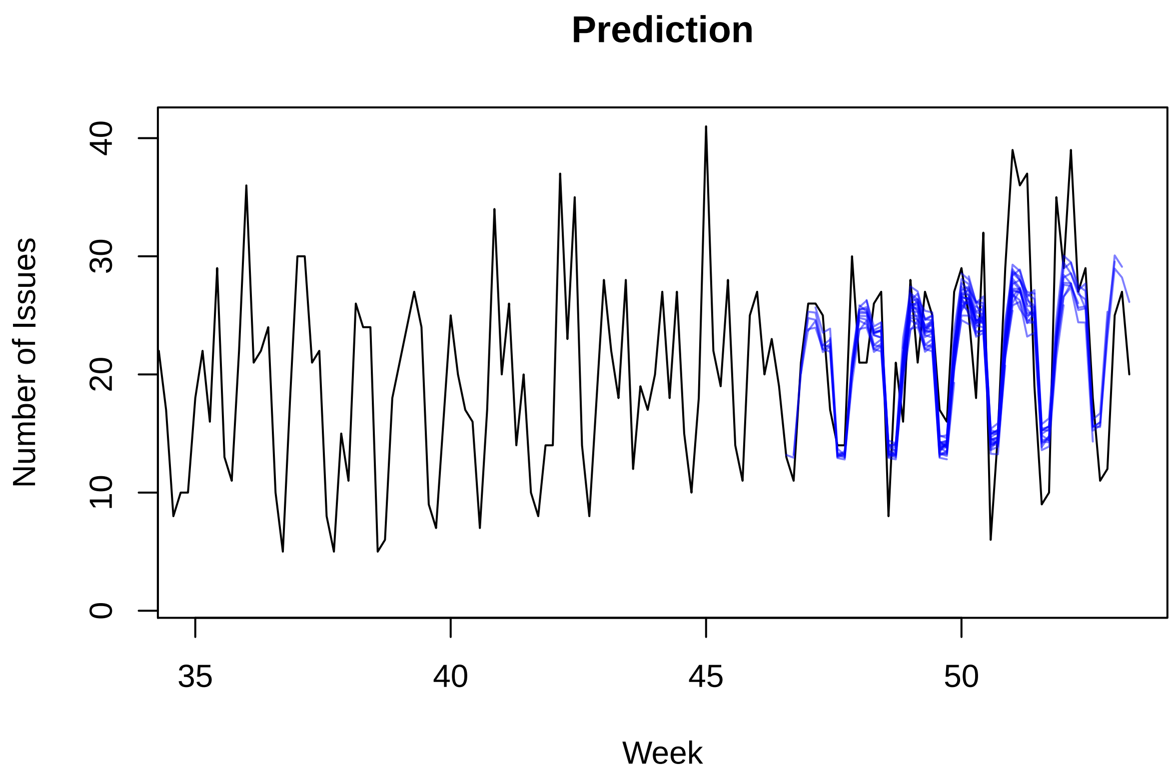
# Smoothing

## Exponential smoothing ZMM

```r
hw.forecast <- function(sample) {
  results <- list()
  results$train <- sample$train.ts
  results$valid <- sample$valid.ts
  results$model <- ets(sample$train.ts, model = "ZMM")
  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))
```

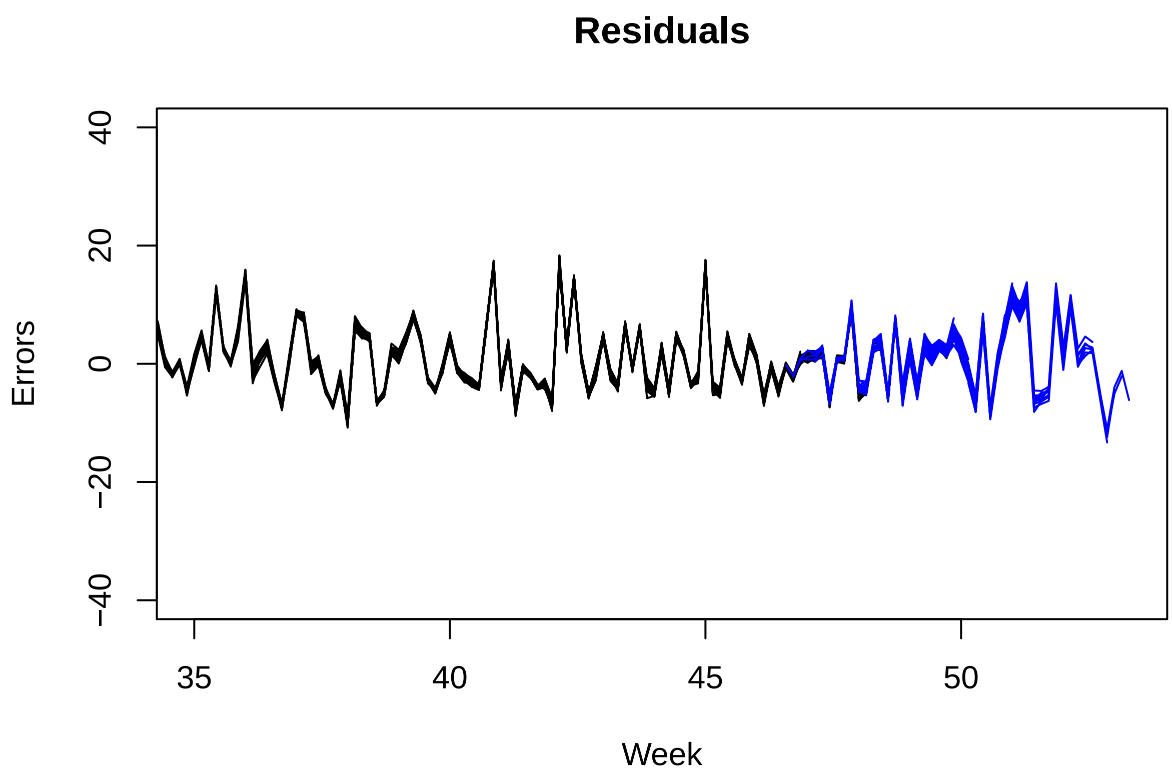
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 0.0268633 | 5.232163 | 4.104117 | -15.944895 | 35.26945 | 0.7470828 | 0.1386387 | NA |
| Test set | 1.3127929 | 5.795947 | 4.917541 | -4.604707 | 26.26229 | 0.8955106 | -0.1160116 | 0.5146231 |

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



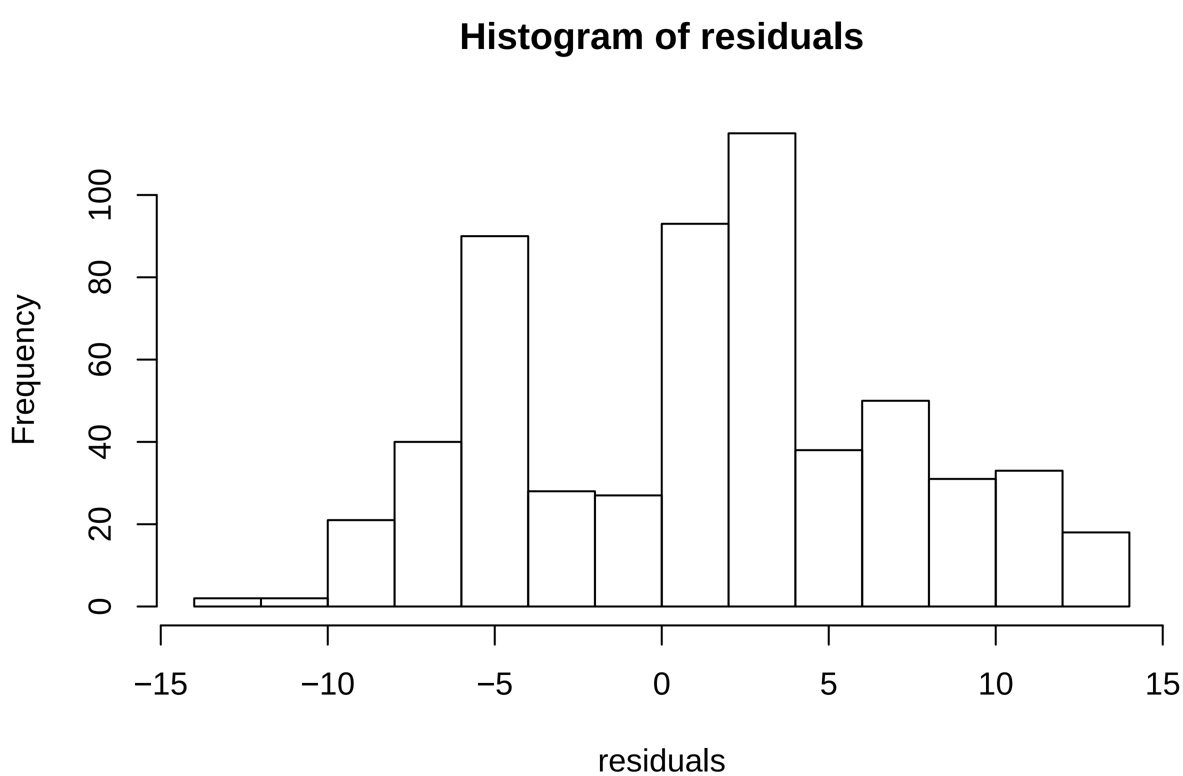

```
## NULL
```

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



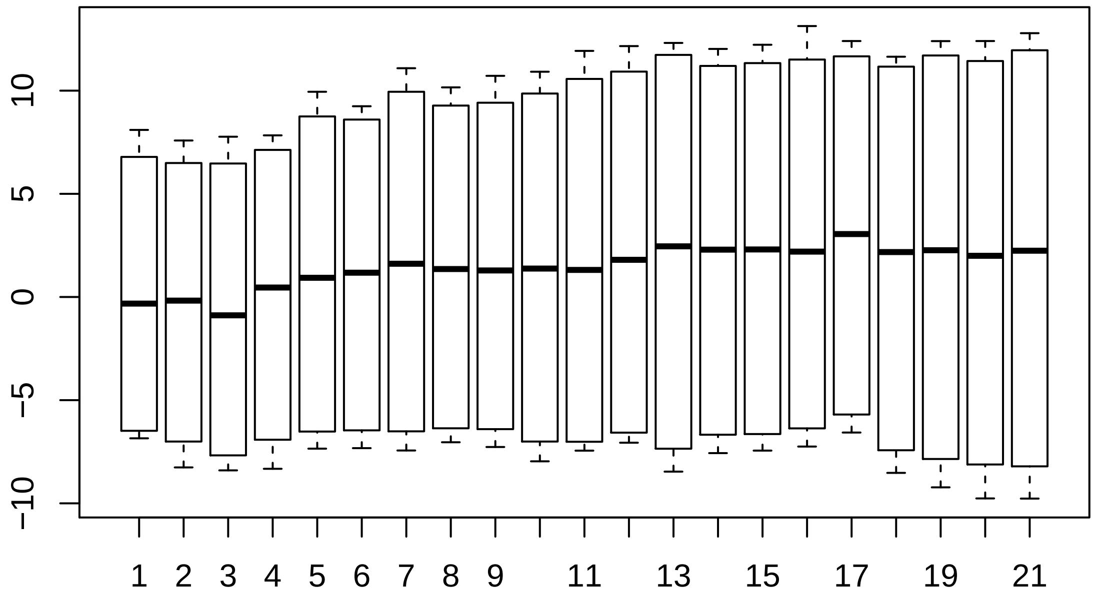

```
## NULL
```

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



```
##     97.5%       90%       10%      2.5%
## 12.223959  9.525994 -6.136258 -8.313395
```

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




```
##     97.5%       90%       10%      2.5%
## 12.223959  9.525994 -6.136258 -8.313395
```

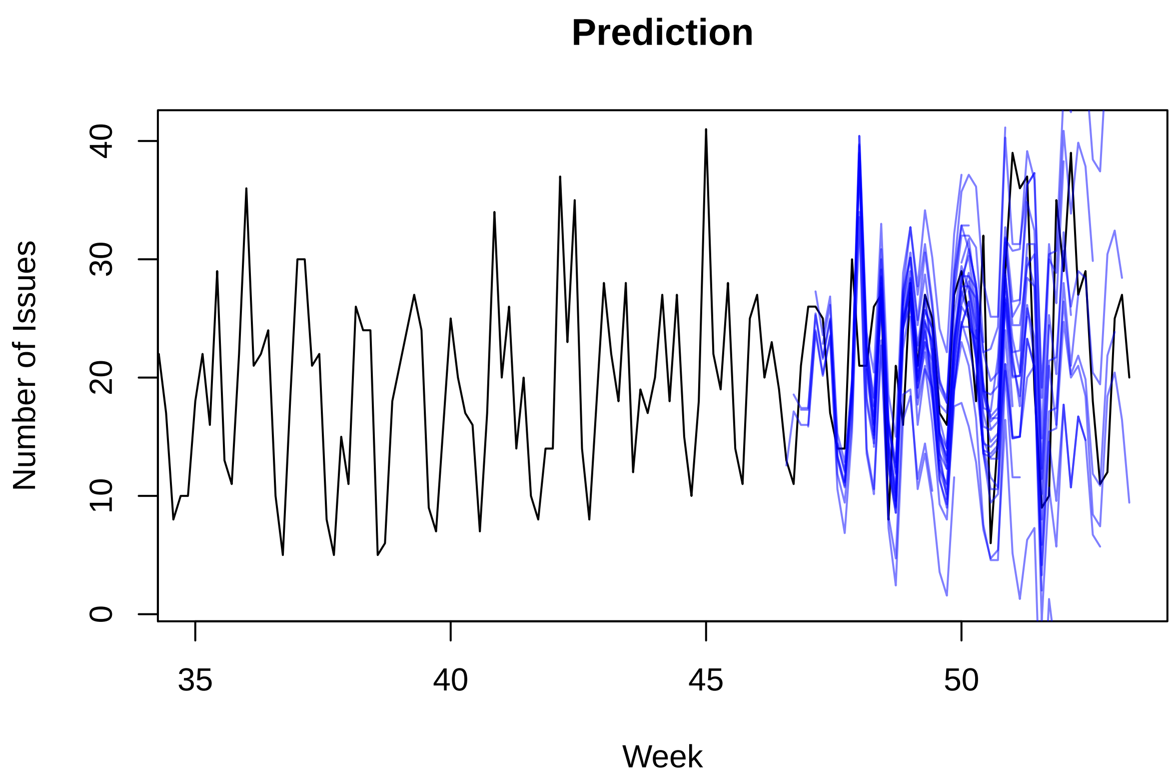
### Double differencing

```
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))
```

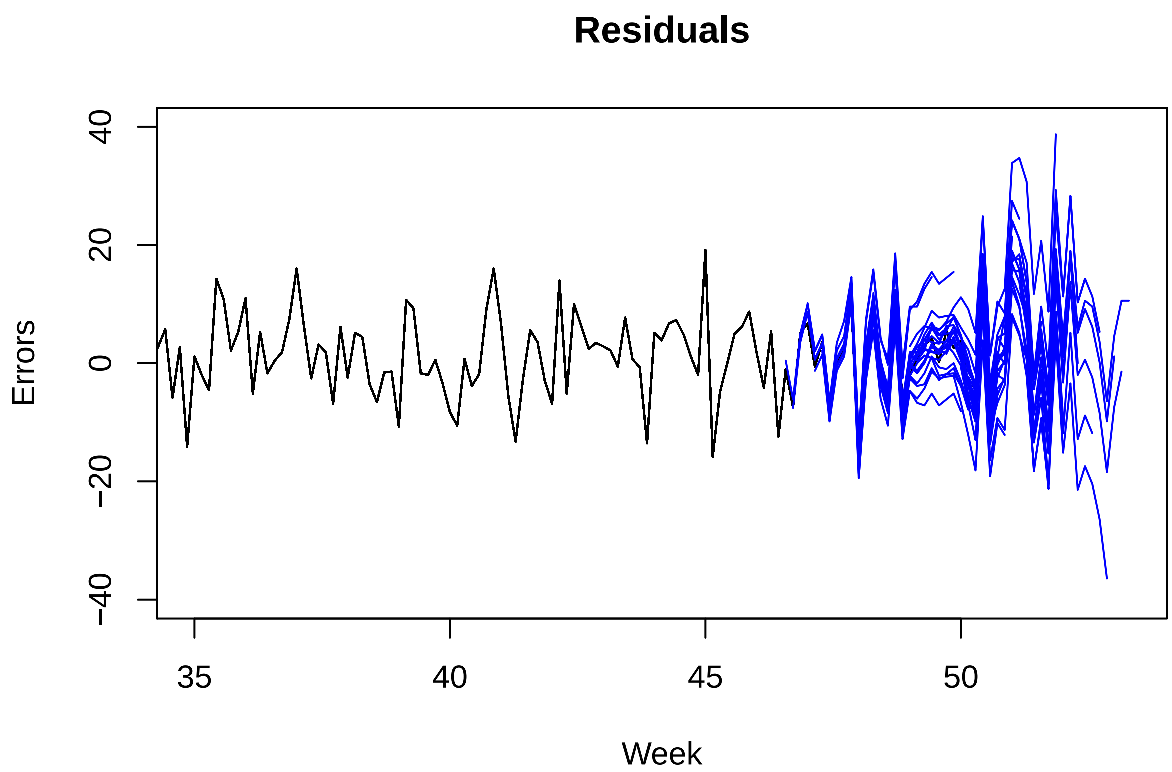
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 0.9008647 | 6.811868 | 5.386710 | -3.838190 | 38.74571 | 0.9559362 | 0.1534380 | NA |
| Test set | 2.0017007 | 9.694547 | 7.654276 | -2.681855 | 39.09663 | 1.3584964 | -0.1009088 | 0.8742482 |

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



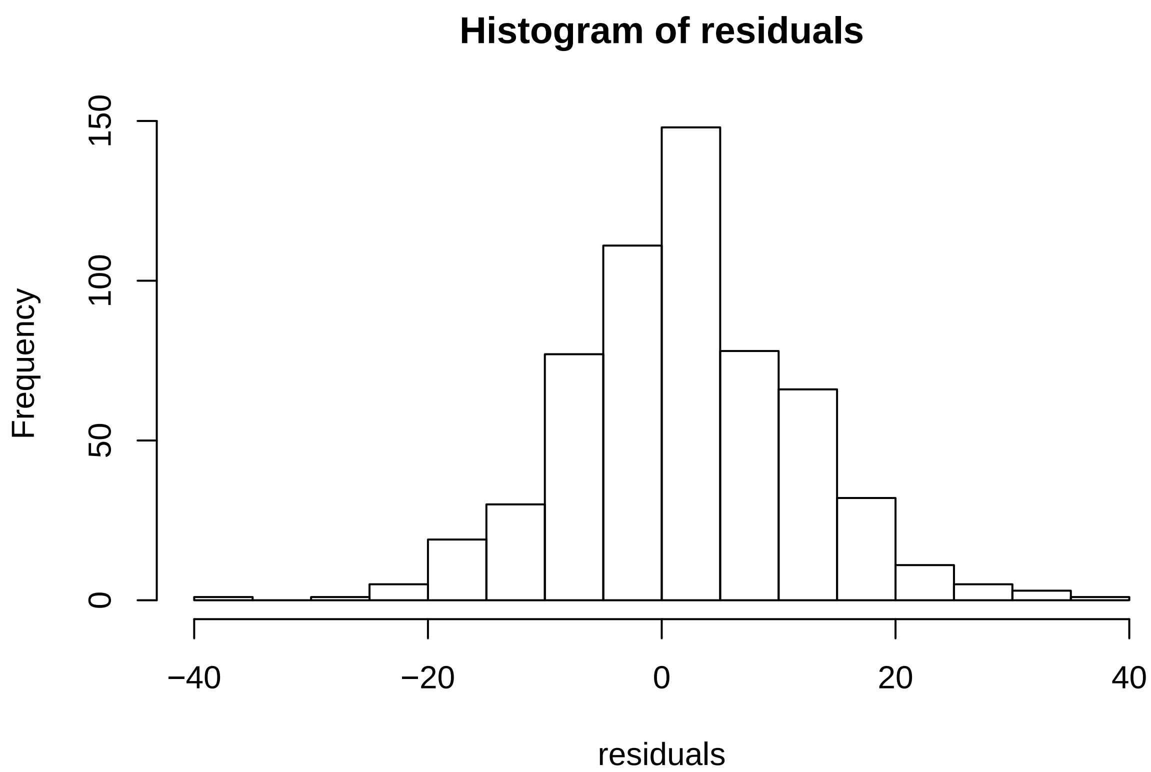

```
## NULL
```

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



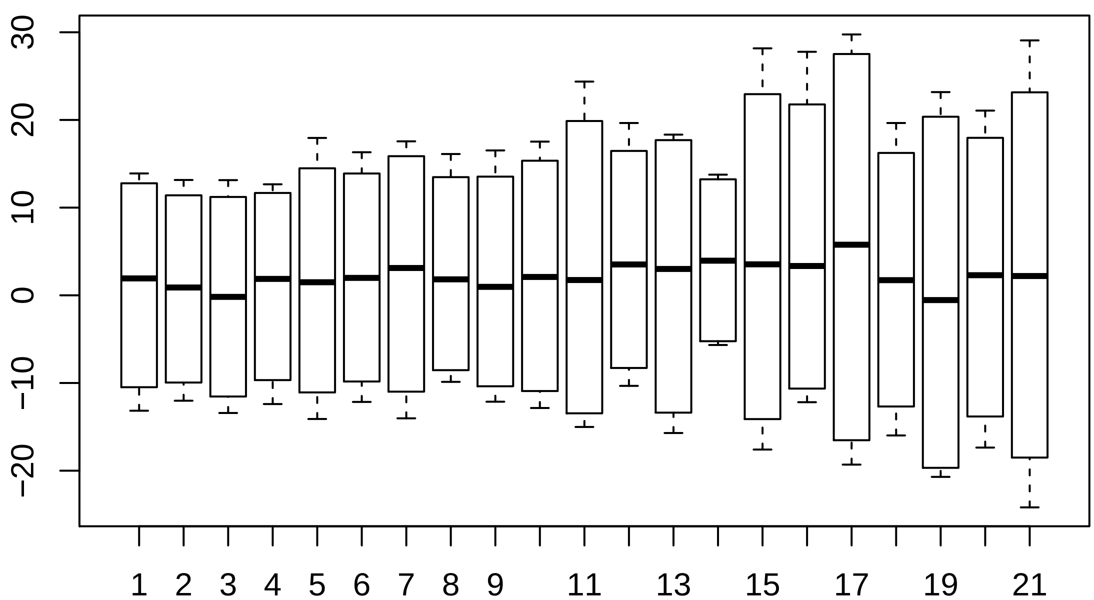

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##  22.264286  14.328571  -9.857143 -18.189286
```

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




```
##      97.5%        90%        10%       2.5%
##  22.264286  14.328571  -9.857143 -18.189286
```

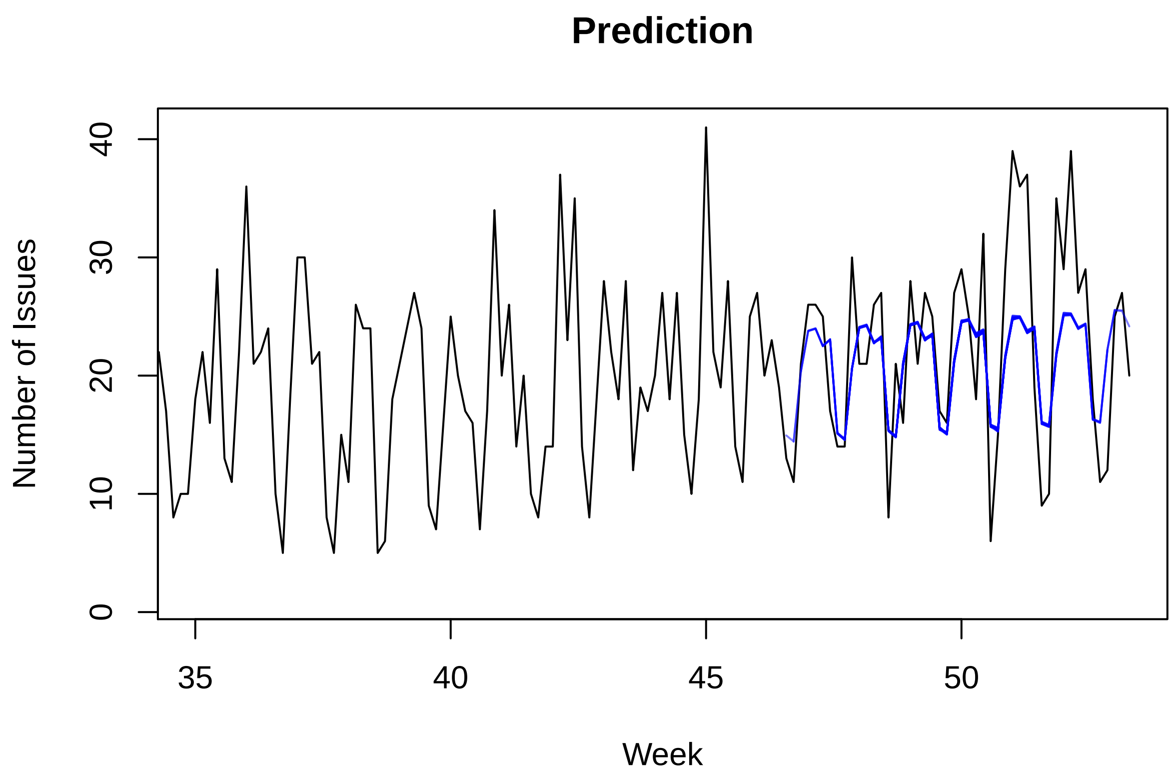
## Regression

### 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 + trend)
  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))
```

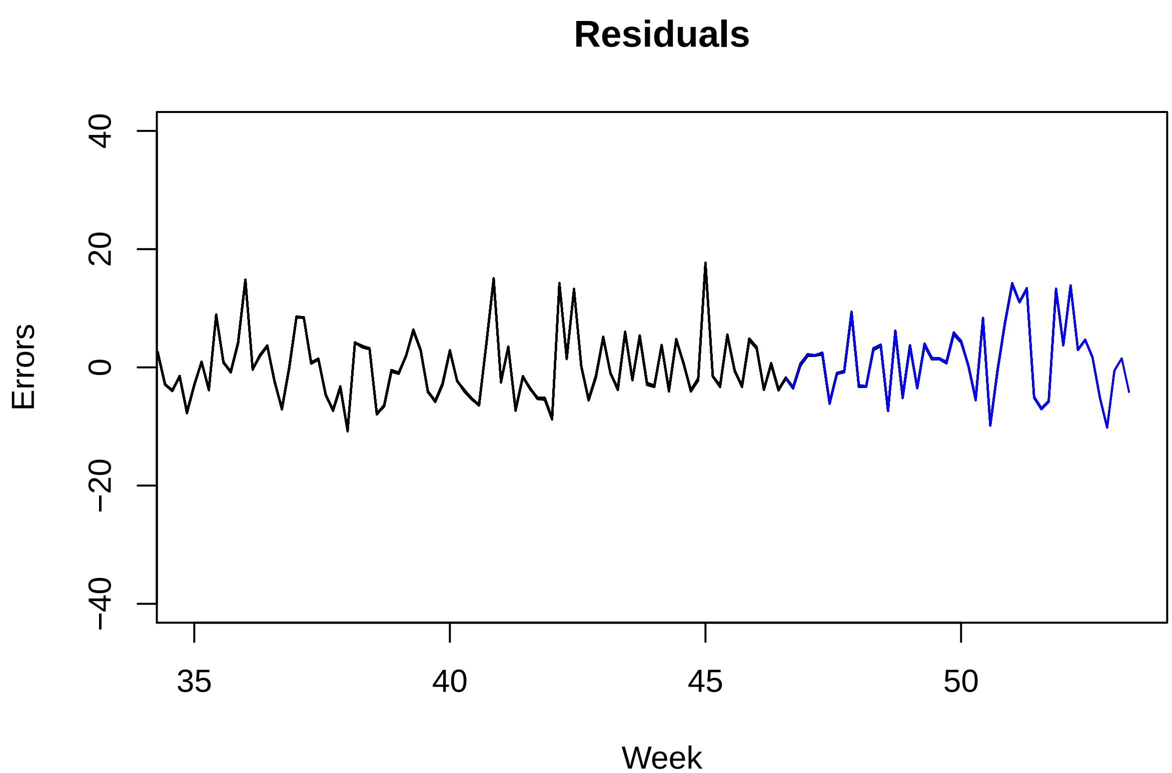
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 0.000000 | 5.250405 | 4.185671 | -14.604929 | 35.13895 | 0.7619330 | 0.2104383 | NA |
| Test set | 1.792356 | 6.280787 | 5.254605 | -4.740692 | 27.92253 | 0.9570341 | -0.0912699 | 0.5195687 |

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



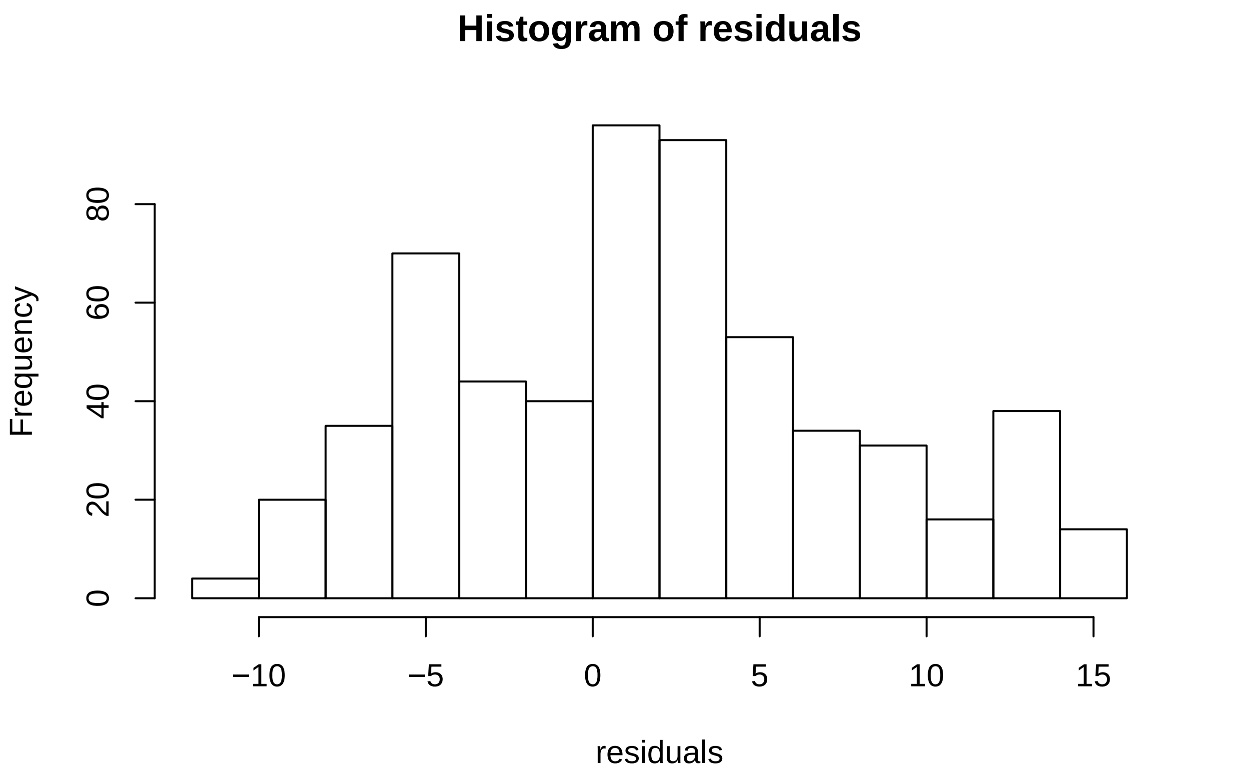

```
## NULL
```

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



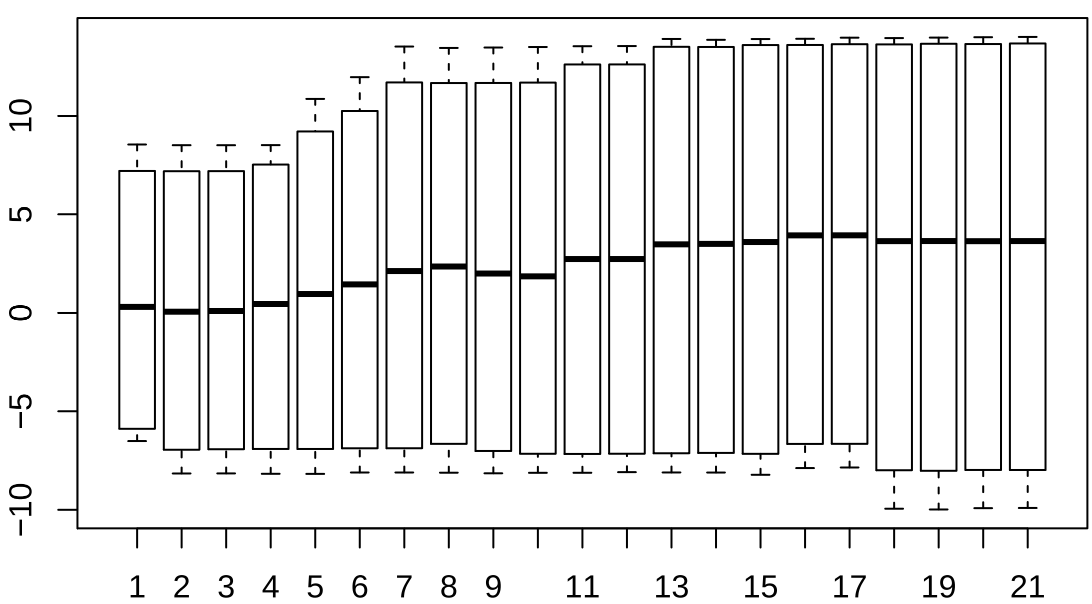

```
## NULL
```

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




```
##      97.5%       90%       10%      2.5%
## 13.949857 11.071882 -5.894789 -9.771909
```

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



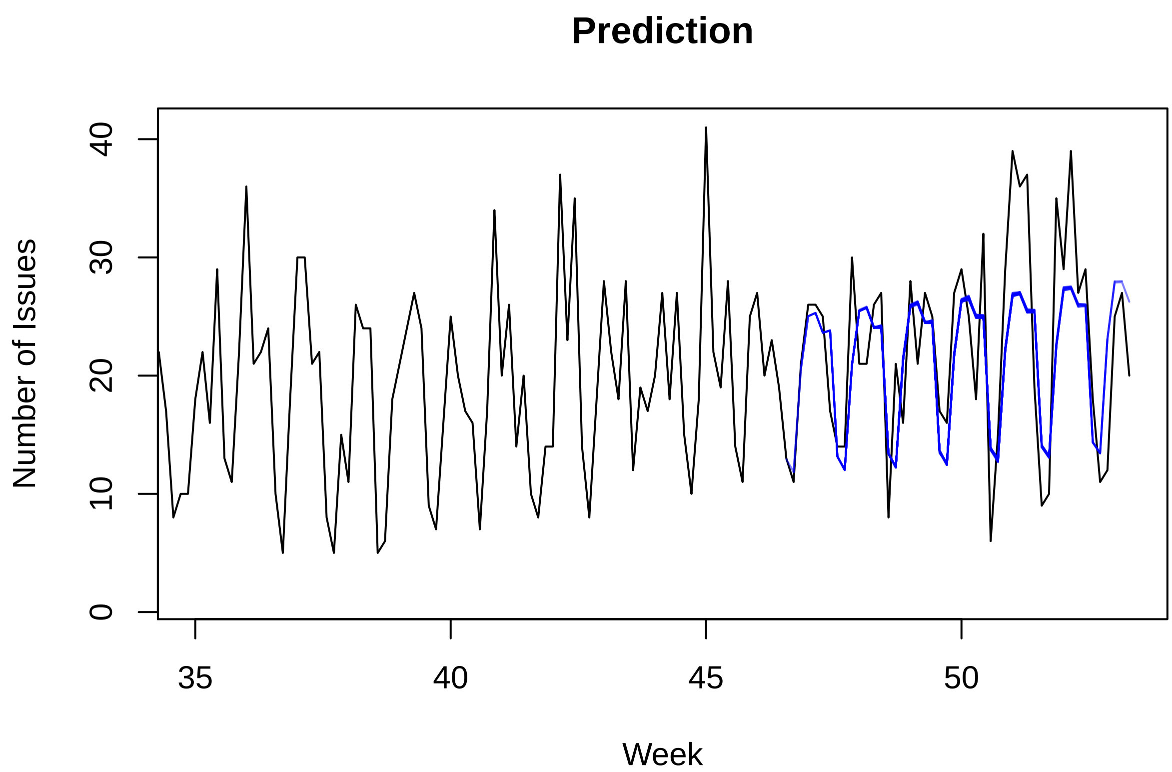

```
##      97.5%       90%       10%      2.5%
## 13.949857 11.071882 -5.894789 -9.771909
```

### linear multiplicative regression

```r
regr.mult.forecast <- function(sample.issues) {
  train.ts <- sample.issues$train.ts
  valid.ts <- sample.issues$valid.ts
  train.lm <- tslm(train.ts ~ season + trend, 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))
```

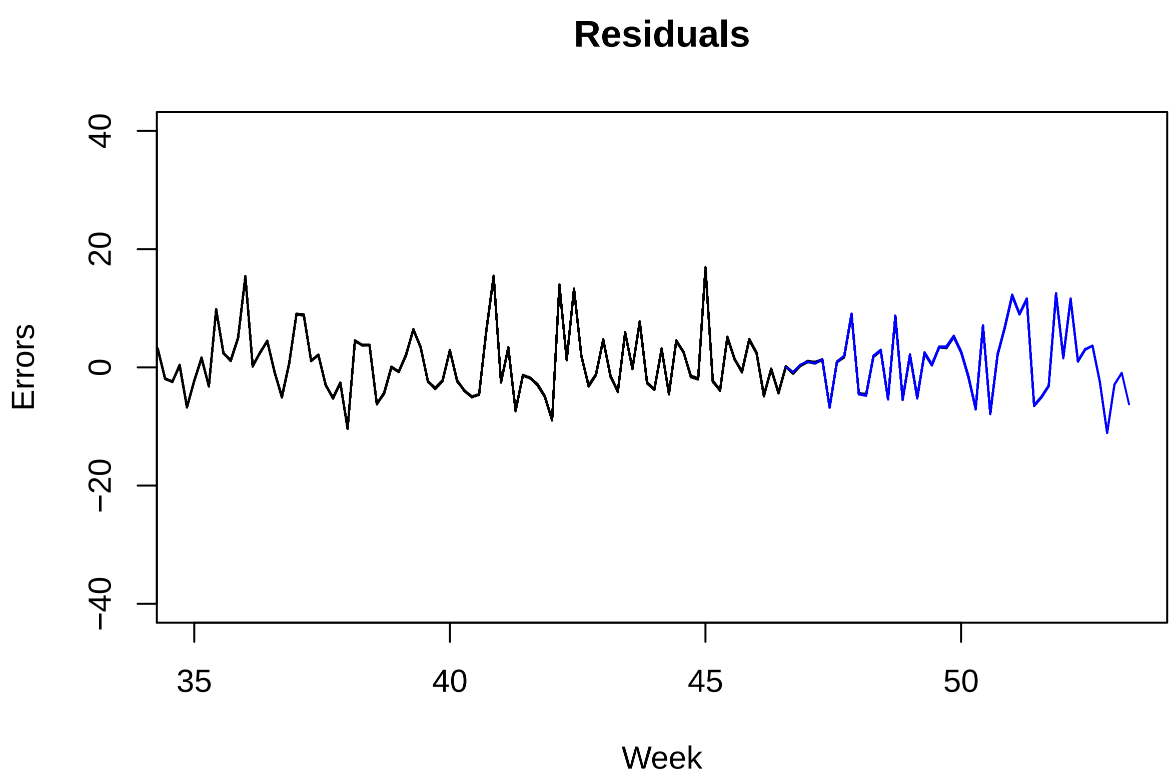
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 0.8836526 | 5.209053 | 4.036375 | -7.784251 | 32.05142 | 0.7347547 | 0.2142670 | NA |
| Test set | 1.4627136 | 5.838436 | 4.964767 | -2.978273 | 26.07463 | 0.9042351 | -0.1013327 | 0.5354692 |

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



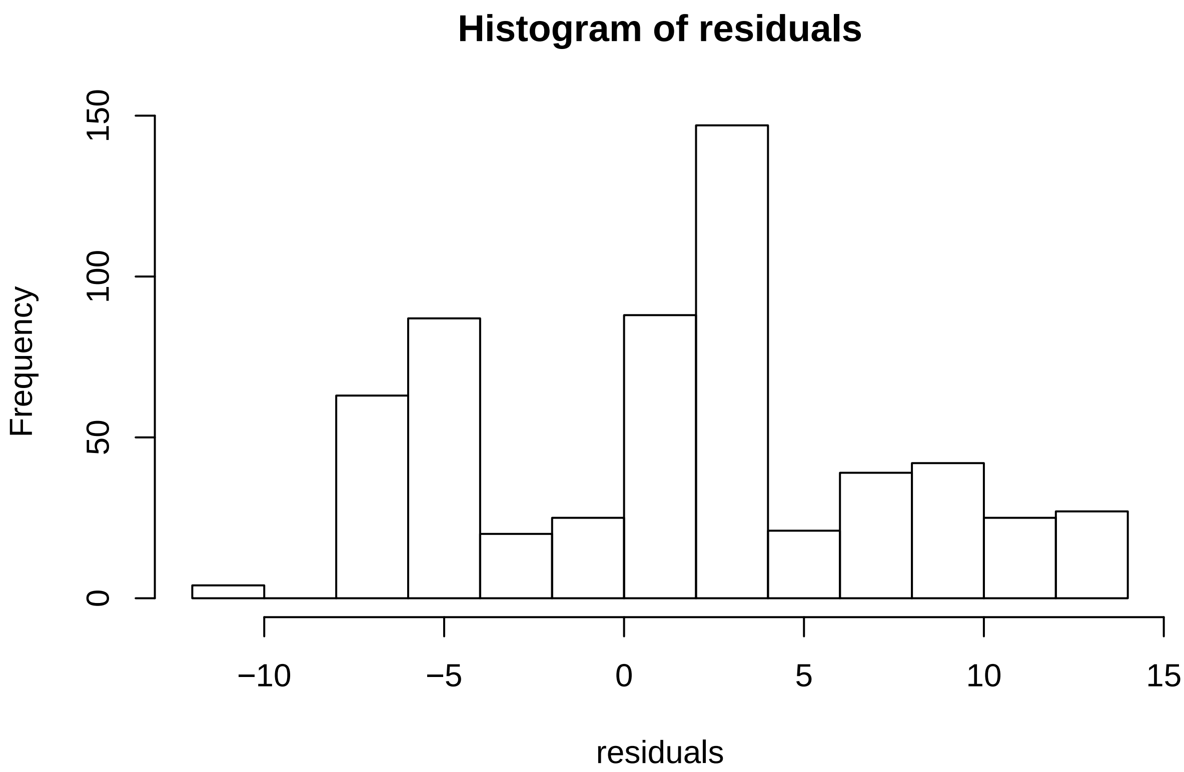

```
## NULL
```

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



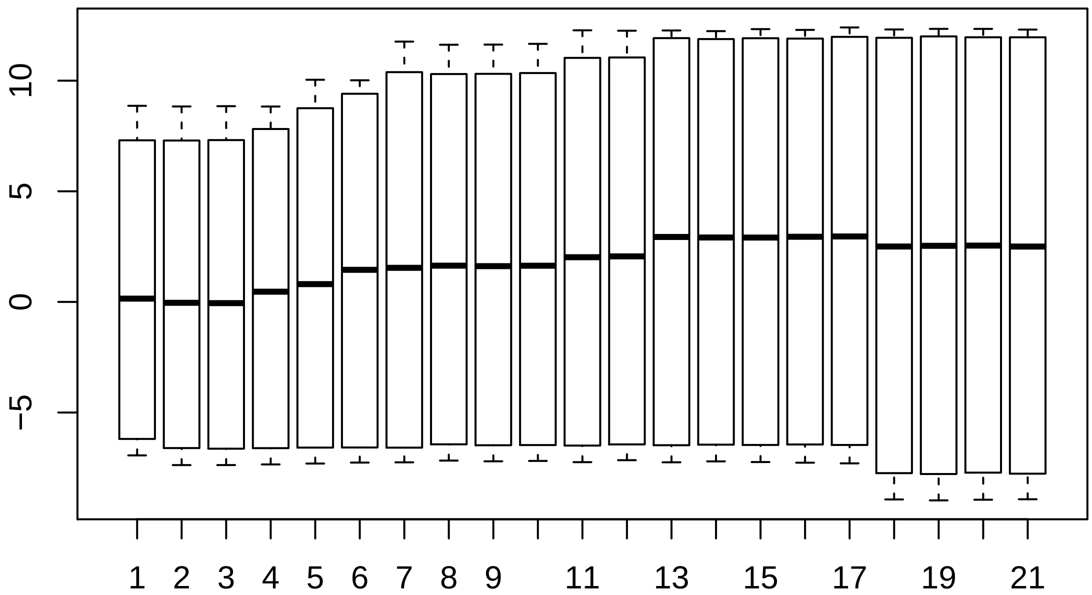

```
## NULL
```

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




```
##     97.5%       90%       10%      2.5%
## 12.244596  9.066479 -6.451632 -7.836283
```

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



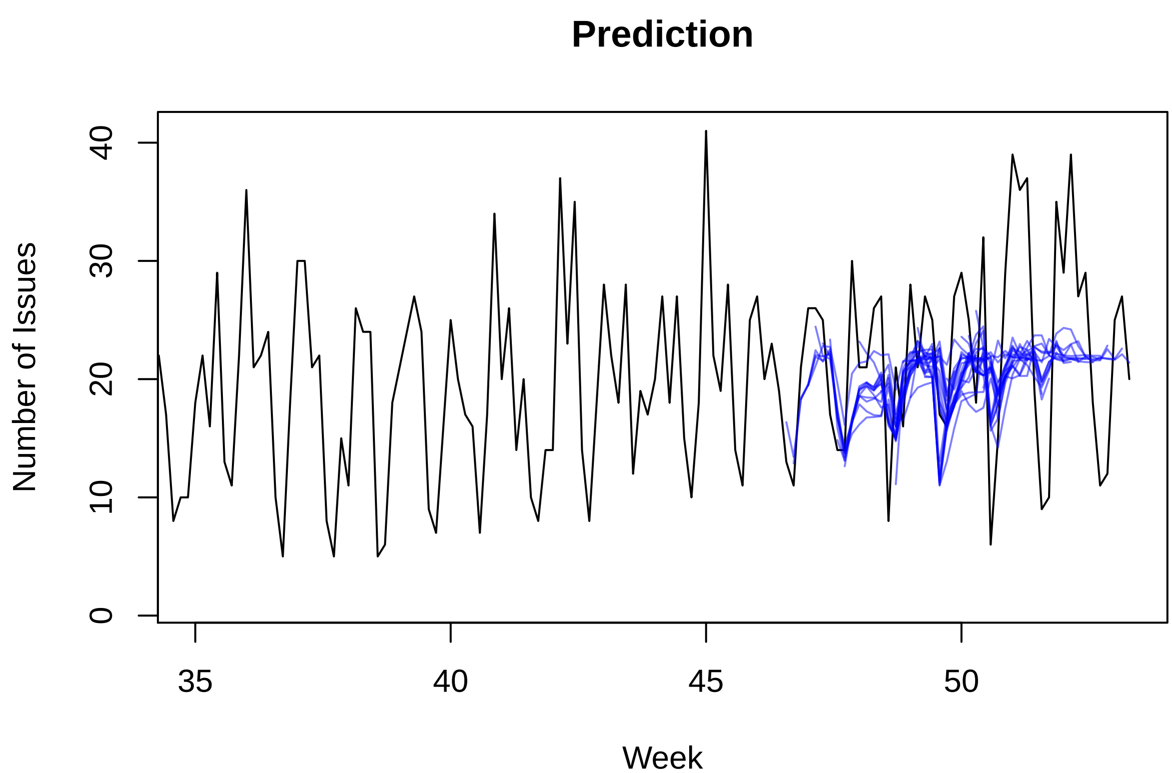

```
##     97.5%       90%       10%      2.5%
## 12.244596  9.066479 -6.451632 -7.836283
```

### 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))
```

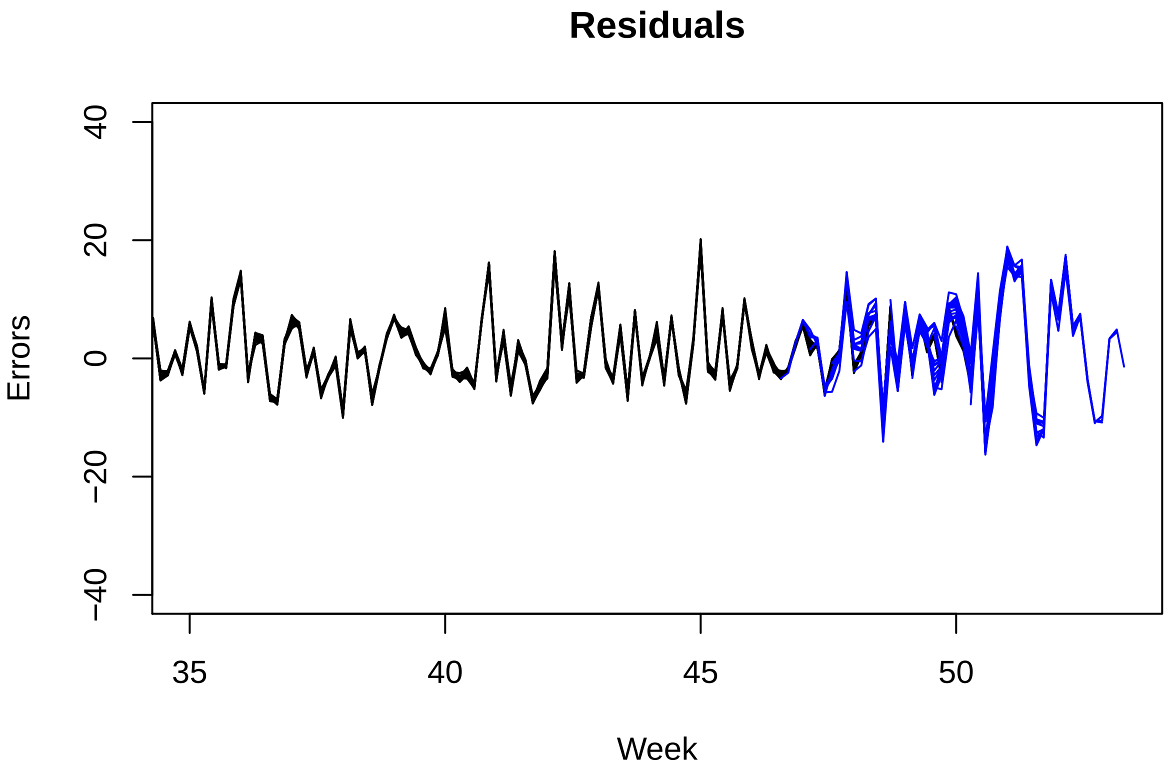
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | -0.0034192 | 5.459053 | 4.353965 | -18.792661 | 38.65364 | 0.7925785 | 0.0126479 | NA |
| Test set | 2.8166244 | 8.276614 | 7.086151 | -5.584128 | 38.73241 | 1.2905682 | 0.0635875 | 0.6748557 |

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



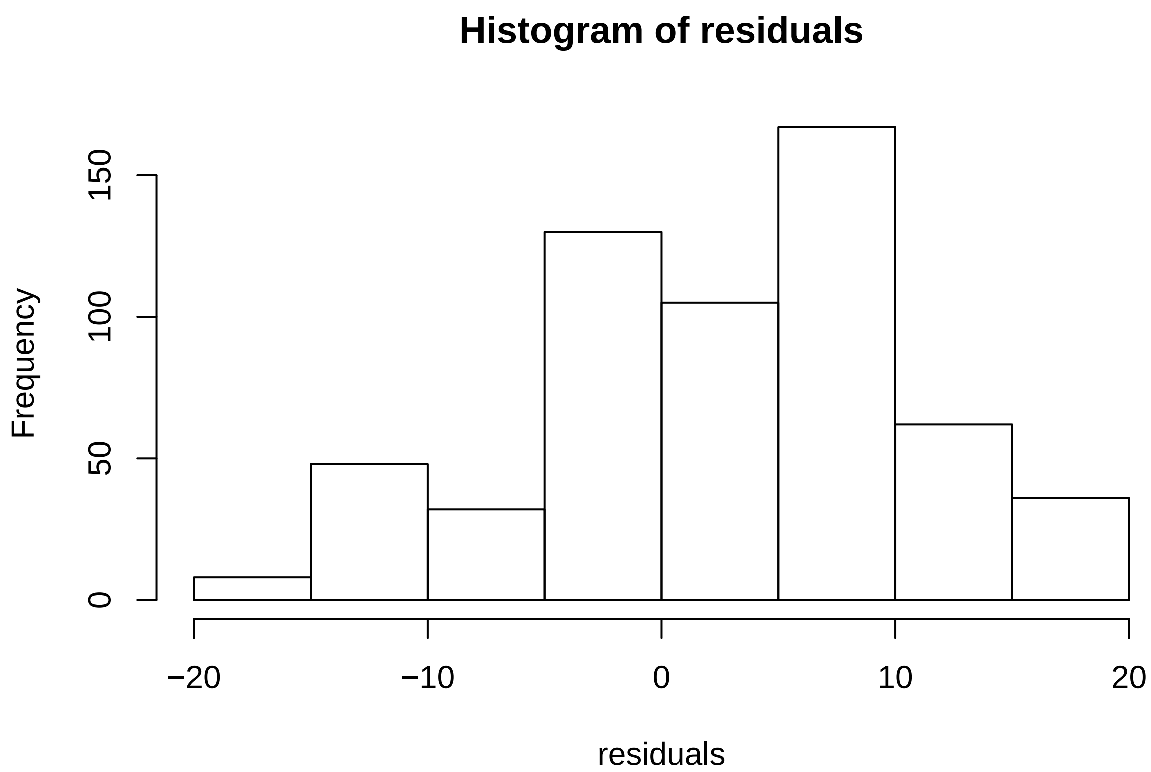

```
## NULL
```

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



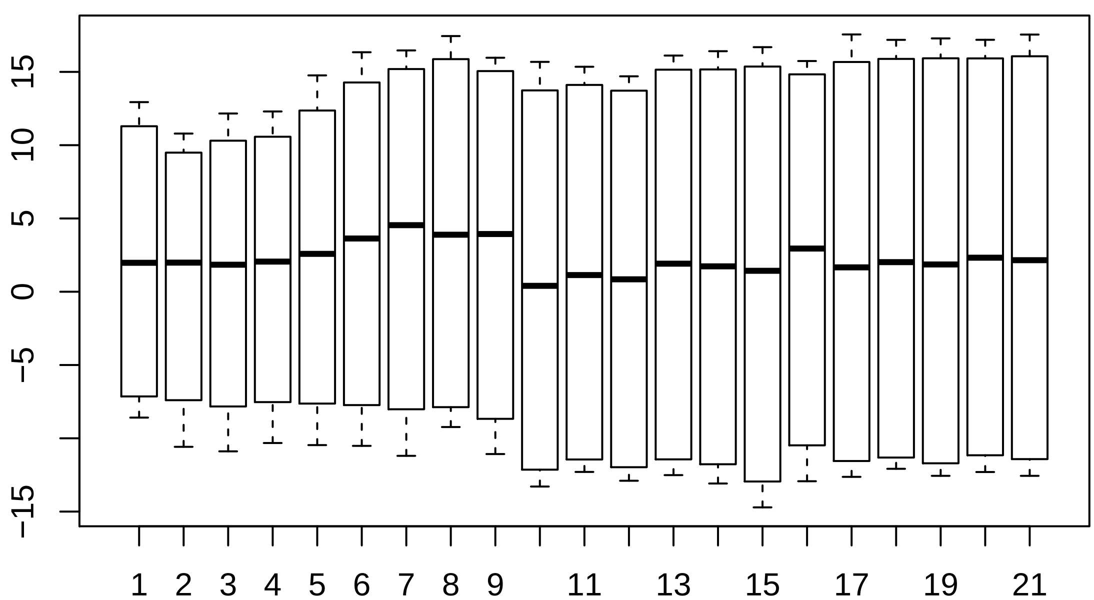

```
## NULL
```

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




```
##       97.5%       90%       10%      2.5%
##   17.134195 13.661779 -9.722986 -14.023674
```

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



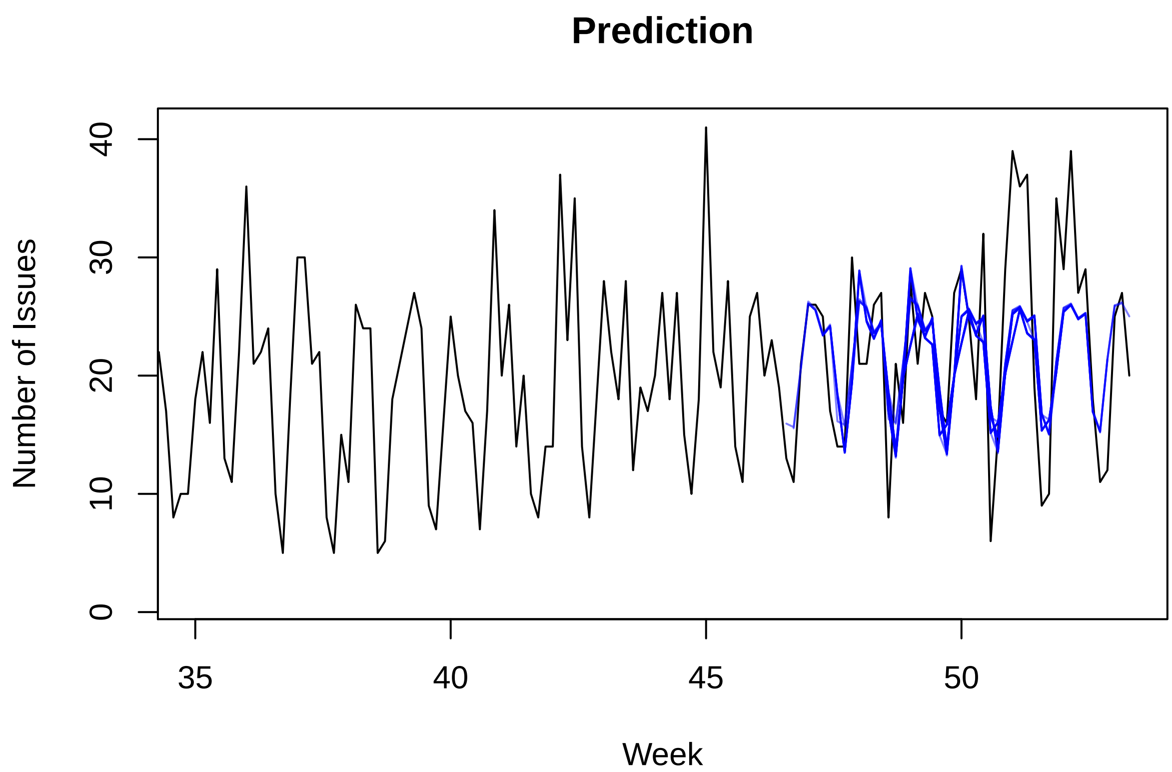

```
##       97.5%       90%       10%      2.5%
##   17.134195 13.661779 -9.722986 -14.023674
```

## 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))
```

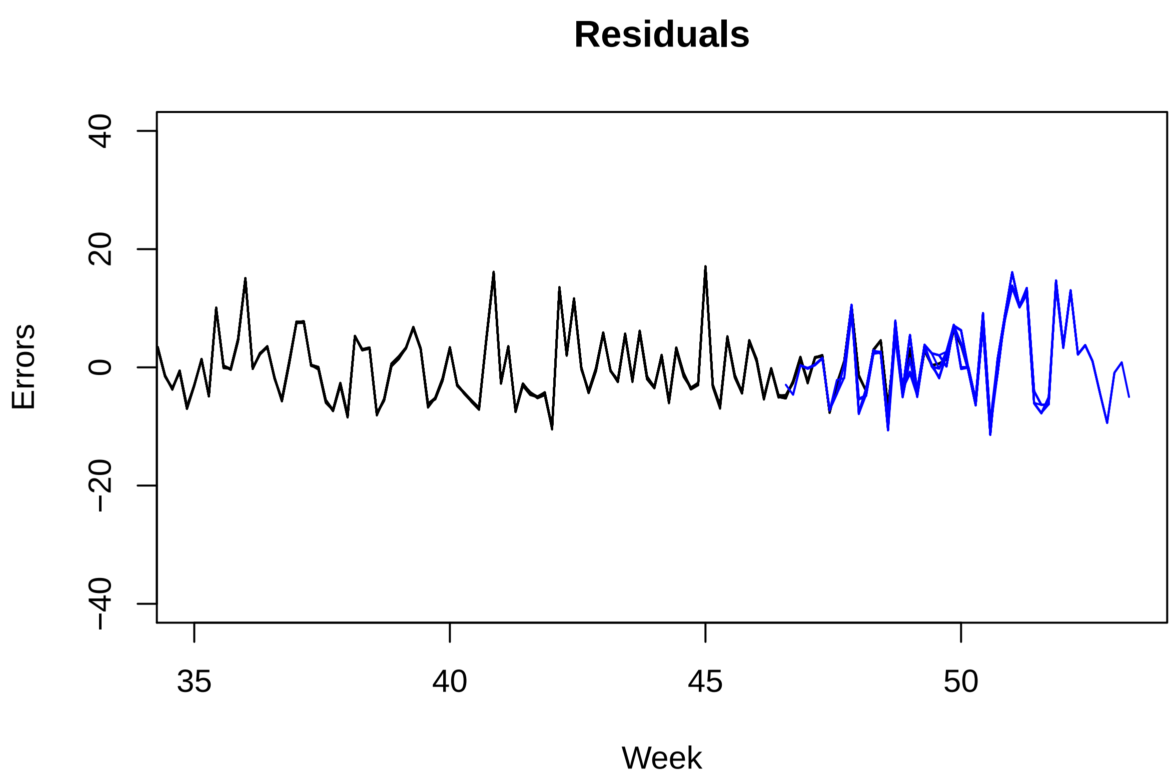
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 0.000000 | 5.145321 | 4.126345 | -14.222141 | 34.84697 | 0.7496657 | 0.1511997 | NA |
| Test set | 1.408232 | 6.627177 | 5.417145 | -7.092694 | 29.66271 | 0.9847067 | -0.0742656 | 0.5656583 |

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



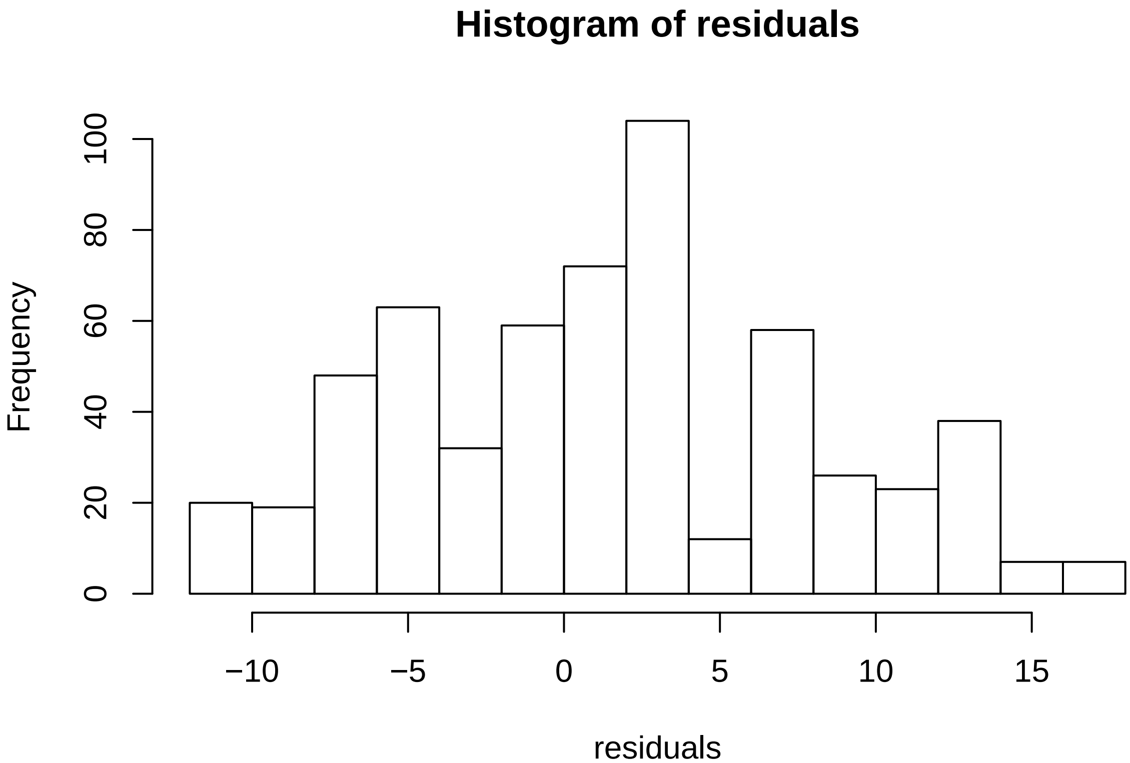

```
## NULL
```

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



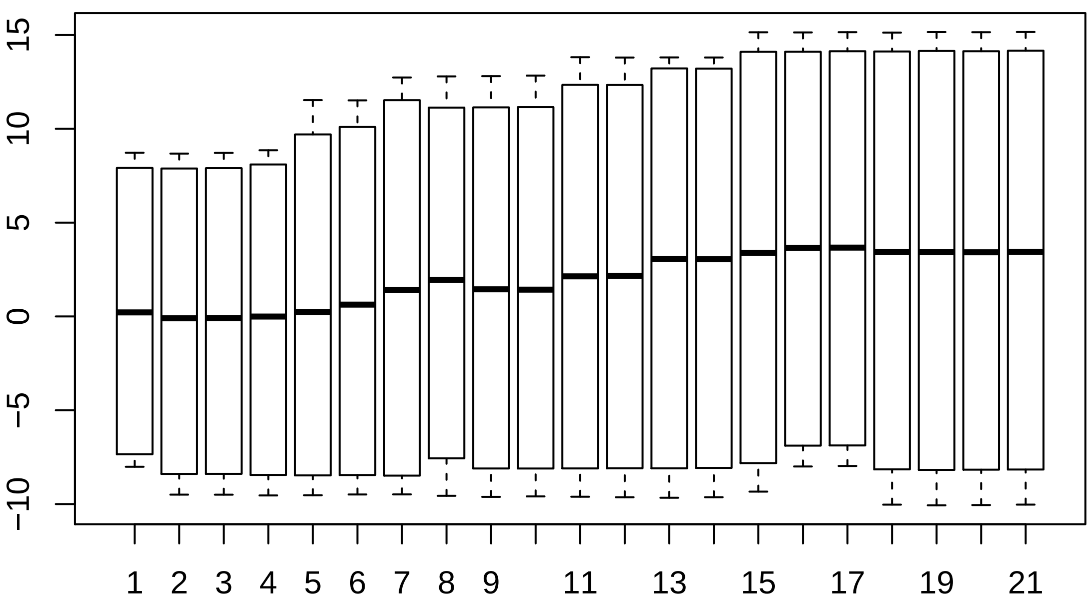

```
## NULL
```

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




```
##      97.5%       90%       10%      2.5%
##   13.83989  10.46163  -6.66626 -10.52706
```

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



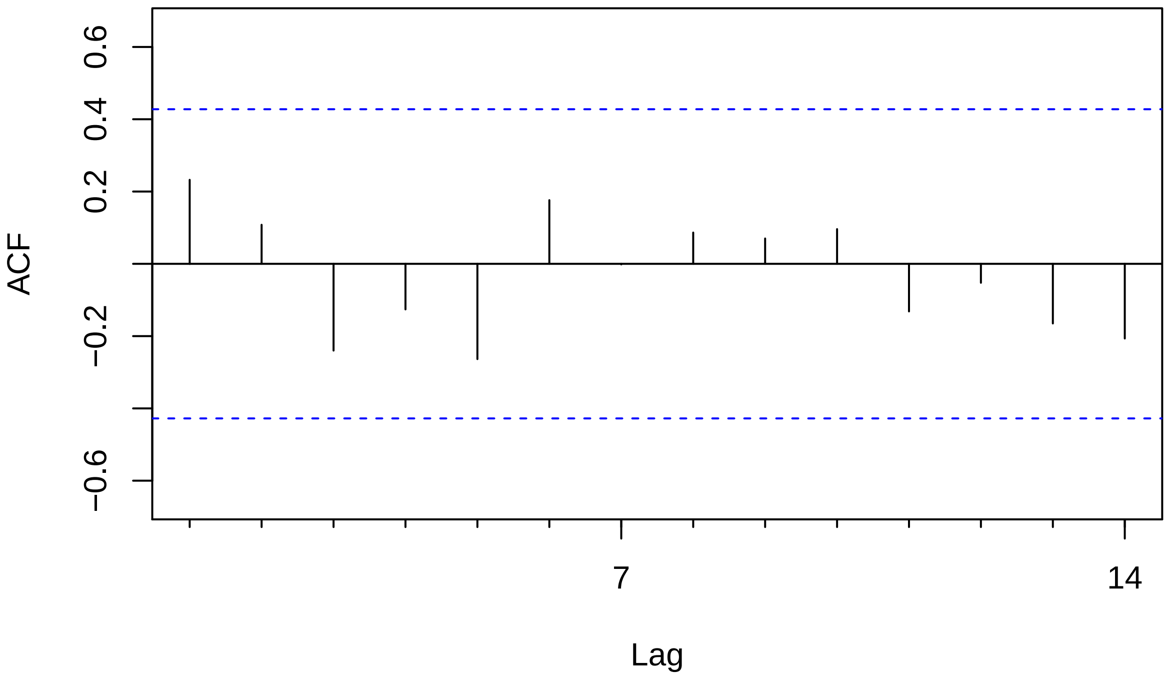

```
##      97.5%       90%       10%      2.5%
##   13.83989  10.46163  -6.66626 -10.52706
```

## 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))
```

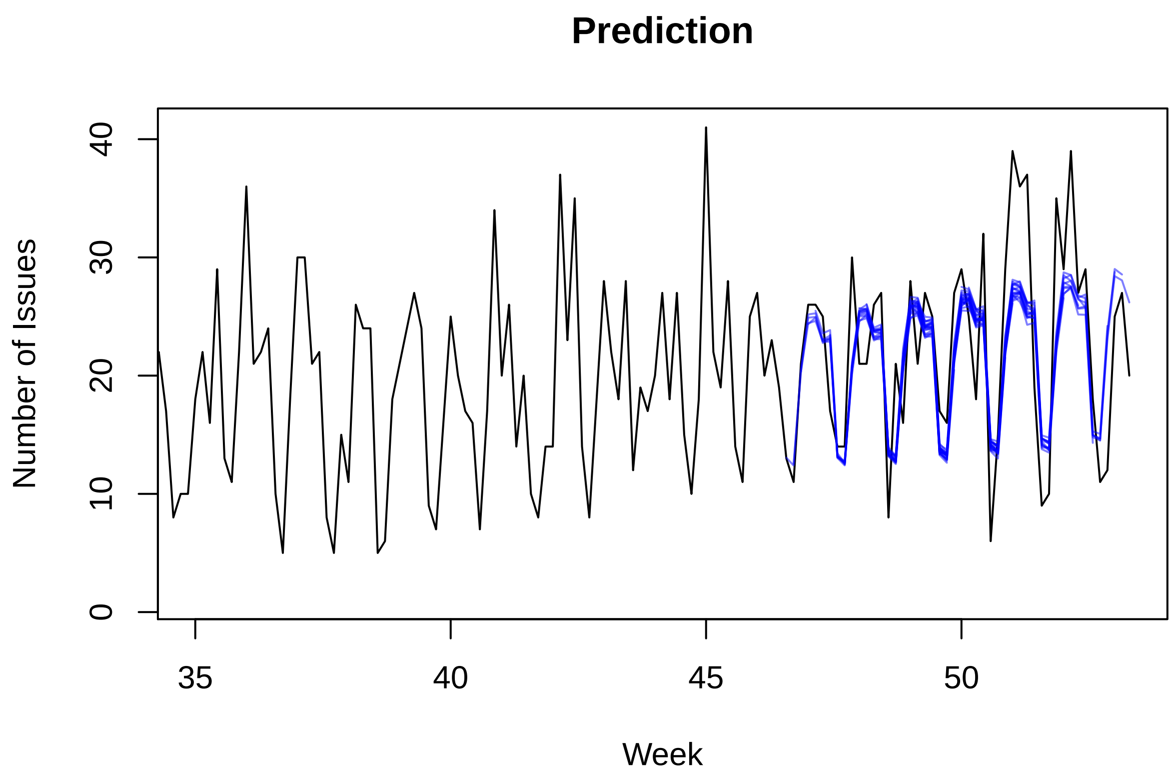
| | ME | RMSE | MAE | MPE | MAPE | MASE | ACF1 | Theil's U |
|---|---|---|---|---|---|---|---|---|
| Training set | 0.6469228 | 8.913922 | 7.355136 | -24.83879 | 61.12687 | 1.3052871 | 0.4323747 | NA |
| Test set | 1.3877533 | 5.790308 | 4.932371 | -3.79149 | 26.12635 | 0.8758456 | -0.1110769 | 0.521521 |

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



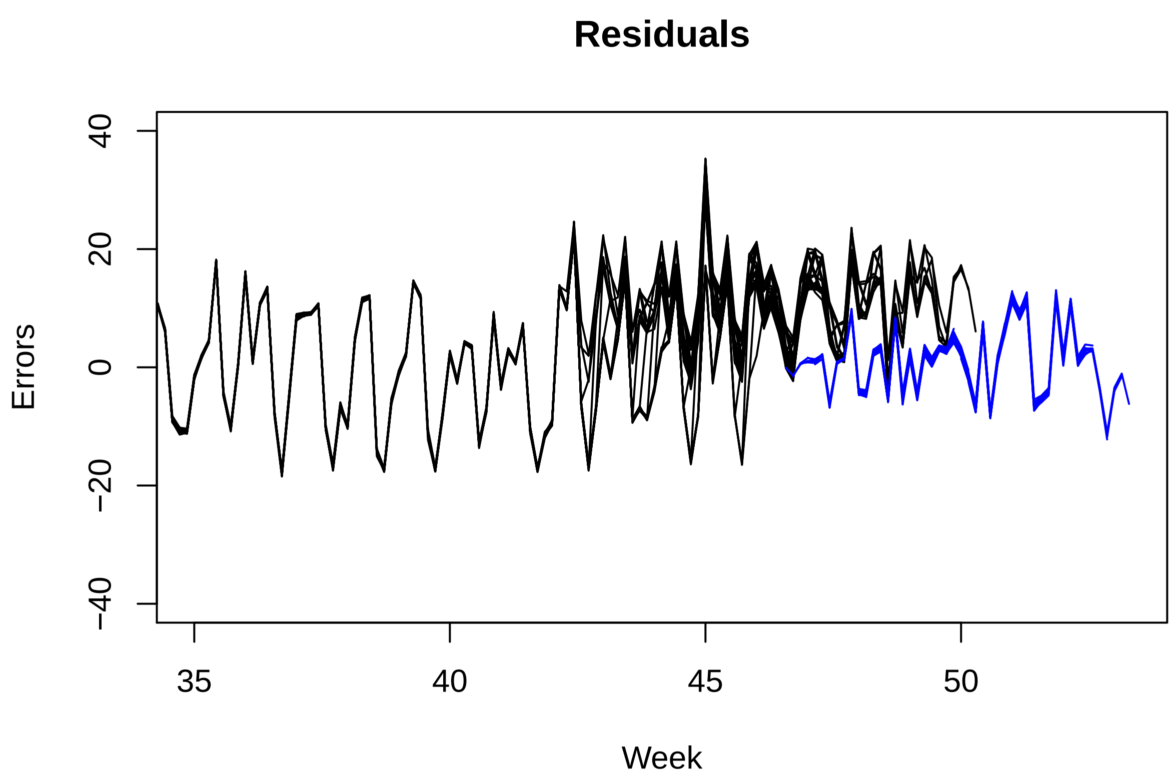

```
test <- list()
test$train.ts <- issues.ts
test$valid.ts <- naive(issues.ts, h=n.valid)$mean

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)
```



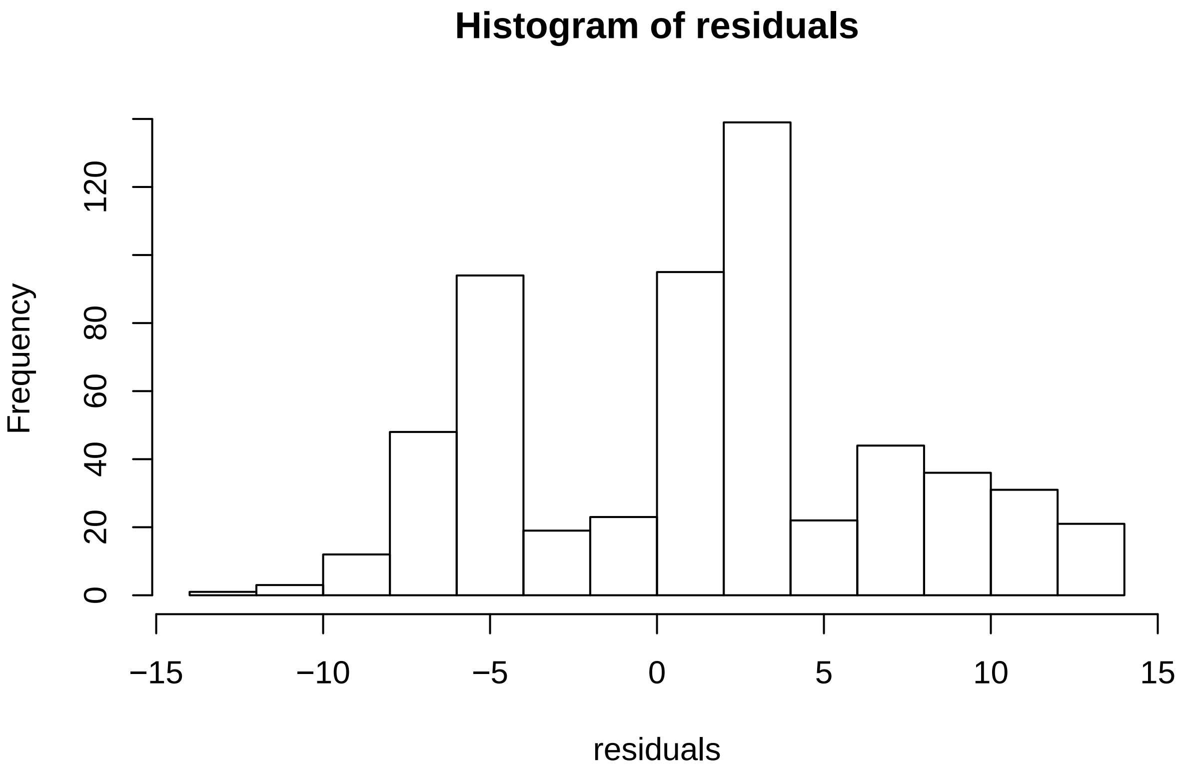

```
## NULL
```

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



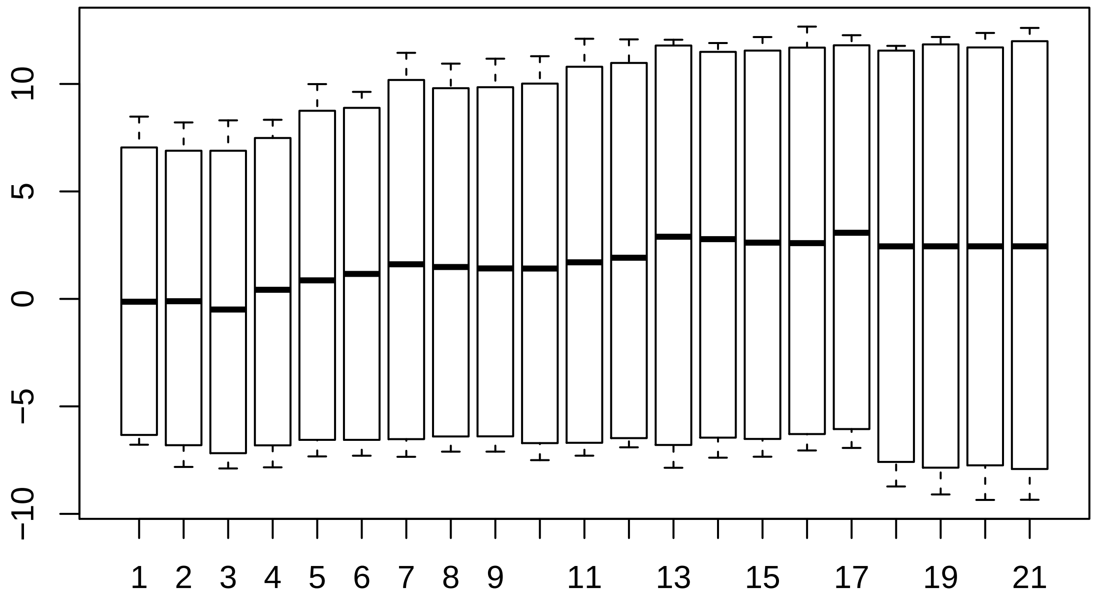

```
## NULL
```

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




```
##      97.5%       90%       10%      2.5%
## 12.109749  9.363365 -6.165749 -8.076651
```

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




```
##      97.5%       90%       10%      2.5%
## 12.109749  9.363365 -6.165749 -8.076651
```

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