Bike-Sharing Data Analysis: Prediction of Daily Bike Rental Counts Based on Multiple Linear Regression

Final Project Report · MA 575 Fall 2021 · C3 · Team #2

Ali Taqi, Hsin-Chang Lin, Huiru Yang, Ryan Mahoney, Yulin Li

12/10/2021

Abstract

In this project, the following question is to be answered: If we have the past history of bike rental counts as well as records of environmental and seasonal conditions, how and how well could we predict the bike rental counts in the future? In this project, such questions are approached by predictive modeling of daily bike rental counts from a 2011-2012 Bike Sharing dataset [1]. The daily bike rental counts are predicted with models based on Multiple Linear Regression (MLS) using the environmental and seasonal variables as predictors. The initial goal of this project is to train the model using only the 2011 data, and then validate the prediction power of the model on the 2012 data. Given the limited time span of available training data, issues are found in the validation process using the 2012 data; the impact of user base on the future predictions is brought to our attention. The initial models are then revisited and corrected to account for the effect of user base. The refined models are expected to have better prediction powers than the initial MLS models, but a full validation would require further availability of bike rental data.

1 Introduction

Bike sharing has become a world-wide phenomenon. Optimization of inventories and dynamic reallocation of bike-sharing resources are of growing interests from both a business and an environmental point of view. Both of these tasks require accurate predictions of bike rental behaviors at least on the daily level.

In this project, we strive to answer the following question:

- If we have the past history of bike rental counts as well as records of environmental and seasonal conditions, how and how well could we predict the bike rental counts in the future?
- In particular, how and how well could we predict for the next whole year, and what about for the next few days?

Such questions are approached by predictive modeling of daily bike rental counts from a 2011-2012 Bike Sharing dataset [1]. The modeling approach is based on Multiple Linear Regression (MLS), and the daily bike rental counts are predicted using the environmental variables (e.g., weather conditions) and seasonal variables (e.g., holiday schedules) as predictors.

2 Background[1]

The aim of this project is to achieve the best model(s) that can be obtained from past data for the use of predictions for the future, preferably predictions one year ahead. To validate the prediction power of models under this setting, the basic goal of this project is to train all models using only the 2011 data, and then test them on the 2012 data.

The response variable to be predicted is the **daily** bike rental count. In the dataset being studied [1], the following 3 types of bike rental counts are recorded:

1. the count of bike rentals by **casual** users
2. the count of bike rentals by **registered** users
3. the **total** count, which is the sum of casual count and registered count.

[1] Ali Taqi: modeling and analysis, figures and plots, writing; Hsin-Chang Lin: modeling and analysis; Huiru Yang: modeling and analysis; Ryan Mahoney: modeling and analysis, figures and plots; Yulin Li: modeling and analysis, figures and plots, writing

Two main types of predictors are included in the dataset, the environmental ones and the seasonal ones[2]:

1. **Environmental Variables[3]:** `weathersit`, `temp`, `atemp`, `hum`, `windspeed`
2. **Seasonal Variables[4]:** `yr`, `mnth`, `season`, `holiday`, `weekday`, `workingday`

3 Modeling & Analysis

3.1 Pre-processing

3.1.1 Type Conversion

To be noticed, the value of categorical variables indicates type labels and has very limited physical meaning in the magnitude of those values, which thus cannot be used in the same way as the numeric variables in MLS models. The categorical variables therefore needs to be recognized before the actual modeling process and to be carefully handled.

The below variables are interpreted as Boolean variables and are transformed into `logical`-type variables in `R`:

- `holiday` (holiday or not)
- `workingday` (working day or not)

The below variables are interpreted as categorical variables and are transformed into `factor`-type variables in `R`:

- `season` (season, from 1 to 4)
- `yr` (year, from 0 to 1)
- `mnth` (month, from 1 to 12)
- `weekday` (weekday, from 0 to 6)
- `weathersit` (weather type, from 1 to 4)[5]

3.1.2 Value Conversion

The recorded values of `temp` (measured temperature), `atemp` (feeling temperature), `hum` (measured humidity) and `windspeed` (measured wind speed) in the data set being studied here are the normalized ones; all recorded values are the ones that have been divided by the maximum of measured values [1]. For example, the recorded values of `temp` (measured temperature) are obtained by dividing the original measured values by 41 (max) and are thus all less than or equal to 1.

In this project, these normalized records are scaled back to their original values for the sake of easier interpretations. For example, the recorded values of `temp` (measured temperature) are multiplied by 41 (max) in the pre-processing process, which recovers the original scale of temperatures in Celsius.

3.2 Variable Selection

3.2.1 Response Transformation

Notably, the behaviors of rental counts from different user types are considerably different.

1. **Patterns with weekdays** (see Figure 1[6], 2[7]): Over the time span of a week, the casual count usually reaches its minimum in the middle of a week (grey dots mostly) and its maximum on weekends (green dots mostly), while the registered count does the opposite.

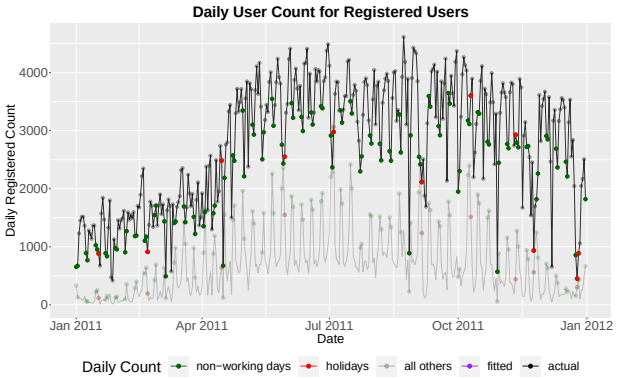


Figure 1: 2011 Registered User Counts

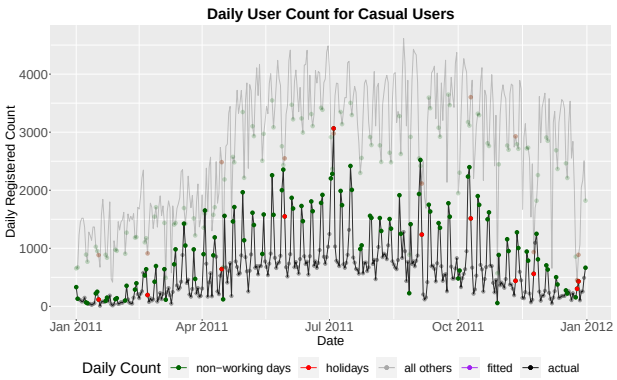


Figure 2: 2011 Casual User Counts

[2]For a full explanation, see Appendix.

[3]Variable meanings (in order): weather type, measured temperature, feeling temperature, humidity, wind speed

[4]Variable meanings (in order): year, month, season, holiday, weekday

[5]Note: larger index indicates worse weather

[6]Transparent plots of the casual counts in the background for easier comparison.

[7]Transparent plots of the registered counts in the background for easier comparison.

2. **Patterns with temperatures** (see Figure 3): On **working** days, the casual count seems more linear in temperature (`atemp` and `temp`), while the registered count seems to be (at least) quadratic most of the time.

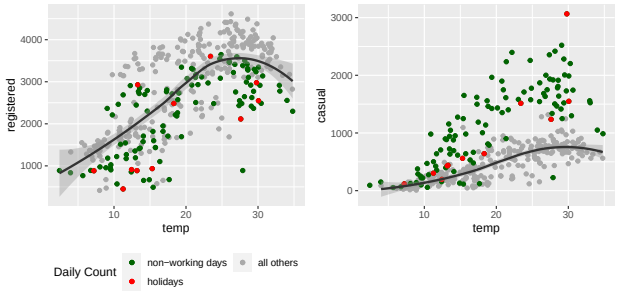


Figure 3: Registered Counts v.s. Temperature (Left Subplot); Casual Counts v.s. Temperature (Right Subplot)

We therefore expect that the registered counts and casual counts will follow different distributions and should thus be predicted by separate models. Furthermore, for the casual count, avoiding unnecessary higher other terms has the benefit of more stable computations and model structures. The prediction of total counts will then be obtained by adding the predicted registered counts and predicted casual counts together.

3.2.2 Predictor Selection

Given the predictive nature of modeling in the current problem setting, the predicted response is of greater interests than the actual value of the parameter estimates, as opposed to that in an inference task. This, to some degree, relaxes the constraint forbidding colinearity in the predictors, since colinearity will only lead to instability in the parameter estimates but not in the predictions; however, we should still seek to minimize colinearity at least in our beginning model, which would lead to clearer model structures as well as better interpretability of model statistics at the early stage of modeling, which could provide us clearer directions in the improvement process that follows.

With the above considerations in mind, the predictors in the beginning model are selected following the 2-step approach below:

1. The scatter plot matrix for the whole set of variables are plotted for the 2011 training dataset, and all predictors that seem to be significant, i.e., predictors with which the response variable (daily rental count, `cnt`) exhibits a notable visual pattern, are selected.
2. From the selected predictors above, all the highly correlated predictors are removed. Within a group of correlated predictors, only the one that has the largest correlation coefficient with the response variable as well as having the strongest causal relation with the response (in the intuitive sense) will be kept.

It is important that the investigation is done for all predictors for the sake of minimal loss of information. Note that in practice, the whole set of predictors is divided into two groups, environmental and seasonal, and plotted separately, for better readability of the large scatter plot matrices. The separation is justified by the fact that most environmental variables, such as weathers, are expected to be independent of the seasonal variables, such as weekdays and holiday schedules.

At last, the above process leaves us with a small subset of the very core predictors for our beginning model: `weathersit`, `atemp` and `weekday`.

3.3 Initial Modeling[8]

In the model building and selection process, we start from the simplest models, which have the minimal number of predictors all in the additive form, as the beginning models.

3.3.1 Beginning Models

1. For **total count**:

$$\text{cnt} \sim \text{wkngday} + \text{weathersit} + \text{atemp} + \text{atemp}^2$$

Model	rmse	n-rmse	% Error	cv-rmse
2011 tot	717.93	0.52	25.83	730.8

Table 1: Evaluating the Beginning Total Model

2. For **registered count**:

$$\text{reg} \sim \text{wkngday} + \text{weathersit} + \text{atemp} + \text{atemp}^2$$

3. For **casual count**:

$$\text{cas} \sim \text{wkngday} + \text{weathersit} + \text{atemp}$$

The percentage error of 2011 predicted total counts computed by the sum of the registered and casual models is less than that computed by the single total model (see Table 1, 2), which demonstrates the

[8]Models evaluated on: 1. RMSE (`rmse` in table); 2. normalized RMSE (RMSE divided by residual standard deviation) (`n-rmse` in table); 3. percentage error (`% Error` in table); 4. LOOCV RMSE (`cv-rmse` in table)

Model	rmse	n-rmse	% Error	cv-rmse
2011 cas	309.97	0.56	72.87	314.86
2011 reg	584.55	0.52	25.37	594.36
2011 tot	722.15	0.52	25.44	734.18

Table 2: Evaluating the Beginning Registered and Casual Models

power of separate modeling for registered and casual users. Therefore, we will continue with the scheme of separate modeling in the later part.

3.3.2 Procedures

Starting from the beginning models, we perform model building and selection in an iterative manner, for the registered and casual counts separately:

- Starting from a (relatively) simpler version of the model, the 2011 fitted response using this model is plotted along with the actual response against time (see, for example, Figure 4, 5); the numeric metrics (RMSE, normalized RMSE, percentage error and LOOCV RMSE) are obtained as well (see, for example, Table 2).

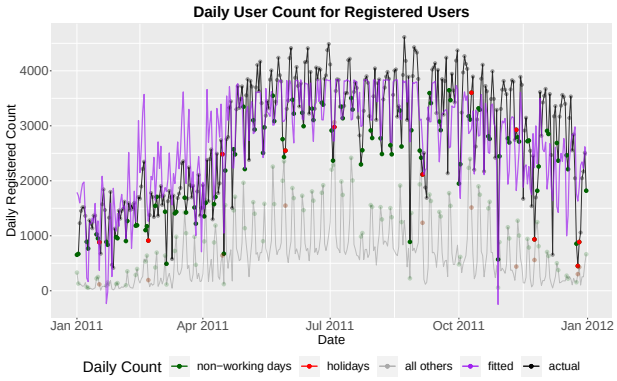


Figure 4: 2011 Fitted Registered Count v.s. Actual (Beginning Model)

- From the 2011 fitted versus actual plot, patterns in the biases can usually be visually recognized; combined with commonsense, this provides us with ideas of new variables potentially needed to account for the unexplained biases.

For example, for the beginning models of registered and casual counts, both response variables are being consistently overestimated in the 2011 spring and underestimated in the later part of the year (see Figure 4, 5), which indicates that the users' response to weather and temperature might differ across seasons. This makes sense because each label of the weather

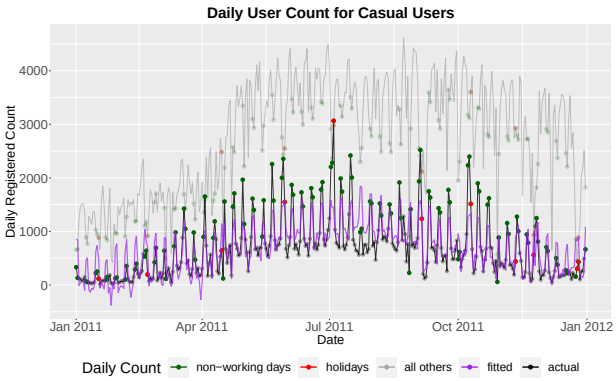


Figure 5: 2011 Fitted Casual Count v.s. Actual (Beginning Model)

type includes several different kinds of weathers (e.g., slightly snowy and slightly rainy days are both labeled as 3), which could cast different level of difficulties on biking activities, and it is still insufficient to fully distinguish between those weathers given only temperature information. Additionally, the level of biases also differ considerably between workdays (grey dots) and weekends (mostly green dots), which indicates that the weekday variables might also be needed.

- The new variables, one at a time, are then added to the simple model to create a more complex model. Both additive terms and interactive terms will be attempted. Then the version with the most significant improvement, according to the fitted versus actual plot and the numeric metrics, will be kept for the next round.

In this iterative process of modeling, we look at the Leave-One-Out-Cross-Validation (LOOCV) RMSE as a proxy for the extent of overfitting. The model building process stops when the LOOCV RMSE starts to ramp up as model complexity increases, typically becoming considerably larger than the RMSE (compared to the previous models).

Note that in this process, the model statistics (e.g., p-values) are also checked but are not relied on as much, out of the following two considerations:

1. There is no guarantee in the normality of residual distributions as well as correct model forms, in which case the summary statistics might thus be invalid at all, especially in the intermediate stage of modeling with the incomplete models.

2. As the model becomes more and more complex in this process, the colinearity issues worsen, which might weaken the significance of inter-correlated predictors (i.e., it might turn out that none of

those predictors are indicated as significant in the summary table), while this is not to say that none of those predictors are necessary for the model to have more accurate predictions.

3.3.3 Final Models

At the end of the iterative modeling process, we arrive at the following final models[9]:

1. For **registered count**:

$$\text{reg} \sim \text{holiday} + \text{season:weathersit} + \text{season:workingday:atemp} + \text{season:workingday:atemp}^2$$

2. For **casual count**:

$$\text{cas} \sim \text{holiday} + \text{season:weathersit} + \text{season:workingday:atemp}$$

Model	rmse	n-rmse	% Error	cv-rmse
2011 cas	238.01	0.43	42.57	264.81
2011 reg	392.24	0.37	14.81	441.33
2011 tot	516.80	0.37	15.72	589.74

Table 3: Evaluating the Final Registered and Casual Models

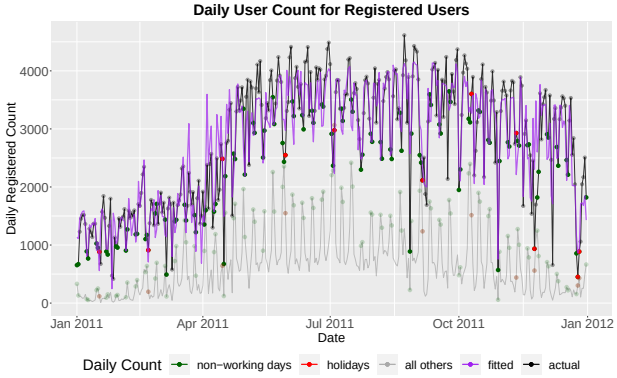


Figure 6: 2011 Fitted Registered Count v.s. Actual (Final Model)

The final models are built with considerations of real-life experience and commonsense. For example, the interaction terms between feeling temperature, working day and seasons indicate people's different reaction to temperature change under different weathers on working days and on weekends.

From this point on, adding any other variable to the model will result in a rise in the LOOCV RMSE,

[9]For a more complete list of attempted models, see Appendix.

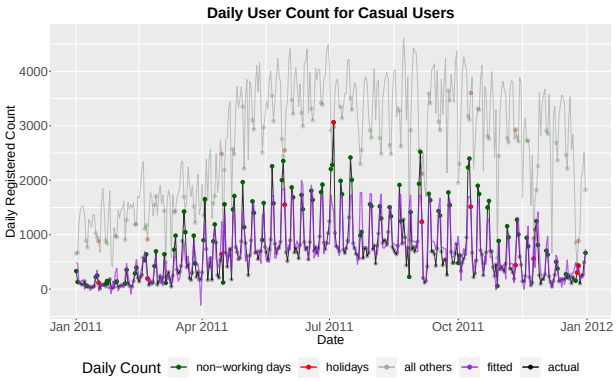


Figure 7: 2011 Fitted Casual Count v.s. Actual (Final Model)

which indicates an overfitting issue that weakens the predictive power of the model. Notably, using the `weekday` variable in place of the `workingday` variable also worsens the performance.

The final models achieve a ~10% improvement in the prediction percentage error in 2011 daily total counts as well as significant improvements in all the other metrics. The LOOCV RMSE is only ~15% higher than the training RMSE, indicating that the overfitting issue is still at a moderate level. Also, the systematic biases in the beginning models, as mentioned in the former section, have been alleviated (see Figure 6, 7).

A diagnostic analysis (see Figure 8, 9) indicates that the constant-variance assumption and normality assumption of the registered model are mostly sound, while those of the casual model are more in doubt. Accordingly, the prediction errors of the casual model is also at a higher level than the registered model (see normalized RMSE and percentage error in Table 3). This situation can be understood intuitively, as the behavior of registered users is generally more regular and can be better captured by the simple MLS model, thus more predictable. Note that the least-square approach for the casual model is still valid, because the residuals are still uni-modal at the very least.

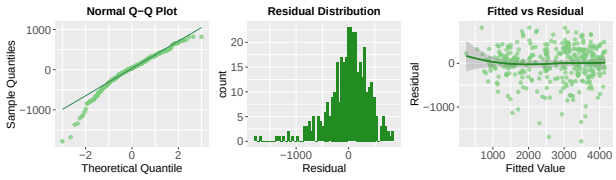


Figure 8: Residual Diagnostic Plots for the Final Registered Model

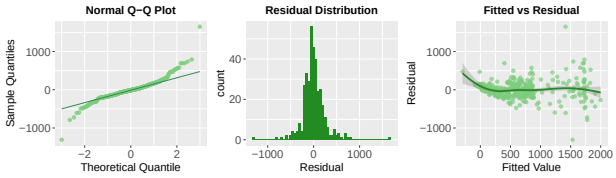


Figure 9: Residual Diagnostic Plots for the Final Casual Model

3.4 Prediction

3.4.1 Initial Predictions

Despite the predictive power demonstrated by cross-validation metrics in the 2011 training set, the final models provide much worse predictions in 2012.

Model	rmse	n-rmse	% Error
2012 cas	506.02	0.67	39.36
2012 reg	1912.08	1.36	39.71
2012 tot	2262.83	1.28	37.94

Table 4: Validating the Final Models (Initial Predictions)

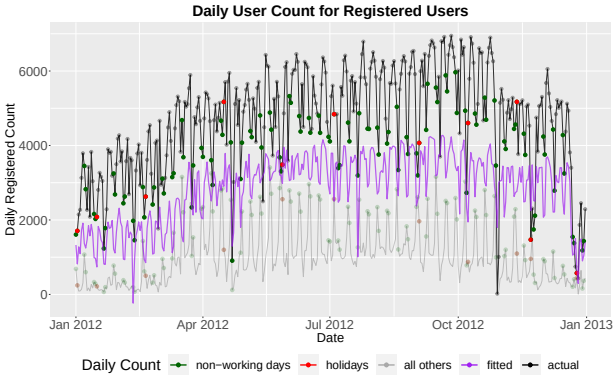


Figure 10: 2012 Predicted Registered Count v.s. Actual (Final Model) (Initial Prediction)

A closer look at the 2012 predictions reveals the fact that the daily rental counts are being consistently underestimated (see Figure 10, 11).

3.4.2 Adjusting Predictions for Growth

A comparison of the 2011 and 2012 rental counts reveals the reason for the systematic prediction error: the 2012 rental counts are generally higher than that in 2011. Furthermore, the 2012 predictions by the models trained in 2011 seem to be generally off by a constant factor.

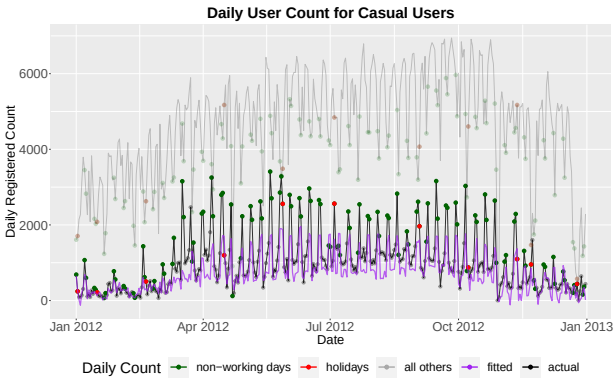


Figure 11: 2012 Predicted Casual Count v.s. Actual (Final Model) (Initial Prediction)

This observation leads to one possible explanation for the prediction error:

Intuitively, the bike-rental behavior of an individual user may be modeled by a random variable, say, X. Then, the total bike rental count from all active users could then be modeled as a sum of independent identically distributed random variables $\sum_{i=1}^{n} X_i$, where $X_i \sim X$ and n is the total number of active bike users. As a consequence, the expected value of total counts will be proportional to the user count, n, and so does the variance.

Note that the effect of user base on the rental counts should be independent of all the other predictors. If the above hypothesis were true, then the correct models for 2012 bike rental counts should be the 2011 models scaled by the growth ratio of user counts.

As such, an estimation of the growth ratio, which we call $\hat{g}$, is very important. By construction, its true value is well estimated by the ratio $\frac{\mathbb{E}_{2012}(C)}{\mathbb{E}_{2011}(C)} \approx (0.608)^{-1} = 1.64$, which is the average counts in 2012 divided those in 2011. Note that this means our user base grew by around 64% between 2011-2012, which is a non-trivial amount. If unaccounted for, that magnitude of growth would (and does in fact) lead to chronic model underfitting.

To verify the above hypothesis, we adjust the initial 2012 predictions by multiplying with $\hat{g}$. As a result, the adjusted predictions for both registered and casual counts now almost perfectly match the actual values (see Figure 12, 13). This provides a strong evidence for our accounts for the role of user base.

To generalize for future predictions in practice, note that this modeling paradigm is based on the assumption that the growth trend within a year will remain the same in the future years as that in the year of

Model	rmse	n-rmse	% Error
2012 cas	422.99	0.56	52.06
2012 reg	774.86	0.55	17.41
2012 tot	977.65	0.55	17.39

Table 5: Validating the Final Models (Adjusted Predictions)

2011. However, this is NOT saying that the user base is supposed to remain unchanged throughout the entire year; the fact that the same scaling factor works at all points in the entire year is due to the fact that the MLS model in the later part of the year, e.g., in fall and winter, are already trained to compensate for rental count growth due to user growth using the environmental and seasonal variables.

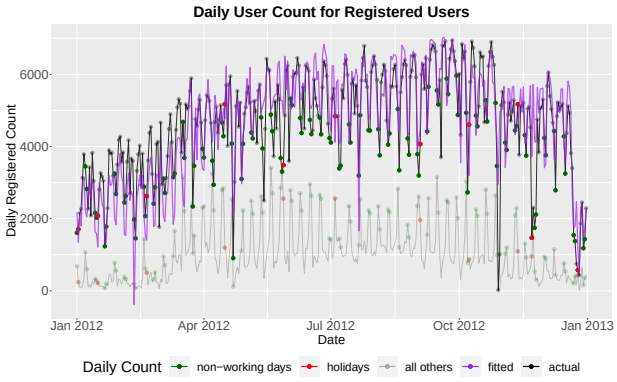


Figure 12: 2012 Predicted Registered Count v.s. Actual (Final Model) (Adjusted Prediction)

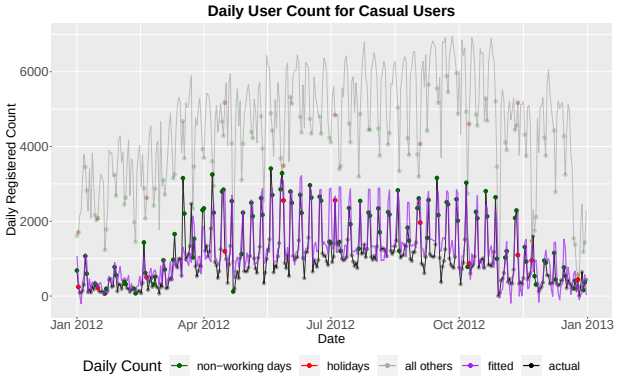


Figure 13: 2012 Predicted Casual Count v.s. Actual (Final Model) (Adjusted Prediction)

4 The Growth Ratio

Adjusting the predictions with the growth ratio leads to significantly improved performance. However, since we are not **allowed** to peek at the 2012 data, we must resort to other creative techniques in estimating $\hat{g}$. In this report, we propose two techniques for estimating $\hat{g}$: the environmental loss function technique (more involved) and the window technique (more simple).

4.1 Method I: Environmental Loss Function

The primary issue of relying solely on one years' worth of data to estimate the growth within that year is that there is great difficulty in factoring out the role environmental factors play in determining the difference in observed counts between any two given days. As such, if we are somehow able to find two days whose weather/environmental conditions are "similar" or "equivalent", the ratio of the counts between the later day and the earlier one is a reasonable "data point" useful for obtaining a sense of the growth. This follows because we asked those two days to be "environmentally equivalent" (we will formalize this notion soon), so *a priori*, if we were somehow able to marginalize out the environmental factors, then any observed difference in the count `cnt` **must** be attributed to user base growth! This motivates our technique, aptly named the "environmental loss function" technique. In brief, here is the strategy: by designing a loss function, we seek to find special pairs of days whose environmental factors are roughly equivalent. By taking the ratio of the counts, we obtain a preliminary estimate for the growth factor since in theory, we have marginalized the environmental factors in the selection process. Aggregating all these ratios and averaging them, we obtain an estimate for $\hat{g}$. That being said, we now slowly develop our technique.

Suppose we have two days (observations) d and d'. Consider the reduced dataset where we only have the three predictors: `atemp`, `hum`, and `windspeed`. In other words, our observations are vectors solely comprised of our *continous environmental variables*. Furthermore, suppose that we **standardize** these variables. This is critical since we don't want our loss functions' units to be arbitrarily inflated/deflated, but rather, unit-agnostic. This is one reason we might prefer continuous variables when implementing our loss function. Also, let's use `holiday` to focus only on non-holidays and removing holiday days from the dataset. We can adjust for environmental factors but comparing two "environmentally equivalent" days without accounting for `holiday` may very well throw off our

estimates of $\hat{g}$. Losing the 11 days which are holidays prove to be trivial, since we are counting pairs and $\binom{365}{2} \approx \binom{354}{2}$. Note that we omit `weathersit` and `workingday` (albeit the latter is not environmental, it is still important) mainly since they are categorical variables. While loss functions can include categorical variables, we will opt for the simplicity, familiarity, and stability of continous variables. In any case, we can justify their omission anyway since `weathersit` is correlated to our chosen environmental variables, so the information loss is not catastrophic. Additionally, since we are using `cnt` and not the subdivisions of `cas` and `reg`, the explanatory power of `workingday` is "canceled" out in the `cnt` variable in a sense since we observe inverse behavior of `cas` and `reg` with respect to `workingday`. Now, we may put the discussion of chosen predictors aside.

That being said, this means we may write $d = (x_1, x_2, x_3)$ and $d' = (x'_1, x'_2, x'_3)$ where x_i represents the **standardized** value of predictor i. Furthermore, let $\vec{\alpha} = (\alpha_1, \alpha_2, \alpha_3)$ be a weight vector, more heavily weighing variables deemed more important in determining "environmental similarity". We deem this to be `atemp`. In practice, we end up weighing `atemp` by $\alpha_1 = \frac{2}{3}$ and the other two predictors, `hum` and `windspeed`, by $\alpha_2 = \alpha_3 = \frac{1}{6}$. Note that this is arbitrary, and subject to scrutiny, but for the sake of simplicity, we may all agree temperature is the major determinant in constituting a notion of "environmental equivalence".

Finally, let δ_i denote the weighted difference between in the values of predictor i between day d and d'. So, we write $\delta_i = \alpha_i(x_i - x'_i)$. Observing the differences among all our predictors, we obtain the difference vector $\vec{\delta} = (\delta_i)_{i=1}^3$. Finally, a very natural notion of "loss function" arises: the magnitude of the difference vector! We restate this as such. Consider the loss function $\mathcal{L} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^+$, which takes two days as input, and outputs the magnitude of weighted vector of the differences in our normalized predictor variables between them. We write the *environmental equivalence loss* of d' with respect to d as follows:

$$\mathcal{L}_d(d') = |\vec{\delta}| \text{ where } \delta_i = \alpha_i(x_i - x'_i)$$

Now, suppose we enumerate **all possible unique ordered pairs** of days, which we denote Π:

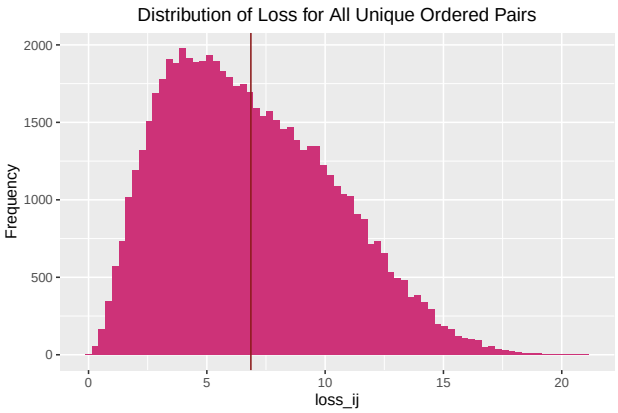
$$\Pi = \{\pi_{ij} = (d_i, d_j) \mid i < j\}$$

Note that we use the lower-triangle scheme, where we choose $i < j$ so that $i - j < 0$. This will prove useful later on since we want day i to preceed day j. We call this the "lower-triangular" scheme because this corresponds to the unique pairs of indices found in the "lower triangle part" of matrix. In any case, we compute the loss for every pair, and call it $l_{ij} = \mathcal{L}(\pi_{ij})$. Then, we obtain the following set:

$$\mathcal{L}(\Pi) = \{l_{ij} = \mathcal{L}(\pi_{ij}) \mid i < j\}$$

This is an exhaustive and computationally expensive procedure, since we compute the loss for $\binom{354}{2} \approx 62,000$ pairs of days! So, in practice we compute this once and store the data in `df_loss.csv`. When imported, we conventionally call the dataframe `df_loss`.

Now, here is the magic! We are not interested in taking pairs π_{ij} which are not very environmentally similar. After all, our quest is to estimate growth between days where we in theory expect a similar amount of users through the marginalization of environmental factors. So, there comes a question, which pairs do we discount as days too disparate to be considered environmentally equivalent? Well, our good foresight in normalizing the continuous variables and weighing them properly (to the best of our abilities) means that we can immediately consult distribution of l_{ij}, which we see below:



A quick note: it should not be surprising that we obtain a distribution that resembles a χ^2 distribution. After all, since we assume our predictors are (roughly) normal, their differences will be too; since our loss is essentially the norm of a multivariate normal, this explains our distribution shape! In any case, since we have done a good job by normalizing our variables (making our loss unit-less), a valid strategy is to **discount all pairs above a pre-specified upper bound**, which we will call B. An appopriate cutoff would probably be anywhere below the mean of our distribution $\mu \approx 6.86$, shown as the red line above. For example, we can take $B = \mu - 1\sigma \approx 3.35$ to be a reasonable cutoff for our upper loss bound B.

So, to summarize, we want to vet our exhaustive list of all possible day pairs Π and throw out bad pairs

after doing some "quality control" to obtain a filtered set of pairs $\tilde{\Pi}$. Let us take a brief moment to recap everything. We are interested in estimating $\hat{g}$, and one way we are interested in doing so is observing the growth in pairs of days where the environmental effects are marginalized. A strategy for doing so is to consider "reasonably good" pairs of days, which we explicitly quantify through filtering Π through an upper bound on the loss B, which we may holistically choose based on the reasoning above.

So, we obtain our filtered/selected set of day pairs $\tilde{\Pi}$, by cutting off the maximum loss at the selected upper bound B. That way, we can write our selected pair set as the output of a function f_{loss} that *filters* Π with respect to the chosen loss upper bound B:

$$f_{loss}(\Pi, B) = \tilde{\Pi}_B = \{\pi_{ij} \mid l_{ij} = \mathcal{L}(\pi_{ij}) \leq B\}$$

Next, we also filter for what we call an "index difference bound". This is our second round of "quality control". Namely, the idea is simple: we don't want to pick days too close to each other. We know days close to each other will have similar environmental conditions, but as to avoid pesky autocorrelation and vaccuous/misleading information, we impose a bound on how close the days can be.

Let $\rho := \min(i - j)$. In our lower-triangle matrix analogy (where we imagine indices as positions of matrix entries), ρ denotes which diagonal band we are considering. In any case, by imposing a lower bound on $i - j$, to pass the index difference filter, every pair must be atleast ρ days apart, or in other words, be at least ρ diagonal bands away from the corner or towards the main diagonal. In practice, we find low values of ρ to produce the most impactful filters, this makes sense, owing to the autocorrelation observed in days close to each other mentioned above. This makes the index difference bound much less critical than the loss bound, as it is much more impactful for low values of ρ. So, to summarize:

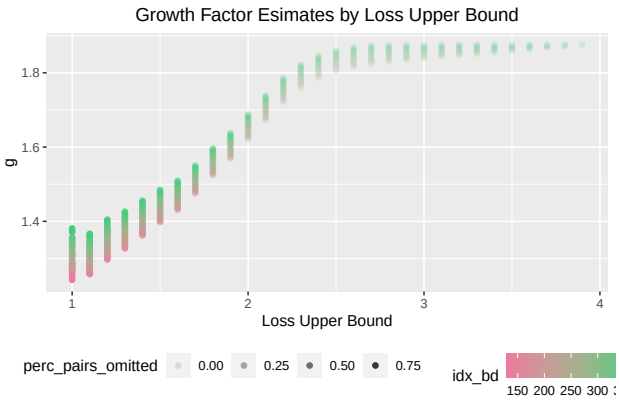
$$f_{idx\_diff}(\Pi, \rho) = \tilde{\Pi}_\rho = \{\pi_{ij} \mid i - j \geq \rho\}$$

Finally... we can compute the estimate derived from the filtered pair set $\tilde{\Pi}$ as follows. Let c_i denote the count of bike users in day i, and c_j those in day j. **Since we have, in theory, marginalized out the environmental factors (and holidays), we may obtain a reasonable estimate for g by taking the observed growth observed between day i and day j, then taking the average over all the pairs in our filtered pair set.** Since by construction, we used lower-triangular indices, this means day j is always after day i. As such, we can imagine each pair, which has marginalized environmental factors, to contain true information about user-base growth since day j is "environmentally equivalent" to day i and is **in the future** of day i! As such, we could in theory attribute this growth to none other than the user base growth itself. To be more explicit, consider a filtered set of pairs $\tilde{\Pi}_{B,\rho}$. Since every pair is considered "good", from each pair π_{ij} we obtain a preliminary estimate which we denote $\gamma_{ij} := \frac{c_j}{c_i}$. In general, the γ_{ij} are slightly unstable, and their variance may be worth studying; this remains outside the scope of the paper. However, since we are aggregating pairs from a set of $n = 62,500$ pairs (in practice we choose less), there is an abundance of data to counterbalance this. So, to summarize, given a filtered set of pairs $\tilde{\Pi}_{B,\rho}$, we obtain $\hat{g}$ as by taking average of the set of estimates γ_{ij} from every $\pi_{ij} \in \tilde{\Pi}_{B,\rho}$:

$$\hat{g} := \mathbb{E}[\gamma_{ij}] \text{ where } \gamma_{ij} = \frac{c_j}{c_i} \text{ and } \pi_{ij} \in \tilde{\Pi}_{B,\rho}$$

Estimator Analysis in Bound Paramaters Space. That being said, with our methodology explained, how does one estimate $\hat{g}$? This is our goal in the end! Well, as suggested by the notation above, the parameter estimate $\hat{g}$ is dependent on our filtering thresholds B and ρ. Of course, any choice of B_0 and ρ_0 would be arbitrary, so our next course of action is to observe and analyze our estimator over the bound parameters space. To be concrete, we say fix some ranges $[B_0, B_1]$ and $[\rho_0, \rho_1]$, then observe the behaviour of the resulting parameter estimate $\hat{g}$. To do so, we use the `df_loss` previously mentioned in conjuction with the `get_df_param` function which takes these ranges as an input and outputs `df_param`. A glimpse into what our table looks like is shown below. Note that n counts the number of pairs that "survive" the filtrations given the thresholds. In principle, we want our thresholds to pick exclusive pairs of days which are *unusually* environmentally similar (low B) but far apart enough (high ρ), with emphasis on the former. That being said, we now plot the results from this dataframe and obtain the exciting results below!

```
##   idx_bd loss_bd        g    n
## 1    130     1.0 1.242667  521
## 2    138     1.0 1.252098  506
## 3    138     1.1 1.258080 3198
## 4    146     1.0 1.256074  497
## 5    146     1.1 1.260398 3149
## 6    146     1.2 1.297205 8044
```



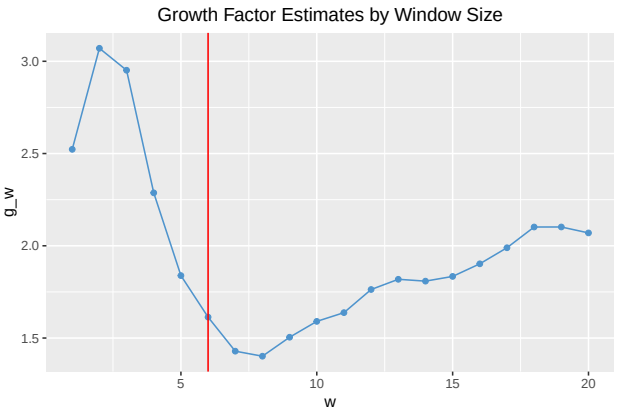
Again, as mentioned previously, the index difference bound is more sensitive when it comes to lower values. Greener points suggest that the index difference bound is high, so our days are quite far apart. Before we delve into the behaviour of the loss upper bound B, note that the opacity corresponds to the "percentage of pairs omitted". Recall that we want an exclusive set of pairs in which few pairs survive our strict filtrations or quality control. This is what the plot is suggesting, the more "faded" estimates of $\hat{g}$ are less well-founded, in fact completely unfounded as $p \to 1$ since we would indiscriminately be considering all pairs! As such, we conclude that the optimal range for B probably lies between $[1, 2]$, as these points have more "clarity" in their predictions due to their exclusively low loss. Otherwise, we also notice an interesting pattern with the index difference bound which makes sense, allowing days closer to each other will suggest "less growth". That being said, this means that in some sense, a potential optimal range for $\hat{g}$ could be that around the values where its variance starts to stabilize with respect to ρ. Visually, this is the range where the "length" of the streaks starts to hold still. For values before the critical $B \in [1, 2]$ range, this corresponds to the values around $\hat{g} \in [1.4, 1.6]$. That being said, this analysis is holistic, and not 100% conclusive. However, seeing this continous picture of estimating g and narrowing down its true value to be close to $\hat{g} \in [1.4, 1.6]$ is not bad!

4.2 Method II: The Window Technique

Jeez, that stuff was too technical. Thankfully, the window technique is much simpler. Remember our goal is to estiamte $\hat{g}$, which is the growth rate from 2011 to 2012. As such, the most elementary method to use based on this principle is to take the ratio of the counts in the first day of 2011 and those in the last day. This is justified because everyone would agree, there is some notion that Dec 31, 2011 $\approx$ Jan 1, 2012. However, the issue with this is that our estimate is based on one pair of points ($n = 1$).

As such, to preserve the spirit of the principle, but generalize as to instill more certainty and stability in our estiamte, we describe the "window technique". Firstly, we will say that the method mentioned above is the window technique, for a window size of $w = 1$. The principle of the window technique is the same as for the method described above, the days at the end of the year of 2011 will be similar to those in the beginning of 2012. So, the best we can do is take an equivalent "window" or sample of days from both the beginning and end of the year of size w. This way, we use more data and hence have more stability. Then, using those selected w days, we compute the average `cnt` in the beginning (first w days) and end (last w days), then take the ratio of those averages to estimate $\hat{g}$. We call this estimate $\hat{g}_w$. In our first example, we were describing $\hat{g}_1$.

In any case, just like before, $\hat{g}$ is an estimator based on a parameter, in this case w. We must note that as w increases, our assmuptions about "day similarity" between the end of 2011 and start of 2012 slowly starts to weaken. As such, we cannot go crazy and choose large w. Below, we exhaustively compute $\hat{g}_w$ for $w = 1, 2, \ldots, 20$, and plot our estimates with respect to w.



There are a few things to consider. As expected, the values $w \in [1, 4]$ exhibit highly unstable behaviour in estimating $\hat{g}$. However, it starts to stabilize around $w = 6$, highlighted in red. As it turns out, $w = 6$, holistically speaking is our best w for a few reasons. Firstly, going beyond $w = 6$ would mean including Christmas, which is a holiday and would thus throw off our estiamtes. Furthermore, the values of $\hat{g}$ start to stablize reasonably well around that value, indicating the marginal benefit of increasing w is almost fully realized at this point; remember, our assumptions

about day similarity break quickly as w grows. Note that in the plot below, we compute $\hat{g}$ after omitting Jan 3rd, due to its high environmental loss value when paired with the other values. Lastly, one reason we consider $w = 6$ to be a good candidate is the fact that the estimate $\hat{g}$ starts to steadily increase again, indicating suspicious behaviour which could be attributed to the fact that as the window increases, we include days whose temperatures are changing in potentially different directions (generally both warming).

That being said, there is one curious piece of information corroborated by the environmental loss function technique. Namely, both graphs seem to suggest discounting the values $\hat{g} \gg 1.6$. In the second graph, this is suggested by the vacously increasing value of $\hat{g}$ with respect to w starting around $w = 10$, indicating that the point in which there is "good" information about $\hat{g}$ probably lies before that value of $w = 10$, corroborated also by the simple principle that smaller w is generally better in terms of upholding assumptions. So, with all this information considered, we settle for $\hat{g} := \hat{g}_6 \approx 1.613$, which is the most compatible value from our holistic analysis of both of our estimation methods.

5 Discussion

Below, we have a plot displaying the results of estimating g. Namely, we discovered that roughly speaking, $\hat{g} \in [1.4, 1.6]$. Using 1.4 as a lower bound (displayed in red) and 1.6 as a upper bound (displayed in green), we display the effect it has on taking the unscale predictions (in blue). Indeed, we notice that most would agree our lower and upper bounds yield substantial improvements and sufficiently addresses the chronic underfitting of our models. In any case, the full validation of this model paradigm will require one more full years' worth of data.

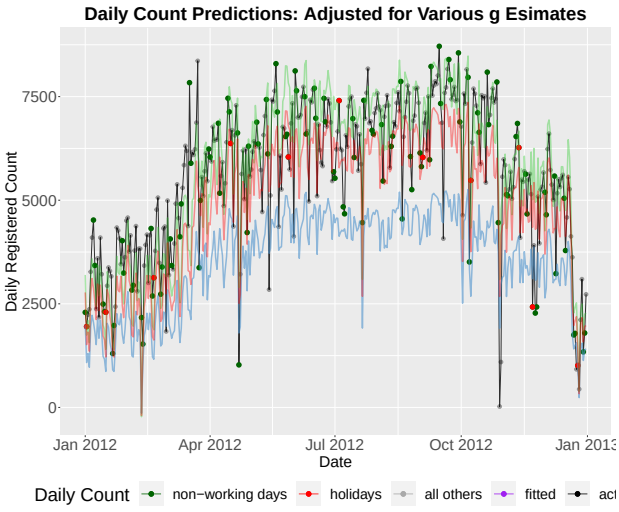


Figure 14: Various g and prediction adjustment

6 References

[1] Fanaee-T, Hadi, and Gama, Joao, "Event labeling combining ensemble detectors and background knowledge", Progress in Artificial Intelligence (2013): pp. 1-15, Springer Berlin Heidelberg, doi:10.1007/s13748-013-0040-3.

[2] C3 Team 2 (2021). bikes575: Prediction of Bike Sharing using MLR, https://github.com/ataqi23/bikes575

7 Appendix

Note: The source code can be found on Github in the repository found in the references. The primary files to consider are the source files found in the "final" directory, which will be refactored and tidied in an ongoing manner.

7.1 Data

- `instant`: record index
- `dteday`: date
- `season`: season (1:spring, 2:summer, 3:fall, 4:winter)
- `yr`: year (0:2011, 1:2012)
- `mnth`: month (1 to 12)
- `hr`: hour (0 to 23)
- `holiday`: weather day is holiday or not (extracted from http://dchr.dc.gov/page/holiday-schedule)
- `weekday`: day of the week
- `workingday`: if day is neither weekend nor holiday is 1, otherwise is 0.
- `weather-sit`:
  - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
  - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
  - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
  - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
- `temp`: Normalized temperature in Celsius. The values are divided to 41 (max)
- `atemp`: Normalized feeling temperature in Celsius. The values are divided to 50 (max)
- `hum`: Normalized humidity. The values are divided to 100 (max)
- `windspeed`: Normalized wind speed. The values are divided to 67 (max)
- `casual`: count of casual users
- `registered`: count of registered users
- `cnt`: count of total rental bikes including both casual and registered

7.2 Wrangling

```
#============================#
#       Wrangling Function     #
#============================#
# Initial wrangling wrapper function
wrangle_init <- function(data, omit_NA = TRUE, omit_idx = TRUE){
  # Boolean variables (from int to logical type)
  data$holiday <- as.logical(data$holiday)          # 0 or 1
  data$workingday <- as.logical(data$workingday)    # 0 or 1
  # Other categorical variables (from int to factor type)
  data$season <- as.factor(data$season)             # 1 to 4
  data$yr <- as.factor(data$yr)                     # 0 to 1
  data$mnth <- as.factor(data$mnth)                 # 1 to 12
  data$weekday <- as.factor(data$weekday)           # 0 to 6
  data$weathersit <- as.factor(data$weathersit)     # 1 to 4
  # Re-scale the normalized measurements
  data$temp <- data$temp * 41
  data$atemp <- data$atemp * 50
  data$hum <- data$hum * 100
  data$windspeed <- data$windspeed * 67
  # Change type of Dates (from char to Date type)
  data$dteday <- as.Date(data$dteday)
  # Remove NAs (if prompted) default value is TRUE
  if(omit_NA) { data <- na.omit(data) }
  # Remove instance column (if prompted) default value is TRUE
  if(omit_idx) { data <- data %>% select(-c("instant")) }
  # Observe christmas
  data$holiday[359] <- T; data$holiday[725] <- T
  # Return the wrangled dataset
  return(data)
}
#========================#
#        Subsetting      #
#========================#
# Filter for the 2011 data
in_2011 <- function(data){ data[(data$dteday >= "2011-01-01" & data$dteday <= "2011-12-31"),] }
# Filter for the 2012 data
in_2012 <- function(data){ data[(data$dteday >= "2012-01-01" & data$dteday <= "2012-12-31"),] }
#======================#
#       Wrangling      #
#======================#
# Import and wrangle data
bike_day <- read.csv("bike-day.csv", header = T)
# Use the wrangling "wrapper" function to clean the data (fxn sourced from wrangle.R)
data <- wrangle_init(bike_day)
# Partition the 2011/2012 data
data2011 <- in_2011(data); data2012 <- in_2012(data)
```

7.3 Modeling

7.3.1 Model Building

```
#============================#
#       Helper Function      #
#============================#
# A helper function that returns a formula in the "lm" syntax
# Takes predictors, a vector of variable name strings as an input
.parseFormula <- function(predictors, response = "cnt"){
  f <- as.formula(
    paste(response,
          paste(predictors, collapse = " + "),
          sep = " ~ "))
  return(f)
}
#====================================#
#       Model "Builder" Functions      #
#====================================#
# Given a vector of variable names, return a linear model fitting cnt
lm.tot <- function(varset, train_data = data2011){
  lm(formula = .parseFormula(predictors = varset, response = "cnt"), data = train_data)
}
# Given a vector of variable names, return a linear model fitting cas
lm.cas <- function(varset, train_data = data2011){
  lm(formula = .parseFormula(predictors = varset, response = "casual"), data = train_data)
}
# Given a vector of variable names, return a linear model fitting reg
lm.reg <- function(varset, train_data = data2011){
  lm(formula = .parseFormula(predictors = varset, response = "registered"), data = train_data)
}
```

7.3.2 Model Formulas

```
# Initial models
mod.cas.1.final <- lm.cas(c("workingday", "weathersit", "atemp"))
mod.reg.1.final <- lm.reg(c("workingday", "weathersit", "atemp", "I(atemp^2)"))
mod.tot.1.final <- lm.tot(c("workingday", "weathersit", "atemp", "I(atemp^2)"))
# Final models
mod.cas.2.final <- lm.cas(c("holiday", "season:weathersit", "season:workingday:atemp"))
mod.reg.2.final <- lm.reg(c("holiday", "season:weathersit",
                            "season:workingday:atemp", "season:workingday:I(atemp^2)"))
```

7.3.3 Growth-Adjusting Predictions

```
# Select some arbitrary model
model_example <- mod.cas.1.final
# Select a parameter estimate for g
g_hat <- 1.61
# Predictions without growth adjustment
preds <- predict(model_example, data2012)
# Predictions after growth adjustment
preds_adj <- preds * g_hat
```

7.4 Diagonostic Analysis

```
residual_QQ <- function(model, title_str = ""){
  # Plot parameters
  fsize0 <- 13; fsize1 <- 13; fsize2 <- 13
  # Plot
  df_res <- data.frame(ep = model$residuals)
  df_res %>%
    ggplot(aes(sample = ep)) +
    geom_qq(alpha = 0.8, color = "palegreen3") +
    stat_qq_line(color = "seagreen") +
    labs(x = "Theoretical Quantile", y = "Sample Quantiles",
         title = paste("Normal Q-Q Plot", title_str, sep = ""))
}
residual_fitted <- function(model, title_str = ""){
  # Plot parameters
  fsize0 <- 13; fsize1 <- 13; fsize2 <- 13
  # Plot
  df_res <- data.frame(fitted_value = model$fitted.values, residual = model$residuals)
  df_res %>%
    ggplot(aes(x = fitted_value, y = residual)) +
    geom_point(alpha = 0.7, color = "palegreen3") +
    stat_smooth(se = T, color = "forestgreen") +
    labs(x = "Fitted Value", y = "Residual",
         title = paste("Fitted vs Residual",title_str, sep = ""))
}
residual_histogram <- function(model, title_str = ""){
  # Plot parameters
  fsize0 <- 13; fsize1 <- 13; fsize2 <- 13
  epsilons <- unname(model$residuals)
  df_res <- data.frame(epsilon = epsilons)
  df_res %>%
    ggplot(aes(x = epsilon)) +
    geom_histogram(fill = "forestgreen", bins = 60) +
    labs(x = "Residual", title = paste("Residual Distribution",title_str, sep = ""))
}
residual_plots <- function(model, title_str){
  p1 <- model %>% residual_QQ(title_str)
  p2 <- model %>% residual_histogram(title_str)
  p3 <- model %>% residual_fitted(title_str)
  (p1 + p2 + p3)
}
```

7.5 Validation and Problemshooting

```
##############################################################################
#                                  LOOCV
##############################################################################

get_loocv_rmse = function(model) {
  loocv_rmse <- sqrt(mean((resid(model) / (1 - hatvalues(model))) ^ 2))
  # Round the numbers
  digits <- 2
  loocv_rmse <- round(loocv_rmse, digits)
  # Return the dataframe row
  return(data.frame('CV_rmse' = loocv_rmse))
}

get_loocv_rmse_tot = function(cas, reg) {
  res_cas <- resid(cas) / (1 - hatvalues(cas))
  res_reg <- resid(reg) / (1 - hatvalues(reg))
  loocv_rmse <- sqrt(mean((res_cas + res_reg) ^ 2))
  # Round the numbers
  digits <- 2
  loocv_rmse <- round(loocv_rmse, digits)
  # Return the dataframe row
  return(data.frame('CV_rmse' = loocv_rmse))
}

##############################################################################
#                        Model Validation Functions
##############################################################################

get_rmse <- function(y, y_hat, name='none'){
  rmse <- sqrt(mean((y - y_hat)^2, na.rm = TRUE))
  normalized_rmse <- sqrt(mean(((y - y_hat)^2), na.rm = TRUE)) / sd(y)
  percentage_error <- mean(abs(y - y_hat) / y, na.rm = TRUE) * 100
  # Round the numbers
  digits <- 2
  rmse <- round(rmse, digits)
  normalized_rmse <- round(normalized_rmse, digits)
  percentage_error <- round(percentage_error, digits)
  # Return the dataframe row
  return(data.frame('name' = name, 'rmse' = rmse,
                    'nrmse' = normalized_rmse, 'prc_err' = percentage_error))
}

get_mod_eval_2011 <- function(cas, reg){
  ret1 <- rbind(get_rmse(data2011$casual,     predict(cas, data2011), '2011 cas'),
                get_rmse(data2011$registered, predict(reg, data2011), '2011 reg'),
                get_rmse(data2011$cnt,
                         predict(cas, data2011) + predict(reg, data2011), '2011 tot'))
  ret2 <- rbind(get_loocv_rmse(cas),
                get_loocv_rmse(reg),
                get_loocv_rmse_tot(cas, reg))
  return(cbind(ret1, ret2))
}
```

```
get_mod_eval <- function(cas, reg, data2011, data2012, scale_2012 = TRUE, include_2011 = F){
  if(scale_2012){G_FACTOR <- 0.608} else{G_FACTOR <- 1}
  if(include_2011){
    return(rbind(
      get_rmse(data2011$casual,     predict(cas, data2011), '2011 cas'),
      get_rmse(data2011$registered, predict(reg, data2011), '2011 reg'),
      get_rmse(data2011$cnt,        predict(cas, data2011) + predict(reg, data2011), '2011 tot'),
      get_rmse(data2012$casual,     predict(cas, data2012)/G_FACTOR, '2012 cas'),
      get_rmse(data2012$registered, predict(reg, data2012)/G_FACTOR, '2012 reg'),
      get_rmse(data2012$cnt,
               predict(cas, data2012)/G_FACTOR + predict(reg, data2012)/G_FACTOR, '2012 tot')))
    }
  return(rbind(
    get_rmse(data2012$casual,     predict(cas, data2012)/G_FACTOR, '2012 cas'),
    get_rmse(data2012$registered, predict(reg, data2012)/G_FACTOR, '2012 reg'),
    get_rmse(data2012$cnt,
             predict(cas, data2012)/G_FACTOR + predict(reg, data2012)/G_FACTOR, '2012 tot')))
  }
get_mod_eval_tot_2011 <-function(tot){
  ret1 <- rbind(get_rmse(data2011$cnt, predict(tot, data2011), '2011 tot'))
  ret2 <- rbind(get_loocv_rmse(tot))
  return(cbind(ret1, ret2))
}
get_mod_eval_tot <-function(tot, data2011, data2012, scale_2012 = TRUE){
  if(scale_2012){G_FACTOR <- 0.608} else{G_FACTOR <- 1}
    return(rbind(get_rmse(data2011$cnt, predict(tot, data2011), '2011 tot'),
                 get_rmse(data2012$cnt, predict(tot, data2012)/G_FACTOR, '2012 tot'),
                 get_loocv_rmse(tot)))
}
```

7.6 Prediction of the Yearly Growth Ratio

```
#========================#
#      Miscellaneous      #
#========================#
# Enumerate all the pairs in the lower-triangular matrix scheme
# In other words, (i < j or i - j < 0) so that day_i preceeds day_j
.unique_pairs_lower <- function(N){
  is <- do.call("c", purrr::map(1:N, function(i){rep(i,N)}))
  js <- rep(1:N, N)
  # Helper function: selects elements only if they are upper triangular
  .LowerTri <- function(i, j){if(i > j) { c(i = i, j = j) }}
  pairs <- do.call("rbind", purrr::map2(is, js, .f = .LowerTri))
  data.frame(pairs)
}

#==================================#
#     Dataframe Wrapper Functions   #
#==================================#
# Given a dataframe of loss values between all possible day pairs
# and a set of idx and loss bounds, compute the g estimates
get_df_param <- function(df_loss, idx_bds, loss_bds){
  # Compute the index pairs
  MIP <- .unique_pairs_lower(length(idx_bds))
  # Use helper function to compute the g estimates for each set of bounds
  .helper <- function(k, df_loss = df_loss){
    idx <- MIP$i[k]; loss <- MIP$j[k]
    results <- g_estimate(df_loss, idx, loss)
    data.frame(idx_bd = idx_bds[idx], loss_bd = loss_bds[loss], g = results[[1]], n = results[[2]])
  }
  # Run helper over 1,...,n where n is the number of pairs surviving bound cutoff
  df_param <- map_dfr(1:nrow(MIP), .helper, df_loss)
  # Return the estimates and number of pairs used for each bound pair
  return(df_param)
}

get_df_loss <- function(data2011){
  # Obtain the continous variables and holiday
  data <- data2011 %>% select(dteday, holiday, atemp, hum, windspeed, "cnt")
  # Normalize the continous variables
  data <- data %>% mutate_at(c("atemp", "hum", "windspeed"), ~(scale(.) %>% as.vector))
  # Take out holidays (to avoid needing to account for its effect, omits only few days of data)
  data <- data %>% filter(!holiday)
  # Enumerate all the possible index pairs (for use in purrr::map2_dfr)
  pairs <- .unique_pairs_lower(nrow(data))
  # Helper function that takes two indices and returns the loss between the days at those indices
  .helper <- function(i,j){data.frame(i = i, j = j, loss_ij = loss_ij(i,j))}
  # Compute the loss function exhaustively for every possible (lower-triangle) pair
  df_loss <- map2_dfr(pairs[["i"]], pairs[["j"]], .f = .helper)
  # Add the index difference column (useful for determining space between days)
  df_loss <- df_loss %>% mutate(idx_diff = i - j)
  # Return the exhaustive dataset of loss values for every unique day ordered pair
  return(df_loss)
}
```

```r
#============================================================================#
#============================================================================#

#=========================#
#      Loss Function      #
#=========================#

# Given two rows (days) from the data, compute the loss function
loss <- function(day0, day1){
  atemp_diff <- day0[["atemp"]] - day1[["atemp"]]
  wind_diff <- day0[["windspeed"]] -  day1[["windspeed"]]
  hum_diff <- day0[["hum"]] - day1[["hum"]]
  norm(as.matrix(c(4*atemp_diff, wind_diff, hum_diff)))
}

# Given two row indices, compute the loss function between those two days
loss_ij <- function(i, j){
  day0 <- data[i,]; day1 <- data[j,]
  loss(day0, day1)
}


growth_ratio <- function(i, j){ data2011$cnt[i]/data2011$cnt[j] }

g_estimate <- function(df_loss, idx_bound, loss_bound){
  # Compute the g-estimate after filtering through bounds
  df_loss <- df_loss %>%
    filter(idx_diff > idx_bound) %>%
    filter(loss_ij < loss_bound) %>%
    mutate(g = growth_ratio(i, j))
  # Return the mean (g^) and number of pairs used (sample size n)
  list(mean(df_loss$g), nrow(df_loss))
}

#=================================#
#     Loss Fxn Technique Plots    #
#=================================#

g_plot <- function(df_param){
  # Filter bounds to obtain a good window frame
  df_param <- df_param %>% filter(idx_bd > 10, loss_bd < 5)
  # Create the proportion of maximum pairs variable (for alpha)
  df_param <- df_param %>% mutate(perc_pairs_omitted = 1 - n/max(df_param$n))
  # Scatterplot
  df_param %>%
    ggplot() +
    geom_point(aes(color = idx_bd, x = loss_bd, y = g, alpha = perc_pairs_omitted)) +
    geom_abline(slope = 0, intercept = 3.35) +
    theme(legend.position = "bottom") +
    labs(x = "Loss Upper Bound", y = "g", title = "Growth Factor Esimates by Loss Upper Bound") +
    #scale_alpha(guide = "none") +
    scale_color_gradient(low = "palevioletred2", high = "seagreen3")
}
```

```r
#============================================================================#
#============================================================================#

#=========================#
#     Window Technique    #
#=========================#

window_g <- function(w, no_outlier = T){
  # Compute the indices of the first and last w days
  lower_idx <- 1:w
  upper_idx <- 366 - (w:1)
  # Obtain the first and last w days
  last_w <- data2011[upper_idx,]
  first_w <- data2011[lower_idx,]
  # Since Jan-3 is environmentally different (by loss function value), we remove it
  if(no_outlier){ first_w <- first_w %>% filter(dteday != "2011-1-3") }
  # Compute and return the difference in means between the first and last w days
  return(mean(last_w$cnt)/mean(first_w$cnt))
}

#==============================#
#     Window Technique Plots    #
#==============================#

plot_window <- function(tbl_window){
  # Plot parameters
  col0 <- "steelblue3"
  # Scatterplot
  tbl_window %>%
    ggplot(aes(w, g_w)) +
    geom_point(color = col0) +
    geom_line(color = col0) +
    geom_vline(xintercept = 6, color = "red") +
    labs(title = "Growth Factor Estimates by Window Size")
}
```

```r
#------------------------------------------------------------------------------#
#==============================================================================#
#                              Primary Code Script
#==============================================================================#
#------------------------------------------------------------------------------#
# Control flow for avoiding computational/time waste
recompute <- F
if(recompute){
  # Obtain the exhaustive dataset of loss values for every unique day ordered pair
  df_loss <- data_2011 %>% get_df_loss
  # Write CSV to avoid future recomputation
  write.csv(df_loss, "df_loss.csv", row.names = F)
} else{
  df_loss <- read.csv("df_loss.csv")
}

#=====================================================#
#   Estimate Performance over Bound Paramater Space
#=====================================================#

# Set a sequence of index bounds
idx_bds <- seq(122, 365, 8)
# Set a sequence of loss bounds: more critical
loss_bds <- seq(1, 4, 0.10)
# Obtain the parameter dataframe of g estimates
df_param <- df_loss %>% get_df_param(idx_bds, loss_bds)
# Plot the estimates over bound parameter space
plot_g <- df_param %>% g_plot()

#========================#
#    Window Technique    #
#========================#

# Obtain the table of g esimates by window paramater w
tbl_window <- purrr::map_dfr(1:20, function(w){data.frame(w = w, g_w = window_g(w))})
# Plot the values
plot_w <- plot_window(tbl_window)
```