LeastSquares

1. Introduction

The dataset for this project is the “Bike Sharing Dataset Data Set” found in the UCI Machine Learning Repository. The dataset contains hourly count of rental bikes for all of 2011 and 2012 (January 1, 2011 to December 31, 2012) in the Capital Bikeshare System of Washington D.C. area (Washington-Arlington-Alexandria, DC-VA-MD-WV metropolitan area). The UCI Machine Learning Repository cites Hadi Fanaee-T from the “Laboratory of Artificial Intelligence and Decision Support (LIAAD), University of Porto” for the compilation of the data.

The dataset is outdated since data is actually available up to November 2020 on Capital Bikeshare’s website (as of December 18, 2020), but this limited dataset will still work for the purposes of demonstrating linear algebra on a real world dataset.

There are two files included in the dataset: a `hour.csv` and a `day.csv`. We will use the `hour.csv` for the regression, since the `day.csv` is simply just a sumamry of the `hour.csv` file. We also made a function that easily converts the `hour.csv` to the `day.csv` called `convert_hour_to_day()`.

There are 14 different variables that are in this dataset that are potentially of interest. Two variables are not useful and immediately thrown out: `instant` (this is simply the row number of the dataset) and `dteday` (date of the year).

Denote x_n as plausible independent variables and denote y_n as plausible dependent variables.

x_1: `season` (1: spring, 2: summer, 3: fall, 4: winter)

x_2: `yr` (0: 2011, 1: 2012)

x_3: `mnth` (1 to 12)

x_4: `hour` (0 to 23)

x_5: `holiday` (whether a holiday (0 or 1) from this list of holidays)

x_6: `weekday` (0 to 6)

x_7: `workingday` (1 if weekday and not holiday, 0 otherwise)

x_8: `weathersit`: Weather conditions (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)

x_9: `temp` (0-1, normalized temperature in Celsius. Divided by 41)

x_{10}: `atemp` (0-1, normalized “feels like” temperature in Celsius. Divided by 50)

x_{11}: `hum` (percent humidity)

x_{12}: `windspeed` (0-1, Normalized wind speed. Divided by 67)

y_1: `casual` (count of casual users)

y_2: `registered` (count of registered users)

y_3: `cnt` (count of sum of casual and registered users)

The following least squares regression exercise will try to predict the `casual`, `registered`, or `cnt` as a function of the independent variables. Also, we will try some principal components analysis (PCA) and k-nearest neighbors (kNN) with these variables.

Data Analysis

Preliminary data analysis shows that the `hour` is by far the most important independent variable for explaining the variation in the dependent variables. Thus, it is important to know how exactly the `hour` variable interacts with `registered`, `casual`, and `cnt`.

We also found that it makes sense to treat `hour` as a categorical variable (treat `hour` as 23 independent dummy variables (0 or 1 for each variable), one for each hour minus the constant term), but in this case, we will try to fit `hour` in terms of a polynomial curve to demonstrate polynomial fitting with linear algebra.

A plot of `hour` on the x-axis and `registered` or `casual` on the y-axis would us some insight of what degree polynomial for `hour` we should be looking for.

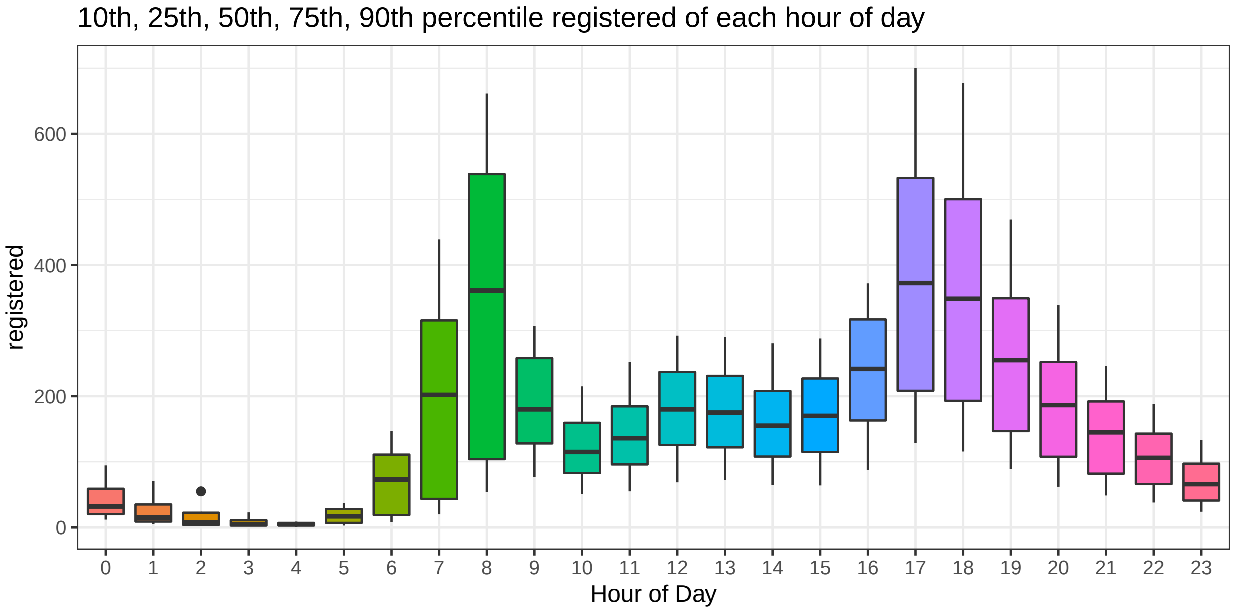

Figure 1: Boxplot of registered users for Jan 1, 2011 to December 31, 2011 for each hour of day. The low whisker represents 10th percentile, the box represents the interquartile range, the top whisker represents the 90th percentile. For example, with 731 days in the datset, the low whisker represents the 73rd lowest value.

Figure 1 for the number of registered users show a trimodal distribution with three peaks throughout the day. A possible interpretation of the three peaks is that there is an early peak for the morning commute, a central peak for lunchtime, and a late peak for the evening commute. This suggests that we need a high-degree polynomial to accurately the number of registered users throughout the day. A six-degree polynomial is the minimum degree that can represent a trimodal distribution.

Figure 2 for the number of casual users show a single peak distribution. A possible reason for this is that casual users are tourists that don't commute. Tourists also don't wake up early in the morning and most things to do for tourists occur in the afternoon and evening. Thus, the peak occurs at around 12 PM-6 PM. A two-degree polynomial is potentially sufficient to represent the number of casual users throughout the day.

Figure 1 and **Figure 2** show significant variation so that it is clear that the hour of the day is not the only variable influencing how many users there are for this bike sharing system. We can plot at the number of users for each day for further information.

Figure 3 shows both variation through the year and an increase in number of users from 2011 to 2012. This makes sense if this Capital Bike-Sharing was still a developing system in 2011 and not yet a mature system where the market is already saturated. Thus, we want to include a variable in our least squares regression model that includes controls for seasonal variation and the year. This would be `season` and `yr` from the list of x_n. It turns out that the `weekday` (day of the week) and `mnth` (month of the year) do not explain a large additional amount of variation, so they won't be included in the model.

But **Figure 3** still shows quite a bit of variation day to day within each season. There are quite a few weather related variables in the list of independent variables in the dataset, which would account for some of the remaining variation.

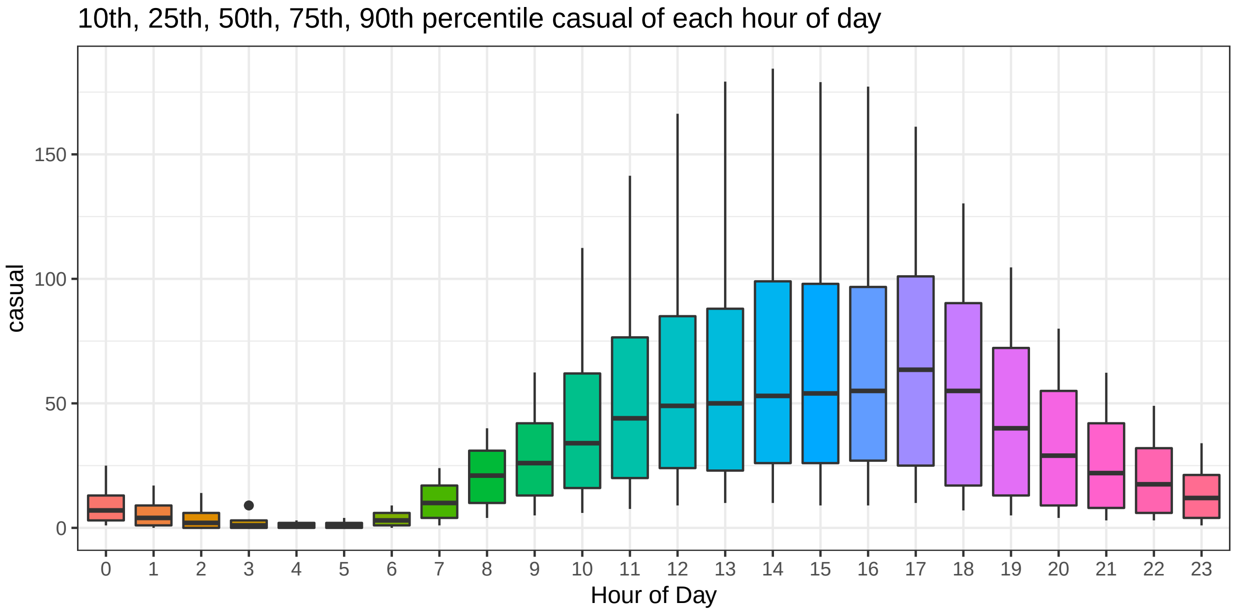

Figure 2: Boxplot of casual users for Jan 1, 2011 to December 31, 2011 for each hour of day. The low whisker represents 10th percentile, the box represents the interquartile range, the top whisker represents the 90th percentile. For example, with 731 days in the datset, the low whisker represents the 73rd lowest value.

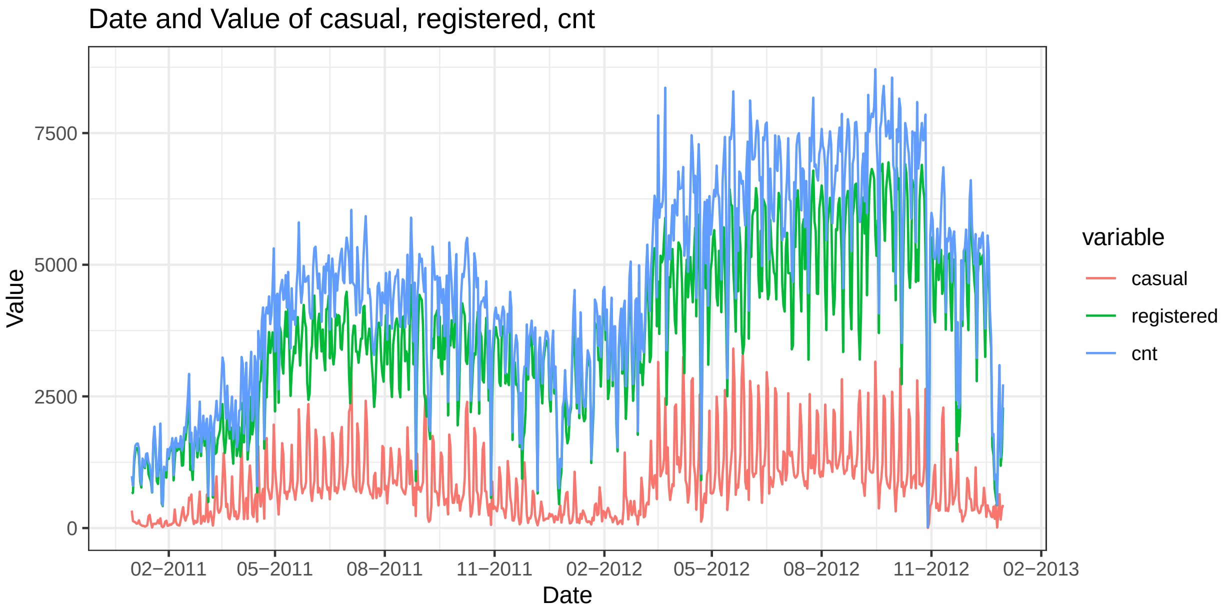

Figure 3: Number of casual, registered, and sum of casual and registered for all 731 days in the dataset

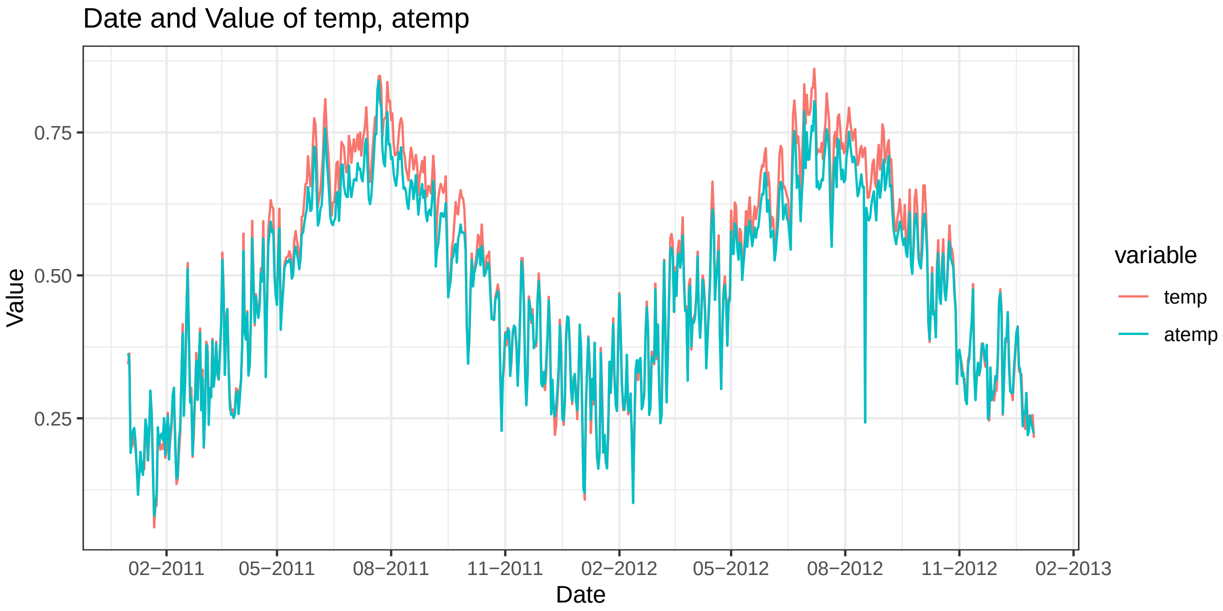

Figure 4: Normalized temperature and "feels like" temperature for all 731 days in the datset

Figure 4 shows the average `temp` (actual temperature) and `atemp` ("feels like" temperature) for each day of the year. A comparison of **Figure 3** and **Figure 4** shows that temperature shows a strong negative relationship with the number of bike-sharing users, which makes sense. People do not want to bike when it is cold outside. These variables have a very high correlation with each other such that we found it to be sufficient to just add `temp`. In fact, `temp` is good enough to explain most of the weather-related variation and other variables such as `wind` (wind speed) and `hum` (humidity) are not necessary.

Thus, we choose just `season`, `yr`, `hr`, and `temp` as the independent variables and omit the rest of the variables for the least squares regression model. These are x_1, x_2, x_4, and x_9.

2. Design Matrix

The design matrix A is the matrix that will be used to solve the equation $Ax = b$, where x are the coefficients, often referred to as the β's and b is the dependent variable. It is the matrix of the explanatory/independent variables.

The design matrix A will be of $\mathbb{R}^{17379\times(n+4)}$, (where n is the maximum degree of the polynomial for x_4 (`hour`)) since there are 17,379 rows (one for each hour in the dataset) and there are four variables other than the `hour` (x_4) variable. The other four variables are the constant term, x_1 (`season`), x_2 (`yr`), and x_9 (`temp`).

The following will be the form of our design matrix $A \in \mathbb{R}^{17379\times(n+4)}$ (Note that m is left the matrix for simplification, but the of $m = 17379$):

$$A = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{14} & x_{14}^2 & \cdots & x_{14}^n & x_{19} \\ 1 & x_{21} & x_{22} & x_{24} & x_{24}^2 & \cdots & x_{24}^n & x_{29} \\ 1 & x_{31} & x_{32} & x_{34} & x_{34}^2 & \cdots & x_{34}^n & x_{39} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 1 & x_{m1} & x_{m2} & x_{m4} & x_{m4}^2 & \cdots & x_{m4}^n & x_{m9} \end{bmatrix}$$

The first value of the index represents the row number and the second value of the index represents the variable number in the matrix. For example x_{32} in the above means the 3rd row of variable x_2.

Now the question remaining is the appropriate value of n. We know that the answer depends on which dependent variable we are using. If b is `registered` or `cnt`, the value of n should be higher than if b is `casual`. One way to find a good for n is calculate the R-squared for each value of n. The R-squared is the proportion of variation in the dependent variable that can be explained by the independent variables. A value of n where $n + 1$ does not increase the R-squared significantly further would be a good value of n.

Table 1: R-squared values for linear regression of each of value of n, maximum power of polynomial for `hour`, including \$x_{1}, x_2, and x_9

	R^2_{cnt}	R^2_{casual}	$R^2_{registered}$
x_4	0.343	0.285	0.291
$x_4 + x_4^2$	0.464	0.371	0.394
$x_4 + x_4^2 + x_4^3$	0.515	0.430	0.431
$x_4 + x_4^2 + \cdots + x_4^4$	0.516	0.440	0.436
$x_4 + x_4^2 + \cdots + x_4^5$	0.530	0.446	0.464
$x_4 + x_4^2 + \cdots + x_4^6$	0.552	0.446	0.497
$x_4 + x_4^2 + \cdots + x_4^7$	0.589	0.446	0.551
$x_4 + x_4^2 + \cdots + x_4^8$	0.599	0.446	0.565
$x_4 + x_4^2 + \cdots + x_4^9$	0.606	0.447	0.575
$x_4 + x_4^2 + \cdots + x_4^{10}$	0.606	0.447	0.575
$x_4 + x_4^2 + \cdots + x_4^{11}$	0.613	0.447	0.585
$x_4 + x_4^2 + \cdots + x_4^{12}$	0.614	0.447	0.585
$x_4 + x_4^2 + \cdots + x_4^{13}$	0.638	0.448	0.619
$x_4 + x_4^2 + \cdots + x_4^{14}$	0.638	0.448	0.619
$x_4 + x_4^2 + \cdots + x_4^{15}$	0.638	0.448	0.619

Table 1 shows the values of R-squared regressed upon the dependent variables `cnt`, `casual`, and `registered` for each value of n for values of 1 to 15. For `cnt` and `registered`, a good value of n appears to be 7. $n = 7$ gives a huge increase in R-squared over $n = 6$ for `cnt` and `registered`. Only small increases of R-squared are seen for values of n above 7 for `cnt` and `registered`. Although these increases are still statistically significant, it is not a good idea to have very high order polynomials as such a regression is likely to overfit

the data. For example, very high order polynomials may give poor predictions for data in 2013 or beyond. For `casual`, a good value of n appears to be 3. Only small increases of R-squared are seen for values of n above 3 for `casual`.

Therefore, the design matrix if b is `cnt` or `registered` is $A \in \mathbb{R}^{17379 \times 11}$, where $m = 17379$:

$$A = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{14} & x_{14}^2 & x_{14}^3 & x_{14}^4 & x_{14}^5 & x_{14}^6 & x_{14}^7 & x_{19} \\ 1 & x_{21} & x_{22} & x_{24} & x_{24}^2 & x_{24}^3 & x_{24}^4 & x_{24}^5 & x_{24}^6 & x_{24}^7 & x_{29} \\ 1 & x_{31} & x_{32} & x_{34} & x_{34}^2 & x_{34}^3 & x_{34}^4 & x_{34}^5 & x_{34}^6 & x_{34}^7 & x_{39} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{m1} & x_{m2} & x_{m4} & x_{m4}^2 & x_{m4}^3 & x_{m4}^4 & x_{m4}^5 & x_{m4}^6 & x_{m4}^7 & x_{m9} \end{bmatrix}$$

The design matrix if b is `casual` is $A \in \mathbb{R}^{17379 \times 7}$, where $m = 17379$:

$$A = \begin{bmatrix} 1 & x_{11} & x_{12} & x_{14} & x_{14}^2 & x_{14}^3 & x_{19} \\ 1 & x_{21} & x_{22} & x_{24} & x_{24}^2 & x_{24}^3 & x_{29} \\ 1 & x_{31} & x_{32} & x_{34} & x_{34}^2 & x_{34}^3 & x_{39} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{m1} & x_{m2} & x_{m4} & x_{m4}^2 & x_{m4}^3 & x_{m9} \end{bmatrix}$$

Table 2: Comparison of speed (in milliseconds) of custom design_matrix() functions with built-in R model.matrix.lm() function

expr	min	lq	mean	median	uq	max	neval
design_matrix()	19.583	21.756	34.720	27.329	37.945	163.543	100
design_matrix_Cpp()	7.528	8.113	12.063	9.119	11.330	43.002	100
model.matrix.lm()	19.591	23.590	41.094	31.195	41.902	243.177	100

We made a custom function called `design_matrix()` in R and a Rcpp (C++) version called `design_matrix_Cpp()` to form the design matrix. This Rcpp tutorial was useful in converting the R functions to C++. The R implementation was slightly faster than the base built-in R version called `model.matrix.lm()` and the C++ version was three times faster than built-in R version. **Table 2** shows a speed comparison using R package `microbenchmark` showing minimum, 25th percentile, 50th percentile, mean, 75th percentile, and max time of 100 replications of the function to form the design matrix with $n = 7$ and $A \in \mathbb{R}^{17379 \times 11}$.

3. Normal Equation

The simplest way to solve $Ax = b$ would be to do use the normal equations.

The solution to $Ax = b$ using the normal equations is $x = (A^T A)^{-1} A^T b$

However, computers cannot represent real numbers exactly. The condition number $\kappa(A)$ of the input matrix A represents how much error there could be in the output. The condition number of $A^T A$ is equal to condition number of A^2. For our example, as n increases (a more complex and higher order polynomial), the condition number $\kappa(A)$ increases significantly. If the condition number is too high, then a computer will treat a non-singular matrix as singular. Then, there will be very large errors in the computation.

Table 3: Condition number of each value of n (maximum power of polynomial for `hour`) and relative error of normal equations versus SVD

	$\kappa(A)^2$	Relative error/error message R	Relative error C++
x_4	6.49×10^3	5.87×10^{-12}	6.17×10^{-12}
$x_4 + x_4^2$	2.09×10^6	-1.1×10^{-14}	-1.7×10^{-14}
$x_4 + x_4^2 + x_4^3$	9.08×10^8	1.18×10^{-12}	1.41×10^{-12}
$x_4 + x_4^2 + \cdots + x_4^4$	4.17×10^{11}	2.87×10^{-12}	2.71×10^{-12}
$x_4 + x_4^2 + \cdots + x_4^5$	2.03×10^{14}	1.52×10^{-8}	1.50×10^{-8}
$x_4 + x_4^2 + \cdots + x_4^6$	1.29×10^{17}	Error, Recripocal $\kappa(A)$: 5.89×10^{-18}	1.24×10^{-9}
$x_4 + x_4^2 + \cdots + x_4^7$	1.12×10^{20}	Error, Recripocal $\kappa(A)$: 6.14×10^{-21}	1.18×10^{-3}
$x_4 + x_4^2 + \cdots + x_4^8$	1.11×10^{23}	Error, Recripocal $\kappa(A)$: 6.35×10^{-24}	5.99×10^{-3}
$x_4 + x_4^2 + \cdots + x_4^9$	1.22×10^{26}	Error, Recripocal $\kappa(A)$: 6.78×10^{-27}	1.06×10^{-1}
$x_4 + x_4^2 + \cdots + x_4^{10}$	1.46×10^{29}	Error, Recripocal $\kappa(A)$: 1.29×10^{-29}	1.02×10^1
$x_4 + x_4^2 + \cdots + x_4^{11}$	1.94×10^{32}	Error, Recripocal $\kappa(A)$: 1.07×10^{-32}	1.06×10^1
$x_4 + x_4^2 + \cdots + x_4^{12}$	2.89×10^{35}	Error, Recripocal $\kappa(A)$: 7.16×10^{-35}	1.03×10^0
$x_4 + x_4^2 + \cdots + x_4^{13}$	4.86×10^{38}	Error, Recripocal $\kappa(A)$: 1.13×10^{-37}	2.12×10^0
$x_4 + x_4^2 + \cdots + x_4^{14}$	3.43×10^{45}	Error, Recripocal $\kappa(A)$: 2.67×10^{-40}	1.00×10^0
$x_4 + x_4^2 + \cdots + x_4^{15}$	2.04×10^{45}	Error, Recripocal $\kappa(A)$: 2.51×10^{-43}	1.09×10^0

Table 3 shows the square of the condition number for matrix A, the relative error/error message when solving $Ax = b$ for each value of n using R, and the relative error using Rcpp (C++). The relative error is calculated as the maximum error of any coefficient of the normal equations when compared to SVD (SVD is considered computationally precise).

However, R gives an error trying to find the inverse if the matrix is close to singular. The error message is the following: 'Error in solve.default(t(A) %*% A): system is computationally singular: reciprocal condition number ='. The reciprocal condition number given in the error message by R is close to the conditional number of A^2. On the other hand, RcppArmadillo does not give an error when finding the inverse of a system with a very high condition number. When $n > 9$ and $\kappa(A^TA) > 10^{29}$, the relative error is 100% or higher. R appears to give an error if the relative error is greater than 0.1% and $\kappa(A^TA) > 10^{17}$. With such large errors possible, it is not recommended to ever use the normal equations when solving $Ax = b$ on a computer.

Table 4: Comparison of speed (in milliseconds) of solving $Ax = b$ where $A \in \mathbb{R}^{17379 \times 7}$ using normal equations implemented in R vs Rcpp (C++)

expr	min	lq	mean	median	uq	max	neval
normal_equations(A, b)	6.047	6.572	9.275	7.478	9.326	28.770	100
normal_equations_Cpp(A, b)	2.244	2.428	2.830	2.648	3.094	7.003	100

Table 4 shows that the Rcpp (C++) implementation is on average 4 times faster than the R implementation of solving the normal equations.

4. QR Decomposition

Another way, a more computationally accurate way, to solve $Ax = b$ to use QR decomposition.

First, find the reduced QR decomposition such that $A = \hat{Q}\hat{R}$, $\hat{Q}$ is an orthogonal matrix, and $\hat{R}$ is an upper triangular matrix. $\hat{Q}$ meets the following condition: $\hat{Q}^T\hat{Q} = \hat{Q}\hat{Q}^T = I$. Also, $\hat{Q} \in \mathbb{R}^{17379 \times 7}$ and $\hat{R} \in \mathbb{R}^{7 \times 7}$

in this case as $A \in \mathbb{R}^{17379 \times 7}$

The solution to $Ax = b$ using QR decomposition is $x = \hat{R}^{-1}\hat{Q}^T b$.

Table 5: Comparison of speed (in milliseconds) of solving $Ax = b$ where $A \in \mathbb{R}^{17379 \times 7}$ using QR decomposition implemented in R vs Rcpp (C++)

expr	min	lq	mean	median	uq	max	neval
qr.solve(A, b)	5.539	6.440	9.661	7.321	9.834	32.641	100
qr_solve_Cpp(A, b)	5.380	6.324	9.964	7.442	9.713	49.032	100

Table 5 shows that the Rcpp (C++) implementation is slightly faster than R implementation of using QR decomposition to solve $Ax = b$.

5. Singular Value Decomposition

A longer way to solve $Ax = b$, but an even more accurate way than QR decomposition is to use Singular Value Decomposition.

First, find the reduced SVD such that $A = \hat{U}\hat{\Sigma}V^T$.

Also, $\hat{U} \in \mathbb{R}^{17379 \times 7}$, $V \in \mathbb{R}^{7 \times 7}$, and $\hat{\Sigma} \in \mathbb{R}^{7 \times 7}$ in this case as $A \in \mathbb{R}^{17379 \times 7}$.

The solution to $Ax = b$ using SVD is $x = V(\hat{U}^T b / \hat{\Sigma})$

Table 6: Comparison of speed (in milliseconds) of solving $Ax = b$ where $A \in \mathbb{R}^{17379 \times 7}$ using SVD implemented in R vs Rcpp (C++)

expr	min	lq	mean	median	uq	max	neval
svd_solve(A, b)	11.051	13.403	22.990	16.567	26.835	91.955	100
svd_solve_Cpp(A, b)	8.607	11.160	17.828	13.372	21.892	45.689	100

Table 6 shows that the Rcpp (C++) implementation is slightly faster than R implementation of using SVD to solve $Ax = b$.

Now, it is time to compare the speed of solving $Ax = b$. If theory is correct, the normal equations should be fastest, followed by QR decomposition, and followed by SVD.

Table 7: Comparison of speed (in milliseconds) of solving $Ax = b$ where $A \in \mathbb{R}^{17379 \times 11}$ using QR decomposition or SVD implemented in R vs Rcpp (C++)

expr	min	lq	mean	median	uq	max	neval
normal_equations_Cpp(A, b)	3.990	4.554	6.474	4.898	6.479	46.761	100
qr.solve(A, b)	10.617	12.831	23.327	17.289	23.969	253.415	100
qr_solve_Cpp(A, b)	12.097	13.886	21.197	16.199	23.211	100.417	100
svd_solve(A, b)	20.816	25.097	40.080	31.263	40.285	139.866	100
svd_solve_Cpp(A,b)	18.233	21.014	31.122	25.653	33.904	127.162	100

Note that **Table 7** uses $n = 7$, the design matrix for `registered` and `cnt` instead of $n = 3$, the design matrix for `casual` in **Table 4, 5, 6**.

Normal equations using Rcpp (C++) is about 3 times faster than QR decomposition using C++. QR decomposition is about 70% faster than SVD. QR decomposition implemented in R is about twice as fast as using SVD implemented in R. Thus, we see that the results from **Table 7** are in line with what we should expect from theory.

6. Final Regression Model

Recall that y_1 = `casual`, y_3 = `cnt`, x_1 = `season`, x_2 = `yr`, x_4 = `hour`, x_9 = `temp` for below.

$y_1 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_4 + \beta_4 x_4^2 + \beta_5 x_4^3 + \beta_6 x_9$

$y_3 = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_4 + \beta_4 x_4^2 + \beta_5 x_4^3 + \beta_6 x_4^4 + \beta_7 x_4^5 + \beta_8 x_4^6 + \beta_9 x_4^7 + \beta_{10} x_9$

The solution to $Ax = b$ for y_1 is:

$$x = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \end{bmatrix} = \begin{bmatrix} -42.145 \\ 0.6875 \\ 12.887 \\ -4.859 \\ 1.275 \\ -0.047 \\ 89.528 \end{bmatrix}$$

The solution to $Ax = b$ for y_3 is:

$$x = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \\ \beta_4 \\ \beta_5 \\ \beta_6 \\ \beta_7 \\ \beta_8 \\ \beta_9 \\ \beta_{10} \end{bmatrix} = \begin{bmatrix} -145.05 \\ 17.18 \\ 88.57 \\ 29.45 \\ -63.199 \\ 24.27 \\ -3.62 \\ 0.258 \\ -0.0088 \\ 0.000116 \\ 243.131 \end{bmatrix}$$

Inputting the solution results in:

$y_1 = -42.145 + 0.6875 x_1 + 12.887 x_2 - 4.859 x_4 + 1.275 x_4^2 - 0.047 x_4^3 + 89.528 x_9$

$y_3 = -145.05 + 17.18 x_1 + 88.57 x_2 + 29.45 x_4 - 63.199 x_4^2 + 24.27 x_4^3 - 3.62 x_4^4 + 0.258 x_4^5 - 0.0088 x_4^6 + 0.000116 x_4^7 + 243.131 x_9$

Note that $\frac{1}{n}\sum_{i=1}^{n} x_1 = \bar{x_1} = 2.50164$, $\frac{1}{n}\sum_{i=1}^{n} x_2 = \bar{x_2} = 0.5025606$, $\frac{1}{n}\sum_{i=1}^{n} x_8 = \bar{x_8} = 0.4970$.

Input the means of x_1, x_2, and x_8 into the formulas of y_1 and y_3.

This results in eliminating all independent variables that aren't `hour` (x_4), resulting in the following two functions:

$y_1 = -4.859 x_4 + 1.275 x_4^2 - 0.047 x_4^3 + 10.54523$

$y_3 = 29.45 x_4 - 63.199 x_4^2 + 24.27 x_4^3 - 3.62 x_4^4 + 0.258 x_4^5 - 0.0088 x_4^6 + 0.000116 x_4^7 + 63.26103.$

This now allows us to graph `casual` (y_1) and `registered` (y_3) as a function of x_4 (`hour`) and see how well the least squares regression fits the actual data.

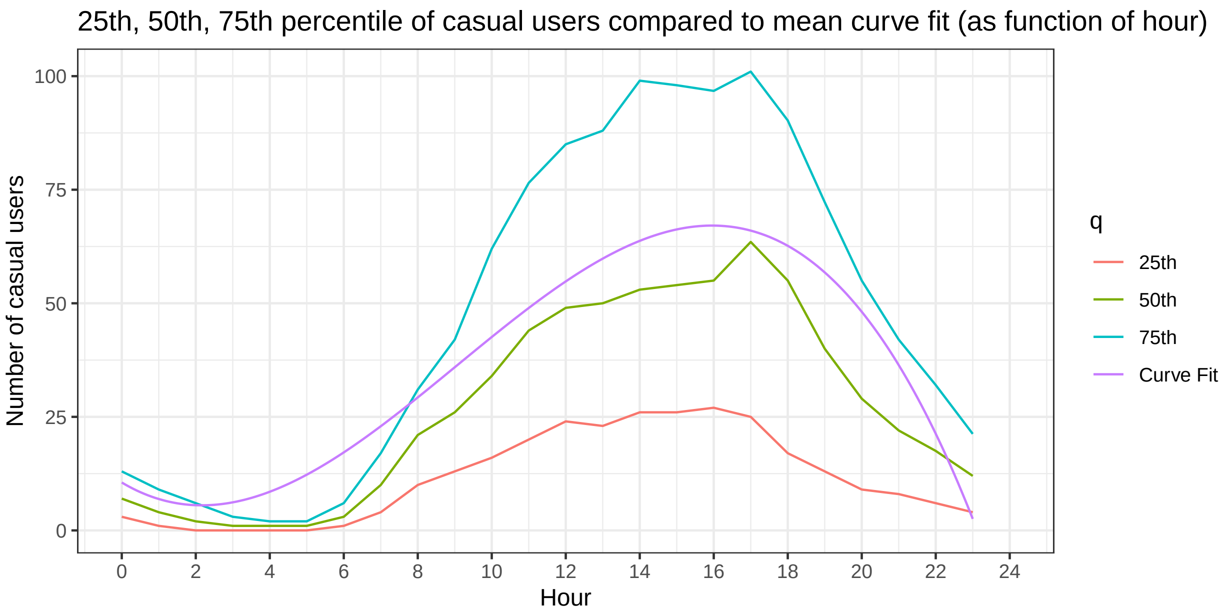

Figure 5: The 25th, 50th, and 75th percentile of casual users (731 days) are graphed for each hour of the day. The mean values of season, yr, and temp are entered into the regression solution to give a constant value. Thus, the regression is now only in terms of hour. Then, the mean curve fit of the regression can be compared to the 25th, 50th, and 75th percentiles of casual users.

Figure 5 shows that the cubic polynomial function is not a great fit for hours 4 to 6 as the mean curve fit is above the 75th percentile. The curve does not dip down far enough from hours 4 to 6. Note that interquartile range (75th percentile minus 25th percentile) for `casual` is very large from **Figure 5** which might make predicting data quite difficult. Also **Figure 5** suggests that `casual` is skewed right since the mean curve fit is almost always above the 50th percentile.

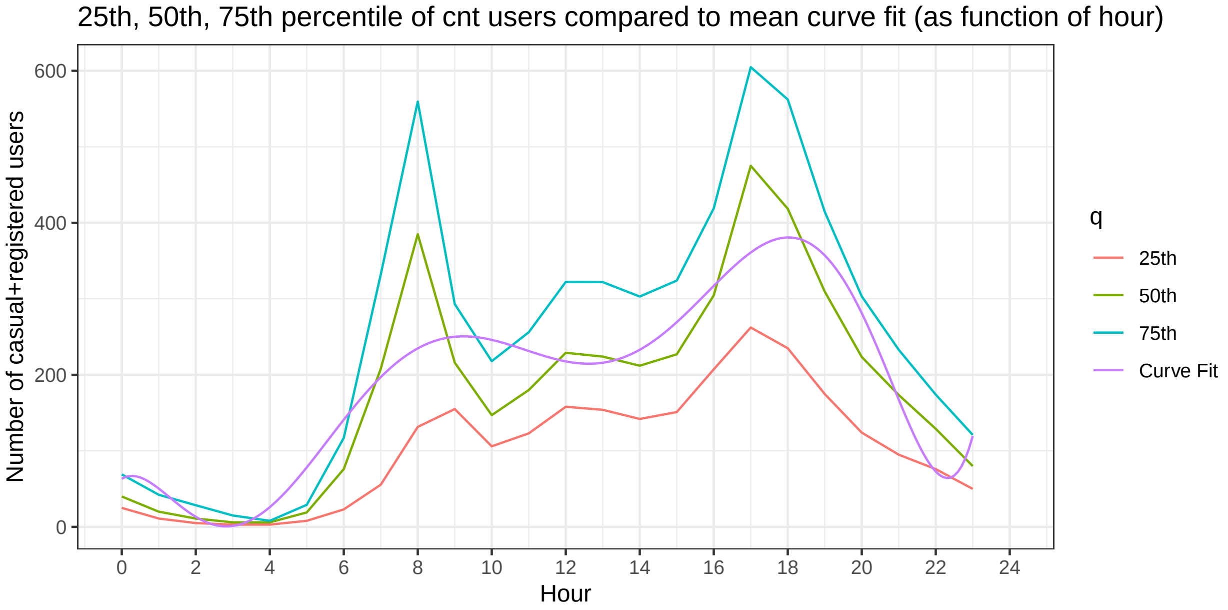

Figure 6: Same description as for Figure 5, except the variable is now cnt (sum of registered and casual users) rather than casual.

Figure 6 shows the 7-degree (septic) polynomial function is quite a good fit for `cnt`. However, near hour 22, the mean curve fit starts to slope upwards again, which is not correct. Also, similar to **Figure 5**, the mean curve fit overestimates the number of users from hours 4 to 6 again.