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.

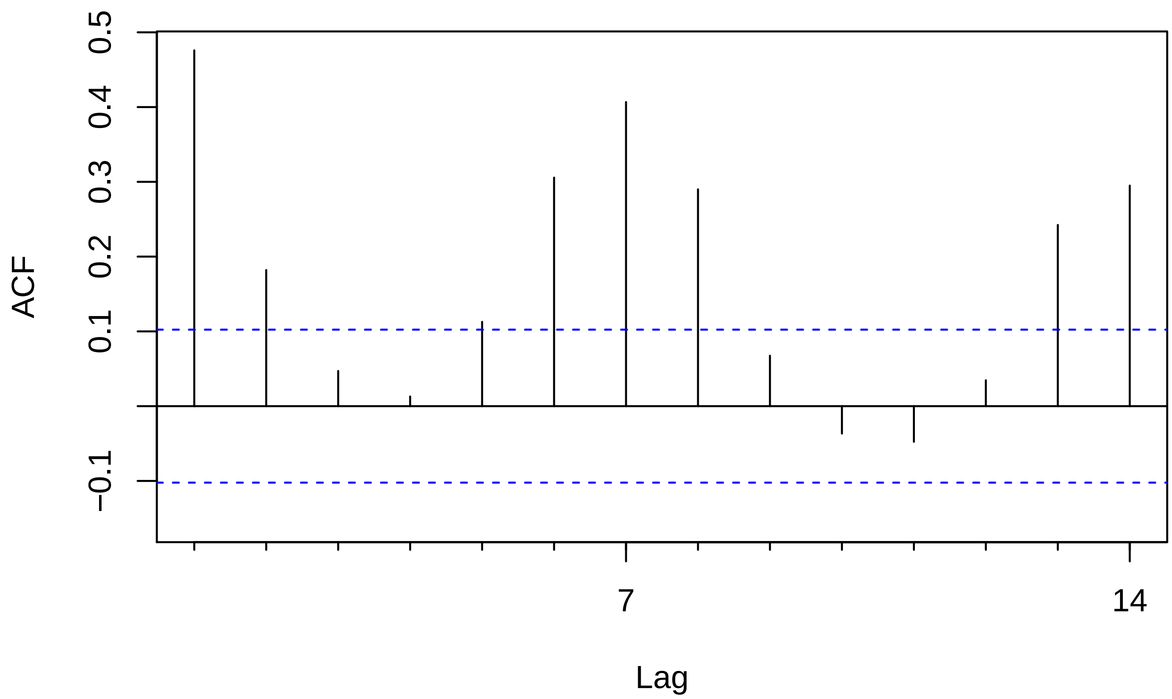
Load the data

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

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

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

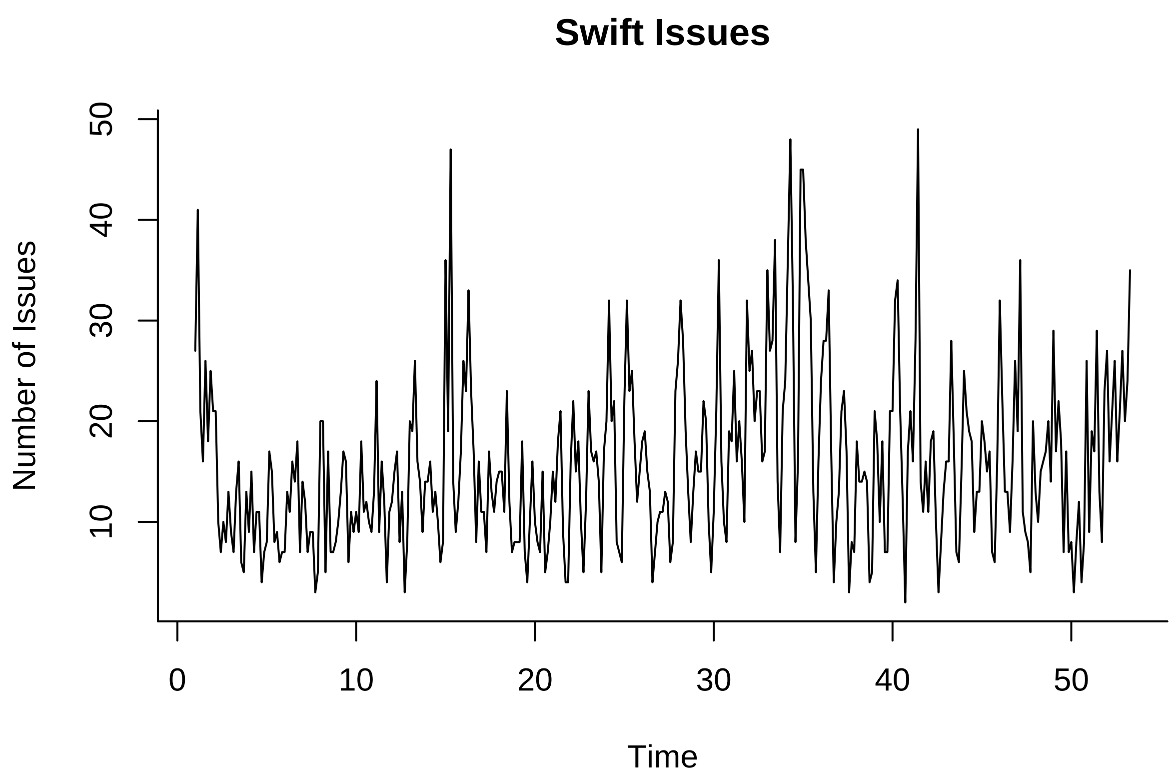
keep the last 12 months

```
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)
```

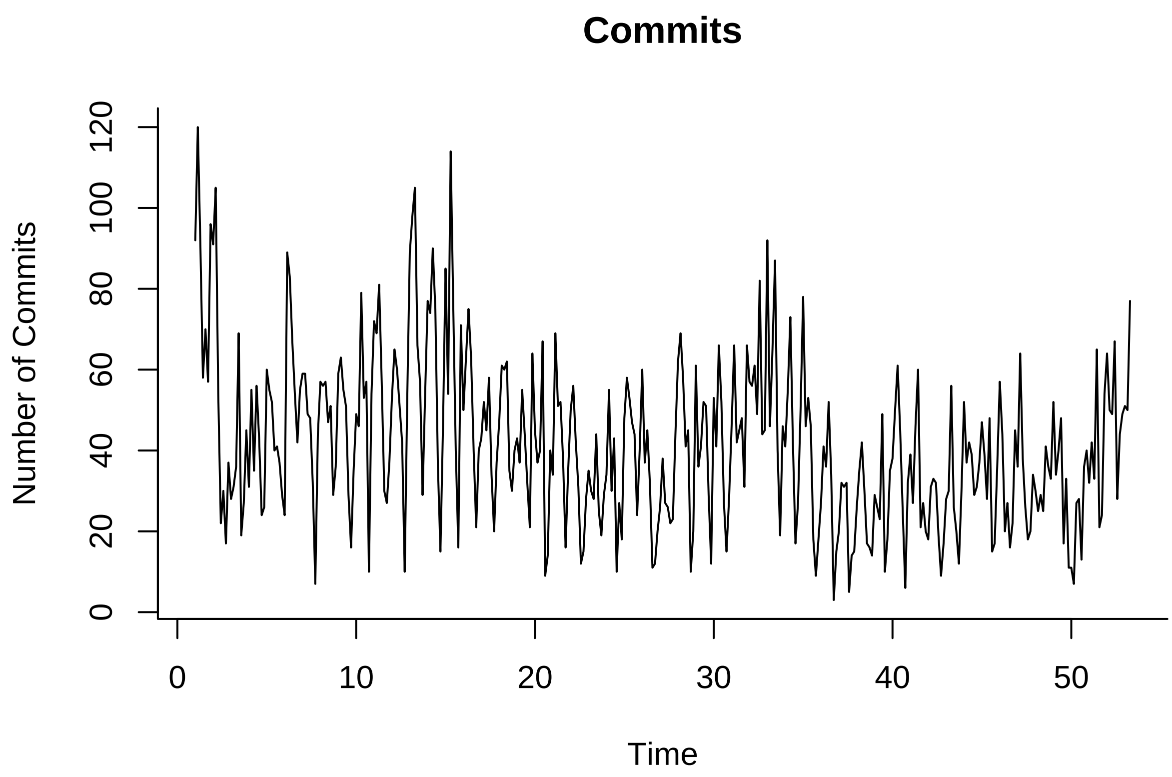
```
#loading issues and commits into a ts object
issues.ts <- ts(repository.csv$number_of_issues, frequency = 7)


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 = 'Swift 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]$train - all.forecast['fitted', i]$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'))
}
```

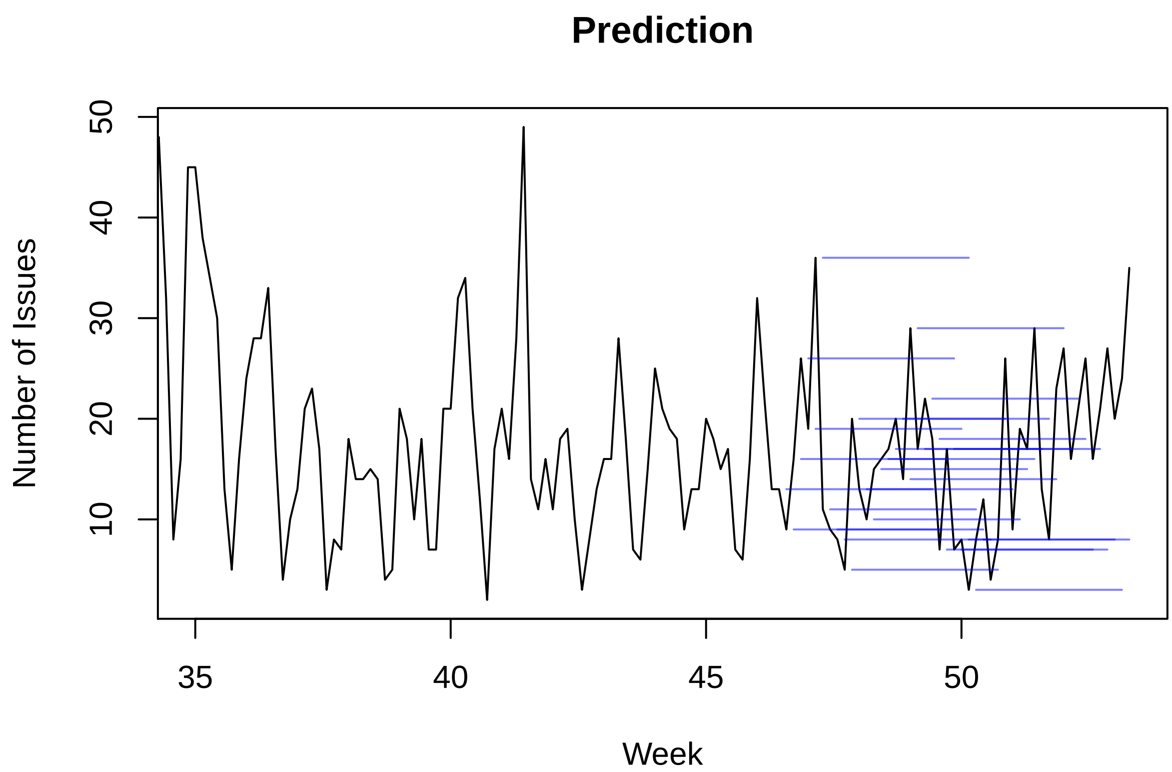
Naive Forecast

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))
```

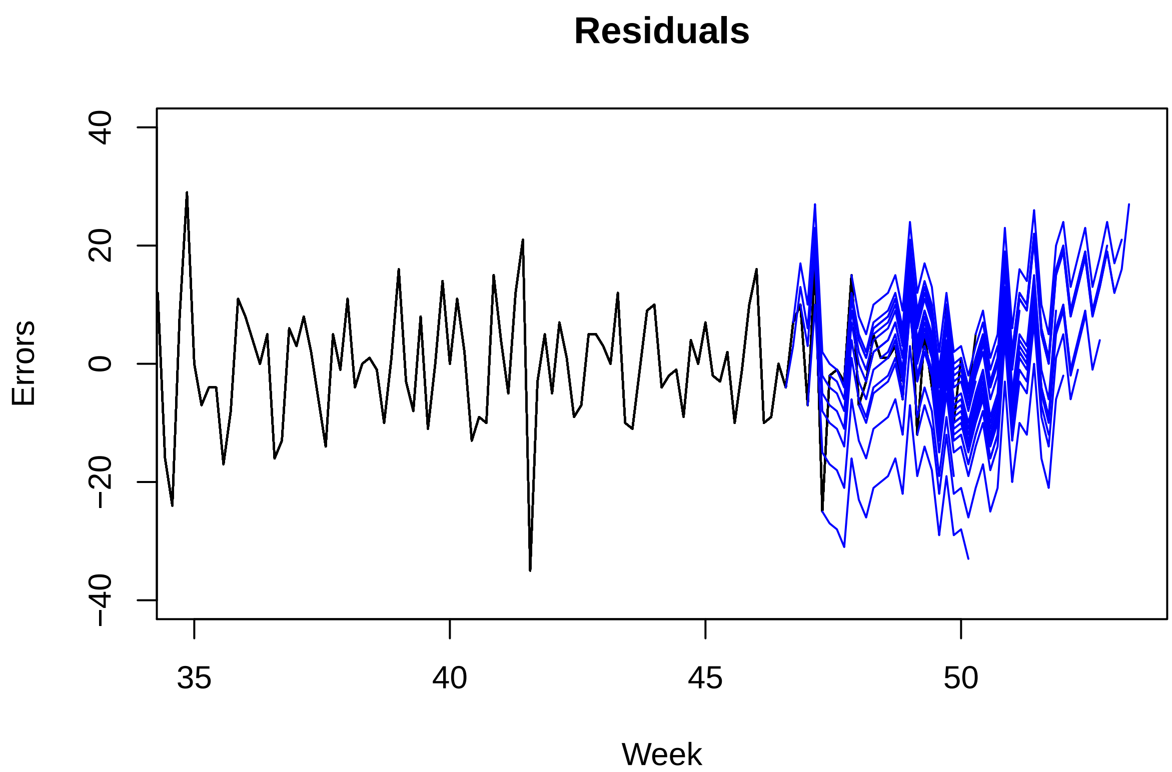
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.0368488	8.601863	6.448758	-20.43692	53.42104	0.9537619	-0.1921831	NA
Test set	0.2585034	10.215274	8.670068	-39.33113	82.14064	1.2809119	0.1848350	1.188885

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



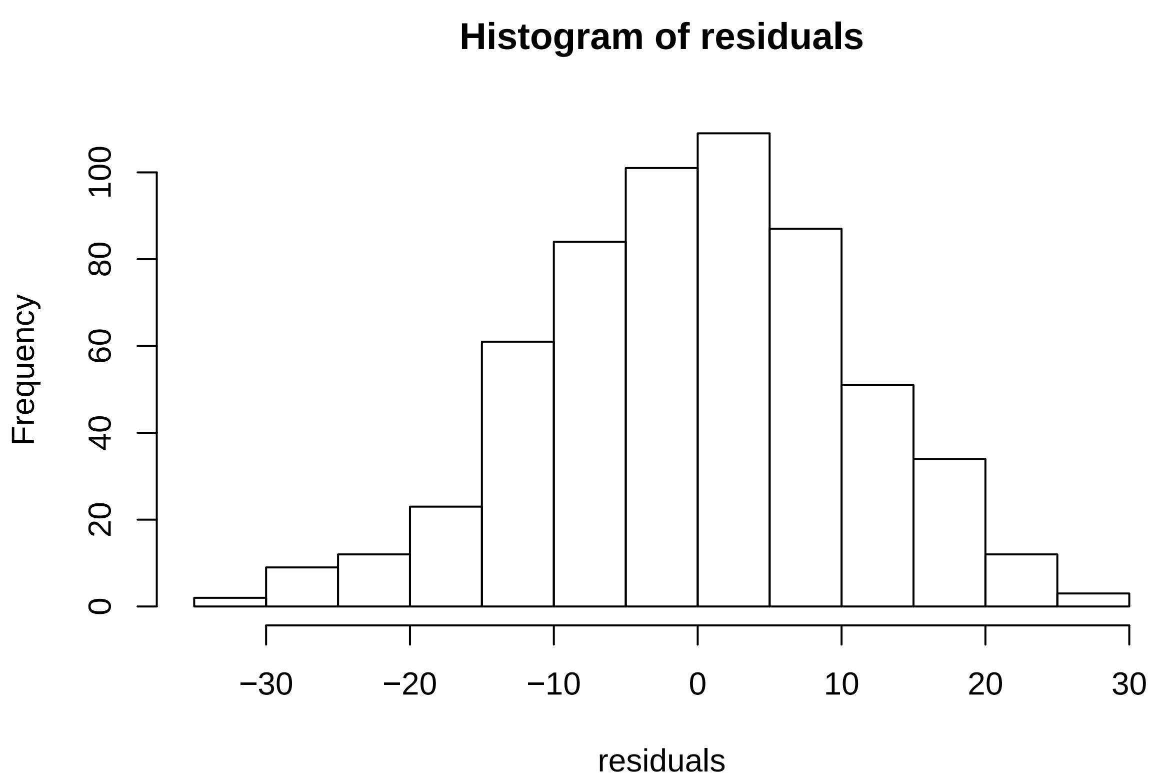

```
## NULL
```

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



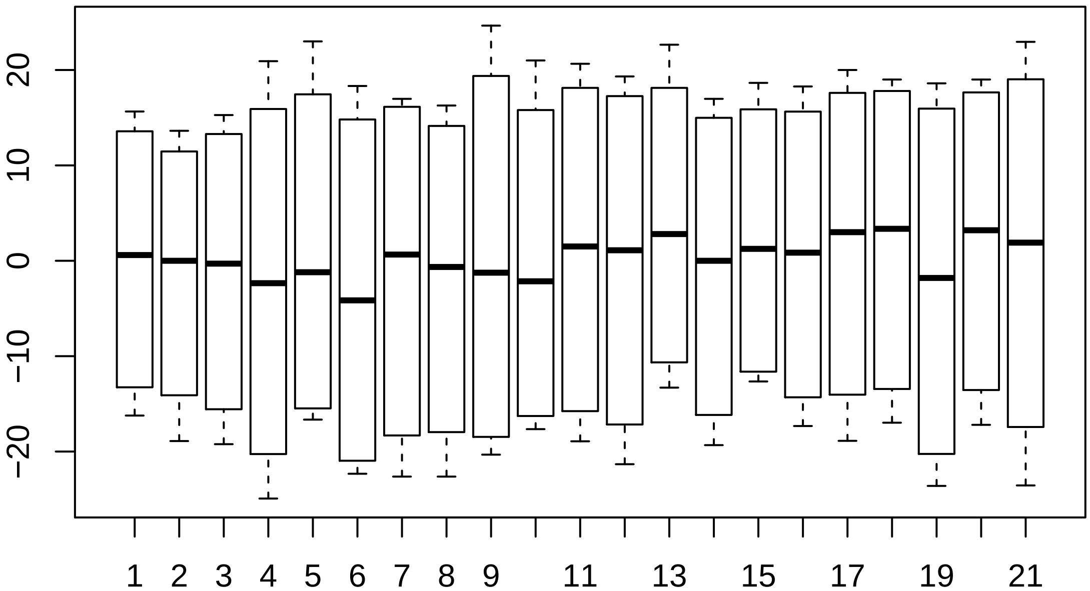

```
## NULL
```

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



```
##    97.5%      90%      10%     2.5%
##   20.325  14.000 -13.000 -21.325
```

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



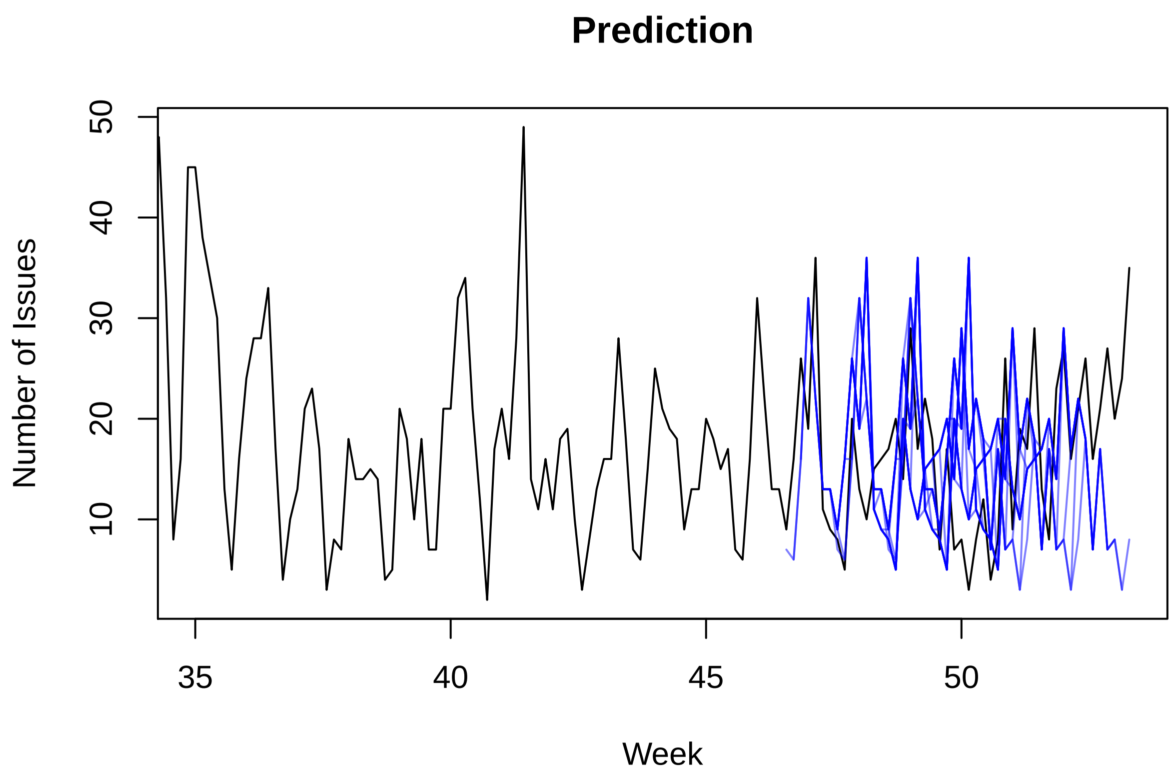

```
##    97.5%      90%      10%     2.5%
##   20.325  14.000 -13.000 -21.325
```

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))
```

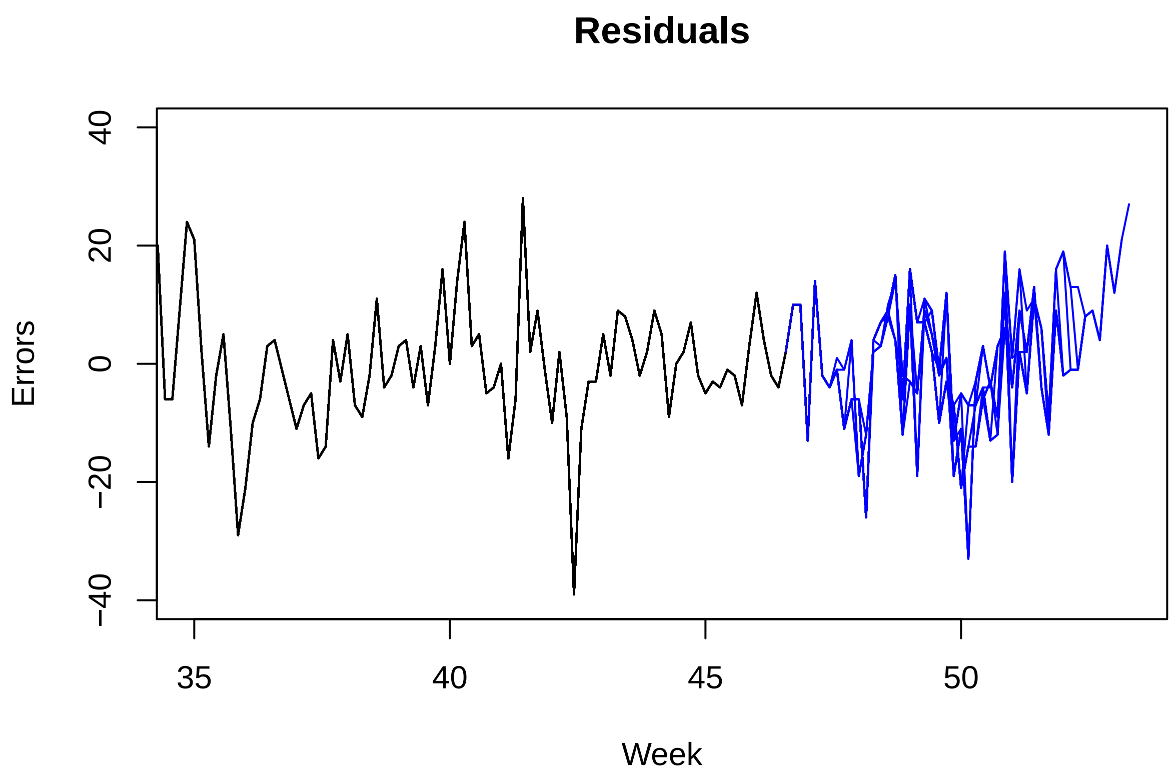
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.1922992	8.950008	6.761770	-20.86898	54.36603	1.00000	0.2962787	NA
Test set	-0.8945578	10.485400	8.615646	-50.77836	87.73850	1.27365	0.0178804	1.245005

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



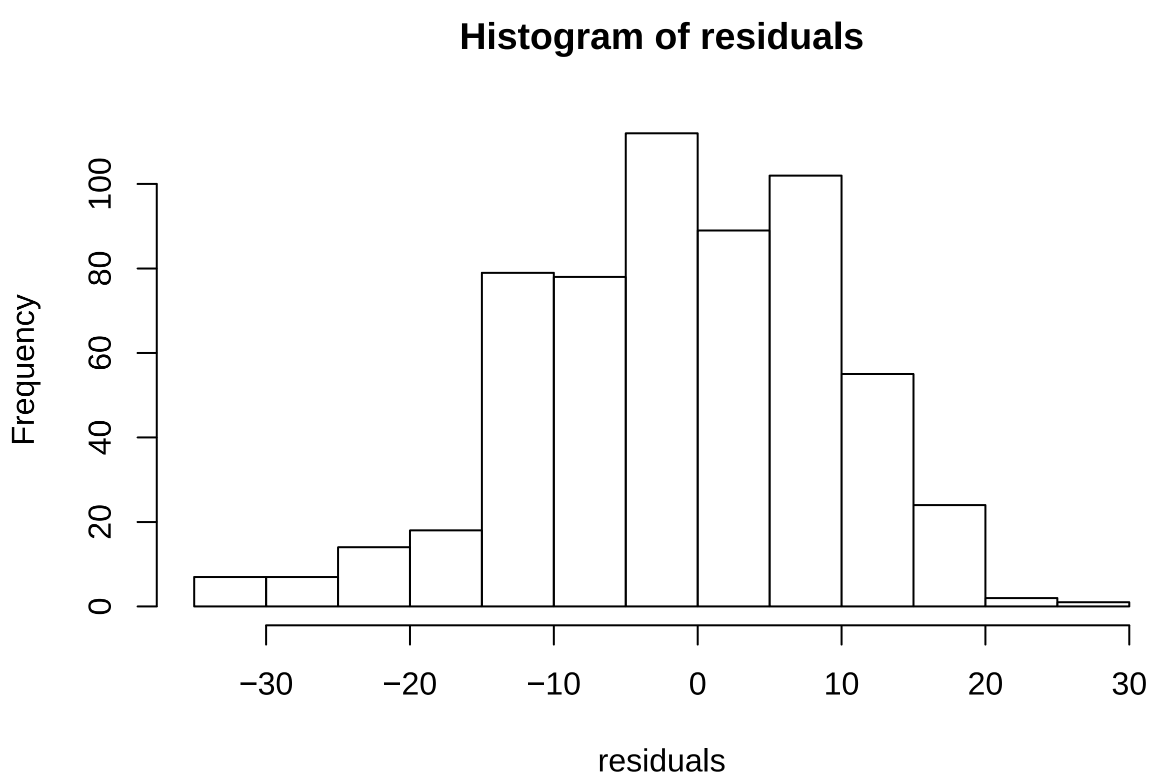

```
## NULL
```

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



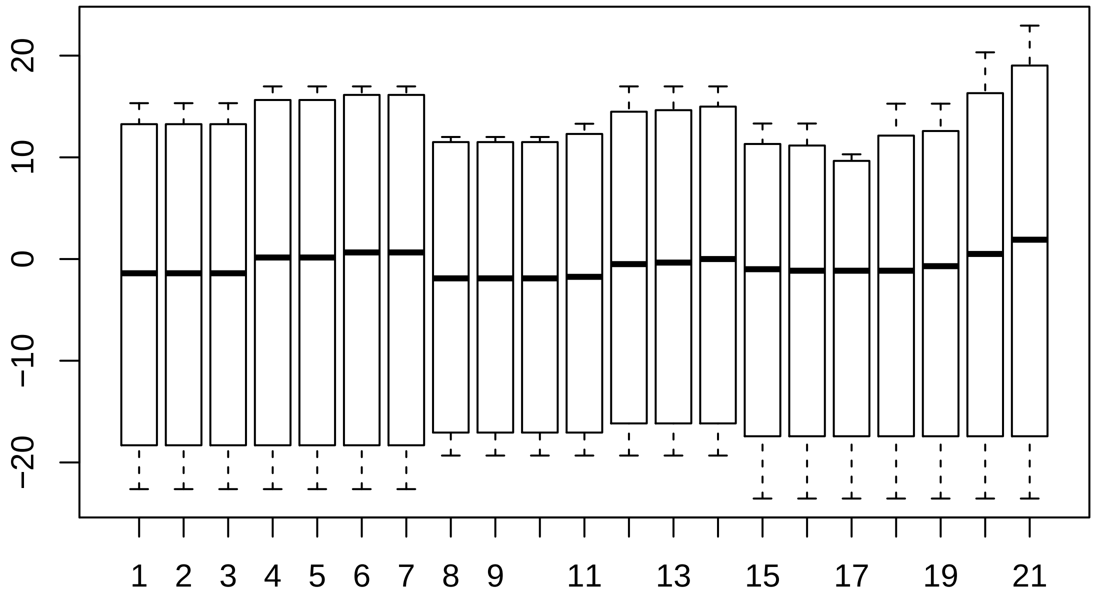

```
## NULL
```

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



```
## 97.5%   90%   10%  2.5%
##    16    12   -14   -21
```

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




```
## 97.5%   90%   10%  2.5%
##    16    12   -14   -21
```

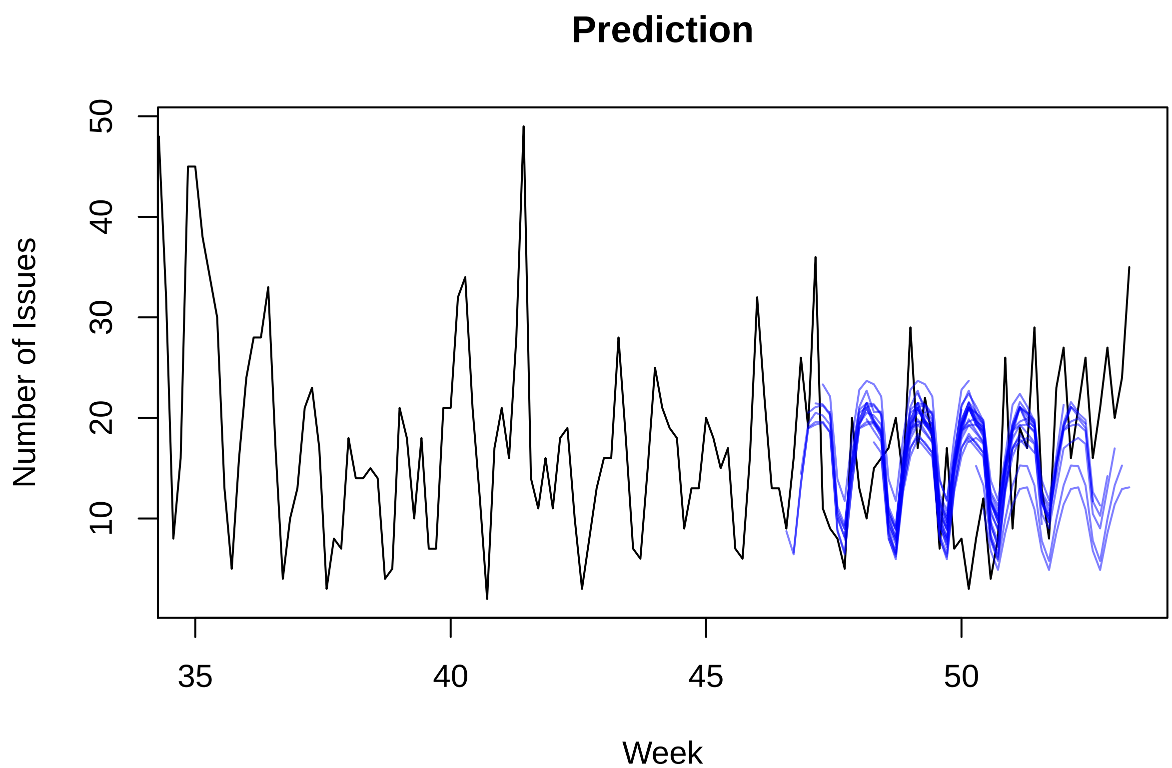
Smoothing

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 = "ZAA")
  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))
```

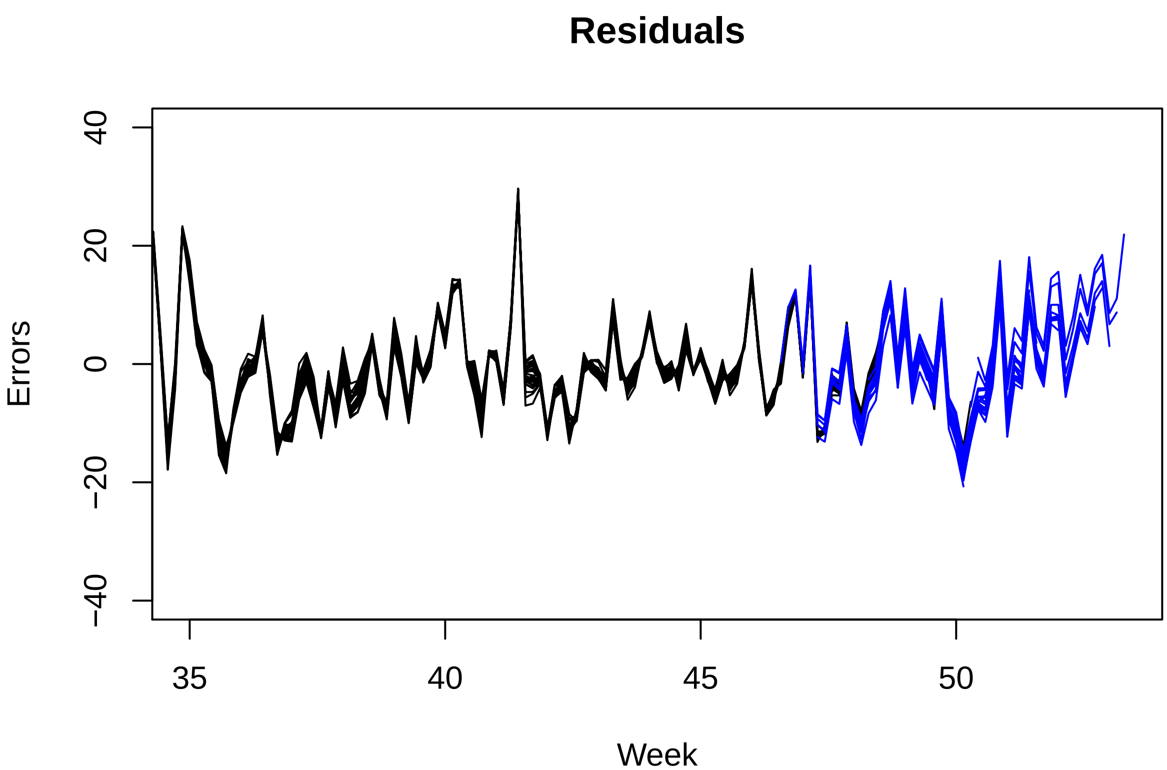
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0077434	6.724095	4.913095	-20.77182	41.83404	0.7265949	0.2487772	NA
Test set	-0.6618274	8.263658	6.881071	-43.83525	71.93529	1.0172088	0.2509245	1.157367

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



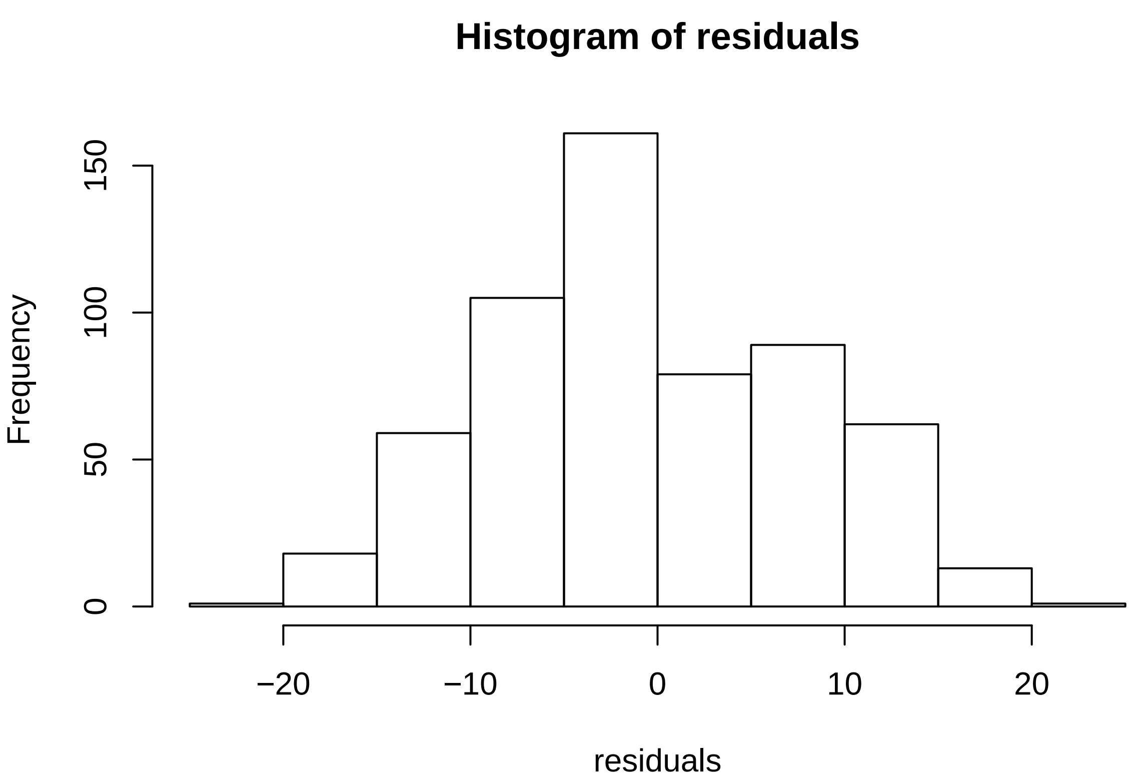

```
## NULL
```

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



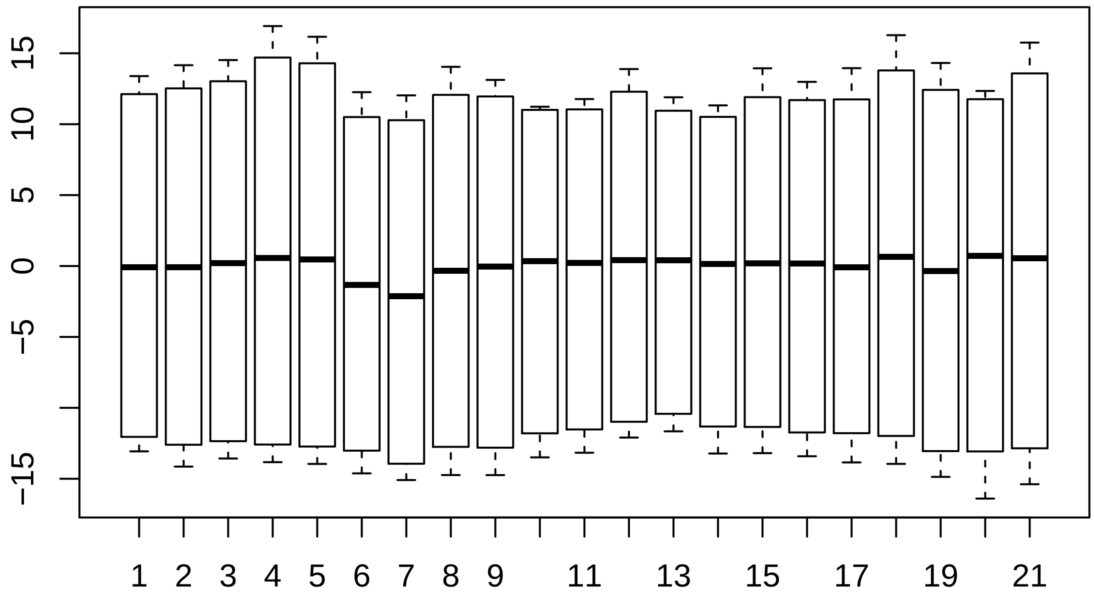

```
## NULL
```

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



```
##      97.5%        90%        10%       2.5%
##   14.67171  10.80649 -11.03764 -16.42613
```

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




```
##      97.5%        90%        10%       2.5%
##   14.67171  10.80649 -11.03764 -16.42613
```

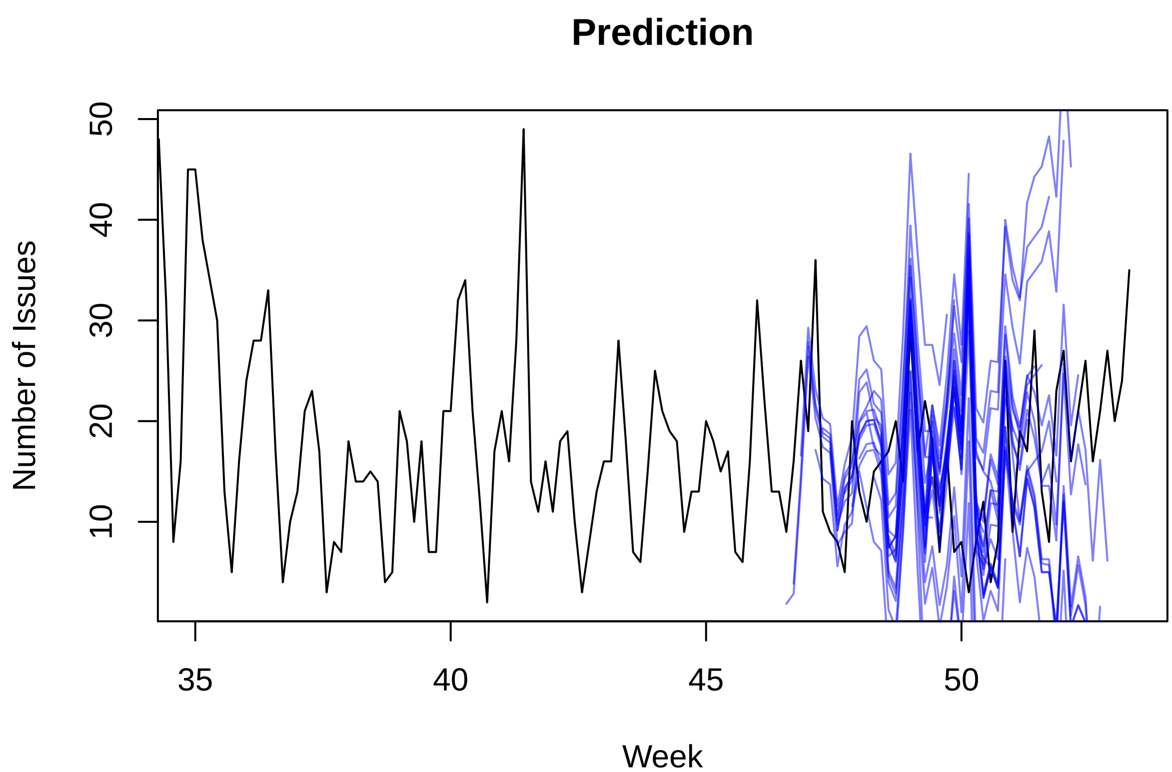
Double differencing

```
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))
```

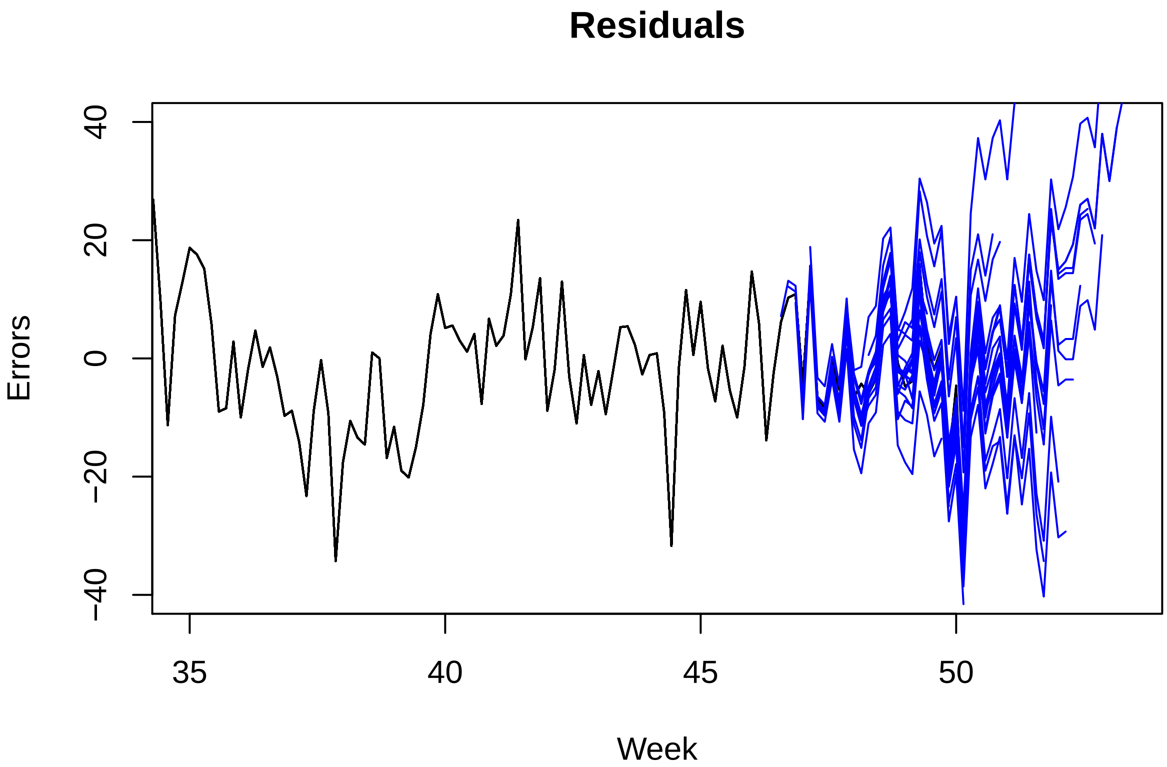
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.2957469	8.692534	6.510425	-16.52078	52.00722	0.9534403	0.3765055	NA
Test set	0.3102527	13.609635	10.848154	-48.18704	115.12905	1.5855099	0.2587500	1.522007

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



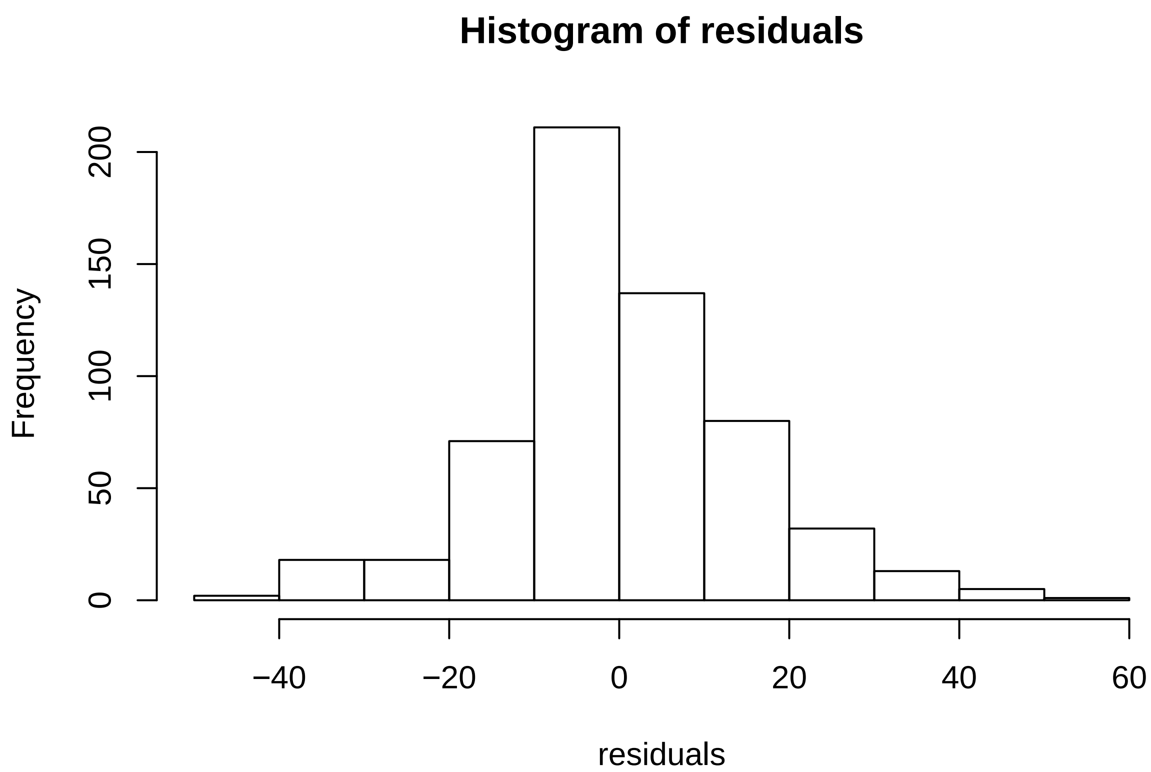

```
## NULL
```

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



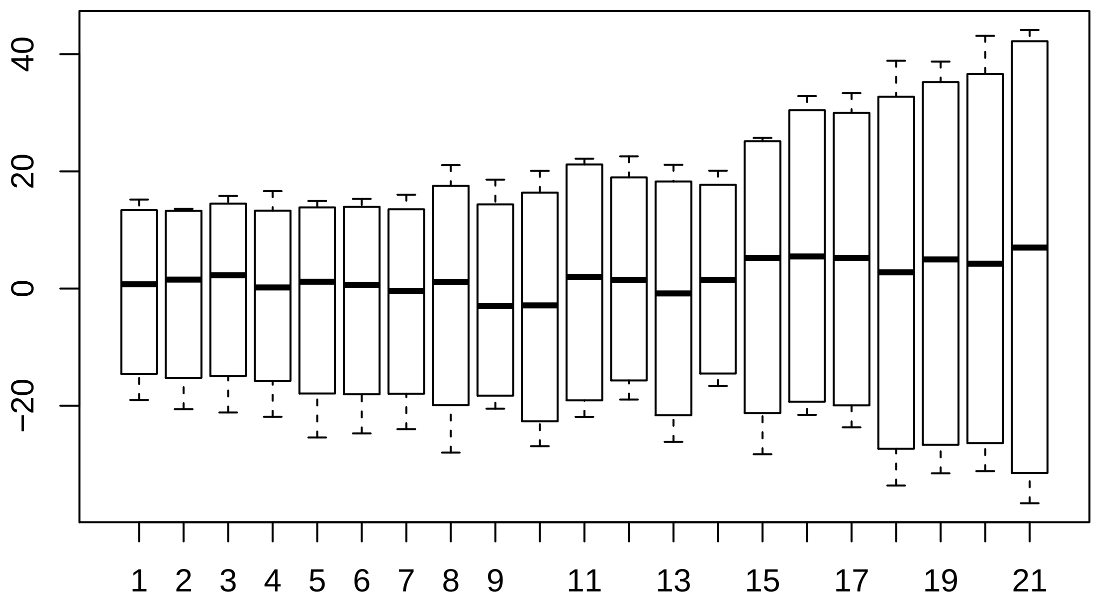

```
## NULL
```

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




```
##      97.5%       90%       10%      2.5%
##   30.52143  17.65714 -15.98571 -30.95000
```

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




```
##      97.5%       90%       10%      2.5%
##   30.52143  17.65714 -15.98571 -30.95000
```

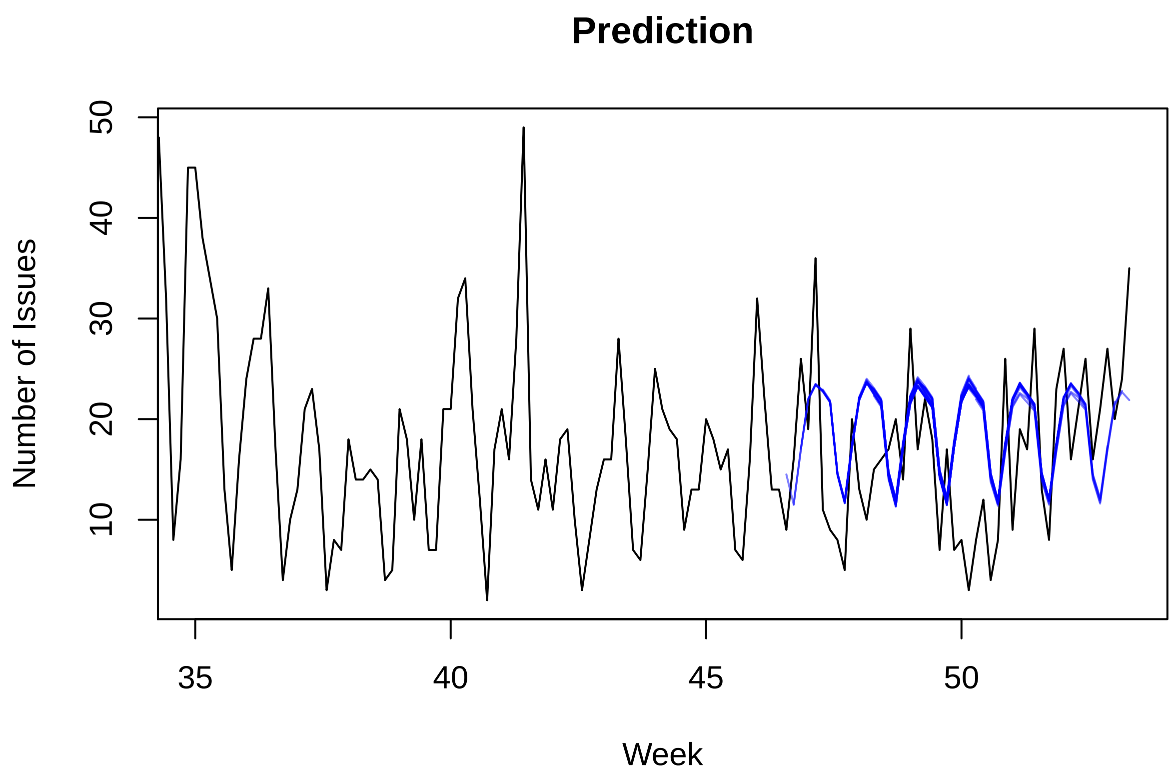
Regression

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))
```

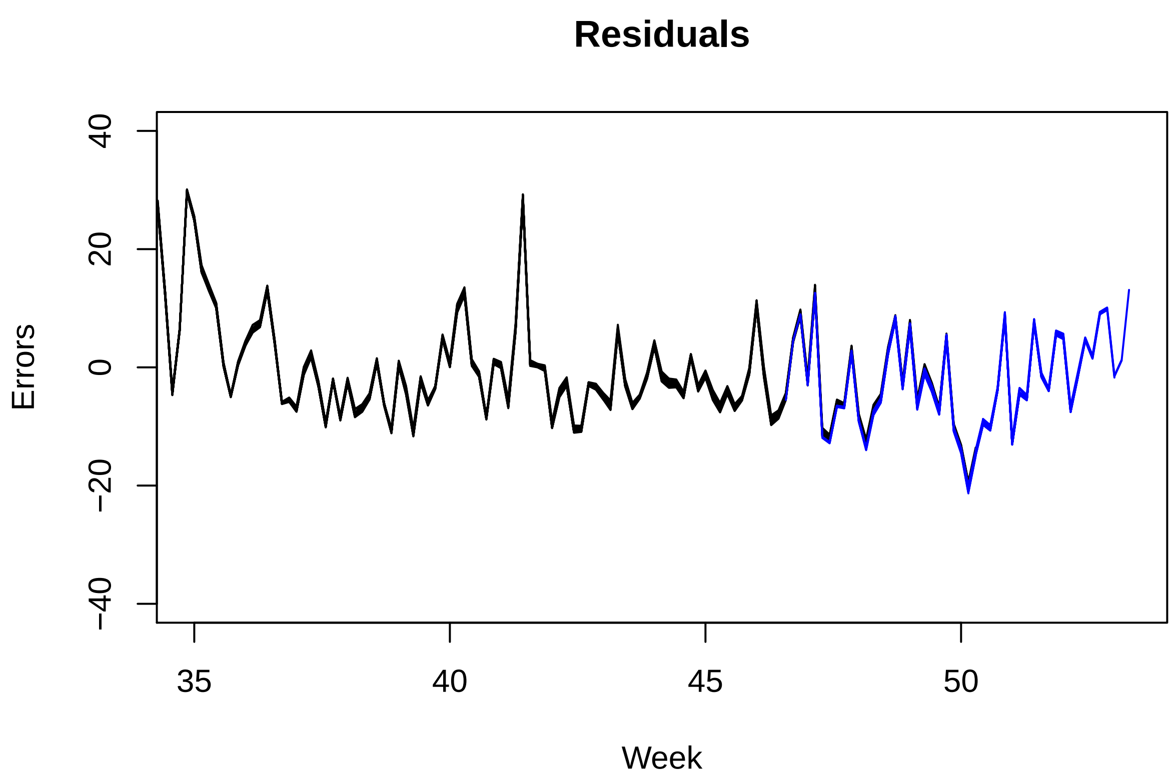
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.000000	7.239035	5.351252	-23.26457	44.73748	0.791423	0.4406588	NA
Test set	-4.000304	8.780990	7.576123	-74.54947	89.71659	1.120802	0.2471670	1.324196

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



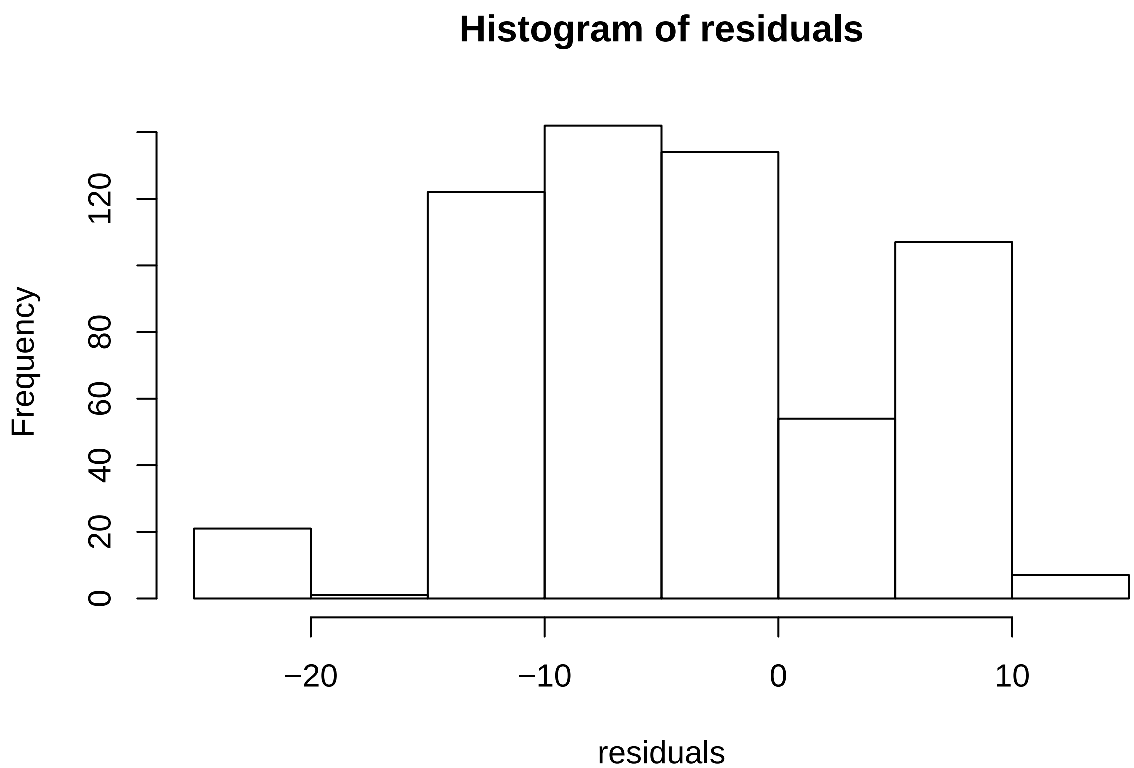

```
## NULL
```

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



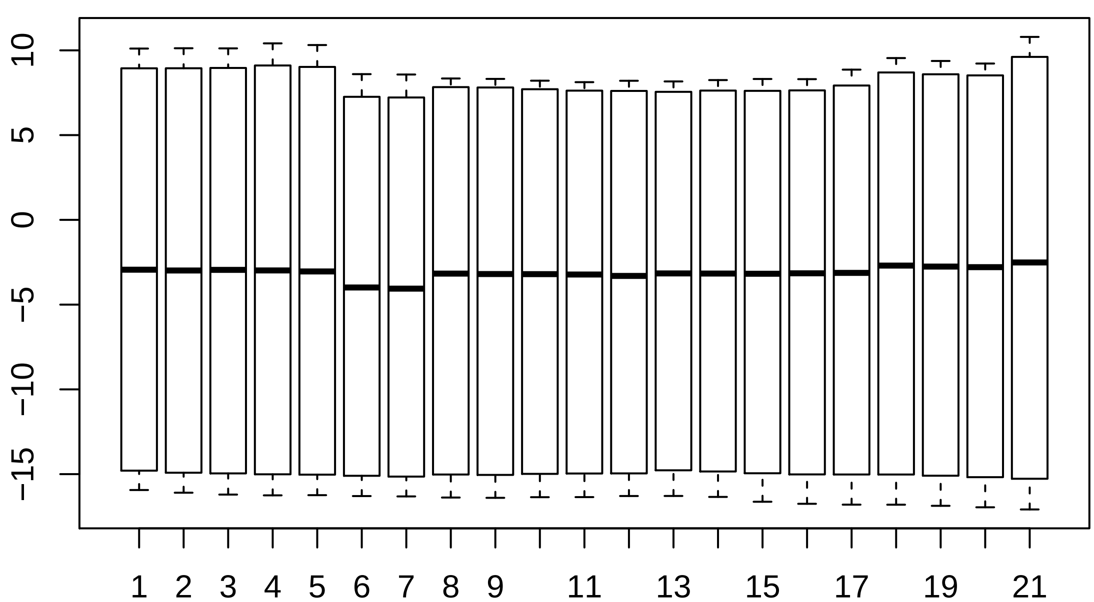

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##   9.058592   7.626598 -13.781981 -20.308204
```

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



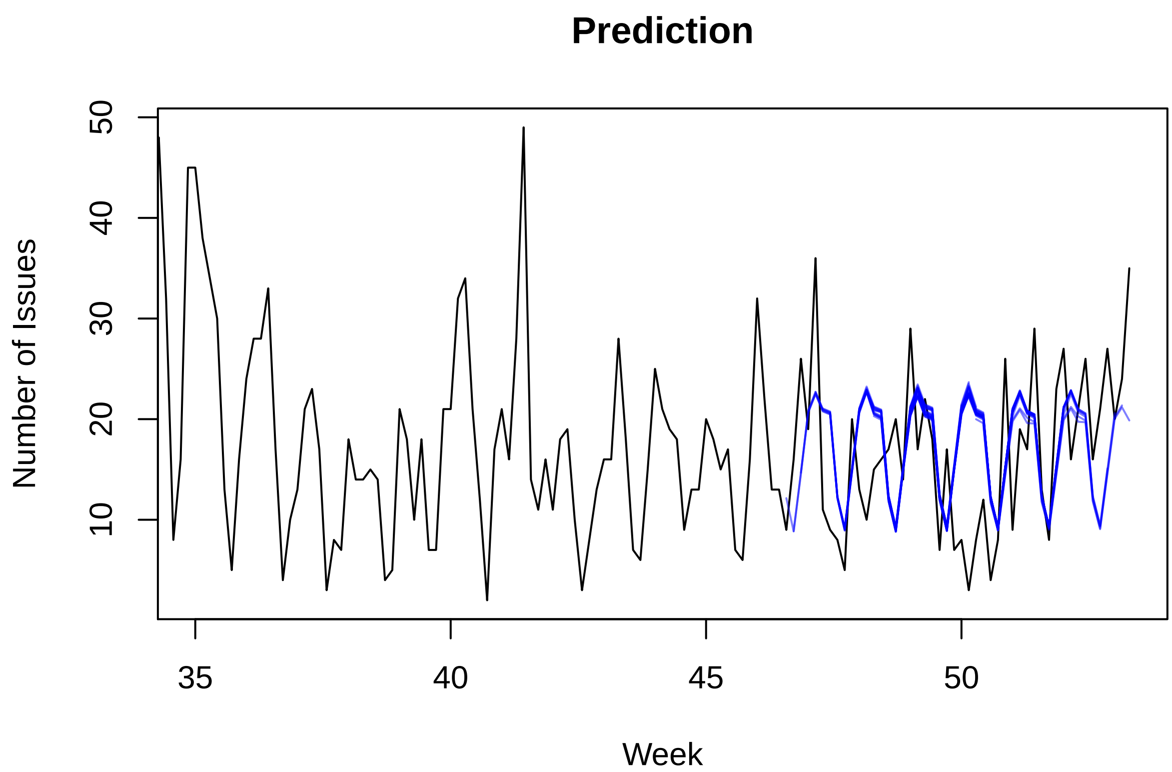

```
##      97.5%        90%        10%       2.5%
##   9.058592   7.626598 -13.781981 -20.308204
```

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 + 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))
```

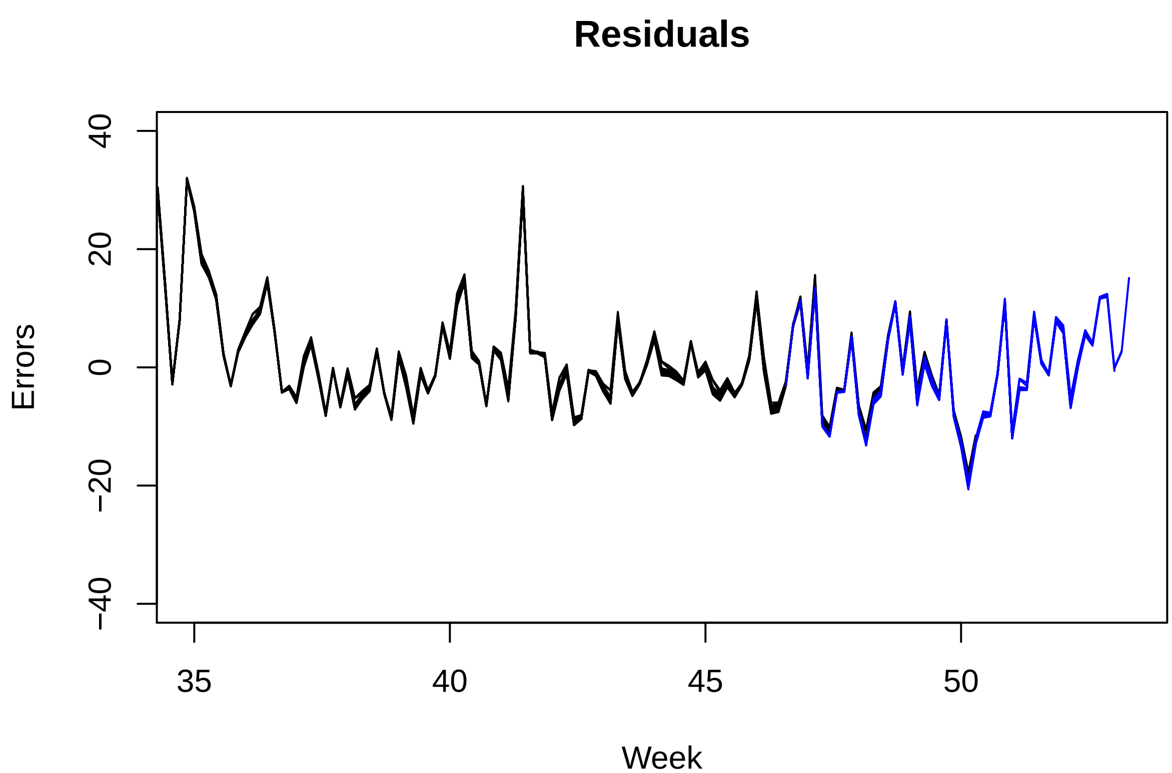
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	1.456774	7.357947	5.233878	-11.28646	39.34291	0.7740503	0.4426447	NA
Test set	-2.251212	8.388616	7.093591	-58.22024	79.51196	1.0493360	0.2639474	1.225677

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



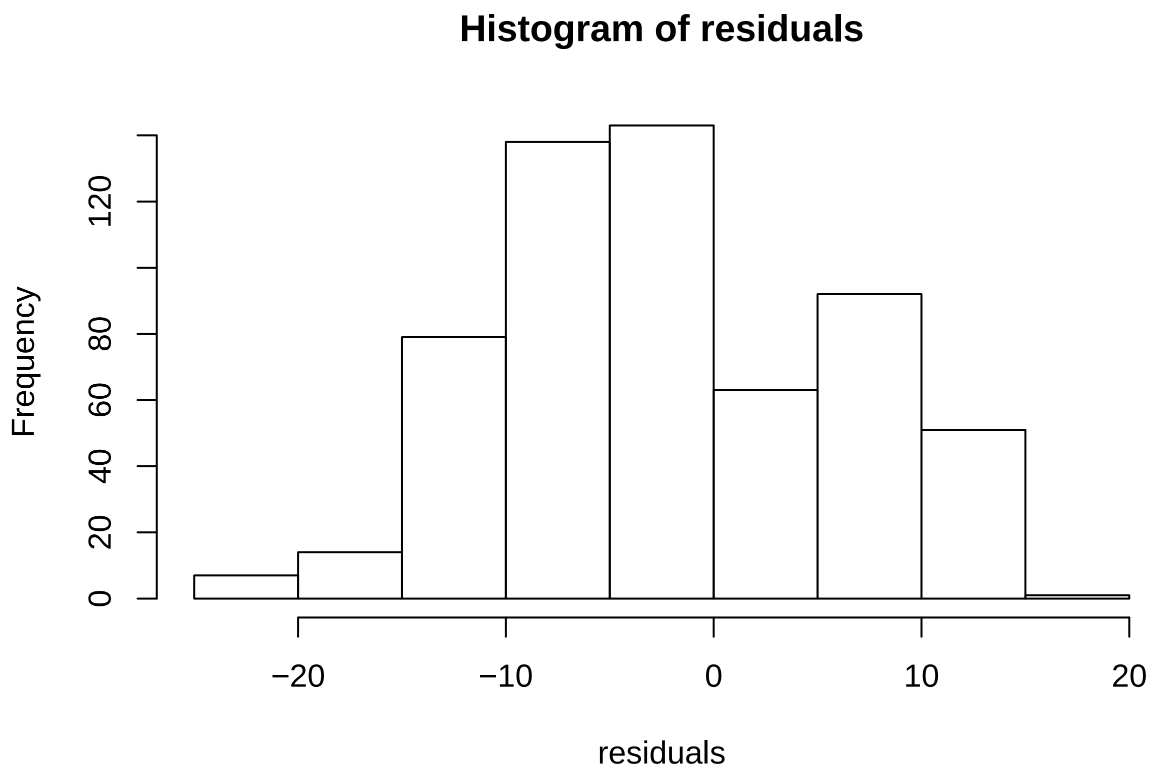

```
## NULL
```

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



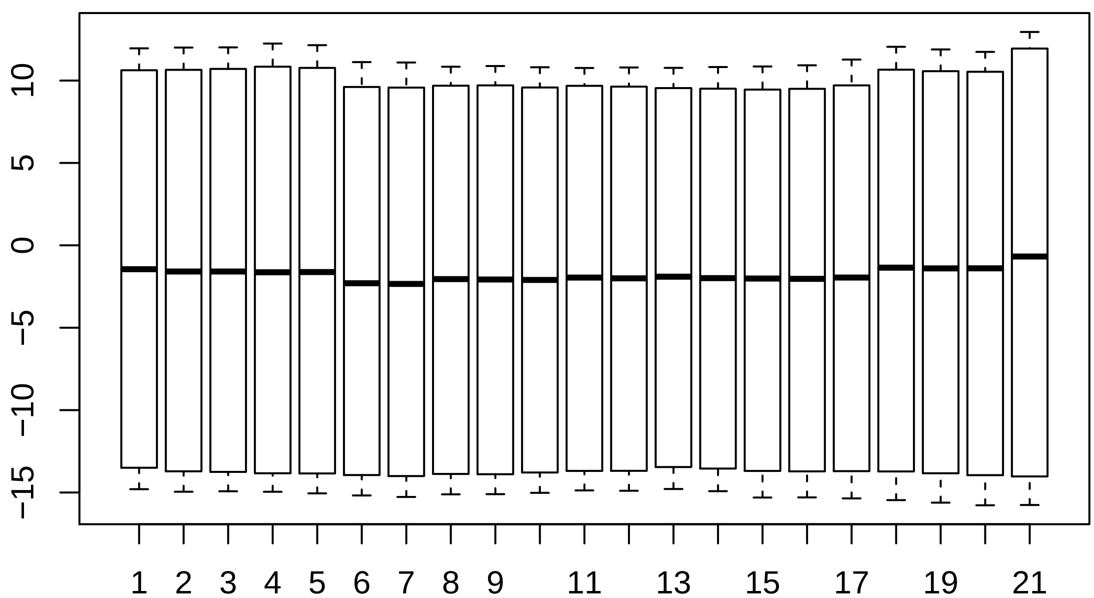

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##  11.553491   8.703288 -12.650962 -19.456856
```

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



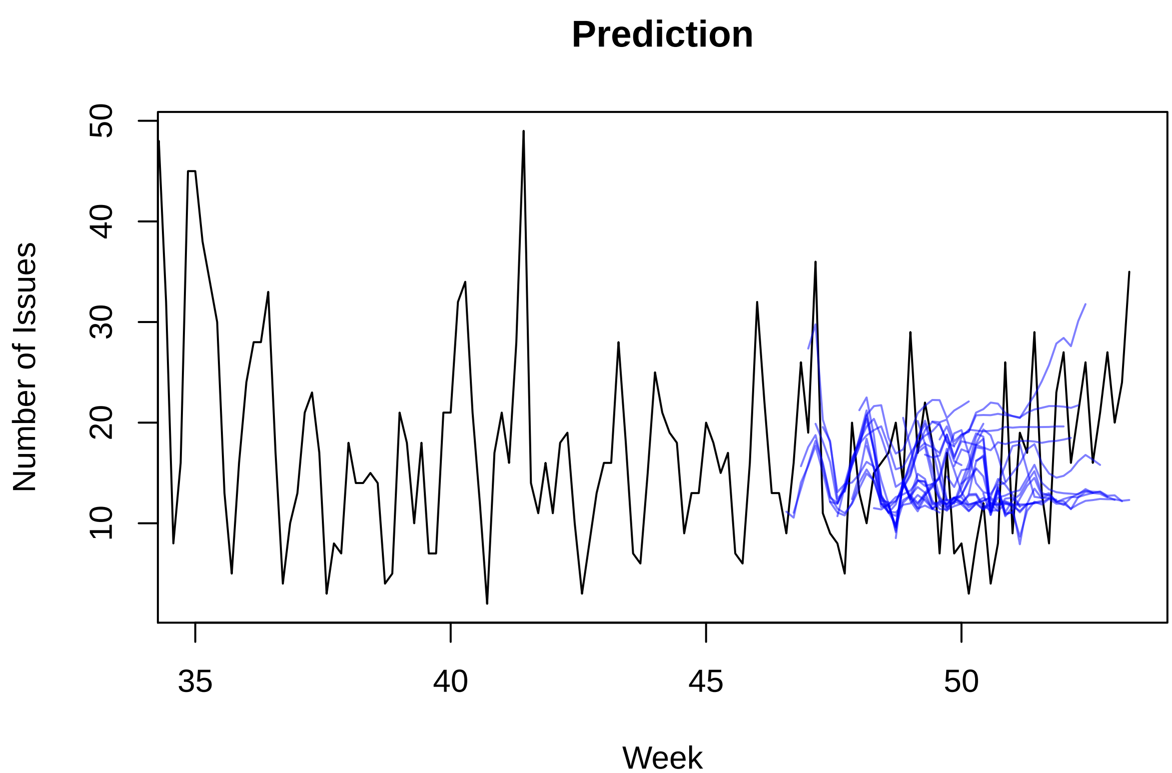

```
##      97.5%        90%        10%       2.5%
##  11.553491   8.703288 -12.650962 -19.456856
```

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))
```

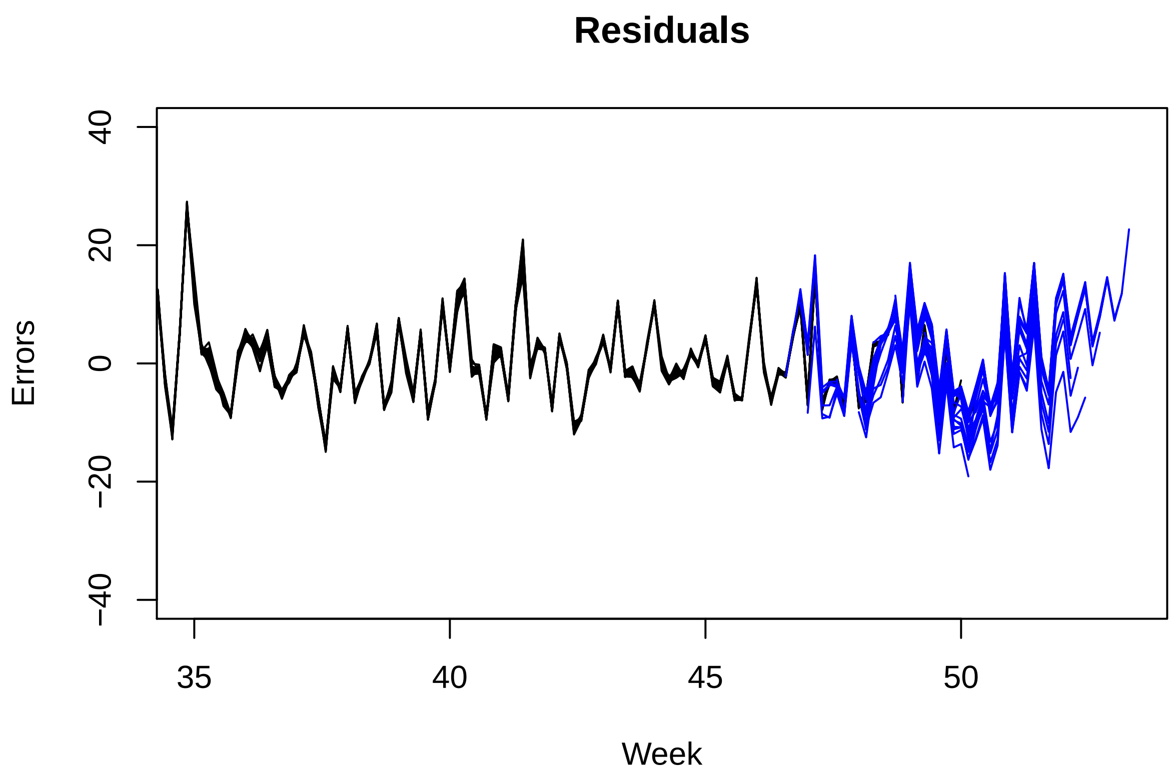
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.0024285	6.276672	4.696565	-22.72812	42.81365	0.694544	0.0564351	NA
Test set	0.0658579	8.118569	6.886711	-38.48416	68.97405	1.017479	0.1697825	0.9982837

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



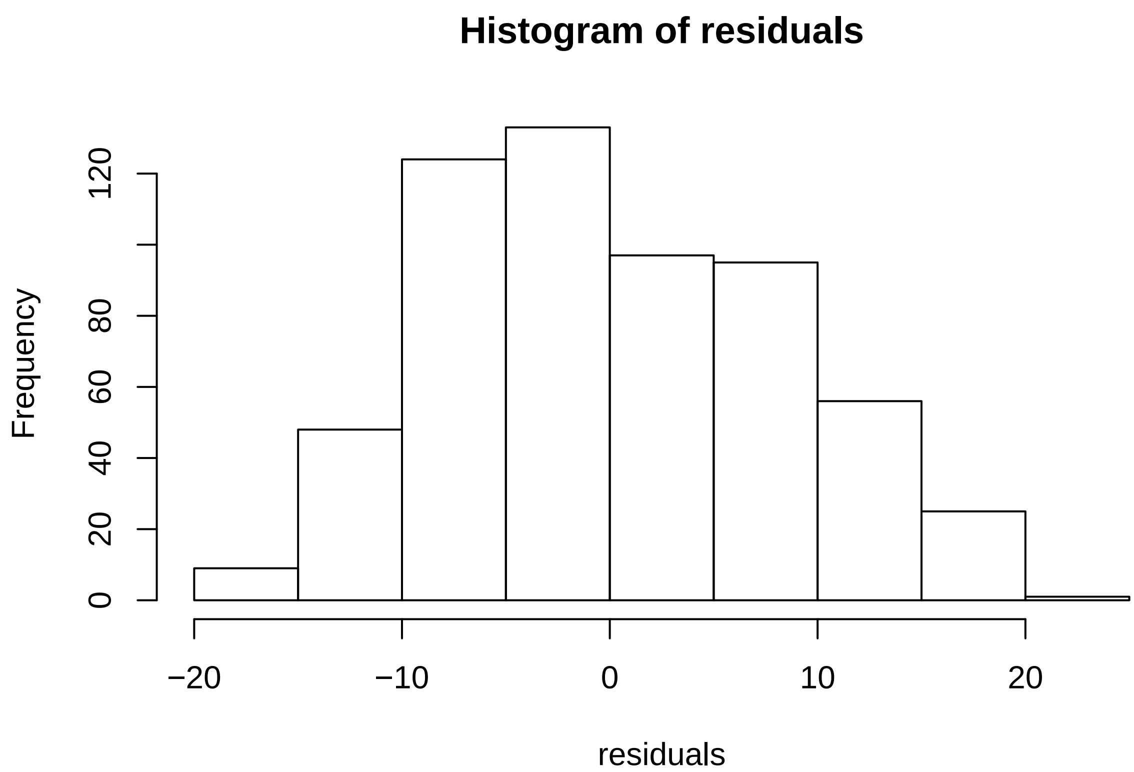

```
## NULL
```

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



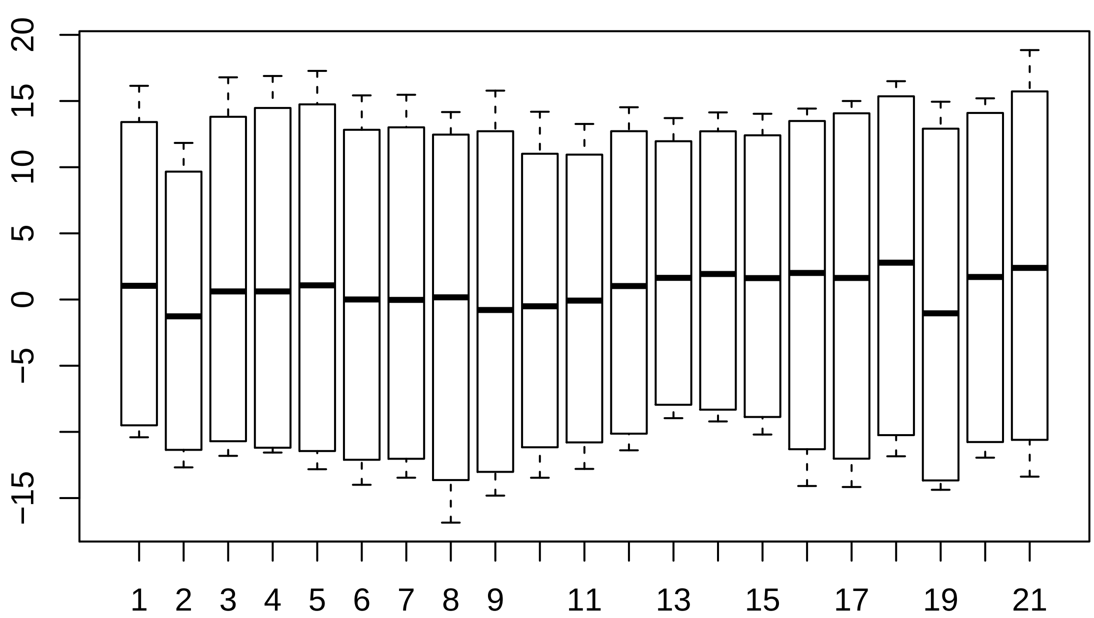

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##  16.270116  11.856330  -9.913945 -13.725596
```

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




```
##      97.5%        90%        10%       2.5%
##  16.270116  11.856330  -9.913945 -13.725596
```

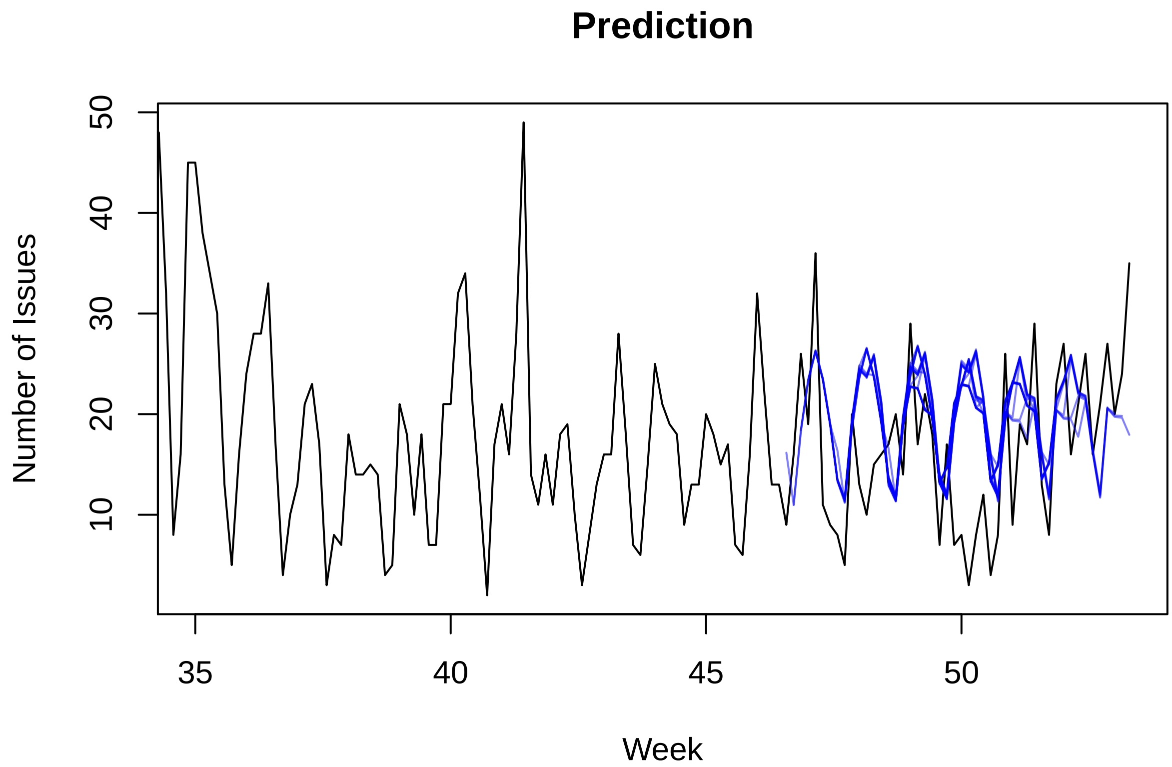
External info Numerical using regression model

```
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	6.908321	5.116976	-20.76774	42.49274	0.756512	0.2542962	NA
Test set	-4.608511	9.045002	7.619735	-79.47292	92.34019	1.127122	0.2677281	1.362637

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

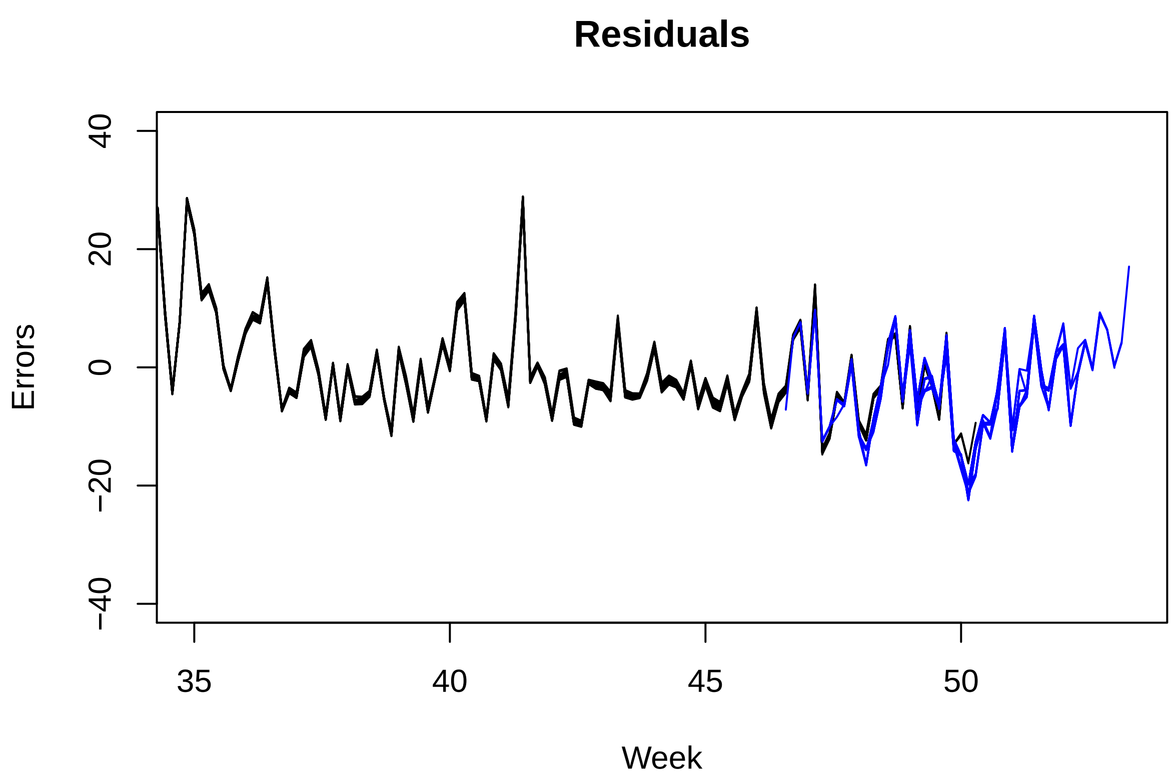
Prediction




```
## NULL
```

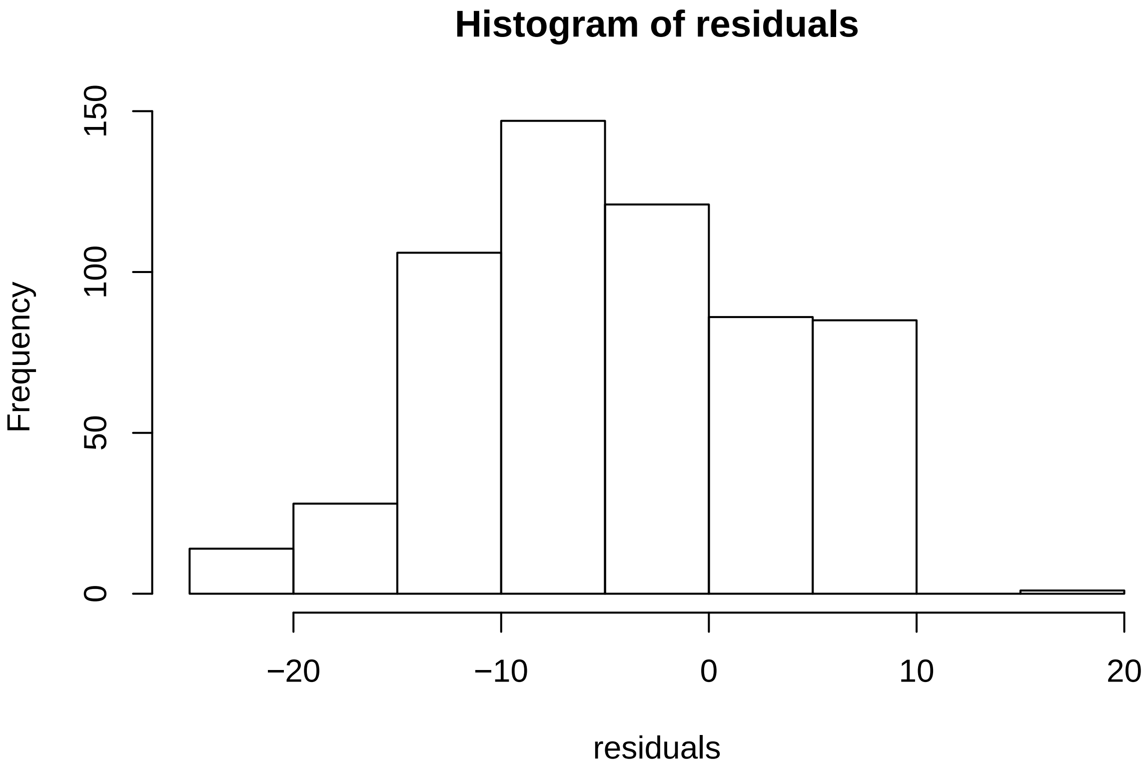
```
plot.all.residuals(all.regr.ext.forecast)
```

Residuals



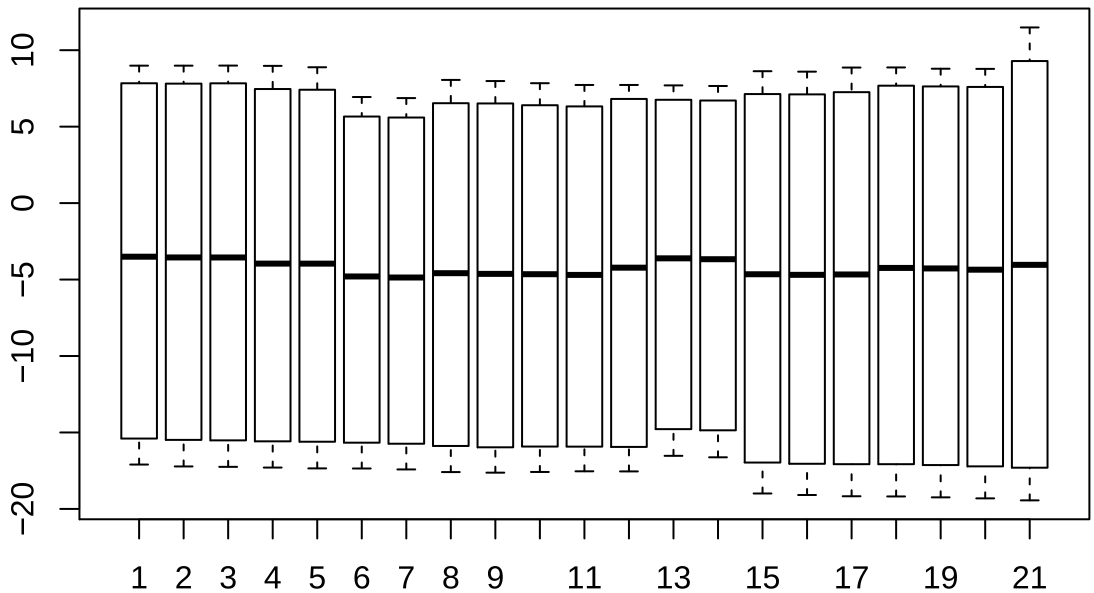

```
## NULL
```

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




```
##      97.5%        90%        10%       2.5%
##   8.706081   6.301365 -14.155572 -19.816631
```

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



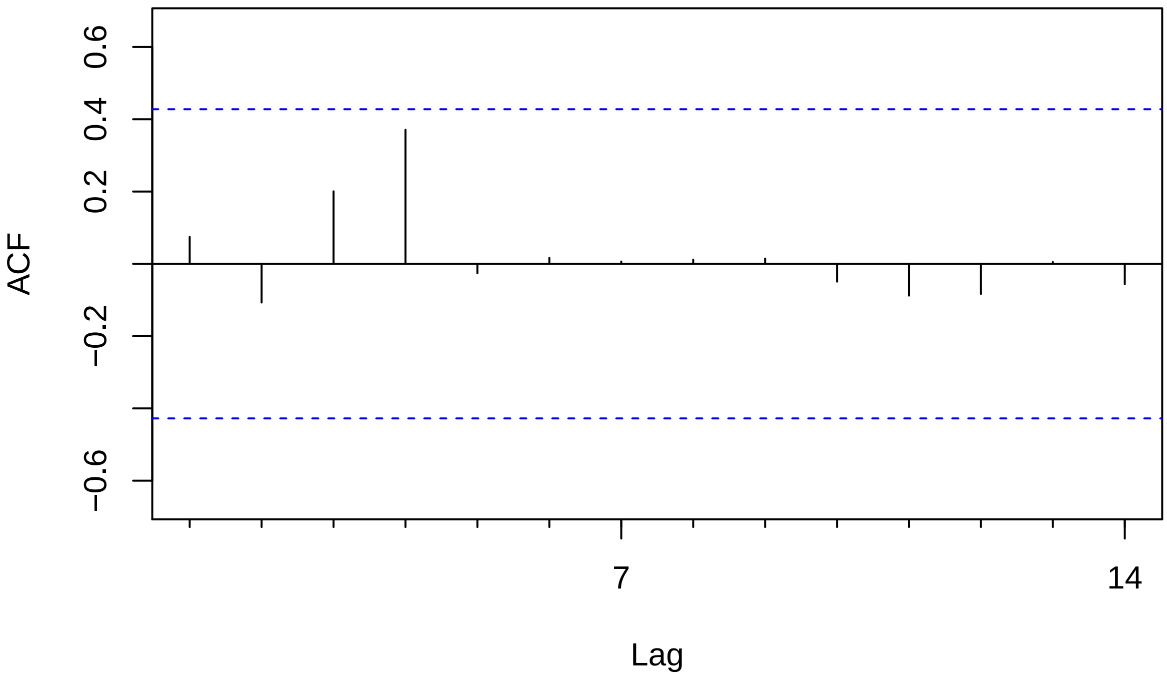

```
##      97.5%        90%        10%       2.5%
##   8.706081   6.301365 -14.155572 -19.816631
```

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))
```

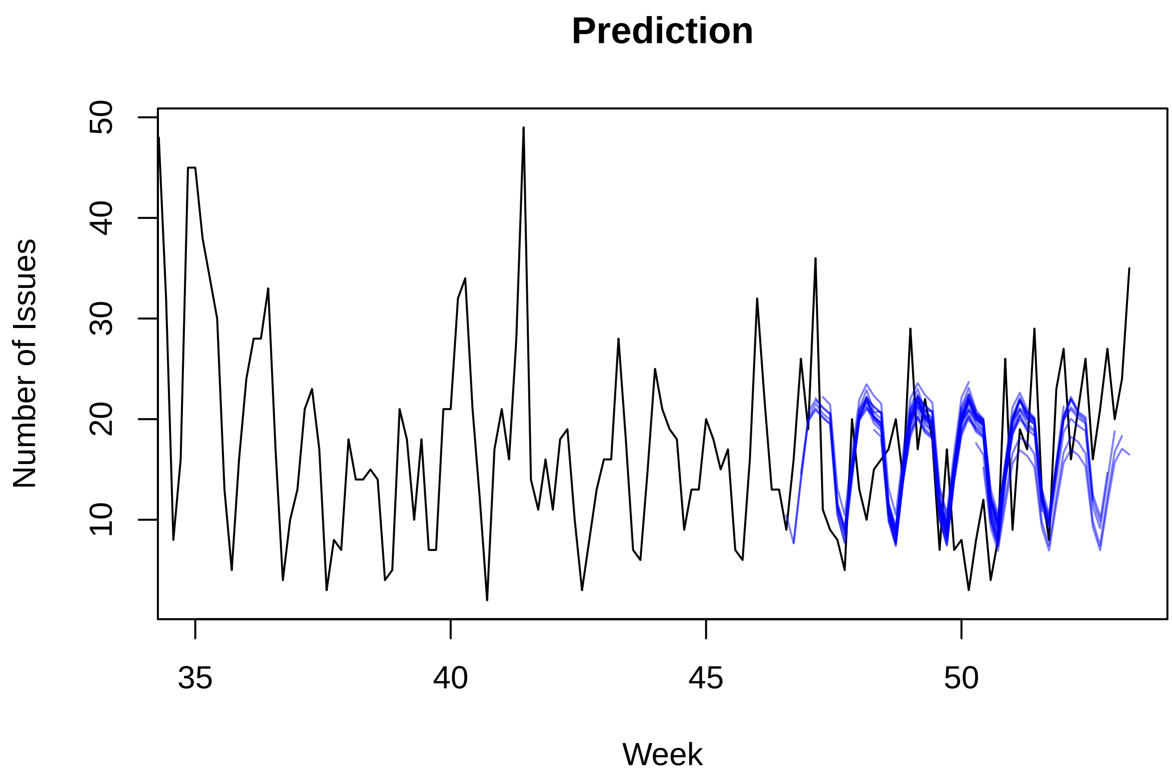
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.7785853	9.954234	7.573165	-33.55217	67.83042	1.109153	0.4841104	NA
Test set	-1.4565195	8.249646	6.923837	-51.02774	75.24937	1.013901	0.2578636	1.185365

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



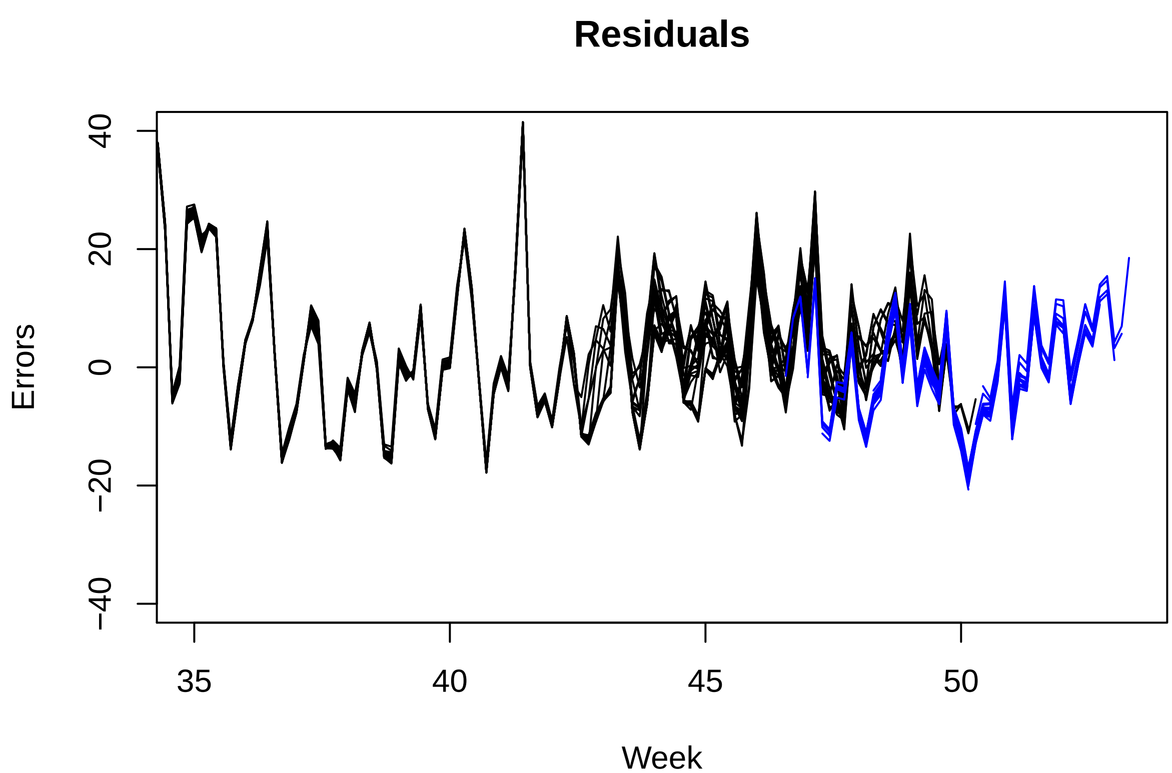

```
test <- list()
test$train.ts <- issues.ts
test$valid.ts <- naive(issues.ts, h=n.valid)$mean

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)
```



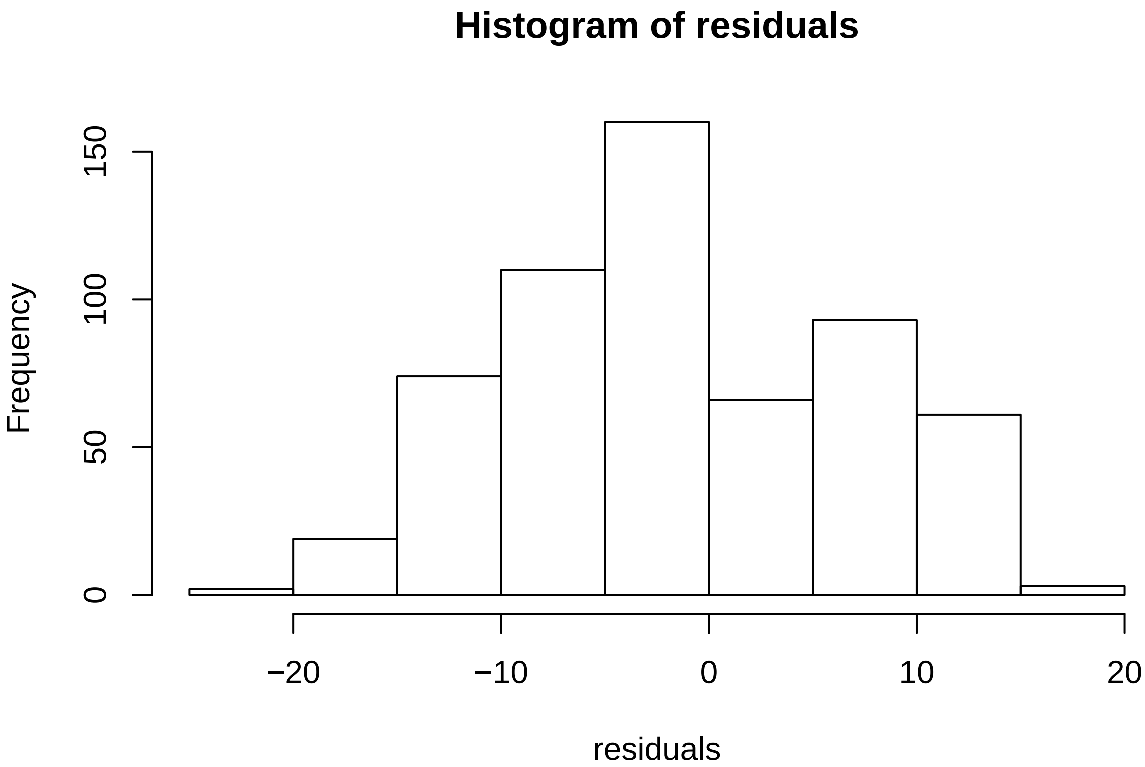

```
## NULL
```

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



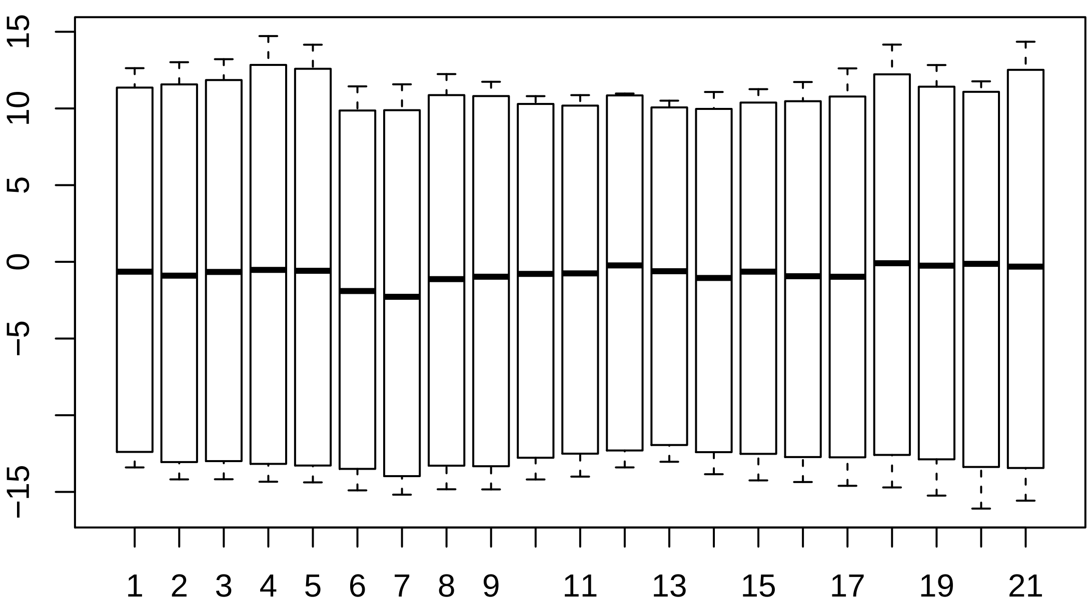

```
## NULL
```

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




```
##     97.5%       90%       10%      2.5%
##  12.53580  10.33512 -11.73825 -17.92391
```

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




```
##     97.5%       90%       10%      2.5%
##  12.53580  10.33512 -11.73825 -17.92391
```

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