Predictive Analytics Homework 7: Forecasting

Goal

We'll return to the task of predicting bike sharing rides. Recall that we noticed that our models performed poorly because they did not account for *distribution shift*: the bike-share system got more popular overall in 2012, but our models from 2011 couldn't handle that.

In this assignment, we'll address the problem in two ways:

1. We'll start halfway into 2012 when making our forecasts. That way we'll have some evidence about what's happening in 2012.
2. We'll apply some modeling approaches that are designed to work with time series.

Recall that none of the models achieved an MAE under 300 for 2012 in the previous assignment. By using these approaches, we'll be able to cut our error by at least a third.

Getting Started

I used the following setup chunk when producing this assignment:

```r
knitr::opts_chunk$set(echo = TRUE)
library(tidyverse)
library(rsample)
```

```r
library(fpp3)
theme_set(theme_bw())
```

Include the following code to load the data file.

```r
daily_rides <- readRDS(url("https://cs.calvin.edu/courses/info/602/14forecast2/hw/bikeshare-
```

Quick EDA (No Exercises)

We spent too little time on the EDA in the previous assignment; here's a few quick additions for this one:

First, `skimr` is your friend. Check it out:

```r
skimr::skim(daily_rides)
```

Table 1: Data summary

Name	daily_rides
Number of rows	731
Number of columns	15
Column type frequency:	
Date	1
factor	7
numeric	7
Group variables	None

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
date	0	1	2011-01-01	2012-12-31	2012-01-01	731

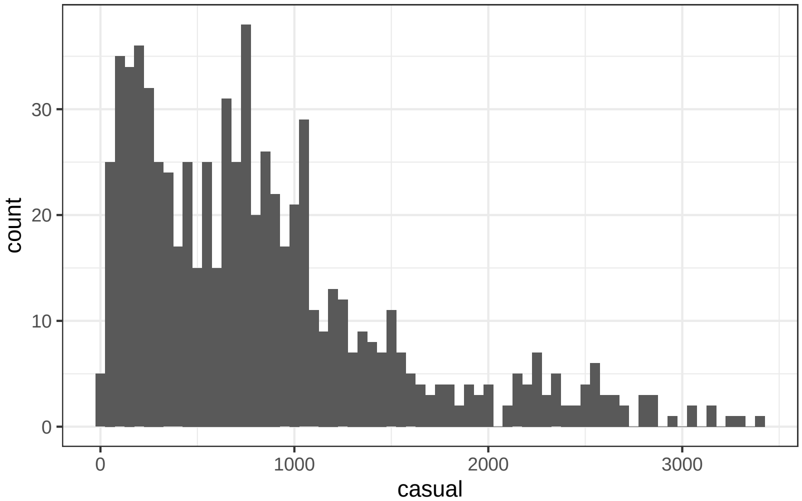
Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
season	0	1	FALSE	4	3: 188, 2: 184, 1: 181, 4: 178
year	0	1	FALSE	2	201: 366, 201: 365
holiday	0	1	FALSE	2	0: 710, 1: 21
weekday	0	1	FALSE	7	Sat: 105, Sun: 105, Mon: 105, Tue: 104
workingday	0	1	FALSE	2	wor: 500, wee: 231
weathersit	0	1	FALSE	3	1: 463, 2: 247, 3: 21
month	0	1	FALSE	12	1: 62, 3: 62, 5: 62, 7: 62

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
temp	0	1	15.28	8.60	-5.22	7.84	15.42	22.80	32.50	
atemp	0	1	15.31	10.76	-10.78	6.30	16.12	24.17	39.50	
hum	0	1	0.63	0.14	0.00	0.52	0.63	0.73	0.97	
windspeed	0	1	0.19	0.08	0.02	0.13	0.18	0.23	0.51	
casual	0	1	848.18	686.62	2.00	315.50	713.00	1096.00	3410.00	
registered	0	1	3656.17	1560.26	20.00	2497.00	3662.00	4776.50	6946.00	
cnt	0	1	4504.35	1937.21	22.00	3152.00	4548.00	5956.00	8714.00	

When starting an analysis, it's really useful to look at the distribution of your response variable. Let's **plot the histogram of `casual`**.

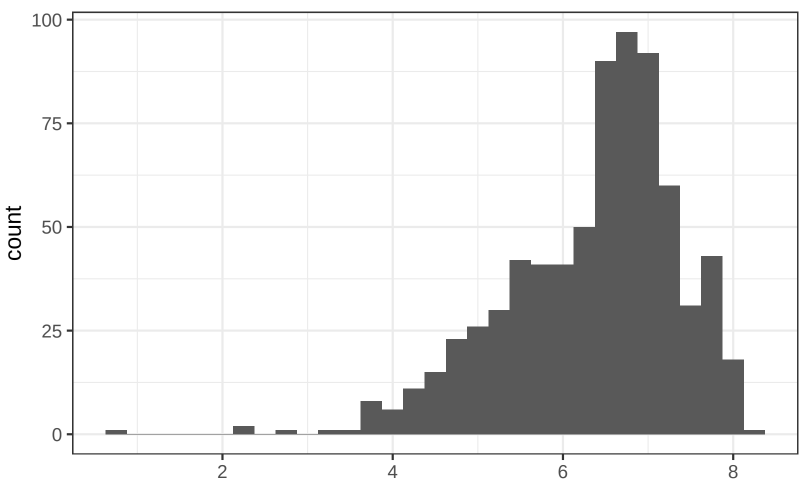


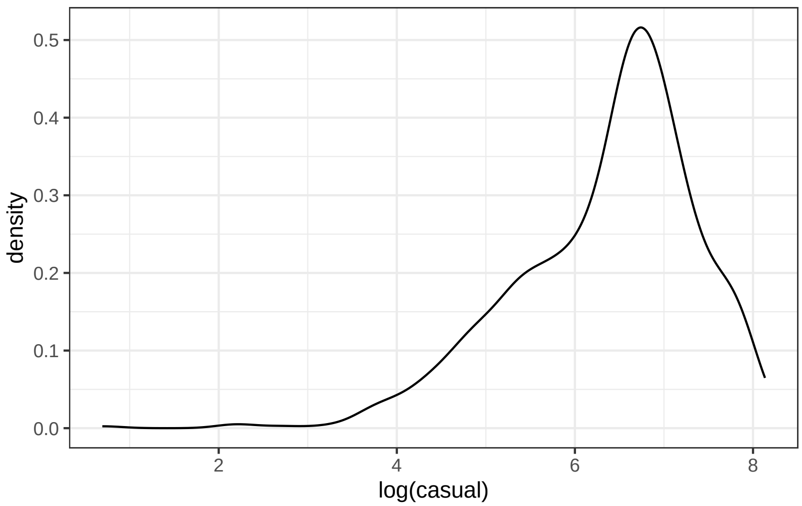
Notice the long tail of high-ridership days. These can often throw off a model. Also notice that there are no ride counts below 0, as should be expected for data of *counts*. But a linear model might end up predicting a negative value, which wouldn't make sense.

We can address both of those problems by transforming our response variable. Instead of trying to predict `casual`, let's try to predict `log(casual)`. That way we'll never predict a negative value for `casual`.

Now, we'll **plot the distribution of `log(casual)`.** Use a histogram or density plot.

Other transformations may work better, but log-transforming data is simple and common. In a linear model, it means that each term contributes *multiplicatively* to the prediction: each degree of temperature increase will *mulitply* the ride count by something (i.e., it goes up by x *percent*), rather than adding to it.





Notice that the data distribution is less skewed, and the clear outliers are now actually the low-ridership days, which may not matter as much anyway. Going forward, let's use this transformed result. (Note that we won't need to explicitly make a new column; the modeling functions will handle the transformation and inverse transformation for us.)

Time Series EDA

We'll first make our data into a time series:

```
rides_ts <- daily_rides %>%
  as_tsibble(index = date)
```

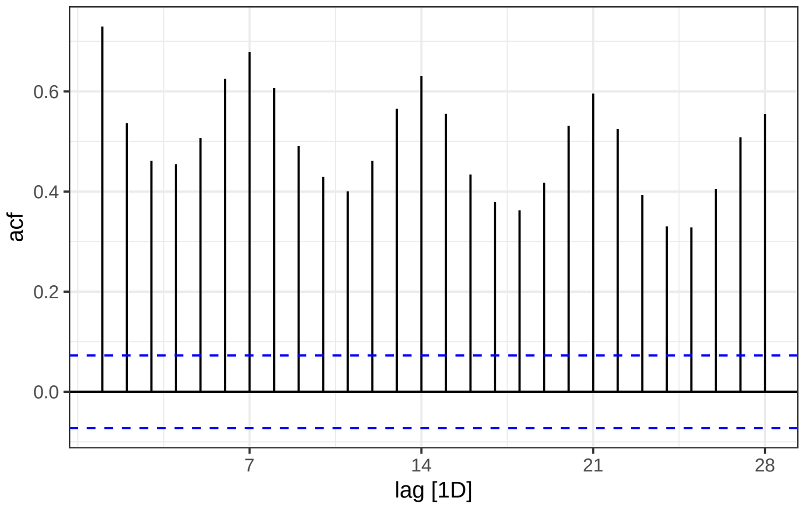
Autocorrelation

Exercise 1 (5pt) Here's a plot of the autocorrelation function (ACF) of `log(casual)`.

a. **What peaks do you observe in the ACF?**
b. **What do those peaks tell us about the data?**

```
rides_ts %>%
  ACF(log(casual)) %>%
```

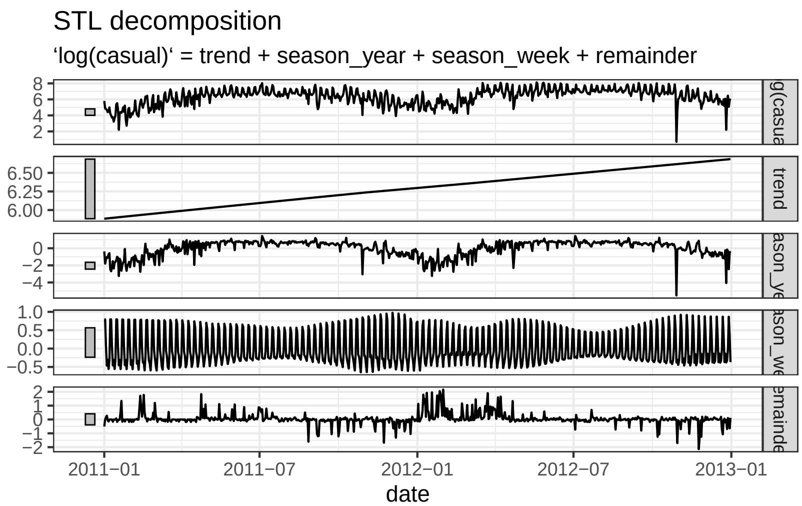
```
  autoplot()
```



Decomposition

Let's decompose the time series into trend, seasonal, and remainder.

```
rides_ts %>%
  model(
    STL(log(casual) ~ trend() + season(), robust = TRUE)
  ) %>%
  components() %>%
  autoplot()
```



Exercise 2 (2pt) Based on what you know or can look up about bike riding, **explain why this time series has both yearly and weekly seasonal components**.

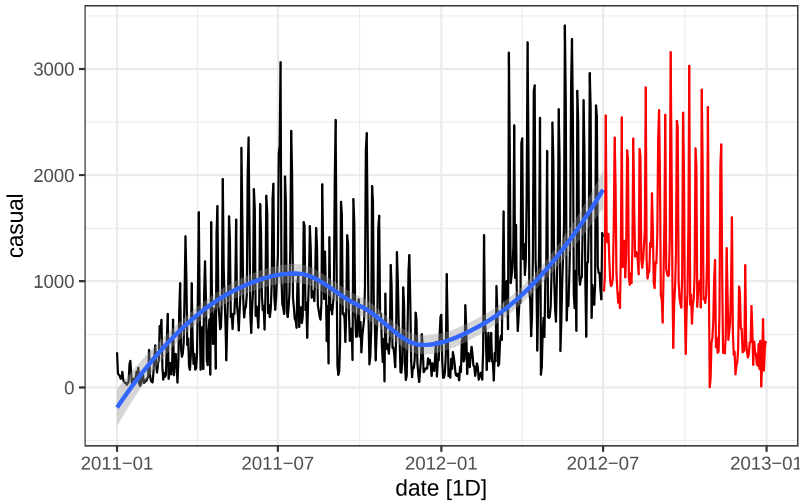
Train-test Split

We'll take the first 3/4 of the data as training set, and last 1/4 as test. We're *not* shuffling the data first; we want to make sure that the test set is actually the *future*.

```
splits <- rides_ts %>% initial_time_split(prep = 3/4)
rides_train <- training(splits)
rides_test <- testing(splits)
```

Let's plot the data, marking the test set separately. I'll give you the code here since it's slightly non-obvious.

```
rides_train %>%
  autoplot(casual) +
    geom_smooth() +
    autolayer(rides_test, casual, color = 'red')
```



Exercise 3 (4pt) Based on what we've seen about the data so far, plus what you might know about bike riding:

a. **What factors work in favor of making good predictions** for the future data?
b. What factors **work against** making good predictions?

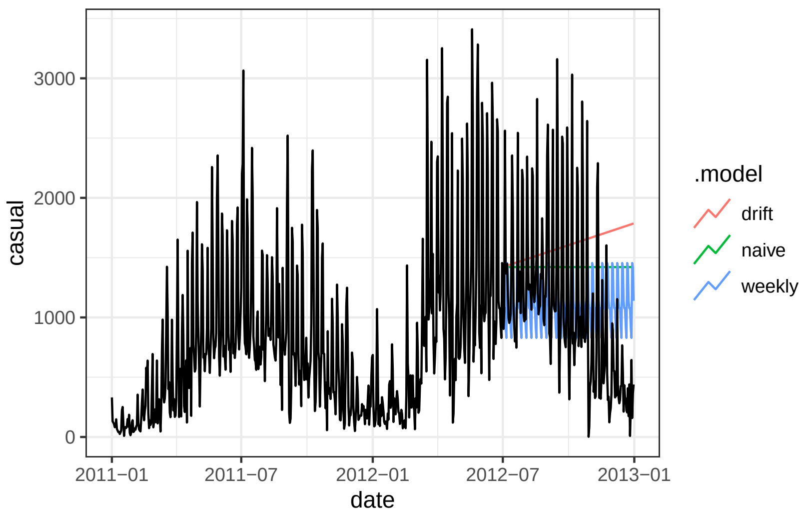
Simplistic Models

Use the following code to predict the ridership on the test set using naive ("random walk"), drift, and weekly seasonal models.

```
# Fit the models.
models <- rides_train %>% model(
  naive = NAIVE(casual),
  drift = NAIVE(casual ~ drift()),
  weekly = SNAIVE(casual ~ lag(7))
)

# Get forecasts on the test set.
forecasts <- models %>%
  forecast(rides_test)
```

```r
# Plot forecasts against the actual data.
# Change `rides_ts` to `rides_test` if you want to zoom in.
# We use `level = NULL` to hide prediction intervals.
forecasts %>%
  autoplot(rides_ts, level = NULL)
```



Exercise 4 (4pt) **Describe the difference between the forecasts made by the three models.** Also, identify which one seems to fit the data best; explain why it does so.

Exercise 5 (2pt) The following code is used to *quantify* the accuracy of the models. **Compare the accuracy numbers you get here with what seemed to fit the data best in the previous exercise.**

```r
forecasts %>%
  accuracy(rides_test) %>%
  select(.model, MAE, MAPE, RMSE)
```

```
# A tibble: 3 x 4
  .model   MAE  MAPE  RMSE
  <chr>  <dbl> <dbl> <dbl>
1 drift   796.  719.  915.
2 naive   663.  600.  768.
3 weekly  492.  463.  617.
```

Regression Models

Exercise 6 (1pt): Copy the code from above but change the models as follows. **First**, keep *only the best-performing* of the simplistic models. Then, **add the following two models**:

- `lm = TSLM(casual ~ temp + workingday + month)` and (don't forget the commas between models)
- `lm_trend = TSLM(casual ~ temp + workingday + month + trend())`

Compare the performance of these models with the previous models based on both the prediction plots and the quantitative error metrics.

> **Note 1**: These declare simple linear models with identical structure to what we fit in the previous homework. There's nothing "time series" about them, except that they do distinguish weekends from weekdays and let the prediction change between months.
>
> **Note 2**: Try to keep the code clean and simple; remove extraneous comments. I used the same variable names so I wouldn't mistakenly forget to, say, change `models` to `models2` when making forecasts, but you may do whatever you prefer as long as it gets the correct results.

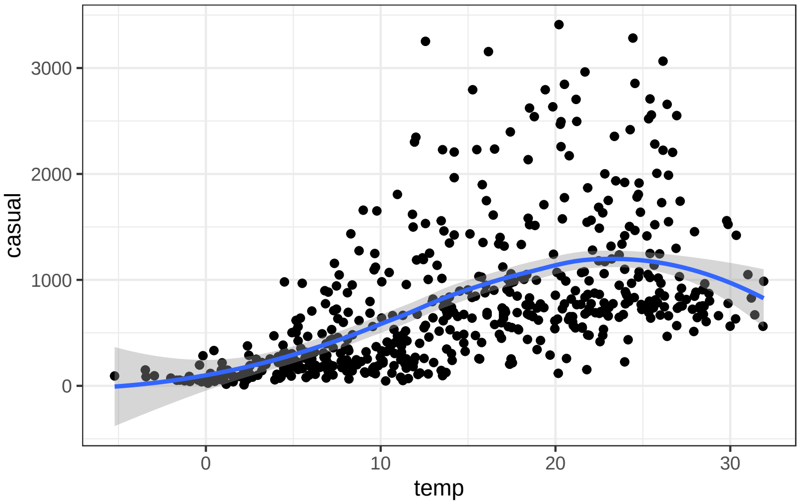
Transformed Targets

Exercise 7 (1pt) Again, copy the above, keeping only the best-performing model (according to MAE). Add a second version of the model, replacing `casual ~` with `log(casual) ~`, so the model works on the log scale.

Exercise 8 (4pt): Looking at the plot, **explain why the model with the best MAE is not the same as the model with the best MAPE**. Which model would you trust more?

More complex models

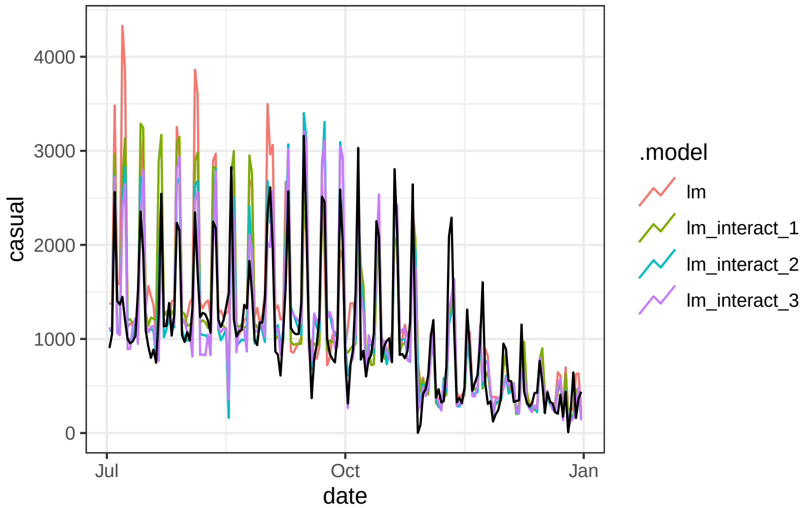
Now, notice that the effect of temperature is not linear.



So let's add some interaction terms. Keep only the `log(casual) ~ temp + workingday + month + trend()` model from above, and make more models, with the following changes:

- Model 1: instead of `temp`, use `temp * atemp`. This is like adding a temp-squared term, but also allows us to pull in the feels-like data from `atemp` (*apparent* temperature).
- Model 2: same, but `temp * atemp * hum` instead.
- Model 3, same, but also add `weathersit` (the type of weather, e.g., sunny or stormy)

```
# A tibble: 4 x 4
  .model          MAE  MAPE  RMSE
  <chr>         <dbl> <dbl> <dbl>
1 lm             343. 258.   537.
2 lm_interact_1  296. 250.   459.
3 lm_interact_2  263. 184.   380.
4 lm_interact_3  261.  90.5  363.
```



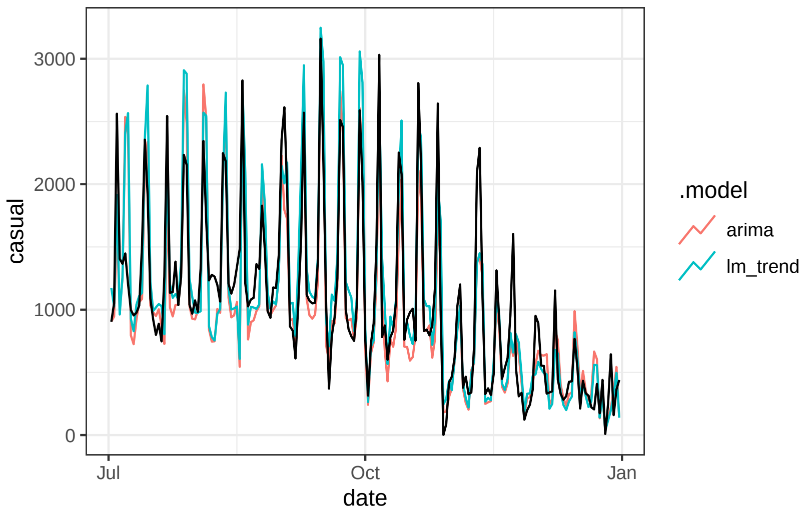
Exercise 9 (2pt): **Describe how the accuracy has changed.**

Dynamic Regression Models

As a preview of what you'd study if you got further into time series analysis than we'll get here, try adding the following model:

`arima = ARIMA(log(casual) ~ temp * hum * atemp + weathersit + holiday + PDQ(period = "1 week"))`

```
# A tibble: 2 x 4
  .model      MAE  MAPE  RMSE
  <chr>     <dbl> <dbl> <dbl>
1 arima      223.  75.5  317.
2 lm_trend   240.  94.6  334.
```



Exercise 10 (5pt): **Reflect on the accuracy results you've seen in this exercise.** Do you think we can make good forecasts? Our latest models look good, but: do you have any reason *not* to be confident in them? **Think about what the habits of *validation* we learned in previous units.**