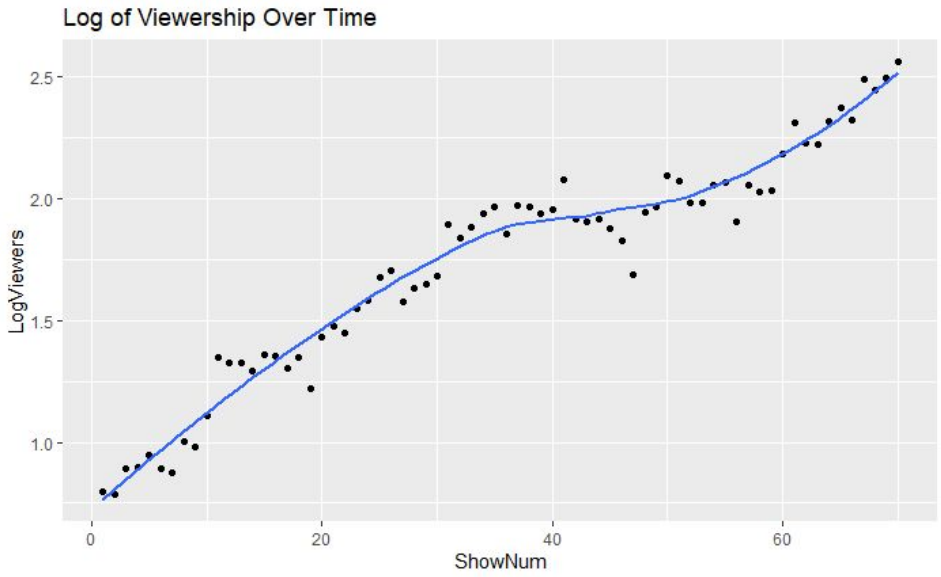
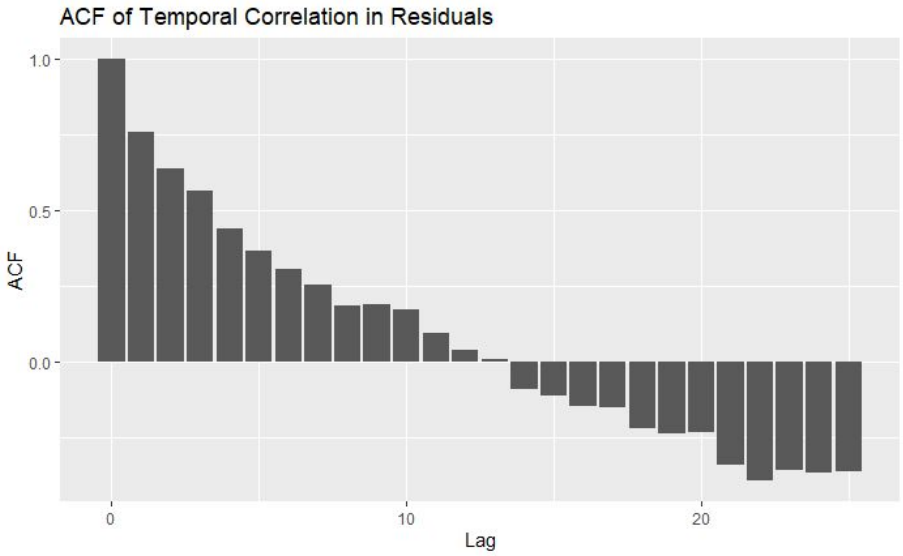
Jacob Andros & Jess Powell

Homework Analysis #3 - TV Viewership

1. The correlation coefficient between the show number (which can be understood as the point in time) and the log of viewership is .956. This indicates that there is a strong, positive, relationship between time and the log of viewership. It also means that according to our data, 91% of the variation in the log of viewership can be explained by time, or show number. The plot below is further evidence of this, and shows that the relationship is approximately linear, although it does look like there may have been a small dip around show 50.




2. We fit a linear model to this data, and it indicated a steady, positive relationship between time and log of viewership. However, we also diagnosed a clear temporal correlation in the residuals of that linear model using a plot of the ACF (auto-correlation function). The plot below shows the strength of the correlation (less than 1) between each point's residual as a "current point" and the future points' residuals. We can see that even after half a entire season (5 shows), there is still a correlation of almost 0.5, and it takes about 12 shows for that correlation to die out.




Since there is a temporal correlation that will needed to be accounted for in the residuals of our model, we conclude that viewership (or the log thereof) is a variable that has momentum - if one episode is really good, then more people are likely to watch the following one, and vice versa. For this reason, we will use a time series model.

3. The seasonal term in this model will have a length of 10, since each season of shows is 10 episodes long. Using the auto.arima() function in R, we determined that the best-fitting time series model would have the following values:

| p = 2 | d = 0 | q = 0 | P = 0 | D = 1 | Q = 1 |
|---|---|---|---|---|---|

The auto.arima() function in R selected these values using a variable selection algorithm. It examined all possible combinations of parameters where d was set to 0, D was set to 1, and the p/P/q/Q parameters were set to be somewhere from 0 to 2. It then chose the model which minimized AIC, and this will help optimize the model for prediction.

4. Our time series model specification is as follows, with the SARIMA parameters following the same order as above:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$
$$\boldsymbol{\epsilon} \sim \mathrm{SARIMA}(2, 0, 0, 0, 1, 1)_{10}$$

The matrix **y** is a vector containing each show's observed log(viewers) value. Since there are 70 shows in this dataset, **y** is a vector of 70 observations, or a 70 by 1 matrix. **X** is a transposed matrix containing each show's number, or each show's explanatory variable, and placeholders for the y-intercept. **β** is a vector containing the slope coefficient estimate for show number and the y-intercept. When we multiply **X** by **β**, the

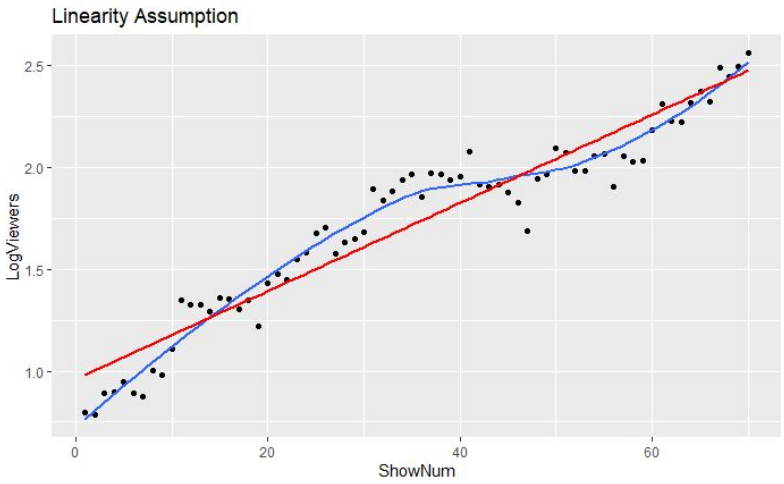
resulting output is a 70 by 1 vector. To get **y**, we add the residual quantities to this vector. Epsilon is a residual or error term for each observation, and it is through these error terms that we will account for temporal correlation between shows.

A time series model will account for this correlation and allow us to determine, through the selection of the parameters (p, d, q) and (P, D, Q), how close and how strong these correlations are. It will allow us to make inferences in deciding whether it would be profitable and beneficial to continue the show (produce another season) by providing predictions of what the log(viewership) of those episodes would be.

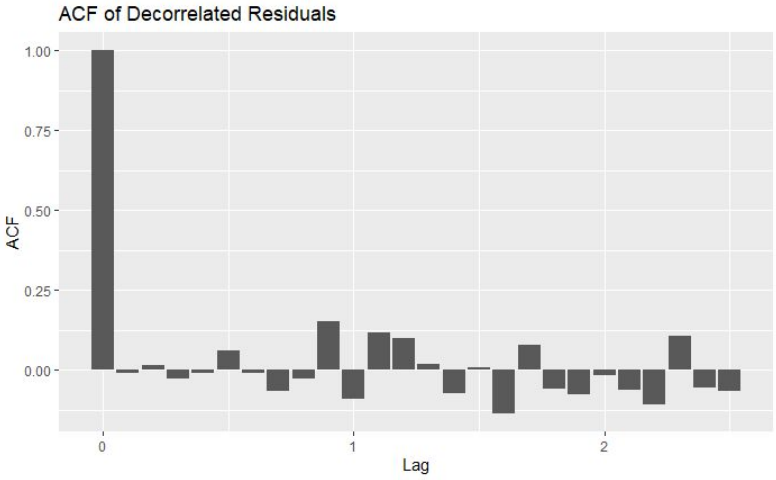
- With the parameter p at 2, we can infer that there is a slowly decaying show-to-show correlation.
- With the parameter d at 0, we can conclude that the differencing in time between shows is not an important variable in this model.
- With the parameter q at 0, we know that there is not any extremely tight adjacent correlation in shows. Although the correlation between shows lasts a long time (it doesn't decay quickly because of p=2), q=0 shows that this correlation begins its decaying process immediately.
- With the parameter P at 0, we infer that there is no progressively decaying season-to-season correlation over time. However, since Q is 1, this means that there is a tight correlation between seasons adjacent in time.
- With the parameter D at 1, we know that the differencing in time between seasons is an important variable in this model.

5. The four assumptions of the model are met as follows:

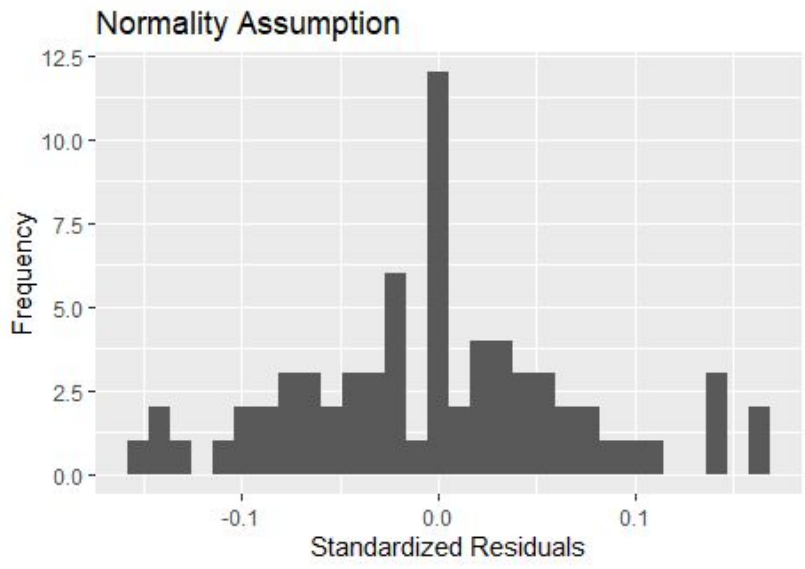
**Linearity:** The exploratory plot of the relationship between show number (time) and log of viewership showed an approximately linear trend. Below is a copy of that plot, this time not only with a smooth curve (blue), but also with a straight line (red). Though there is a temporary dip in log of viewership, it seems that this relationship is linear enough for the model assumption to be met.



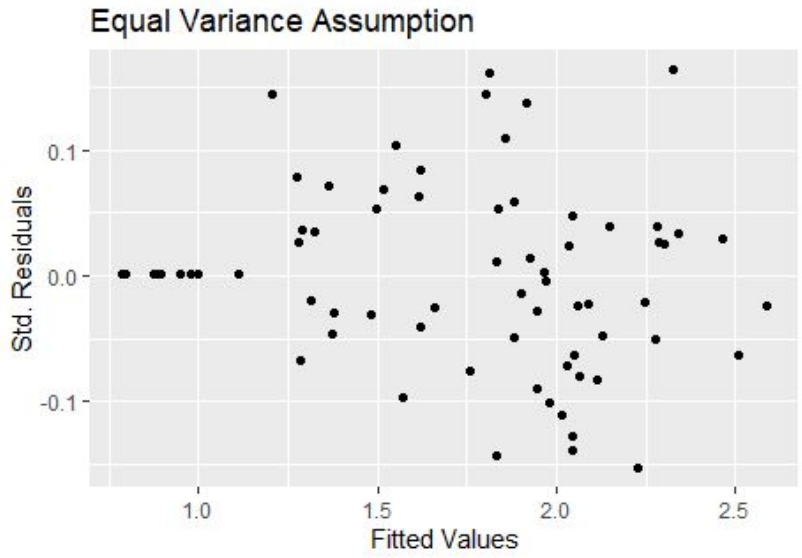

**Independence:** We can use the plot of decorrelated residuals to determine if the data meet the assumption of independence. The first bar of the plot shows that each observation is perfectly correlated with itself, which is to be expected. After the first bar we see very little correlation between the residuals, so we can conclude that the data are independent.



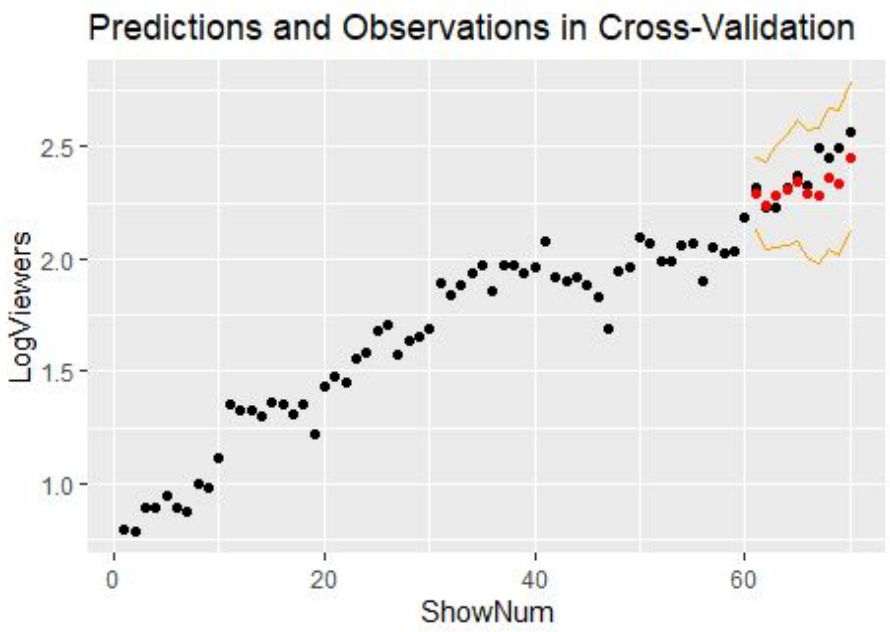

**Normality of residuals:** The standardized residuals must be normally distributed for an accurate model. The plot of the standardized (Pearson) residuals below shows an approximately normal distribution. We have no evidence to conclude that the data is not normally distributed.




**Equal variance:** The plot of standardized residuals vs. fitted values below shows whether our model meets the assumption of equal variance. Based on the plot, it appears as though there is an approximately even spread of standardized residuals over the fitted values. Therefore, we can say that our model has equal variance.




6. Using a Monte Carlo cross-validation, we obtain an RPMSE of 0.098 and coverage of 1. The RPMSE of 0.098 means that on average, predictions using this model are about 0.098 units off of the actual amount of log viewership. The RPMSE is only 5% of the total range of data for log viewership. This means that our model has high predictive accuracy. The coverage of 1 (for 95% confidence intervals) shows that the prediction intervals are functioning as they should be -- each of the prediction intervals created under this model contained the actual log(viewers) of the show number from season 7.




7. To determine if viewership is truly increasing or decreasing, we want to test (using GLHT) whether the coefficient associated with time (ShowNum) is greater than zero.

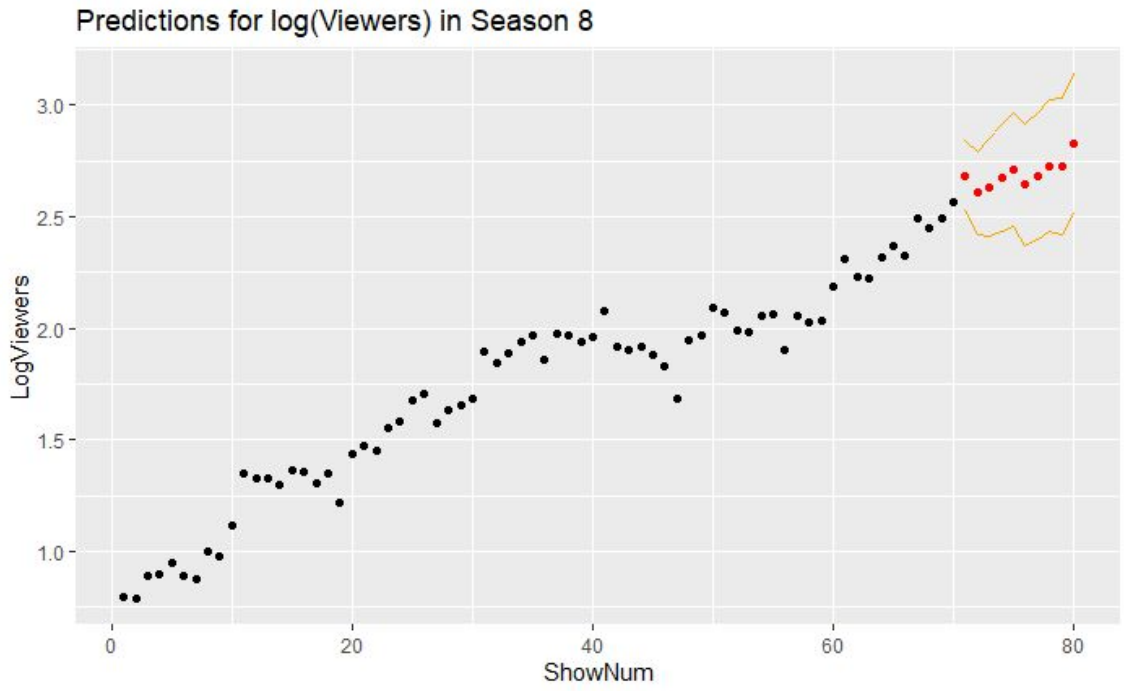
   $H_0: \beta_1 = 0$
   $H_a: \beta_1 > 0$

   The p-value for the hypothesis test described above is virtually zero. Since this is less than alpha of .05, we reject the null hypothesis and conclude that $\beta_1$ is greater than 0. Thus, we can conclude that the log of viewership is increasing with time, or with show number.

   However, a hypothesis test tells us very little about how much viewership is increasing. We can go a step further to create a confidence interval for how much viewership is increasing. We are 95% confident that the log of viewership increases between 0.017 and 0.034 for each additional show number, or each additional “point in time”. Because this confidence interval does not include zero, we can say with confidence that viewership is increasing.

8. The table below shows the forecasted log viewers for season 8 of the show. Executives can use these predictions to gauge if the show should continue into a ninth season. Looking at the table, we can see that our model predicts viewership to steadily increase during season 8. Based on these predictions, we would recommend that the show continue into a ninth season.

| Show Number | Predicted Viewership | 95% Confidence Interval |
|---|---|---|
| Season 8 Episode 1 | 2.68 | (2.53 - 2.84) |
| Season 8 Episode 2 | 2.61 | (2.42 - 2.79) |
| Season 8 Episode 3 | 2.63 | (2.41 - 2.85) |
| Season 8 Episode 4 | 2.68 | (2.44 - 2.91) |
| Season 8 Episode 5 | 2.71 | (2.45 - 2.97) |
| Season 8 Episode 6 | 2.64 | (2.37 - 2.91) |
| Season 8 Episode 7 | 2.68 | (2.40 - 3.02) |
| Season 8 Episode 8 | 2.73 | (2.43 - 3.02) |
| Season 8 Episode 9 | 2.73 | (2.42 - 3.03) |
| Season 8 Episode 10 | 2.83 | (2.52 - 3.14) |



# HW 4 Code Appendix

Jacob Andros/Jess Powell

March 5, 2020

```
# Libraries
library(ggplot2)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------- tidyverse
1.3.0 --
```

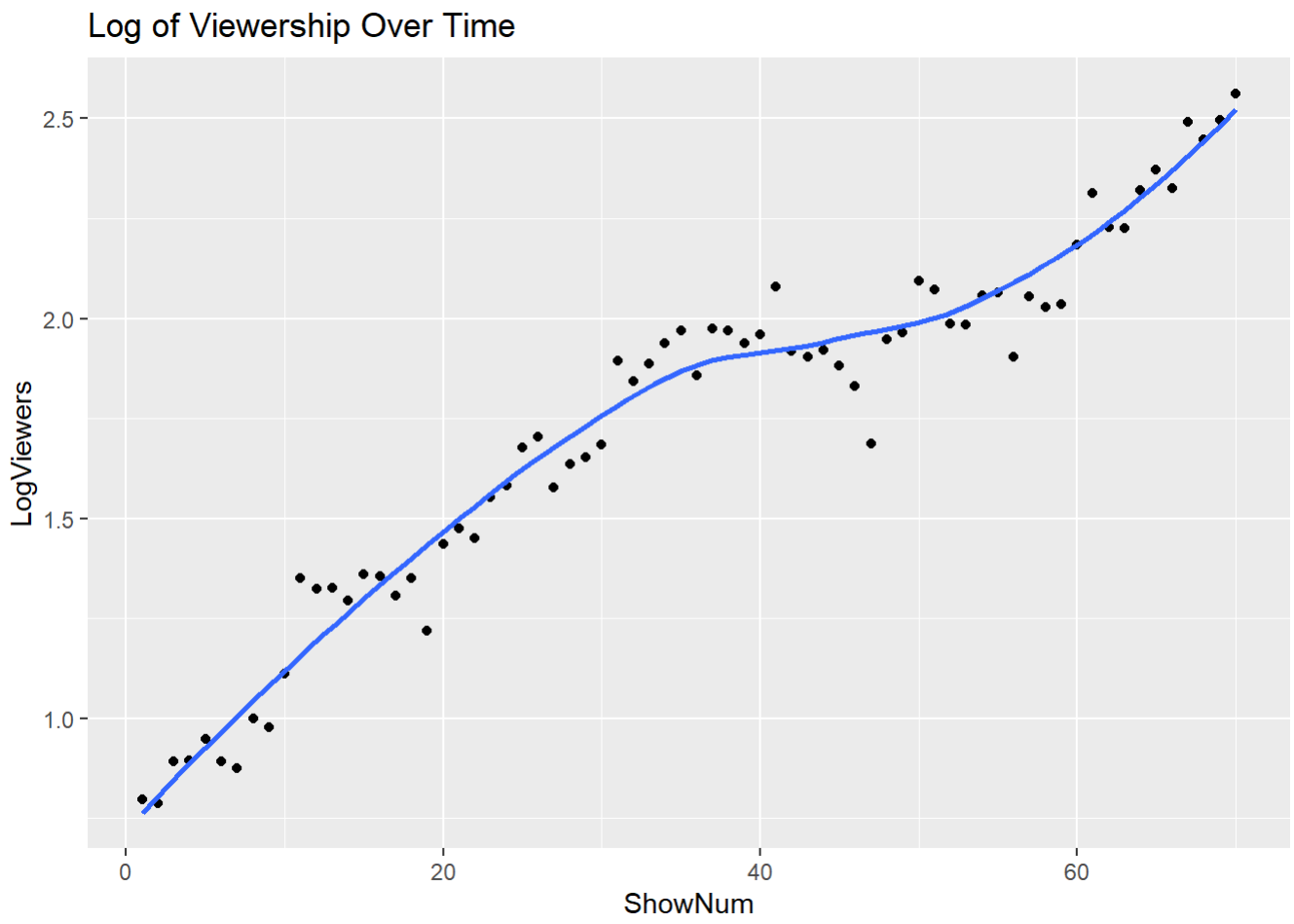
```
## v tibble  2.1.3     v dplyr   0.8.4
## v tidyr   1.0.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3
```

```
## -- Conflicts ------------------------------------------------------------------ tidyverse_confli
cts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
# Get data, make LogViewers variable
tv <- read.csv("https://mheaton.byu.edu/docs/files/Stat469/Topics/2%20-%20TemporalCorrelation/1%
20-%20TimeSeries/HWCaseStudy/Data/Viewership.txt", sep='', header=TRUE)
tv$LogViewers <- log(tv$Viewers)

# Exploratory graphic(s)
ggplot(data=tv, aes(x=ShowNum, y=LogViewers)) + geom_point() + geom_smooth(se=FALSE) + ggtitle(
"Log of Viewership Over Time")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
(r <- cor(tv$ShowNum, tv$LogViewers))
```

```
## [1] 0.9556104
```

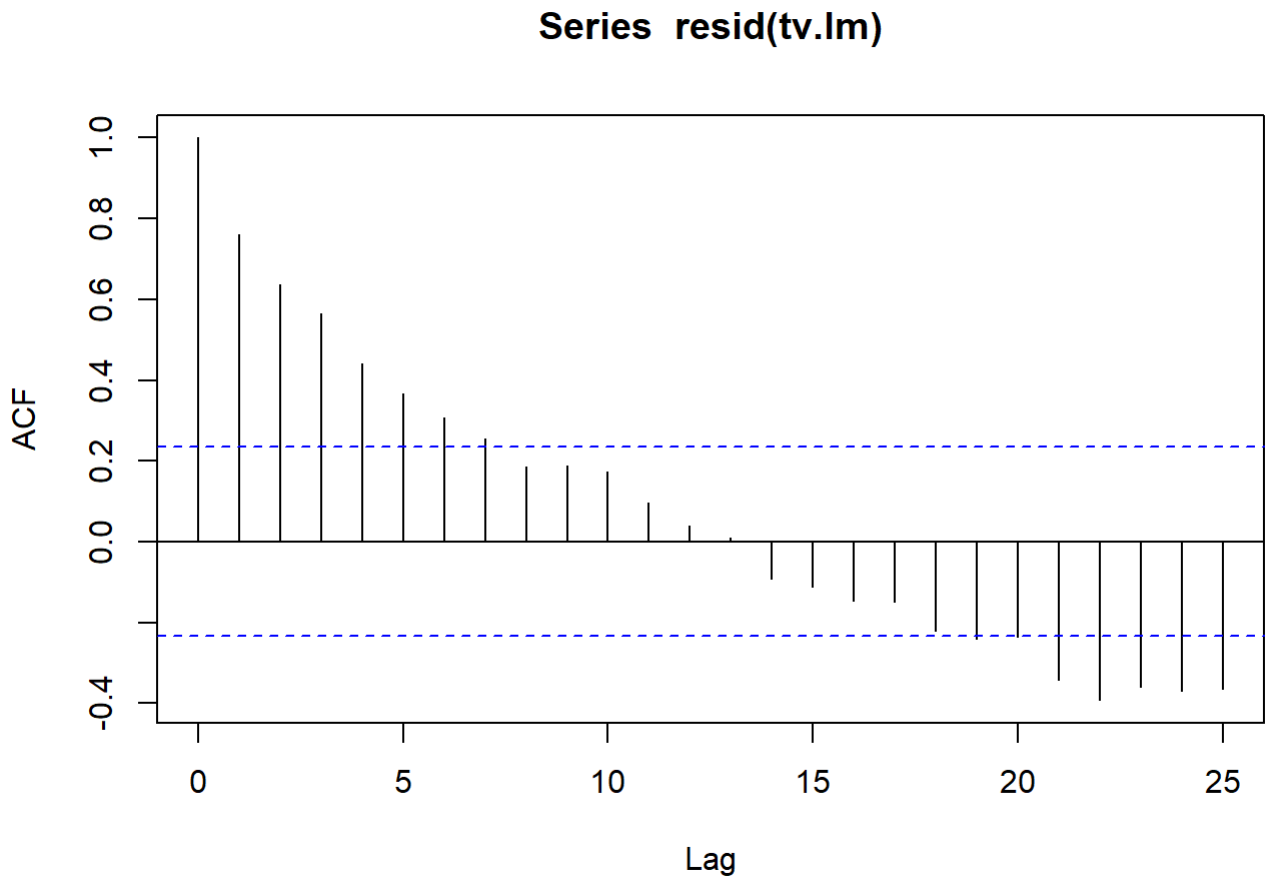
```
r^2
```

```
## [1] 0.9131912
```

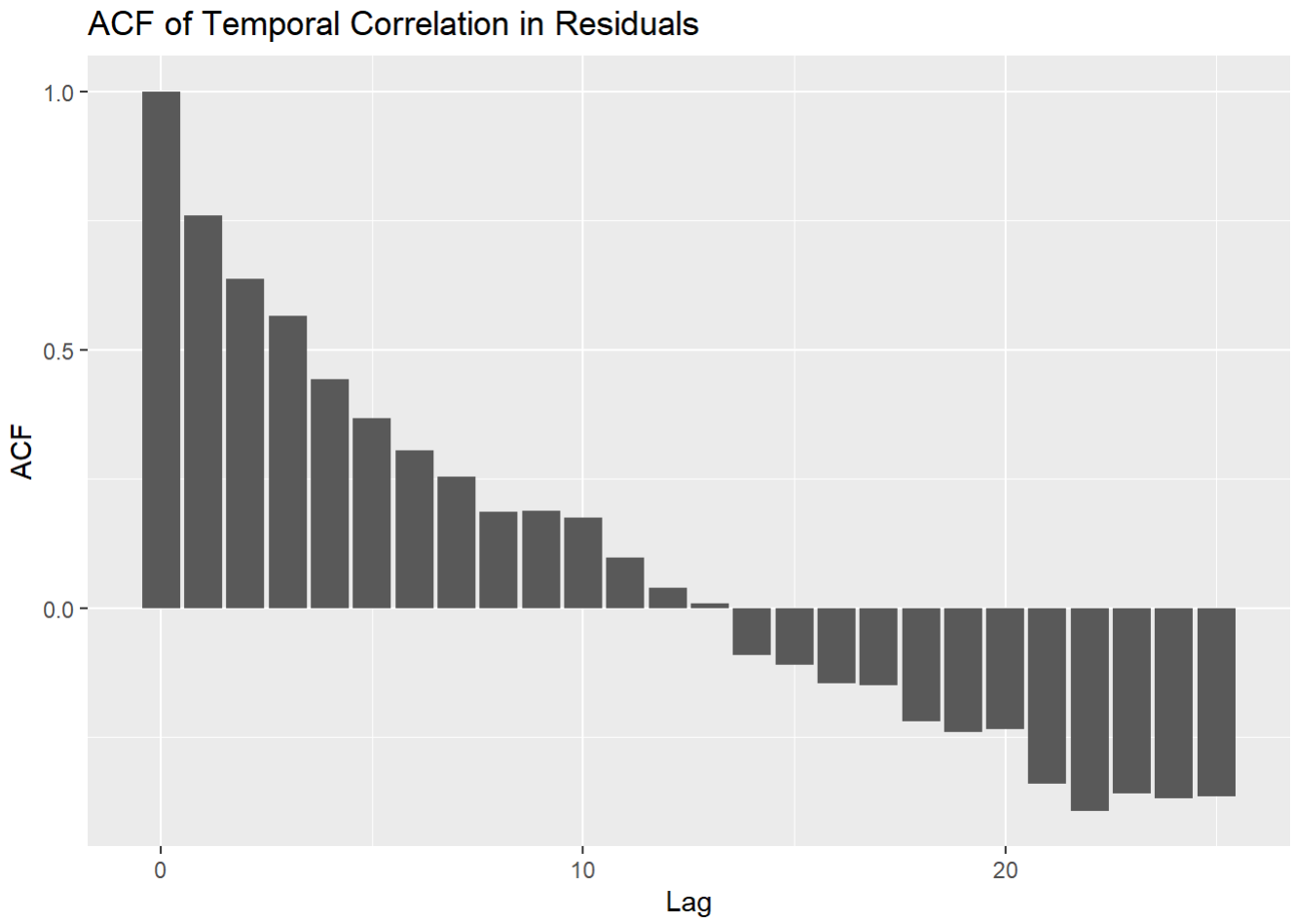
```
# Initial LM and explore correlation in residuals
summary(tv.lm <- lm(data=tv, LogViewers~ShowNum))
```

```
##
## Call:
## lm(formula = LogViewers ~ ShowNum, data = tv)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.29302 -0.10173  0.01179  0.08494  0.26075
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 0.959945   0.033127   28.98   <2e-16 ***
## ShowNum     0.021691   0.000811   26.75   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1371 on 68 degrees of freedom
## Multiple R-squared:  0.9132, Adjusted R-squared:  0.9119
## F-statistic: 715.3 on 1 and 68 DF,  p-value: < 2.2e-16
```

```
ACF.resid <- acf(resid(tv.lm), lag.max=25)
```




```
ACF.resid.dframe <- data.frame(Lag=ACF.resid$lag, ACF=ACF.resid$acf)
ggplot(data=ACF.resid.dframe, aes(x=Lag, y=ACF)) + geom_col() + ggtitle("ACF of Temporal Correla
tion in Residuals")
```

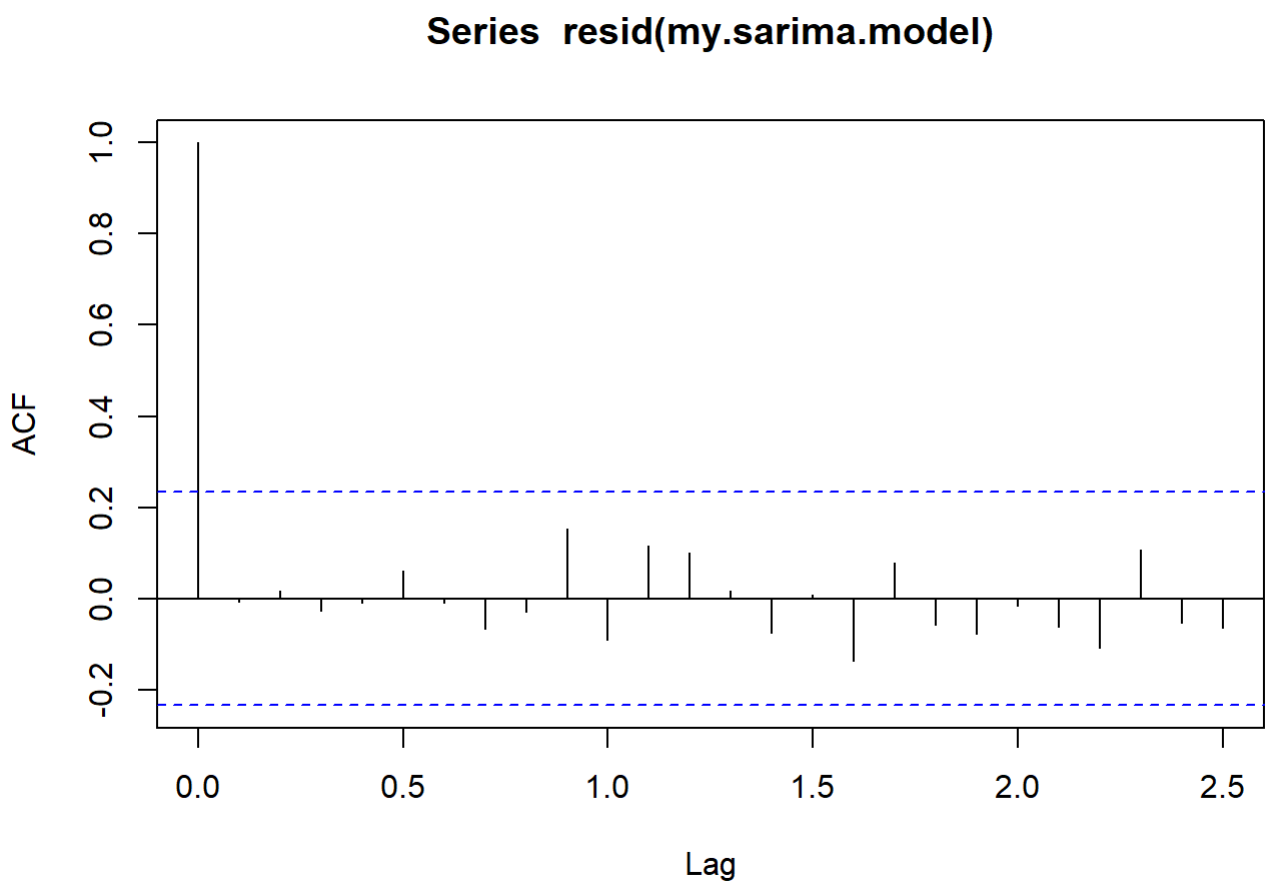



```
# TS object and Arima mode fit
tv.ts <- ts(data=tv$LogViewers, start=c(1,1), frequency=10)
x <- model.matrix(LogViewers~-1+ShowNum, data=tv)
auto.arima(tv.ts, max.p=2, max.q=2, max.P=1, max.Q=1, d=0, D=1, ic="aic", stepwise=FALSE, xreg=
x)
```

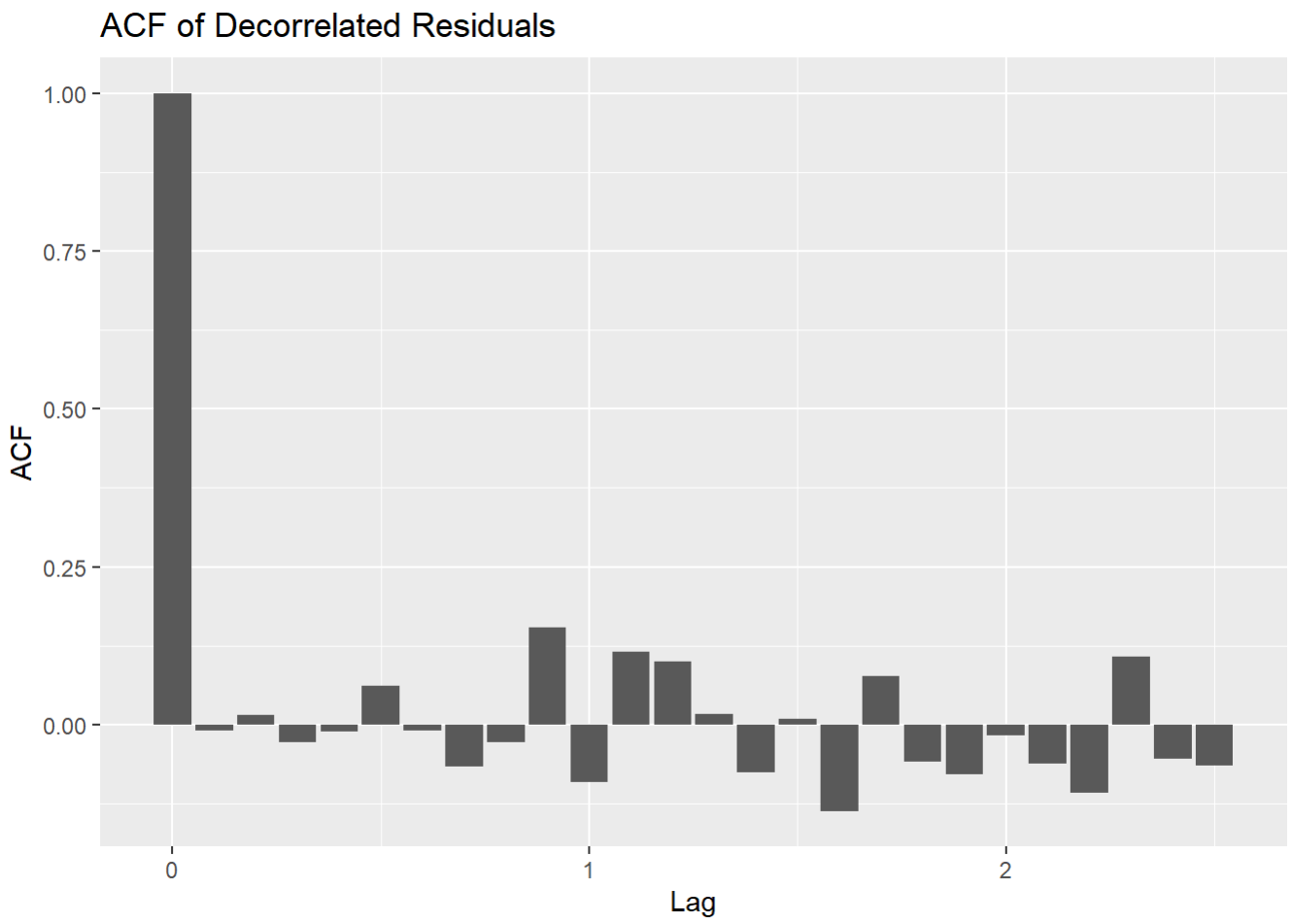
```
## Series: tv.ts
## Regression with ARIMA(2,0,0)(0,1,1)[10] errors
##
## Coefficients:
##          ar1     ar2     sma1  ShowNum
##       0.6406  0.2856  -0.7828   0.0254
## s.e.  0.1236  0.1251   0.2268   0.0043
##
## sigma^2 estimated as 0.006398:  log likelihood=63.58
## AIC=-117.15   AICc=-116.04   BIC=-106.68
```

```
my.sarima.model <- Arima(tv.ts, order=c(2,0,0), seasonal=c(0,1,1), xreg=x)

# Show that residuals have been decorrelated
ACF.decorr <- acf(resid(my.sarima.model), lag.max=25)
```



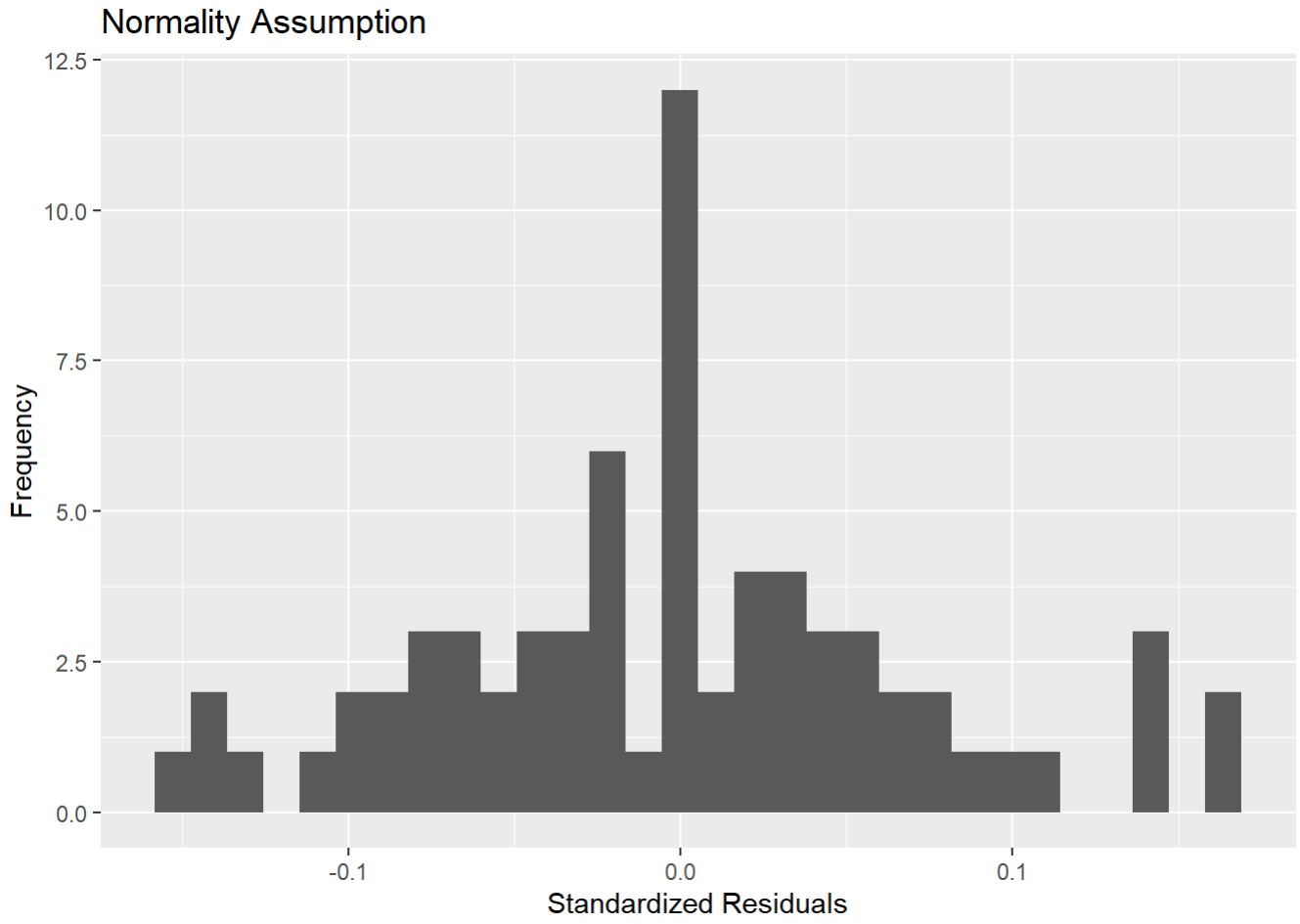

```
ACF.dframe.decorr <- data.frame(Lag=ACF.decorr$lag, ACF=ACF.decorr$acf)
ggplot(data=ACF.dframe.decorr, aes(x=Lag, y=ACF)) + geom_col() + ggtitle("ACF of Decorrelated Re
siduals")
```




```
# Validate assumptions
ggplot()+geom_histogram(mapping=aes(x=resid(my.sarima.model))) + xlab("Standardized Residuals")
 + ylab("Frequency") + ggtitle("Normality Assumption")
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

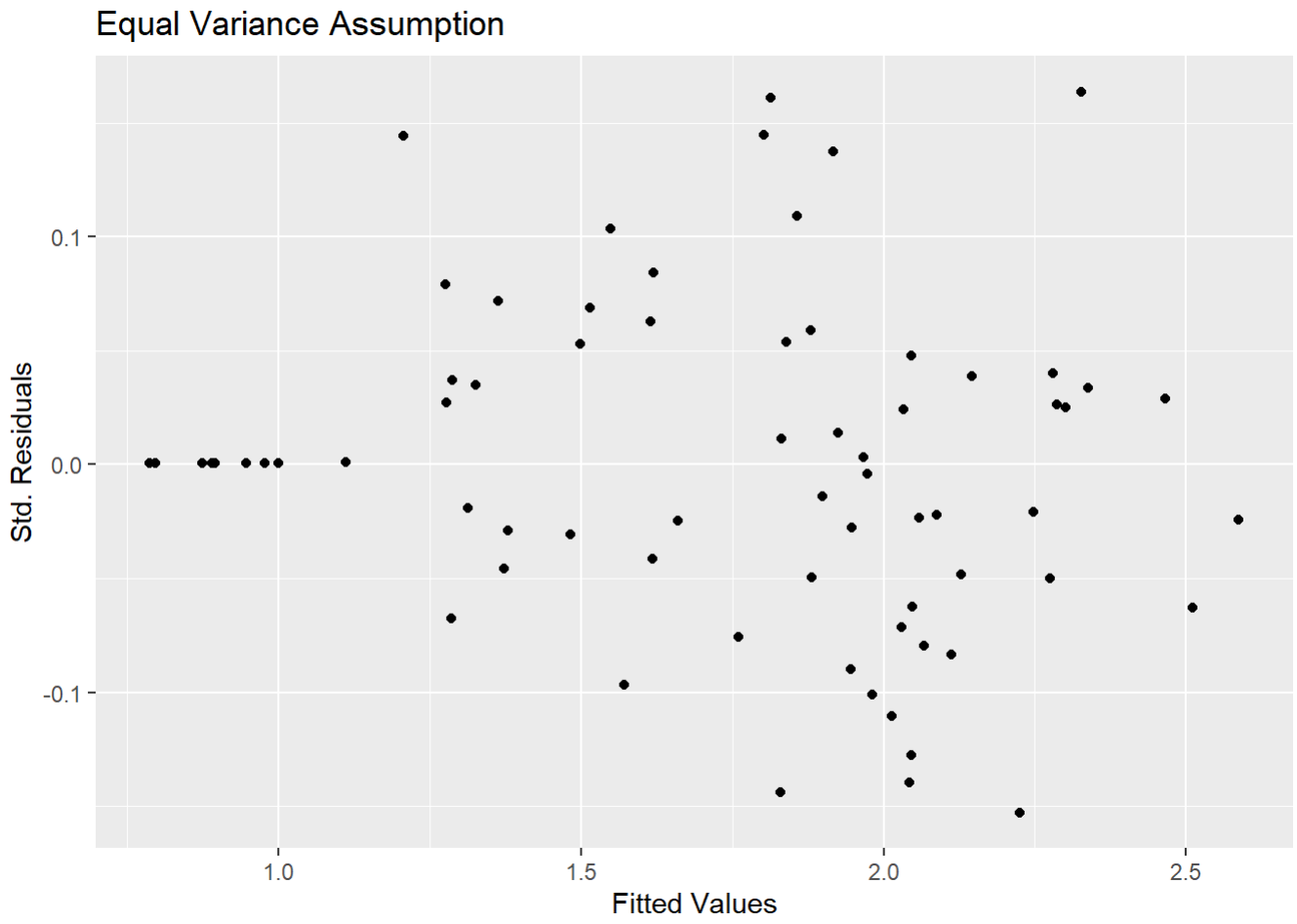
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(mapping=aes(y=resid(my.sarima.model), x=fitted(my.sarima.model))) + geom_point() + xlab(
"Fitted Values") + ylab("Std. Residuals") + ggtitle("Equal Variance Assumption")
```

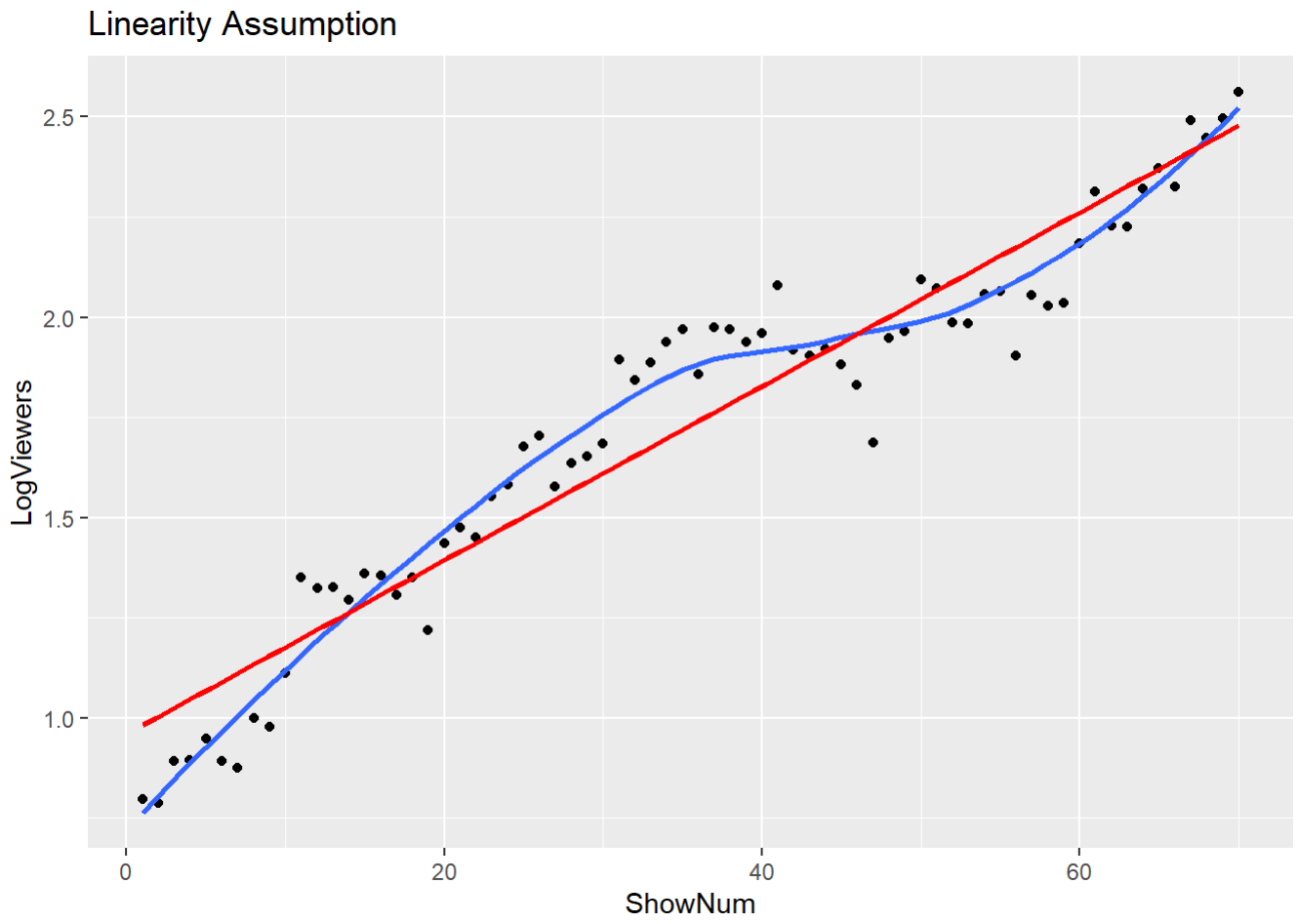
```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```




```
ggplot(data=tv, aes(x=ShowNum, y=LogViewers)) + geom_point() + geom_smooth(se=FALSE) + geom_smoo
th(se=FALSE, method=lm, col='red') + ggtitle("Linearity Assumption")
```

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
# Cross-validation
test.ts <- tv[61:70,]
train.ts <- tv[1:60,]
ts.cv <- ts(data=train.ts$LogViewers, start=c(1,1), frequency=10)
test.x <- x[61:70,]
train.x <- x[1:60,]
#auto.arima(ts.cv, max.p=2, max.q=2, max.P=2, max.Q=2, d=0, D=1, ic="aic", stepwise=FALSE,xreg=t
rain.x)
sarima.cv <- Arima(ts.cv, order=c(2,0,0), seasonal=c(0,1,1), xreg=train.x)
preds <- forecast(sarima.cv, h=10, xreg=test.x, level=.95)
rpmse <- (test.ts[['LogViewers']]-preds$mean)^2 %>% mean() %>% sqrt()
cvg <- ((test.ts[['LogViewers']] > preds[['lower']]) & (test.ts[['LogViewers']] < preds[['upper'
]])) %>% mean()
rpmse
```

```
## [1] 0.09846837
```

```
cvg
```

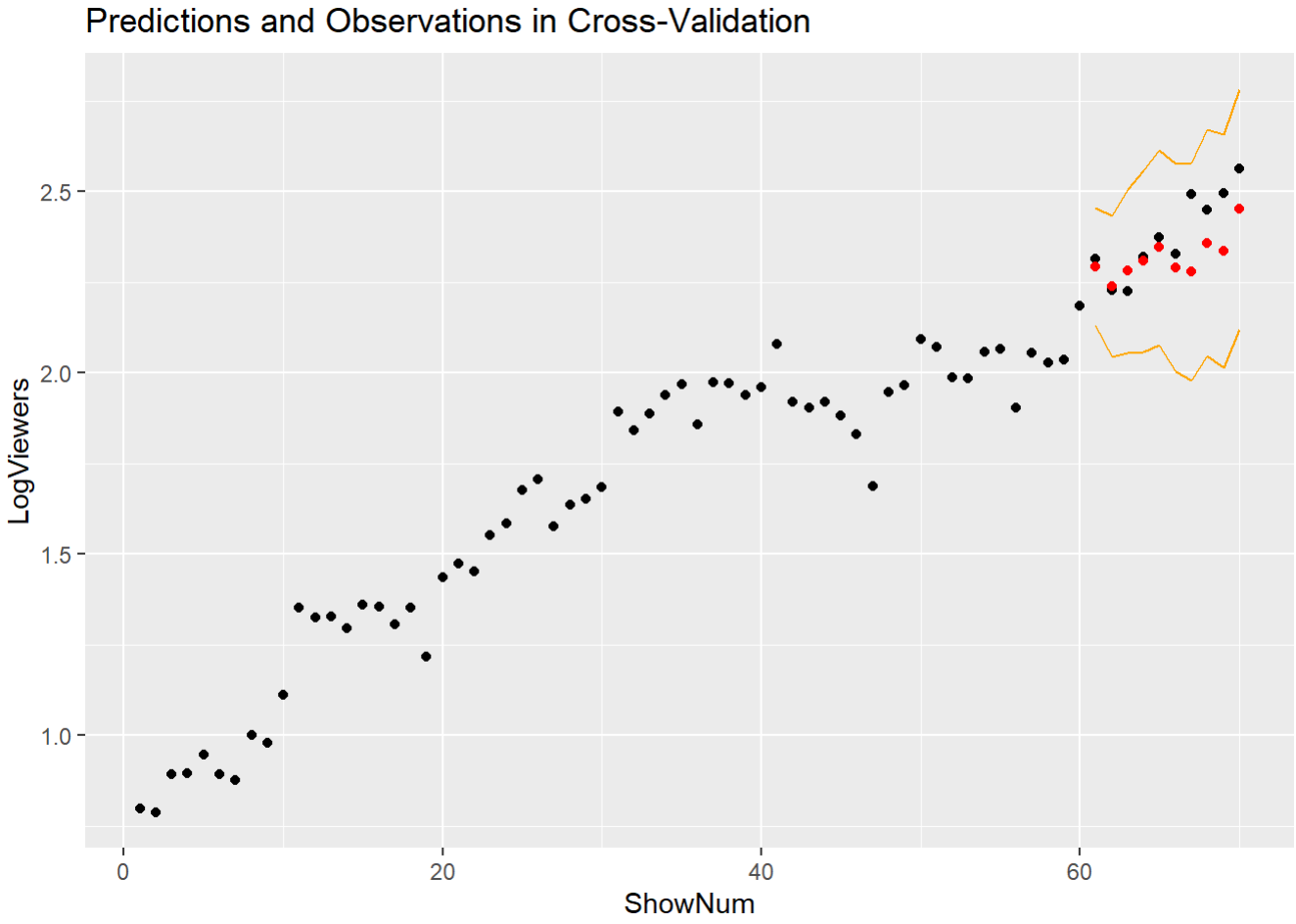
```
## [1] 1
```

```
# Plot of CV vs. observations
a <- rep(NA, 60)
plot.prep <- data.frame(tv$ShowNum, c(a, preds[['mean']]), c(a, preds[['lower']]), c(a, preds[[
'upper']]))
colnames(plot.prep) <- c("ShowNum", "Mean", "Lower", "Upper")
ggplot(data=tv, aes(x=ShowNum, y=LogViewers)) + geom_point() + geom_point(data=plot.prep, aes(x=
ShowNum, y=Mean), col='red') + geom_line(data=plot.prep, aes(x=ShowNum, y=Lower), col='orange')
 + geom_line(data=plot.prep, aes(x=ShowNum, y=Upper), col='orange') + ggtitle("Predictions and O
bservations in Cross-Validation")
```

```
## Warning: Removed 60 rows containing missing values (geom_point).
```

```
## Warning: Removed 60 rows containing missing values (geom_path).

## Warning: Removed 60 rows containing missing values (geom_path).
```




```
# Hypothesis testing
deg.f <- nrow(tv) - 4
se <- (vcov(my.sarima.model) %>% diag() %>% sqrt())[[4]]
1-pt(coef(my.sarima.model)[[4]] / se, deg.f)
```

```
## [1] 6.440135e-08
```

```
coef(my.sarima.model)[[4]] + qt(.975, deg.f) * c(-1,1) * se
```

```
## [1] 0.01686218 0.03403663
```

```
# Predictions
ShowNum <- c(71:80)
x.prep <- data.frame(ShowNum)
x.preds <- model.matrix(~-1+ShowNum, data=x.prep)
preds.season <- forecast(my.sarima.model, h=10, xreg=x.preds, level=.95)
preds.season
```

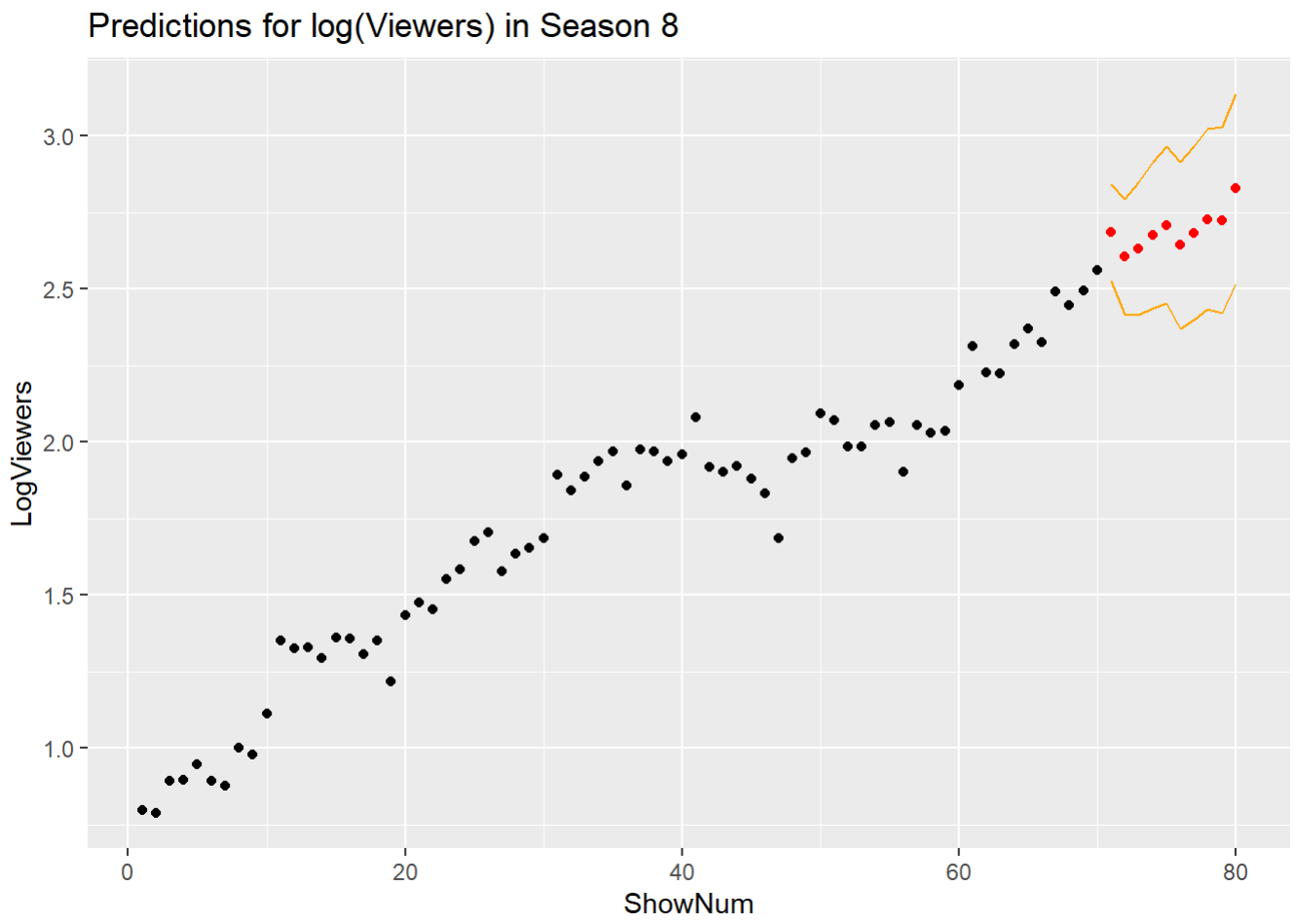
```
##      Point Forecast    Lo 95    Hi 95
## 8.00       2.684463 2.526335 2.842591
## 8.10       2.607017 2.419502 2.794532
## 8.20       2.631389 2.414286 2.848493
## 8.30       2.676295 2.437790 2.914800
## 8.40       2.709451 2.452962 2.965939
## 8.50       2.643435 2.372123 2.914748
## 8.60       2.682187 2.398349 2.966025
## 8.70       2.728309 2.433838 3.022779
## 8.80       2.725808 2.422250 3.029367
## 8.90       2.828278 2.516909 3.139647
```

```
# Plot predictions
b <- rep(NA, 70)
plot.prep2 <- data.frame(c(tv$ShowNum, 71:80), c(b, preds.season[['mean']]), c(b, preds.season[[
'lower']]), c(b, preds.season[['upper']]))
colnames(plot.prep2) <- c("ShowNum", "Mean", "Lower", "Upper")
ggplot(data=tv, aes(x=ShowNum, y=LogViewers)) + geom_point() + geom_point(data=plot.prep2, aes(x
=ShowNum, y=Mean), col='red') + geom_line(data=plot.prep2, aes(x=ShowNum, y=Lower), col='orange'
) + geom_line(data=plot.prep2, aes(x=ShowNum, y=Upper), col='orange') + ggtitle("Predictions for
log(Viewers) in Season 8")
```

```
## Warning: Removed 70 rows containing missing values (geom_point).
```

```
## Warning: Removed 70 rows containing missing values (geom_path).

## Warning: Removed 70 rows containing missing values (geom_path).
```


Predictions for log(Viewers) in Season 8
LogViewers
3.0
2.5
2.0
1.5
1.0
0
20
40
60
80
ShowNum