# William Stults - Time Series Forecasting (D213 Task 1)

March 27, 2023

---

## 1 Part I: Research Question

### 1.1 Research Question

My data set for this time series forecasting exercise includes data on a teleco company's revenue measured daily. Data analysis performed on the dataset will be aimed with this research question in mind: can a time series model be used to forecast future revenue values, and if so, how accurately?

---

### 1.2 Objectives and Goals

Conclusions gleaned from the analysis of this data can benefit stakeholders by revealing information and insights on expected revenue increases and decreases over upcoming time periods. My goal will be to determine how accurately a time series forecast might be able to predict this data.

---

## 2 Part II: Method Justification

### 2.1 Time Series Models

A time series can be defined as "a sequence of various data points that occurred in a successive order for a given period of time" (Pandian, 2023). In my specific case, I have been given a sequence of revenue values, measured daily over a period of 731 days.

The one assumption of time series modeling is that the data in the time series is stationary over time. Stationarity is one of multiple measures that can be used to evaluate a time series, and for the purpose of time series forecasting it is assumed that the point in time at which you begin your forecast will not matter statistically. In more common terms, when the time series is visualized it should appear as though you can draw a straight horizontal line through the middle, rather than a diagonal line (indicative of a trending time series) or a wavy line (may be indicative of a seasonal time series).

In addition to stationarity, the auto-correlation function of a time series is also critical to time series analysis. Analyzing the auto-correlation function plot for a time series can help detect patterns and data exhibiting a trend, as well as confirming the lack of randomness in the data. It is a comparison of a time series with a "lagged" version of itself, helping to determine any correlation between a current data point and previous data points. For a time series that is stationary and non-random, we would expect to see an ACF plot's range narrow significantly when read from left to right, with points along the plot displaying positive and negative variance within a shown confidence interval (Monigatti, 2022).

---

## 3 Part III: Data Preparation

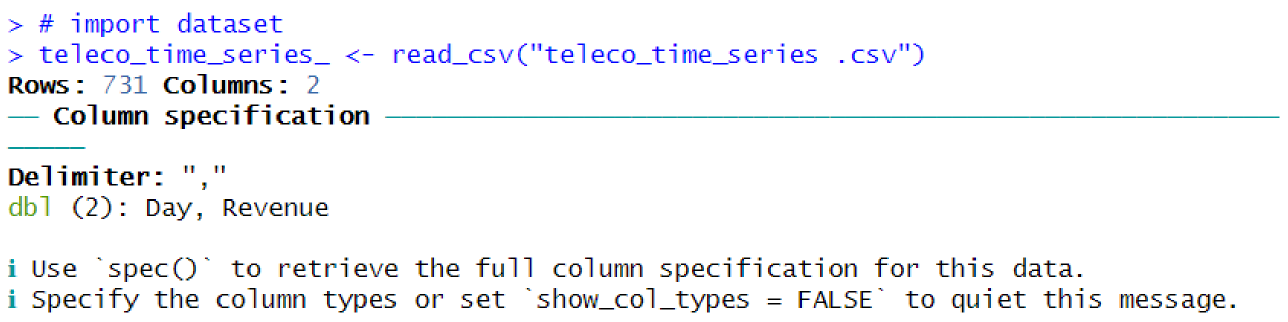
My first steps will be to specify my working directory, import some libraries known to be useful for time series analysis, import the data set which will be the foundation of the time series, and finally inspect the data set visually.

```
# set working directory
setwd('D:/Logos/Dev/Rdata/d213t1scratch')

# import libraries
library(tseries)
library(forecast)
library(readr)
library(ggplot2)
library(zoo)
```

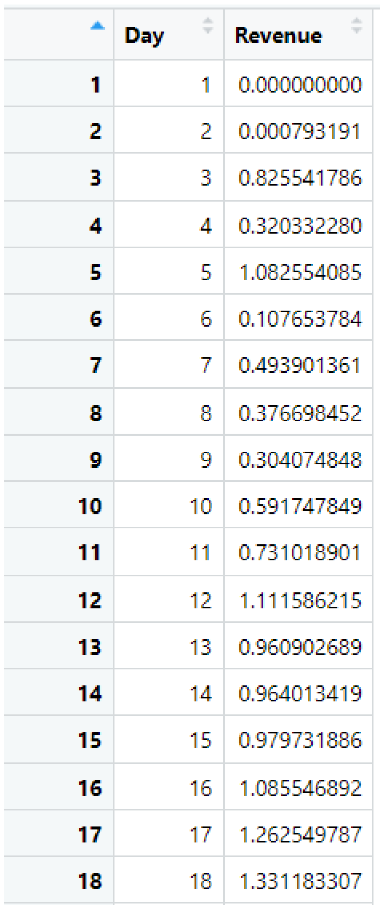
```
> # set working directory
> setwd('D:/Logos/Dev/Rdata/d213t1scratch')
> # import libraries
> library(tseries)
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo

    'tseries' version: 0.10-53

    'tseries' is a package for time series analysis and computational finance.

    See 'library(help="tseries")' for details.

> library(forecast)
> library(readr)
> library(ggplot2)
> library(zoo)

Attaching package: 'zoo'

The following objects are masked from 'package:base':

    as.Date, as.Date.numeric
```

```
# import dataset
teleco_time_series_ <- read_csv("teleco_time_series .csv")

View(teleco_time_series_)
```

---



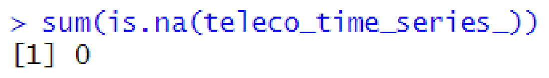
```
> # import dataset
> teleco_time_series_ <- read_csv("teleco_time_series .csv")
Rows: 731 Columns: 2
── Column specification ──────────────────────────────────────────────────────────
────
Delimiter: ","
dbl (2): Day, Revenue

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```



| | Day | Revenue |
|---|---|---|
| 1 | 1 | 0.000000000 |
| 2 | 2 | 0.000793191 |
| 3 | 3 | 0.825541786 |
| 4 | 4 | 0.320332280 |
| 5 | 5 | 1.082554085 |
| 6 | 6 | 0.107653784 |
| 7 | 7 | 0.493901361 |
| 8 | 8 | 0.376698452 |
| 9 | 9 | 0.304074848 |
| 10 | 10 | 0.591747849 |
| 11 | 11 | 0.731018901 |
| 12 | 12 | 1.111586215 |
| 13 | 13 | 0.960902689 |
| 14 | 14 | 0.964013419 |
| 15 | 15 | 0.979731886 |
| 16 | 16 | 1.085546892 |
| 17 | 17 | 1.262549787 |
| 18 | 18 | 1.331183307 |

Before beginning my analysis, I will inspect the data to ensure no null or missing values exist.

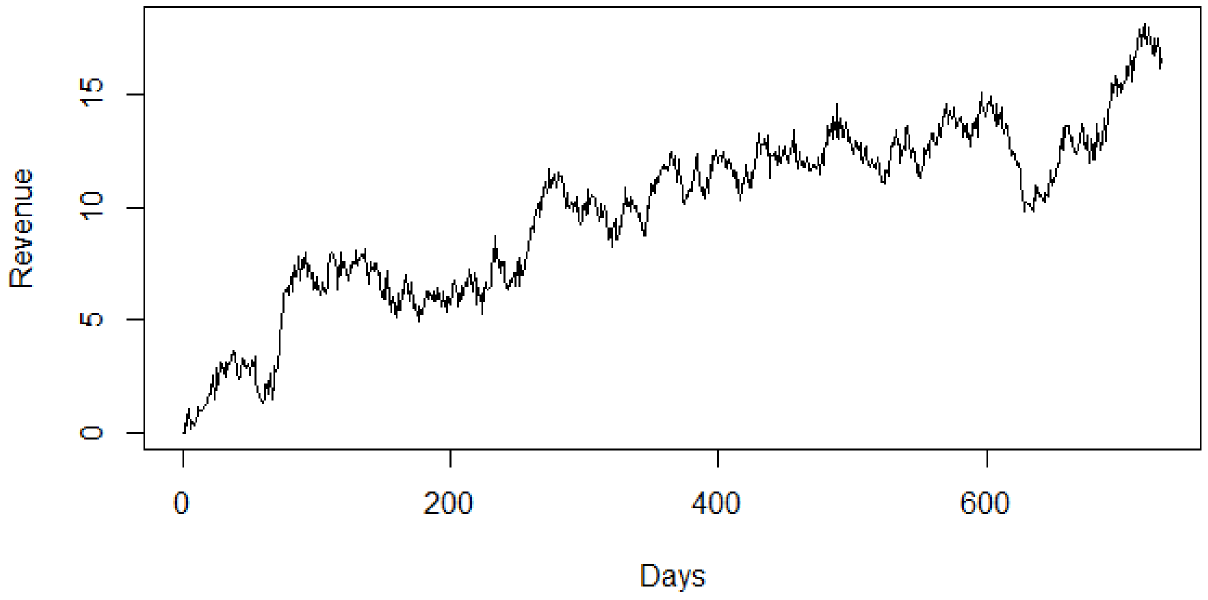
```
# checking for null values
sum(is.na(teleco_time_series_))
```



```
> sum(is.na(teleco_time_series_))
[1] 0
```

I can now convert the data to a time series object and create a plot visualization of the time series.

```
# convert to a time series
teleco_ts <- ts(teleco_time_series_$Revenue)

# plot visualization (part C1)
plot(teleco_ts, ylab = "Revenue", xlab = "Days")
```



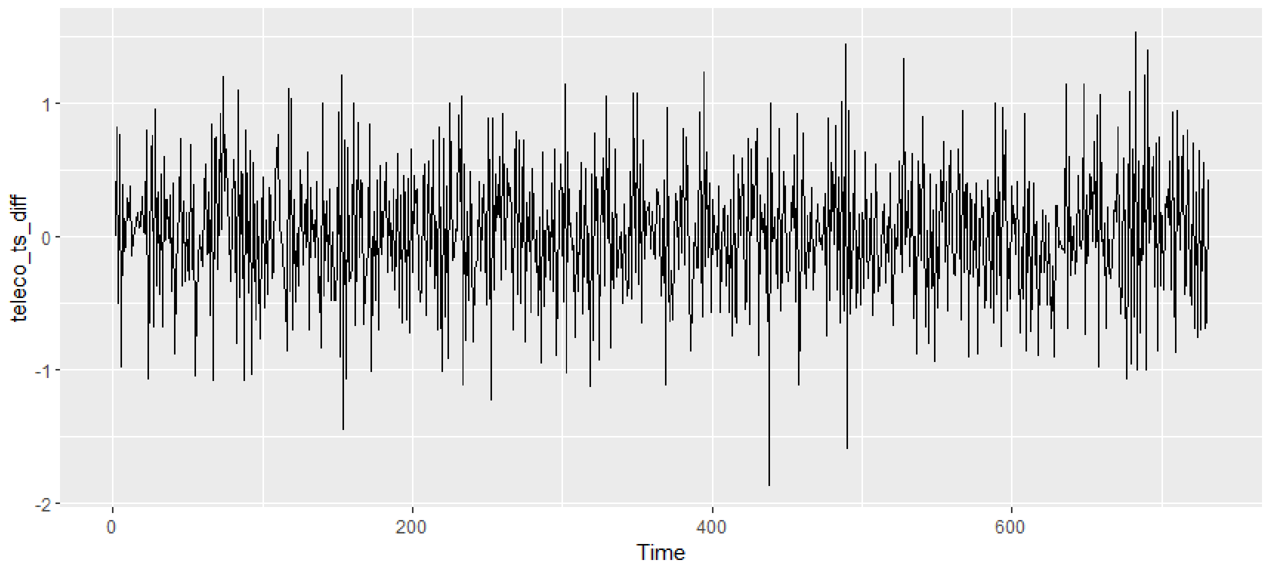
Before going further, I will check the time series data for outliers that may need corrective action.

```
# automatic detection of outliers
teleco_ts_out = tsoutliers(teleco_ts)
teleco_ts_out
```

```
> teleco_ts_out = tsoutliers(teleco_ts)
> teleco_ts_out
$index
integer(0)

$replacements
numeric(0)
```

I can easily see via the previous time series plot that the time series is not very stationary, as the values trend higher the further into the series we look. To get the time series data more stationary I will use differencing, which converts each data point value into the difference between itself and the value previous. This should result in a more "horizontal" looking visualization, indicating greater stationarity.

```
# differencing the time series
teleco_ts_diff <- diff(teleco_ts)
autoplot(teleco_ts_diff)
```



---

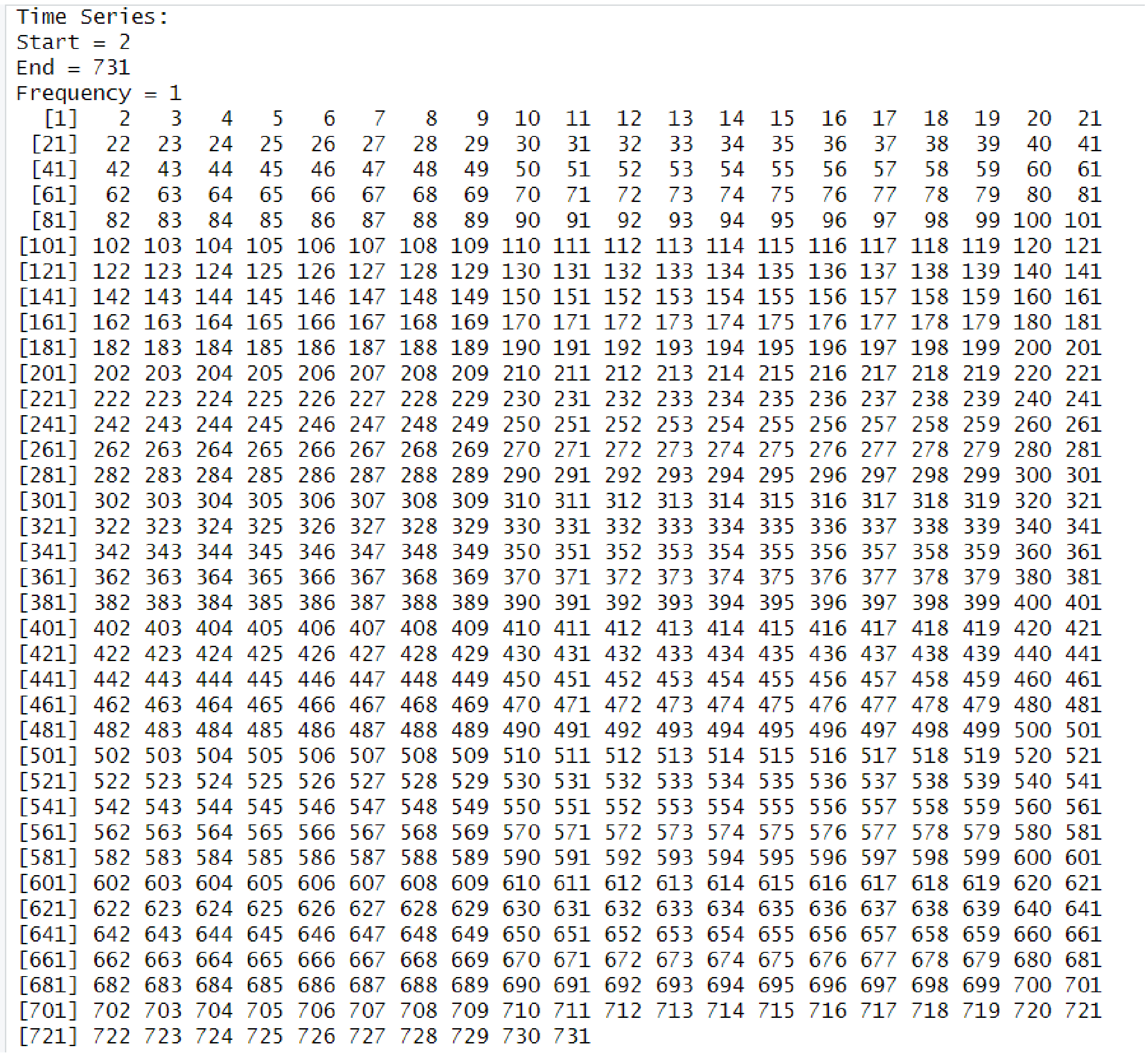
As an additional check for stationarity, I used the Augmented Dickey-Fuller test, a popular method for confirming a time series' stationarity.

```
# Augmented Dickey-Fuller test for stationarity
adf.test(teleco_ts_diff)
```

```
> adf.test(teleco_ts_diff)

	Augmented Dickey-Fuller Test

data:  teleco_ts_diff
Dickey-Fuller = -8.6354, Lag order = 8, p-value = 0.01
alternative hypothesis: stationary
```

With no outliers present, I can inspect the differenced time series formatting. Doing so reveals that there are no gaps or missing days in the measurement period (day 1 is dropped from the series as it has no preceding day to generate a differenced value from), and that the time series is 730 days in length.

```
# checking time step formatting (part C2)
time(teleco_ts)
```



```
Time Series:
Start = 2
End = 731
Frequency = 1
  [1]   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20  21
 [21]  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41
 [41]  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61
 [61]  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81
 [81]  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 101
[101] 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121
[121] 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141
[141] 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161
[161] 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181
[181] 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201
[201] 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221
[221] 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241
[241] 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261
[261] 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281
[281] 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301
[301] 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321
[321] 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341
[341] 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361
[361] 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381
[381] 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401
[401] 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421
[421] 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441
[441] 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461
[461] 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481
[481] 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501
[501] 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521
[521] 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541
[541] 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561
[561] 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581
[581] 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601
[601] 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621
[621] 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641
[641] 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661
[661] 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681
[681] 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701
[701] 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721
[721] 722 723 724 725 726 727 728 729 730 731
```

### 3.1 Training Data and Test Data

I can now split the differenced time series into 2 separate time series', one to be utilized in training the ARIMA model, and another for the purpose of evaluating my results. I will use an 80%/20% split of the available data, so the "train" set will contain days 2-585, and the "test" set will contain days 586-731. Then I will export each set as a csv file.

```
# train set and test set split
teleco_ts_train <- window(teleco_ts_diff,start=2,end=585)
teleco_ts_test  <-  window(teleco_ts_diff,start=586,end=731)

# Export train and test sets to csv
write.csv(teleco_ts_train, file="train.csv")
write.csv(teleco_ts_test,   file="test.csv")
```

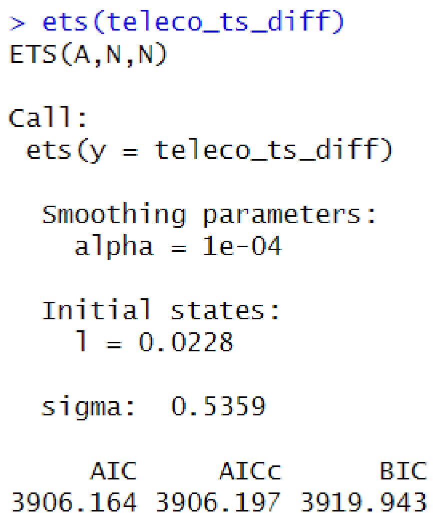
## 4 Part IV: Analysis

### 4.1 Seasonality

Before performing decomposition on my differenced time series, I need to check the series for seasonality to determine the appropriate method of decomposition to use. Seasonality is indicated by an upward or downward swing in the time series at a fixed interval (every X number of days, for example). There is a distinct pattern the swings follow when seasonality is present, and the visualized time series will often appear as a "wave" or "saw" pattern (Pandian, 2023).

I will utilize the ets() function and inspect the output to determine if seasonality is present in the time series.

```
# check for seasonality using ets()
ets(teleco_ts_diff)
```



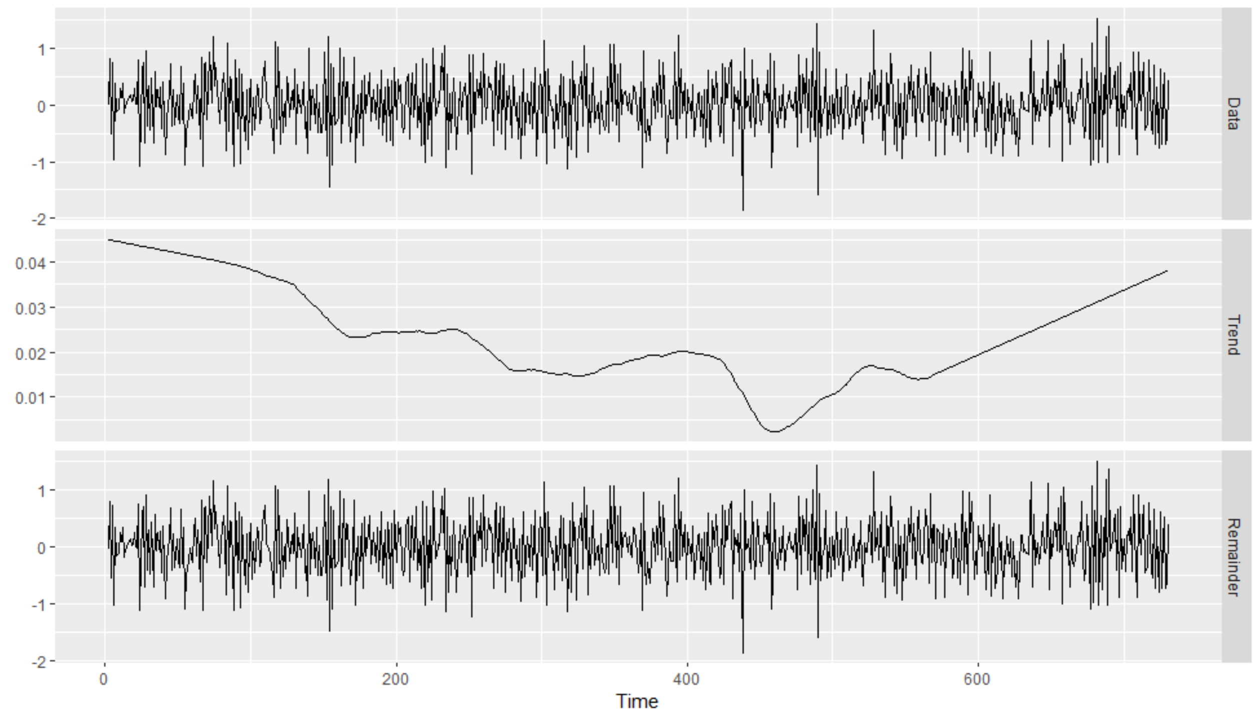
```
> ets(teleco_ts_diff)
ETS(A,N,N)

Call:
 ets(y = teleco_ts_diff)

  Smoothing parameters:
    alpha = 1e-04

  Initial states:
    l = 0.0228

  sigma:  0.5359

     AIC     AICc      BIC
3906.164 3906.197 3919.943
```

The ets() function returns a summary value containing three results: error, trend, and seasonality. My third returned value was N, indicating no seasonality component to the time series data.

---

For time series' without a seasonal component, the mstl() function is the chosen method for decomposition.

```
# Time Series Decomp using mstl()
teleco_comp <- mstl(teleco_ts_diff)
autoplot(teleco_comp)
```



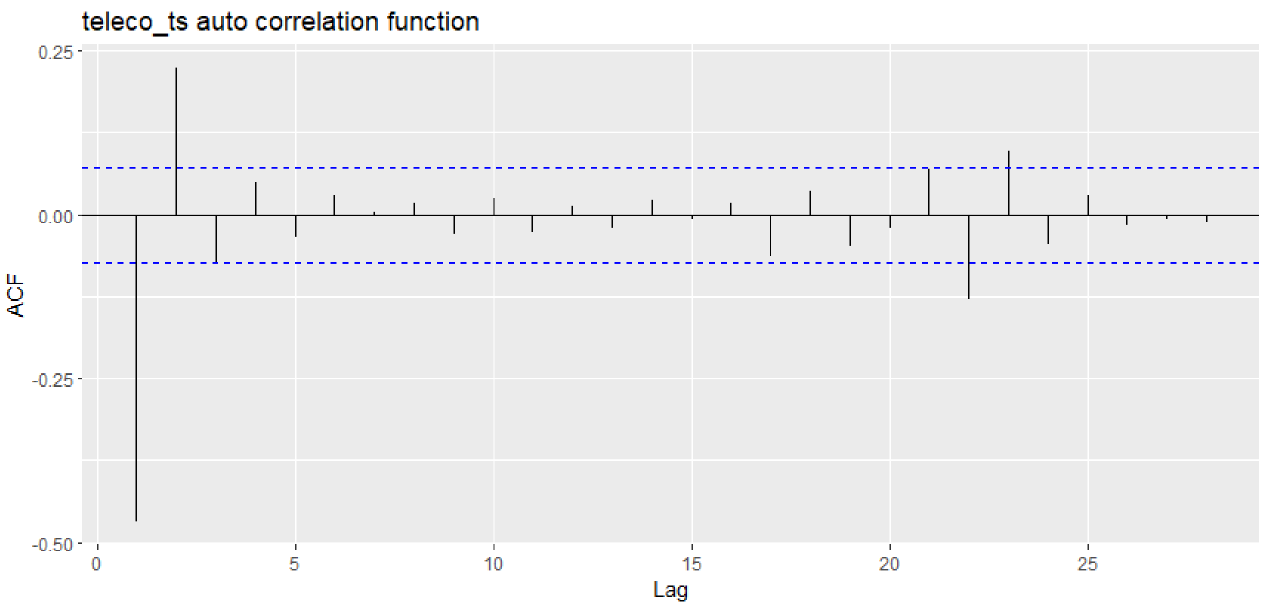
The above visualization confirms for us that there is no trending aspect to our time series, as the plotted line does not travel directly from one corner of the plot to the opposite corner, but rather oscillates within a small range both upward and downward.

The residuals plot (labeled Remainder in the graphic) also confirms an absence of a trending element in the time series' residuals.

---

Using ACF (auto-correlation function), I can further confirm the absence of a trending element in the time series, as well as determining the non-random nature of the time series data.

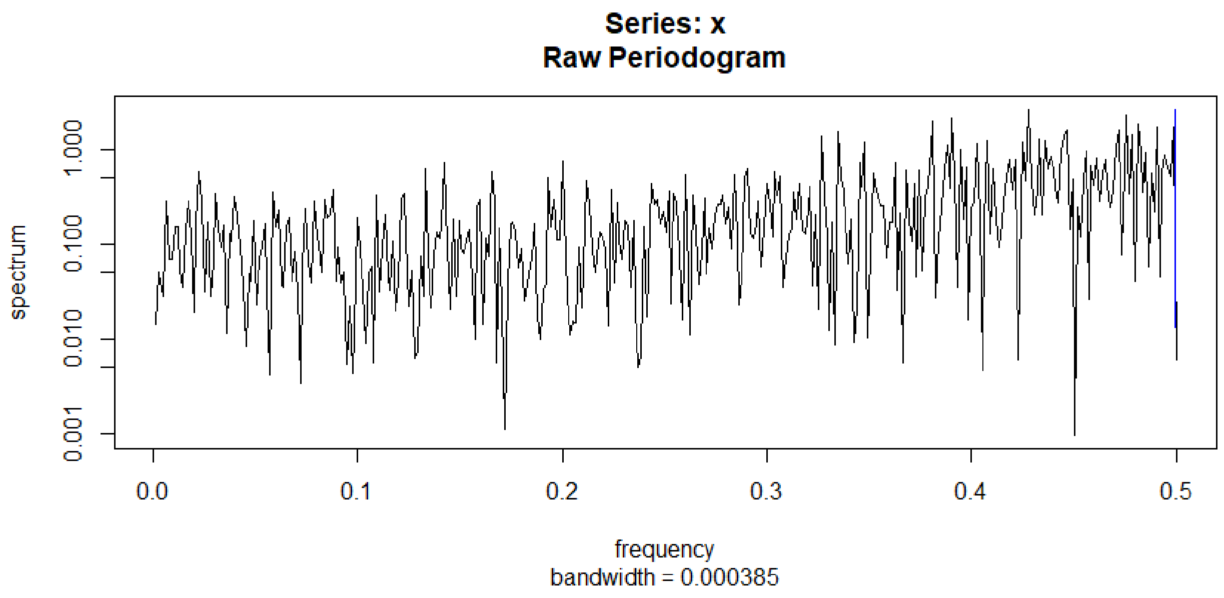
```
# auto correlation function
ggAcf(teleco_ts_diff) + ggtitle("teleco_ts auto correlation function")
```



---

The spectral density of the time series can be observed in the below illustration.

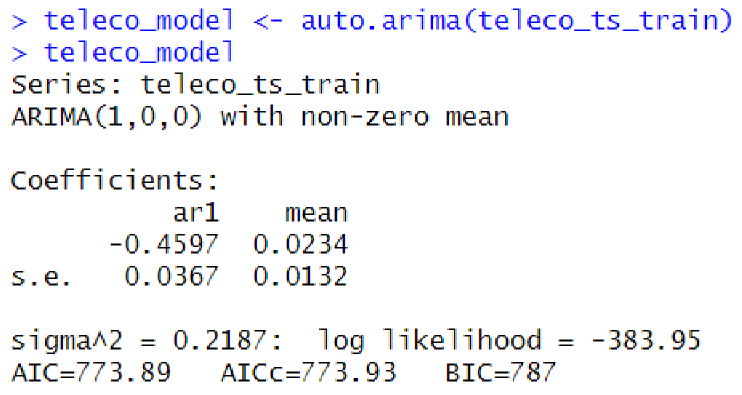
```
# spectral density plot
spectrum(teleco_ts_diff)
```



## 4.2 ARIMA

I can use the auto.arima() function to help determine an appropriate ARIMA model upon which a forecast may be generated.

```
# create arima model
teleco_model <- auto.arima(teleco_ts_train)
teleco_model
```



```
> teleco_model <- auto.arima(teleco_ts_train)
> teleco_model
Series: teleco_ts_train
ARIMA(1,0,0) with non-zero mean

Coefficients:
          ar1    mean
      -0.4597  0.0234
s.e.   0.0367  0.0132

sigma^2 = 0.2187:  log likelihood = -383.95
AIC=773.89   AICc=773.93   BIC=787
```

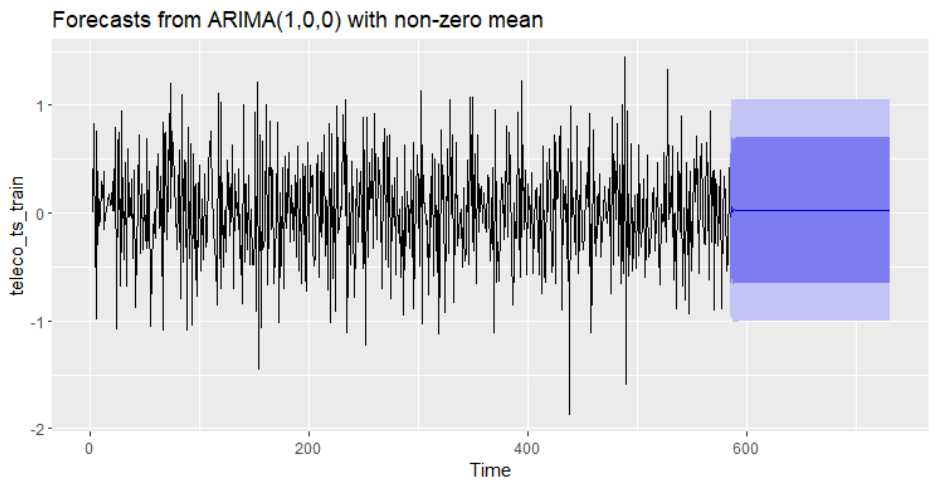
Auto.arima() indicates a model with characteristics 1,0,0 is appropriate. To elaborate, the forecasting model will use:

- (AR) 1 period of lag
- (I) 0 differencing adjustment (likely because I already differenced the time series)
- (MA) 0 moving average window (expected, as moving average is not needed for stationary series)

## 4.3 Forecasting

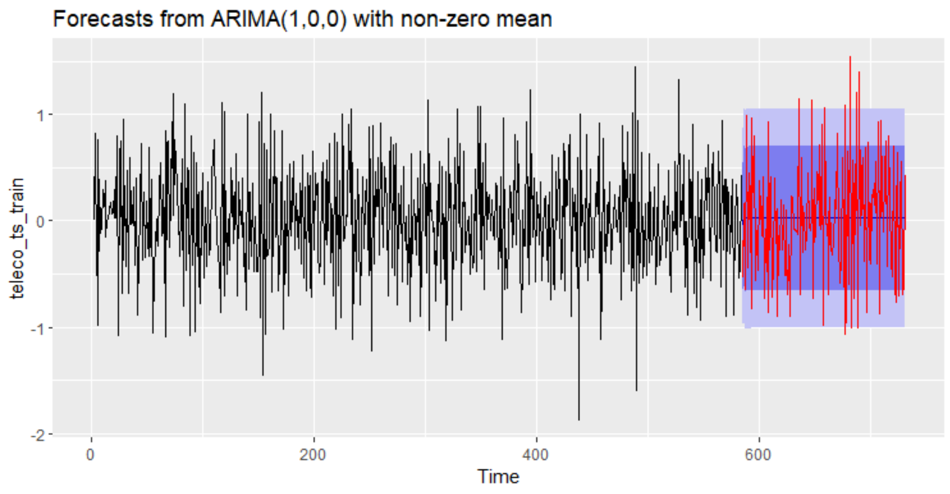
When forecasting, I will specify a period length of 146 periods, or days, as there are 146 days' worth of data points in my test set. This will make it easy to compare the forecast to actual data points from those 146 days.

```
# create and plot forecast
forecast <- forecast(teleco_model, h = 146)
autoplot(forecast)
```



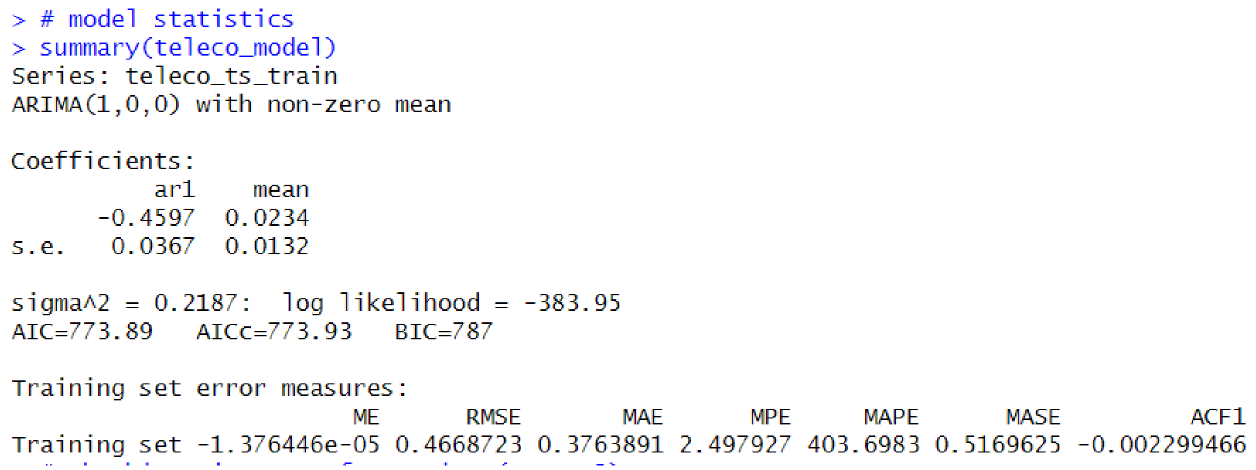
With the forecast plot created, I can overlay the data points from the test set to get a visual impression of how the forecast has performed.

```
# plot forecast with test data overlay
autoplot(forecast) +
  geom_line(
    aes(
      x = as.numeric(time(teleco_ts_test)),
      y = as.numeric(teleco_ts_test)
    ),
    col = "red"
  )
```



I can also use the summary() function to get some metrics indicating the model's performance.

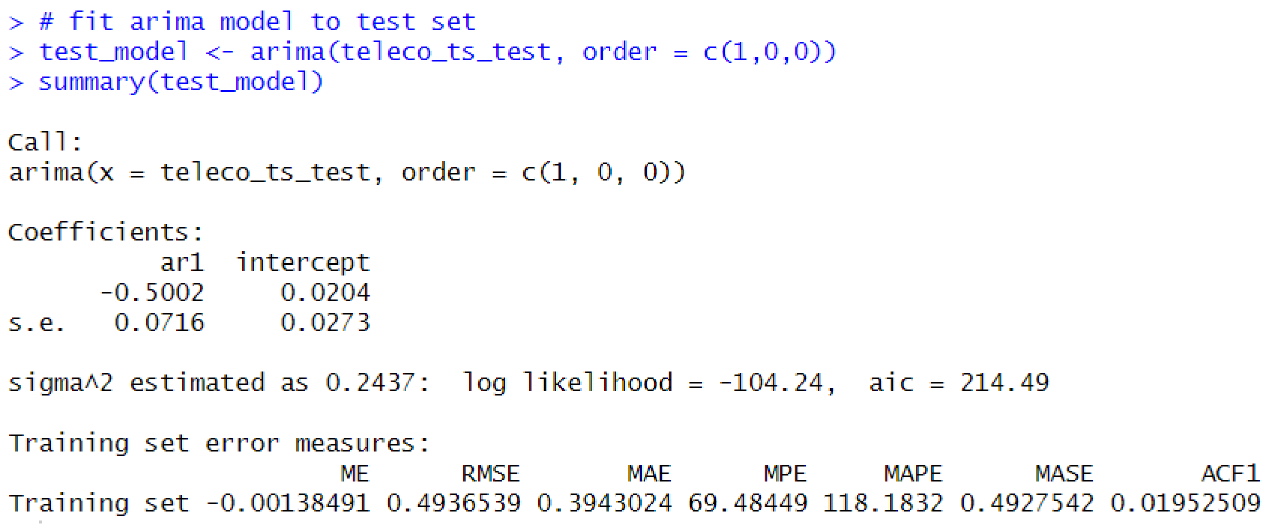
```
# model statistics
summary(teleco_model)
```



```
> # model statistics
> summary(teleco_model)
Series: teleco_ts_train
ARIMA(1,0,0) with non-zero mean

Coefficients:
          ar1    mean
      -0.4597  0.0234
s.e.   0.0367  0.0132

sigma^2 = 0.2187:  log likelihood = -383.95
AIC=773.89   AICc=773.93   BIC=787

Training set error measures:
                        ME      RMSE       MAE      MPE     MAPE      MASE         ACF1
Training set -1.376446e-05 0.4668723 0.3763891 2.497927 403.6983 0.5169625 -0.002299466
```

Fitting the ARIMA model to the test set and reviewing the summary gives me additional data on the model's performance.

```
# fit arima model to test set
test_model <- arima(teleco_ts_test, order = c(1,0,0))
summary(test_model)
```



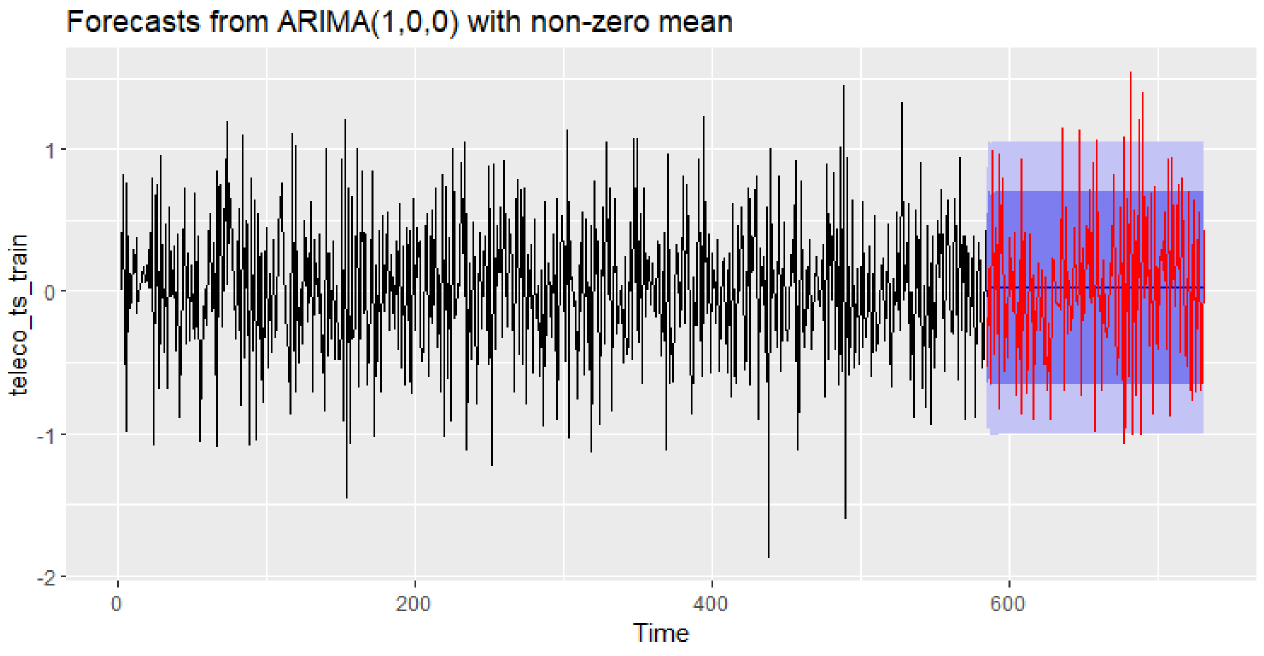
```
> # fit arima model to test set
> test_model <- arima(teleco_ts_test, order = c(1,0,0))
> summary(test_model)

Call:
arima(x = teleco_ts_test, order = c(1, 0, 0))

Coefficients:
          ar1  intercept
      -0.5002     0.0204
s.e.   0.0716     0.0273

sigma^2 estimated as 0.2437:  log likelihood = -104.24,  aic = 214.49

Training set error measures:
                      ME      RMSE       MAE      MPE     MAPE      MASE       ACF1
Training set -0.00138491 0.4936539 0.3943024 69.48449 118.1832 0.4927542 0.01952509
```

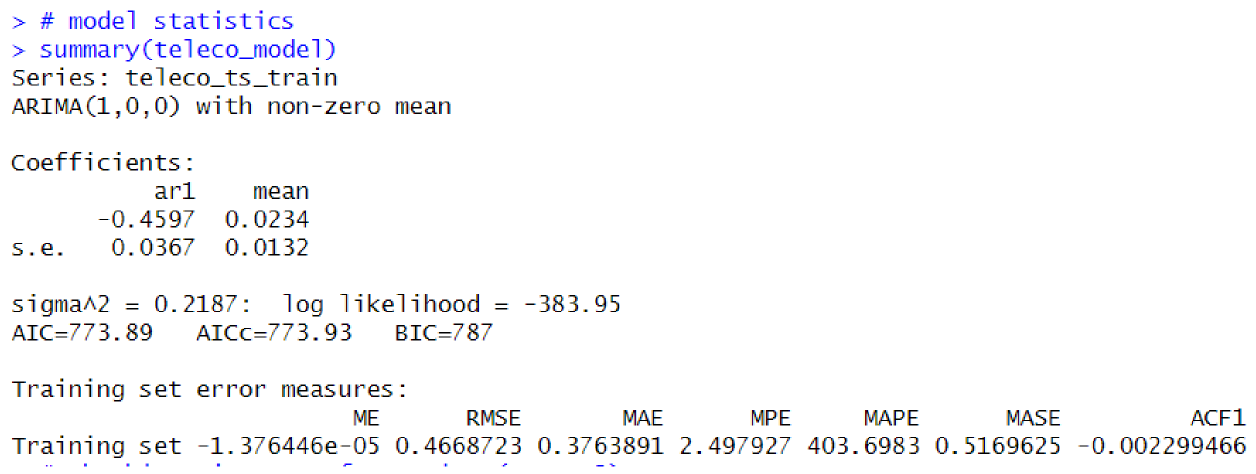
## 5 Part V: Data Summary and Implications

ARIMA model selection was carried out via the auto.arima() function. As stated previously, the result indicated a model with characteristics 1,0,0 is appropriate:

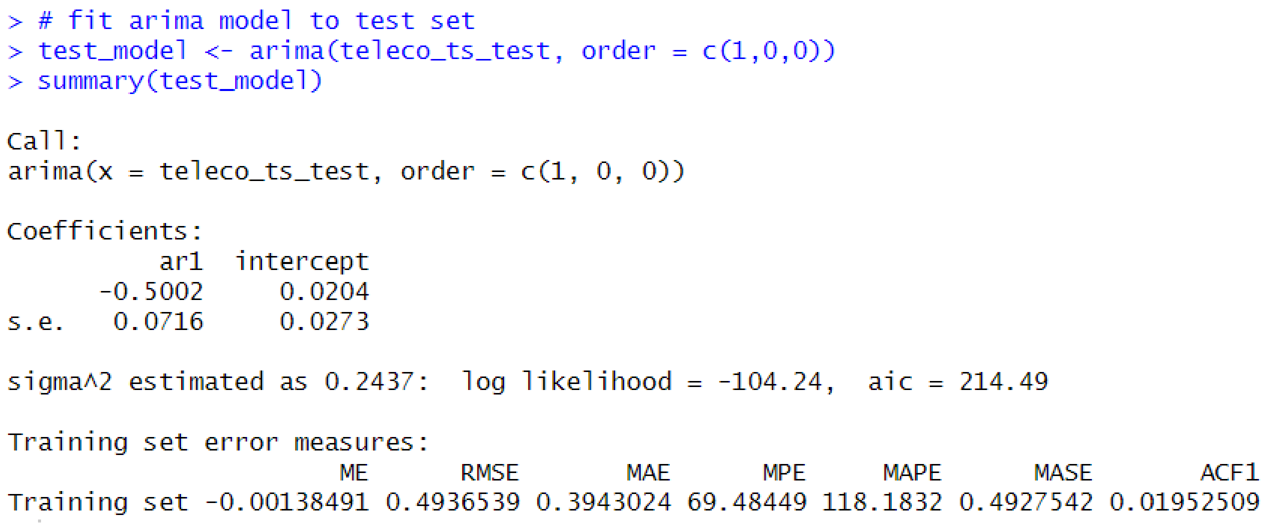
- (AR) 1 period of lag
- (I) 0 differencing adjustment (likely because I already differenced the time series)
- (MA) 0 moving average window (expected, as moving average is not needed for stationary series)

The prediction interval of my forecast was one prediction per day, for 146 days (the length of the test set). I evaluated the model by both visually observing the forecast compared with actual data points, and by comparing the error metrics of the model when applied to both train and test data sets.





```
> # model statistics
> summary(teleco_model)
Series: teleco_ts_train
ARIMA(1,0,0) with non-zero mean

Coefficients:
          ar1    mean
      -0.4597  0.0234
s.e.   0.0367  0.0132

sigma^2 = 0.2187:  log likelihood = -383.95
AIC=773.89   AICc=773.93   BIC=787

Training set error measures:
                        ME      RMSE       MAE      MPE     MAPE      MASE         ACF1
Training set -1.376446e-05 0.4668723 0.3763891 2.497927 403.6983 0.5169625 -0.002299466
```



```
> # fit arima model to test set
> test_model <- arima(teleco_ts_test, order = c(1,0,0))
> summary(test_model)

Call:
arima(x = teleco_ts_test, order = c(1, 0, 0))

Coefficients:
          ar1  intercept
      -0.5002     0.0204
s.e.   0.0716     0.0273

sigma^2 estimated as 0.2437:  log likelihood = -104.24,  aic = 214.49

Training set error measures:
                      ME      RMSE       MAE      MPE     MAPE      MASE       ACF1
Training set -0.00138491 0.4936539 0.3943024 69.48449 118.1832 0.4927542 0.01952509
```

Visually, the forecast appears to have somewhat accurate predictions, with most real data points falling within the 80% confidence interval, some falling very close to precise prediction line plot, and a smaller number exceeding the 50% confidence interval, representing some extreme variance on those points.

The RMSE for both the fitted train set and test set are both close to 0 and similar to each other, supporting a judgement of the model being consistent and valuable for prediction. The MAE for both sets is also similar, and there is little variance when comparing both sets' ACF1.

My recommendation would be that the model is useful for forecasting but may also benefit from additional tuning. Some adjustments during the data preparation steps, like log transformation, may ultimately result in a better performing model and a more reliable forecast.

## 6 Web Sources

https://community.rstudio.com/t/how-to-compare-a-forecast-model-to-actual-data-and-what-is-uncertainty/23598/3

https://rpubs.com/Mentors_Ubiqum/Tutorial_ARIMA_models

https://stackoverflow.com/questions/28847118/how-can-i-export-a-time-series-model-in-r

https://www.youtube.com/watch?v=iwRtpJDDw5M

https://arxiv.org/pdf/2107.13462.pdf

https://openforecast.org/adam/ETSConventional.html

## 7 References

Monigatti, L. (2022, August 2). *Interpreting ACF and PACF Plots for Time Series Forecasting.* Towards Data Science. https://towardsdatascience.com/interpreting-acf-and-pacf-plots-for-time-series-forecasting-af0d6db4061c

Pandian, S. (2023, February 20). *Time Series Analysis and Forecasting | Data-Driven Insights (Updated 2023).* Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-to-time-series-analysis/