NYC Taxi & Limosuine Commission

Automatidata has been commissioned to analysis the 2017 activities of the Newy York City Taxi and Limosuine Commission. in this document we will provide descriptive statistics, A/B hypothesis testing and visualizations to assist in data-driven decision making and regression modelin for predictions of the `total_amount` of expected customer charges. The model input variables should be obtainable at the time of accepting th ride service in the form of trip distance, trip duration (in minutes), payment type (cash or credit card) and Rate ID.

The TLC data for 2017 contains a total of 18478 records of 22 variables that span Jan 1st through Dec 31st. The data in Table 1 below shows descriptive statistics of the cleaned dataset provided by NYC TLC. See the Data Cleaning Record in Appendix A for more details.

Table 1: Cleaned data

		VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store_and_fwd_fl
1	3227	2	2017-01-01T00:08:25	2017-01-01T00:17:20	1	0.52	1	"N"
2	9506	1	2017-01-01T00:26:35	2017-01-01T00:36:12	1	0.9	1	"N"
3	5951	1	2017-01-01T00:43:10	2017-01-01T00:57:21	4	4.9	1	"N"
4	5119	1	2017-01-01T01:21:59	2017-01-01T01:34:12	3	2.1	1	"N"
5	9310	1	2017-01-01T01:32:44	2017-01-01T01:47:41	2	5.4	1	"N"
6	9488	1	2017-01-01T01:33:12	2017-01-01T01:44:09	1	2.2	1	"N"
7	9763	2	2017-01-01T01:36:30	2017-01-01T01:42:17	1	0.88	1	"N"
8	4442	2	2017-01-01T01:46:05	2017-01-01T01:57:33	5	2.12	1	"N"
9	9315	2	2017-01-01T01:59:22	2017-01-01T02:12:59	5	5.57	1	"N"
10	5273	2	2017-01-01T02:15:23	2017-01-01T02:22:51	6	0.75	1	"N"

Table 2: Dataset description

	variable	mean	std	min	max	sum	nmissing	nun
1	:ID	5.67774e7	3.26907e7	12127	113467934	1049133197034	0	notl
2	:VendorID	1.55585	0.496884	1	2	28749	0	notl
3	:tpep_pickup_datetime	nothing	nothing	2017-01-01T00:08:25	2017-12-31T22:28:54	nothing	0	184
4	:tpep_dropoff_datetime	nothing	nothing	2017-01-01T00:17:20	2017-12-31T22:37:29	nothing	0	184
5	:passenger_count	1.65386	1.28879	1	6	30560	0	notl
6	:trip_distance	3.40632	3.86683	0.27	33.96	62942.0	0	notl
7	:RatecodeID	1.03599	0.240315	1	5	19143	0	notl
8	:store_and_fwd_flag	nothing	nothing	"N"	"Y"	nothing	0	2
9	:PULocationID	161.068	66.1152	4	265	2976218	0	notl
10	:DOLocationID	159.849	70.1784	1	265	2953696	0	notl
	more							
22	:cost_per_mile	7.11531	2.945	2.23025	24.6667	1.31477e5	0	notl

Selection deleted

Payment Type Impact Analysis

- Business Task: Answer whether or not customers that pay with a credit card pay higher fare amounts on average than customers who pay with cash.

Cash vs Credit Card Hypothesis Testing

- H_0: There is no difference in the mean total fare collected from customers who pay with cash and those who pay with a credit card
- H_a: Customers who pay with a credit card do not pay the same total fare amount as those who pay with cash.

```
total_amount_mean = 18.13
```

```
cash_mean = 15.46
```

```
cc_mean = 19.31
```

```
mean_difference = 3.85
```

There is a mean difference between cash and credit card customers of **$3.85**. We will use a significance level of **5%** and calculate the p-value for this analysis using a Two Sample Z-test.

```
p_value = 2.5558229343878966e-81
```

```
    Two sample z-test (unequal variance)
    ------------------------------------
    Population details:
        parameter of interest:   Mean difference
        value under h_0:         0
        point estimate:          3.85179
        95% confidence interval: (3.457, 4.247)

    Test summary:
        outcome with 95% confidence: reject h_0
        two-sided p-value:           <1e-80

    Details:
        number of observations:   [12723,5669]
        z-statistic:              19.099363816438345
        population standard error: 0.20167114771822192
```

Conclusion

The p-value for the two sample z-test is very small and we confidently reject the null hypothesis. **The $3.85 mean increase in fares of customer paying with a credit card is statistically significant.**

Next Steps

NYC TLC should encourage customers to use their credit cards for payment whenever possible. This behavior could be encouraged by

- Allow online payments with credit cards.
- Offer rewards or discounts (below the mean increase of $3.85) for customers who regularly pay with their credit card.

Selection deleted

Total Charge Modeling

We will create a multi-variate regression model to predict the **`total_charge`** as our dependent variable with **`trip_distance`**, **`RatecodeID`**, anc **`payment_type`** as our independent variables.

Variable Analysis

The variables `fare_amount`, `extra`, `mta_tax`, `tip_amount`, `tolls_amount`, and `improvement_surcharge` are summed to calculate the target variable `total_amount` and are not independent from `total_amount`, and thus excluded from the model.

The variables `PULocationID` and `DOLocationID` are (presumably) used to calculate the `trip_distance` and thus highly colinear and excluded from modeling. It is shown that the calculated variable `trip_duration_minutes` is also highly colinear with `trip_distance` but has a higher variance and so is excluded from modeling.

The `tpep_pickup_datetime` and `tpep_dropoff_datetime` variables are used to calculate the `trip_duration_minutes` variable and have been included in the model.

The `hour_of_day`, `day_of_week`, and `VendorID` variables show no linear relationship with `total_amount` according to Graph 8.

The `store_and_fwd_flag` variable is used for record storage and no relationship is expected between `total_amount`.

For scatter plots showing the graphical relationship between several variables see Graphs 6 - 8 in Appendix A.

Table1 & Table2: Train / Test Datasets

Using random sampling with an 80 / 20 split. `RatecodeID` has been coerced into a String datatype for treatment as a categorical variable by th model. Removed "No Charge" and "Dispute" groups from `payment_type` variable as not applicable records for modeling.

(

	ID	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	s
1	111133227	2	2017-01-01T00:08:25	2017-01-01T00:17:20	1	0.52	"1"	"
2	111139506	1	2017-01-01T00:26:35	2017-01-01T00:36:12	1	0.9	"1"	"
3	111145951	1	2017-01-01T00:43:10	2017-01-01T00:57:21	4	4.9	"1"	"
4	111055119	1	2017-01-01T01:21:59	2017-01-01T01:34:12	3	2.1	"1"	"
5	111059310	1	2017-01-01T01:32:44	2017-01-01T01:47:41	2	5.4	"1"	"
6	111059488	1	2017-01-01T01:33:12	2017-01-01T01:44:09	1	2.2	"1"	"
7	111060763	2	2017-01-01T01:36:30	2017-01-01T01:42:17	1	0.88	"1"	"
8	111064442	2	2017-01-01T01:46:05	2017-01-01T01:57:33	5	2.12	"1"	"
9	111069315	2	2017-01-01T01:59:22	2017-01-01T02:12:59	5	5.57	"1"	"
10	111075273	2	2017-01-01T02:15:23	2017-01-01T02:22:51	6	0.75	"1"	"
more								
14856	110627349	2	2017-12-30T18:48:37	2017-12-30T19:03:48	3	7.97	"1"	"

Some splits have a single or no representative of some class.

Selection deleted

Linear Regression Model

`lr` =
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredChol{Float64, LinearAlgebra.Cholesky

total_amount ~ 1 + trip_distance + trip_duration_minute + payment_type + RatecodeID

Coefficients:

	Coef.	Std. Error	t	Pr(>|t|)	Lower 95%	Upper 95%
(Intercept)	3.39628	0.0636133	53.39	<1e-99	3.27159	3.52097
trip_distance	2.86815	0.0118966	241.09	<1e-99	2.84483	2.89147
trip_duration_minute	0.167591	0.00264066	63.47	<1e-99	0.162415	0.172767
payment_type: Credit Card	2.94471	0.0634128	46.44	<1e-99	2.82041	3.06901
RatecodeID: 2	-0.178009	0.237529	-0.75	0.4536	-0.643595	0.287577
RatecodeID: 3	31.4129	0.658629	47.69	<1e-99	30.1219	32.7039
RatecodeID: 4	12.2969	1.36142	9.03	<1e-18	9.62832	14.9654
RatecodeID: 5	46.1238	0.736176	62.65	<1e-99	44.6808	47.5668

StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.DensePredChol{Float64, LinearAlgebra.CholeskyP

total_amount ~ 1 + trip_distance + trip_duration_minute + payment_type + RatecodeID

Coefficients:

	Coef.	Std. Error	t	Pr(>|t|)	Lower 95%	Upper 95%
(Intercept)	3.39628	0.0636133	53.39	<1e-99	3.27159	3.52097
trip_distance	2.86815	0.0118966	241.09	<1e-99	2.84483	2.89147
trip_duration_minute	0.167591	0.00264066	63.47	<1e-99	0.162415	0.172767
payment_type: Credit Card	2.94471	0.0634128	46.44	<1e-99	2.82041	3.06901
RatecodeID: 2	-0.178009	0.237529	-0.75	0.4536	-0.643595	0.287577
RatecodeID: 3	31.4129	0.658629	47.69	<1e-99	30.1219	32.7039
RatecodeID: 4	12.2969	1.36142	9.03	<1e-18	9.62832	14.9654
RatecodeID: 5	46.1238	0.736176	62.65	<1e-99	44.6808	47.5668

```
MLJ.save("NYC_TLC_lr.jls", lr)
```

[1.60918, 1.31252, 1.0009, 1.07295]

```
GLM.gvif(lr, scale=true) # Generalized Variance Inflation Factor
```

(0.937416, 0.937386)

```
r²(lr), adjr²(lr)  # training dataset coefficient of determination
```

Linear Model Metrics on Test Dataset

```
# testing dataset prediction
begin
    ŷ = GLM.predict(lr, select(test, Not(:total_amount))) |> disallowmissing
    y = test.total_amount
    r = ŷ - y
end;
```

Coefficients of Determination

(0.934378, 0.934303)

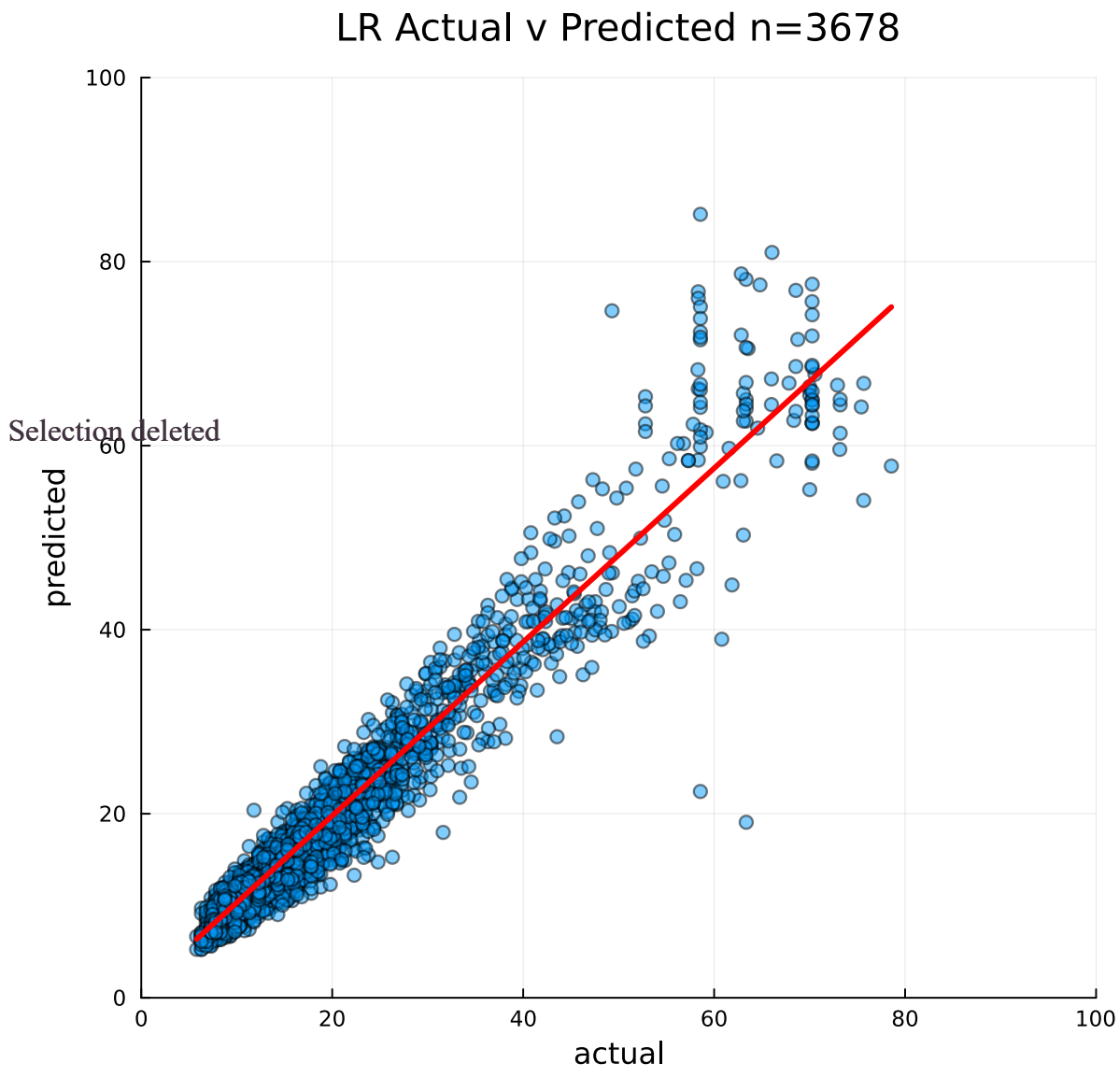
```
R², Adjusted_R² = r2_score(ŷ, y), adjusted_r2_score(ŷ, y, 4)
```

Mean Error Metrics

(1.84376, 8.80483, 2.96729)

```
# Mean Absolute Error, Mean Squared Error, Root Mean Squared Error
MLJ.mae(ŷ, y), mse(ŷ, y), √mse(ŷ, y)
```

Graph 1: Linear Model Predictions

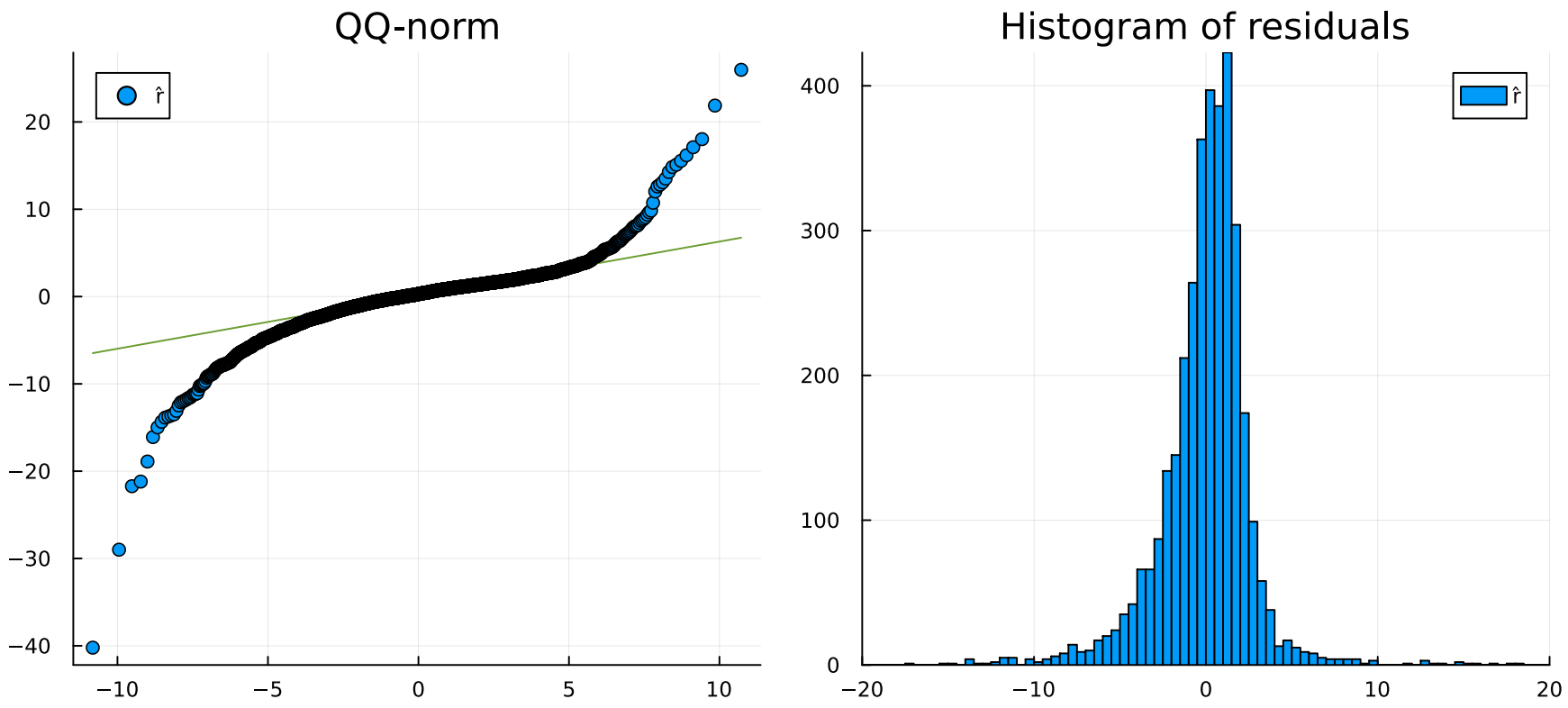


Selection deleted

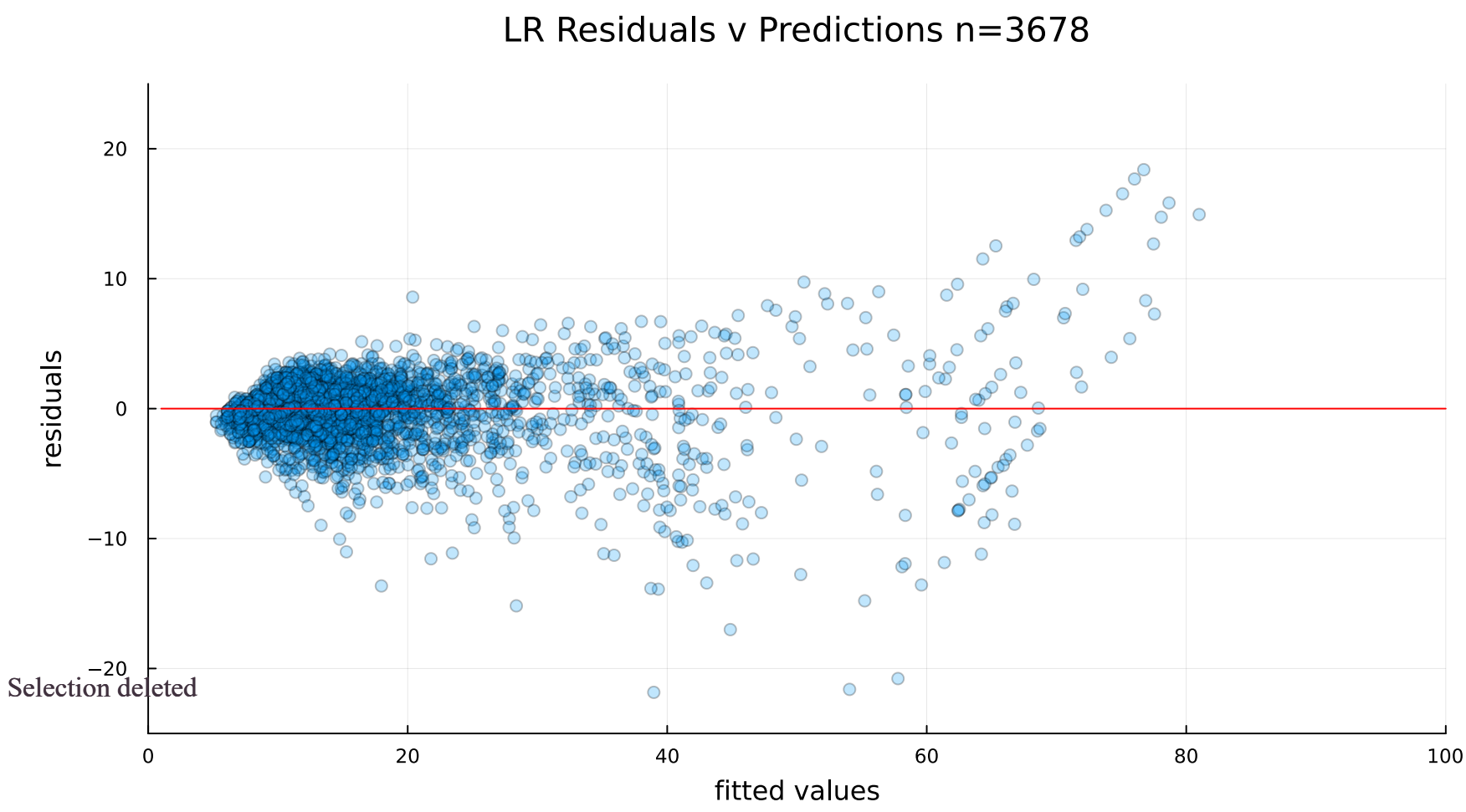
Checking Assumptions of a model validity

1. Independent observations
2. Linear relationships between variables
3. Normality (Q-Q plot)
4. Homoscedasticity (scatter plot of residuals to fitted values to resemble a random cloud of data... no recognizable patterns should be present)

Graph 2: Normality of Residuals



Graph 3: Constant Variance



Selection deleted

Not bad, actually... The R-value shows the model is able to explain much of the variances, however, there is some weirdness in the variance of the residuals with two apparent, seperate effects in the lower and upper values of the fitted `total_amount` predictions. This may be due to some systemic affect on the way the total amount is calculated and may be due to tolls, taxes, and other additional fees independent of the tri duration and rate code.

Random Forest Modeling

A 'Random Forest' is a technique of training a collection of models on boot-strapped data and aggegrating the results into a final prediciton that tends to be more accuracte than any single model. The drawback with this modeling method is that the model is much less explicable than linear regression and fewer conclusions and recommendations can be drawn from an analysis of the model.

We will be using this model in production and to validate the findings of the linear regression model (and the recommendations that flow fror the analysis of the coefficients).

```
rf = MLJDecisionTreeInterface.RandomForestRegressor
```

```
r1 = NominalRange(n_trees = 5, 10, 15, ...)
```

```
r2 = NominalRange(min_samples_leaf = 2, 5, 10, ...)
```

```
DeterministicTunedModel(
  model = RandomForestRegressor(
        max_depth = -1,
        min_samples_leaf = 1,
        min_samples_split = 2,
        min_purity_increase = 0.0,
        n_subfeatures = -1,
        n_trees = 100,
        sampling_fraction = 0.7,
        feature_importance = :impurity,
        rng = Random._GLOBAL_RNG()),
  tuning = Grid(
        goal = nothing,
        resolution = 5,
        shuffle = true,
        rng = Random._GLOBAL_RNG()),
  resampling = StratifiedCV(
        nfolds = 5,
        shuffle = false,
        rng = Random._GLOBAL_RNG()),
  measure = Metrics.mse,
  weights = nothing,
  class_weights = nothing,
  operation = nothing,
  range = MLJBase.NominalRange{Int64, 5}[NominalRange(n_trees = 5, 10, 15, ...), NominalRange(min_samples_leaf = 2, 5, 10,
  selection_heuristic = MLJTuning.NaiveSelection(nothing),
  train_best = true,
  repeats = 1,
  n = nothing,
  acceleration = CPUThreads{Int64}(6),
  acceleration_resampling = CPU1{Nothing}(nothing),
  check_measure = true,
  cache = true)
```

```
rf_mach = trained Machine; does not cache data
          model: DeterministicTunedModel(model = RandomForestRegressor(max_depth = -1, …), …)
          args:
```

```
RandomForestRegressor(
  max_depth = -1,
  min_samples_leaf = 2,
  min_samples_split = 2,
  min_purity_increase = 0.0,
  n_subfeatures = -1,
  n_trees = 20,
  sampling_fraction = 0.7,
  feature_importance = :impurity,
  rng = Random._GLOBAL_RNG())
```

Random Forest Model Metrics on Test Dataset

Selection deleted

```
([36.3043, 43.9794, 46.2378, 67.4659, 44.3449, 43.4426, 43.5249, 68.6245, 39.9288,  more ,10.7777], [38.21, 45.99, 51.41,
```

```
1 rf_ŷ, rf_y = MLJ.predict(rf_mach,
2     select(test, [:trip_distance, :trip_duration_minute, :payment_type, :RatecodeID])), test.total_amount
```

Coefficients of Determination

```
(0.985935, 0.985919)
```

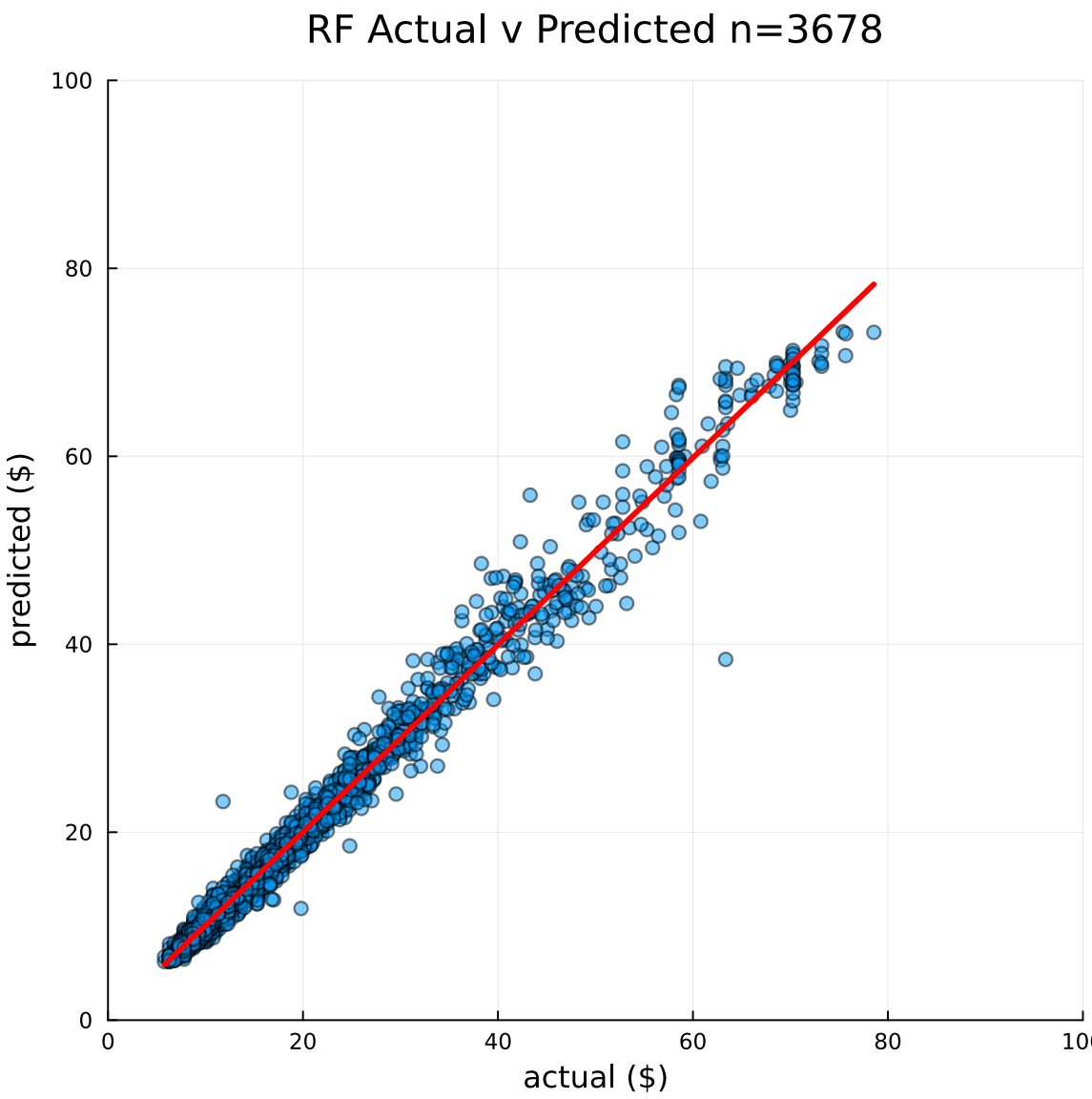
```
1 r2_score(rf_ŷ, rf_y), adjusted_r2_score(rf_ŷ, rf_y, 4)  # test datset coefficient of determination
```

Mean Error Metrics

```
(0.79062, 1.88721, 1.37376)
```

```
1 MLJ.mae(rf_ŷ, rf_y), mse(rf_ŷ, rf_y), √mse(rf_ŷ, rf_y)
```

Graph 4: Random Forest Model Predictions



Model persistance

Model persistance is turned off by default, enable the cell below to re-save the model.

```
1 MLJ.save("NYC_TLC_rf_model.jls", mach)
```

Conclusion

Recommendations

The Random Forest model outperforms the linear regression model on all calculated metrics including the R^2, Adjusted R^2, MSE and MAE scores and should be used in production by NYC TLC development engineers for fare prediction applications.

The final model is the result of a five-fold cross-validation process and grid search of two model hyper-parameters: `n_trees` and `n_sample_leaf` , constructing a total of twenty-five models. **The champion model performs quite well on the test dataset with an average prediction error of $1.15 or 6.0% of the mean total fare amount**. For a visual represention of final model performance, see Graph 4

The inputs expected by the model are (in order):

`trip_distance` ::Int

Selection deleted

`trip_duration_minute` ::Int

`payment_type` ::Categorical

`RatecodeID` ::Categorical

The persisted model will be delivered alongside this report as `NYC_TLC_rf_model.jls` . For any problems or questions please contact:

Dustin Irwin

dusty.irwin@gmail.com

Generous Tip Modeling

We will attempt to predict whether or not the customer will leave more than a 20% tip.

XGBoost Model

```
xgb = XGBoostClassifier
```

```
(
  1:  CategoricalArrays.CategoricalVector{String, UInt32, String, CategoricalArrays.CategoricalValue{String, UInt32}, U
```

2:

	trip_distance	trip_duration_minute	payment_type	RatecodeID	day_of_week	hour_of_day
1	0.52	9.0	1.0	1.0	7	0
2	0.9	10.0	0.0	1.0	7	0
3	4.9	14.0	1.0	1.0	7	0
4	2.1	12.0	1.0	1.0	7	1
5	5.4	15.0	1.0	1.0	7	1
6	2.2	11.0	0.0	1.0	7	1
7	0.88	6.0	0.0	1.0	7	1
8	2.12	11.0	1.0	1.0	7	1
9	5.57	14.0	1.0	1.0	7	1
10	0.75	7.0	0.0	1.0	7	2
	more					
14782	2.67	23.0	1.0	1.0	5	10

```
  3:  CategoricalArrays.CategoricalVector{String, UInt32, String, CategoricalArrays.CategoricalValue{String, UInt32}, U
```

4:

	trip_distance	trip_duration_minute	payment_type	RatecodeID	day_of_week	hour_of_day
1	10.3	62.0	0.0	1.0	4	16
2	0.8	22.0	0.0	1.0	4	17
3	1.0	7.0	0.0	1.0	4	17
4	1.3	21.0	0.0	1.0	4	18
5	0.8	13.0	0.0	1.0	4	18
6	10.8	44.0	0.0	1.0	4	19
7	2.15	18.0	0.0	1.0	4	19
8	8.04	24.0	0.0	1.0	4	19
9	1.52	10.0	0.0	1.0	4	20
10	1.9	12.0	0.0	1.0	4	20
	more					
3696	1.5	9.0	0.0	1.0	7	22

```
)
```

Selection deleted

names	scitypes	types
trip_distance	Continuous	Float64
trip_duration_minute	Continuous	Float64
payment_type	Continuous	Float64
RatecodeID	Continuous	Float64
day_of_week	Count	Int64
hour_of_day	Count	Int64

```
  (Dict("G" ⇒ 7170, "NG" ⇒ 7612), AbstractVector{OrderedFactor{2}})
```

```
tr1 = NominalRange(eta = 0.1, 0.2, 0.3, ...)
```

```
tr2 = NominalRange(max_depth = 1, 2, 3, ...)
```

```
txgb =
ProbabilisticTunedModel(
  model = XGBoostClassifier(
        test = 1,
        num_round = 100,
        booster = "gbtree",
        disable_default_eval_metric = 0,
        eta = 0.3,
        num_parallel_tree = 1,
        gamma = 0.0,
        max_depth = 6,
        min_child_weight = 1.0,
        max_delta_step = 0.0,
        subsample = 1.0,
        colsample_bytree = 1.0,
        colsample_bylevel = 1.0,
        colsample_bynode = 1.0,
        lambda = 1.0,
        alpha = 0.0,
        tree_method = "auto",
        sketch_eps = 0.03,
        scale_pos_weight = 1.0,
        updater = nothing,
        refresh_leaf = 1,
        process_type = "default",
        grow_policy = "depthwise",
        max_leaves = 0,
        max_bin = 256,
        predictor = "cpu_predictor",
        sample_type = "uniform",
        normalize_type = "tree",
        rate_drop = 0.0,
```

```
xgb_mach =
untrained Machine; caches model-specific representations of data
  model: XGBoostClassifier(test = 1, …)
  args:
    1:  Source @826 ⏎ ScientificTypesBase.Table{Union{AbstractVector{ScientificTypesBase.Continuous}, AbstractVector{Scient:
    2:  Source @725 ⏎ AbstractVector{ScientificTypesBase.OrderedFactor{2}}
    3:  Source @207 ⏎ ScientificTypesBase.Count
```

```
trained Machine; caches model-specific representations of data
  model: XGBoostClassifier(test = 1, …)
  args:
    1:  Source @826 ⏎ ScientificTypesBase.Table{Union{AbstractVector{ScientificTypesBase.Continuous}, AbstractVector{Scient:
    2:  Source @725 ⏎ AbstractVector{ScientificTypesBase.OrderedFactor{2}}
    3:  Source @207 ⏎ ScientificTypesBase.Count
```

Feature Importances

```
 Pair{Symbol, Float32}[
    1:       :payment_type => 519.971
    2:       :trip_distance => 1.60723
    3:       :trip_duration_minute => 1.54039
    4:       :day_of_week => 1.51641
    5:       :hour_of_day => 1.47199
    6:       :RatecodeID => 1.2518
]
```

The most important predictive feature for determining a generous tipper is whether the customer is paying with cash or credit card.

```
PerformanceEvaluation object with these fields:
  model, measure, operation, measurement, per_fold,
  per_observation, fitted_params_per_fold,
  report_per_fold, train_test_rows, resampling, repeats
Extract:
```

measure	operation	measurement	1.96*SE	per_fold …
LogLoss(tol = 2.220446049250313e-16)	predict	0.455	0.00152	[0.456, 0.4 … …
MisclassificationRate()	predict_mode	0.228	N/A	[0.227, 0.2 …
BrierLoss()	predict	0.313	0.000988	Float32[0.3 …
Accuracy()	predict_mode	0.772	N/A	[0.773, 0.7 …
FScore(β = 1.0, rev = nothing)	predict_mode	0.742	N/A	[0.73, 0.73 … … …
TruePositiveRate(rev = nothing)	predict_mode	0.635	N/A	[0.623, 0.6 … …

1 column omitted

Selection deleted

```
xgb_ŷ =
 ["G", "G", "G", "G", "G", "G", "G", "G", "G", "G", "G", "G", "G", "G", "G", "NG", "NG", "G", "G", "G",    more ,"NG", "NG", "N
```

```
1 xgb_ŷ = [ i.prob_given_ref[1] > i.prob_given_ref[2] ? "G" : "NG" for i in MLJ.predict(xgb_mach, Xtest) ]
```

	Ground Truth	
Predicted	G	NG
G	1662	729
NG	131	1174

```
MLJ.confusion_matrix(xgb_ŷ, ytest)
```

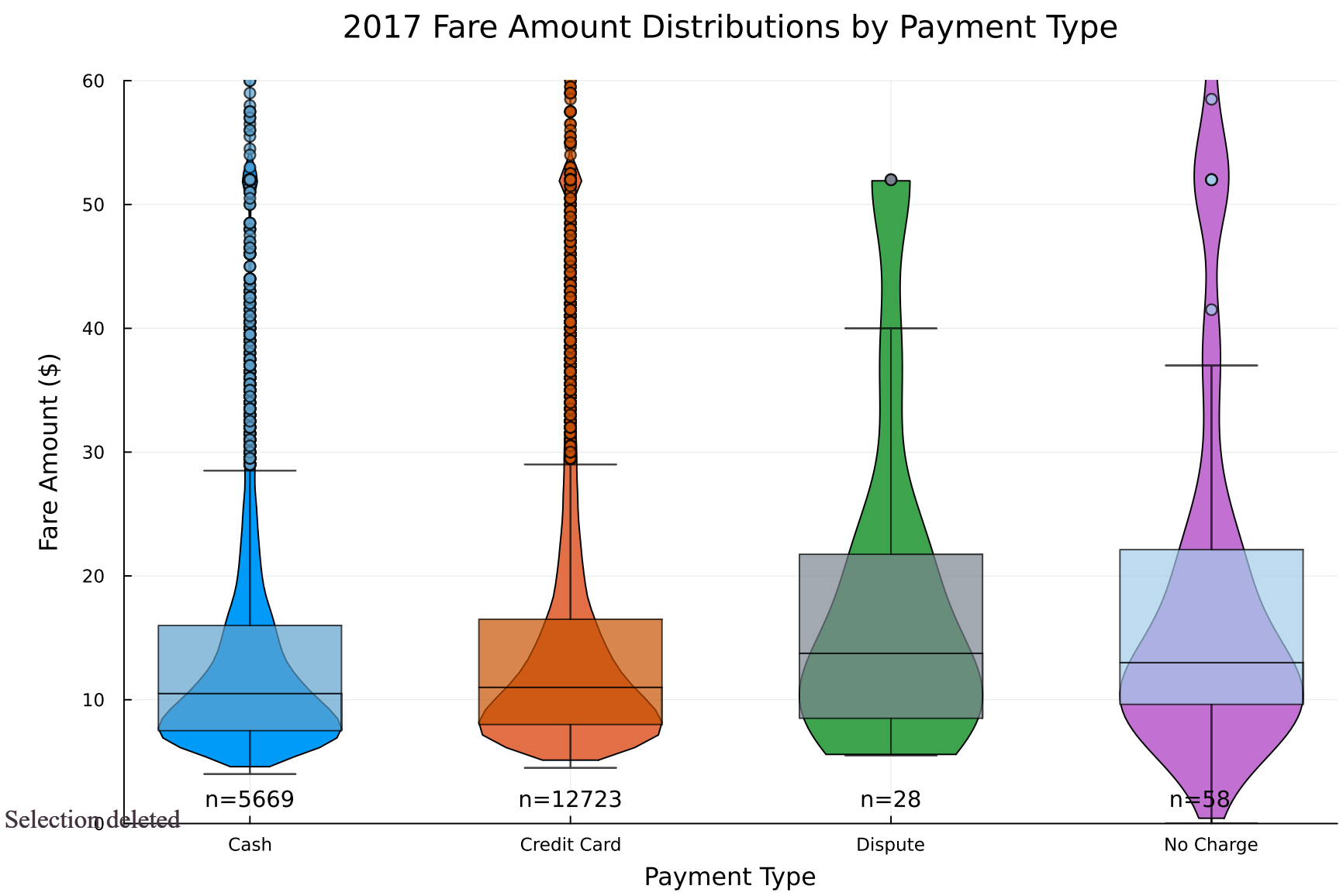
Conclusion

Predicting whether a customer will be a generous tipper (greater than 20% of the billed amount) largely depends on whether the customer is paying in cash or credit. The model is much more accurate at predicting non-generous tippers than generous tippers. Depending on the priorities of NYC TLC, the model may want to be retrained to prefer more accurately predicting generous tippers.

Appendix A

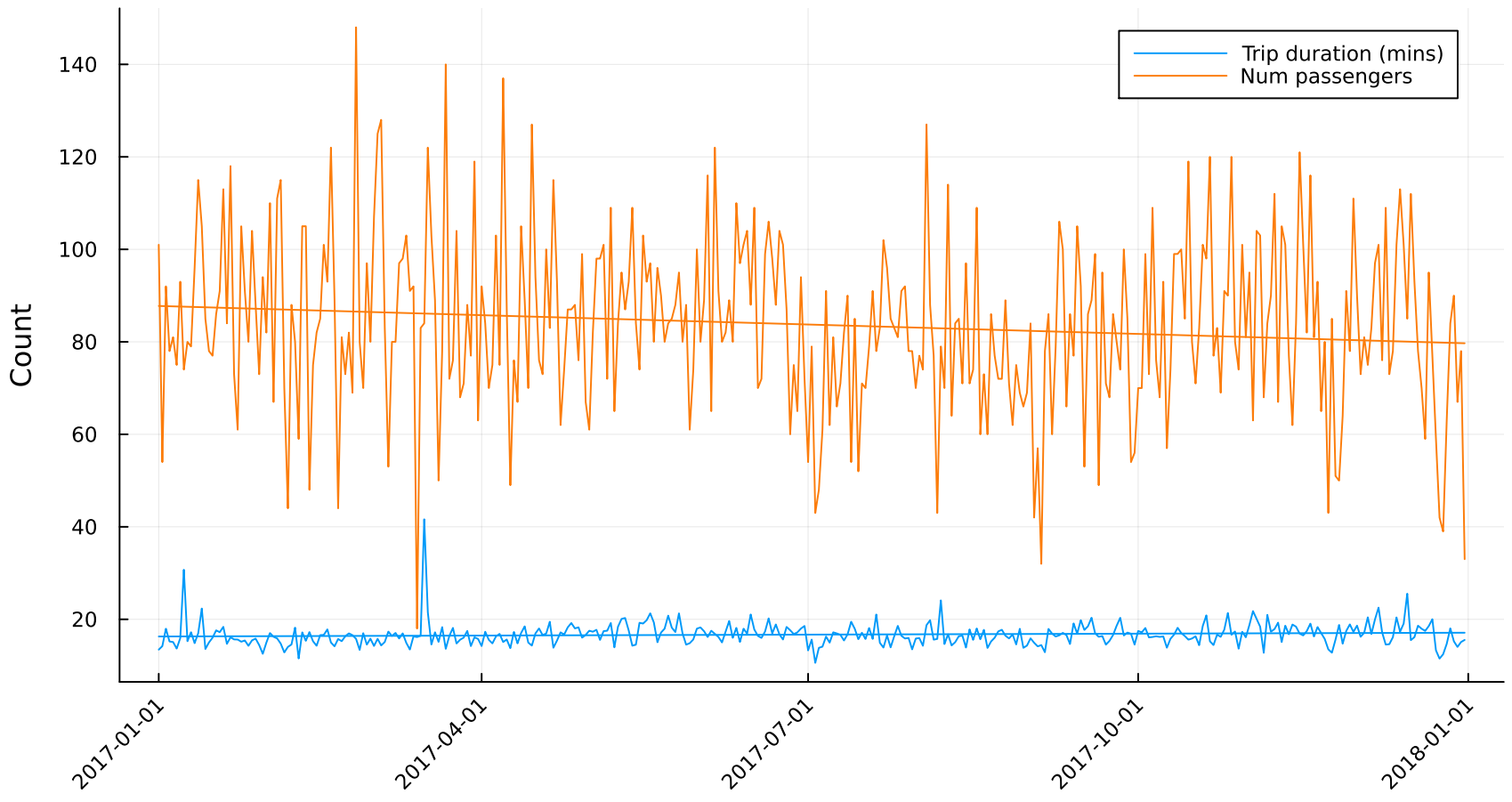
EDA Visualizations

Graph 1:

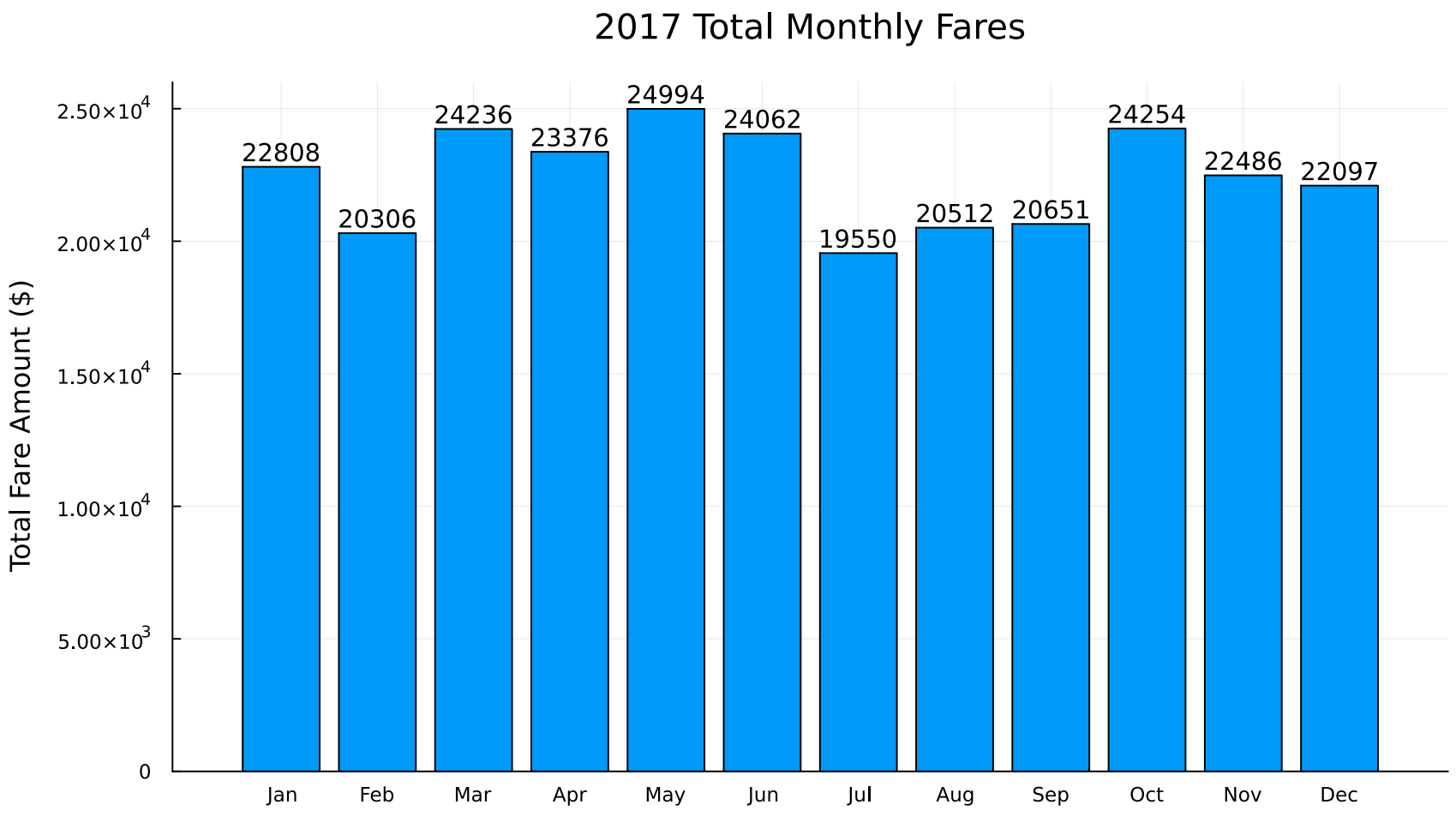


Selection deleted

Graph 2:

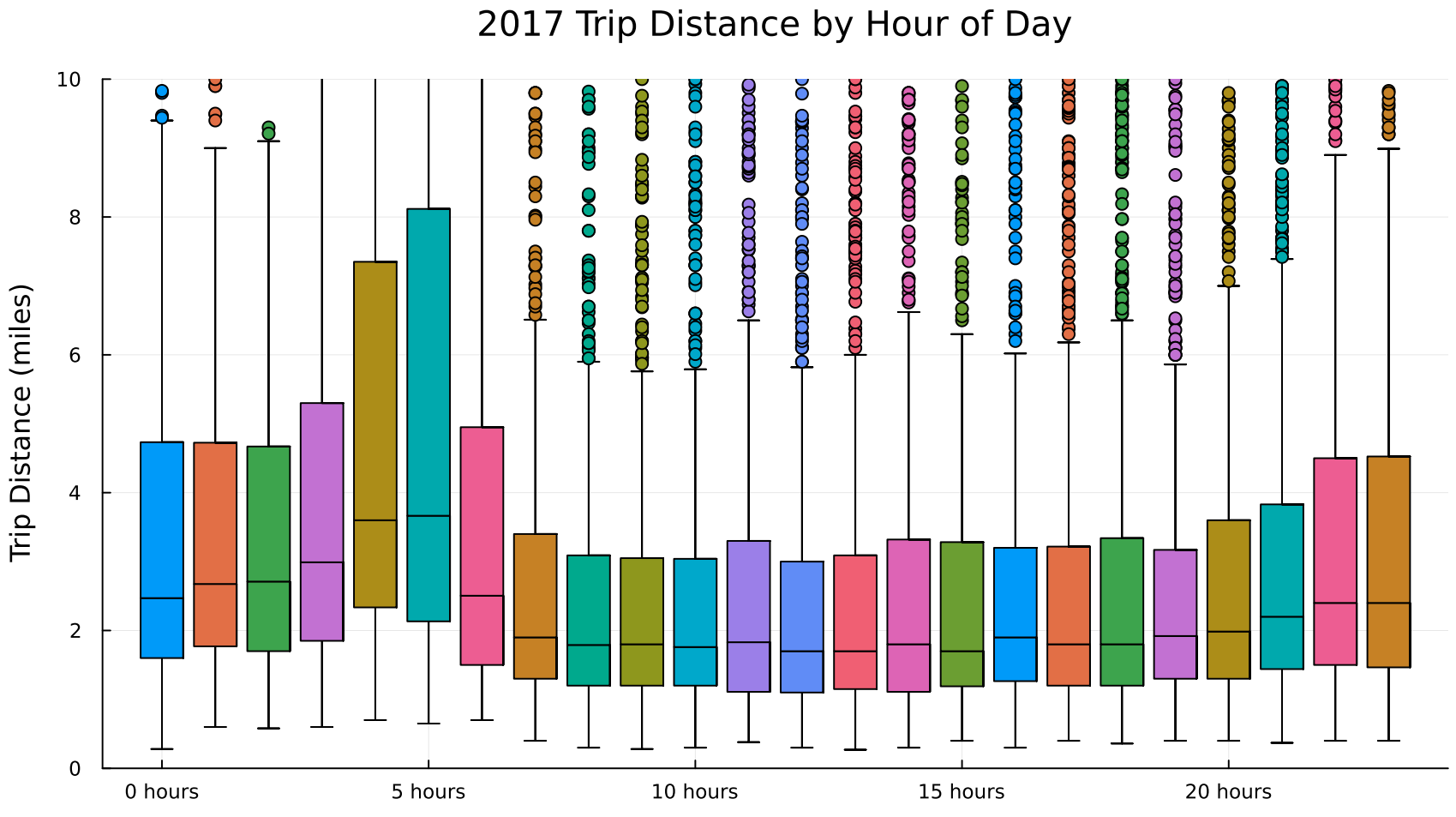


Graph 3:



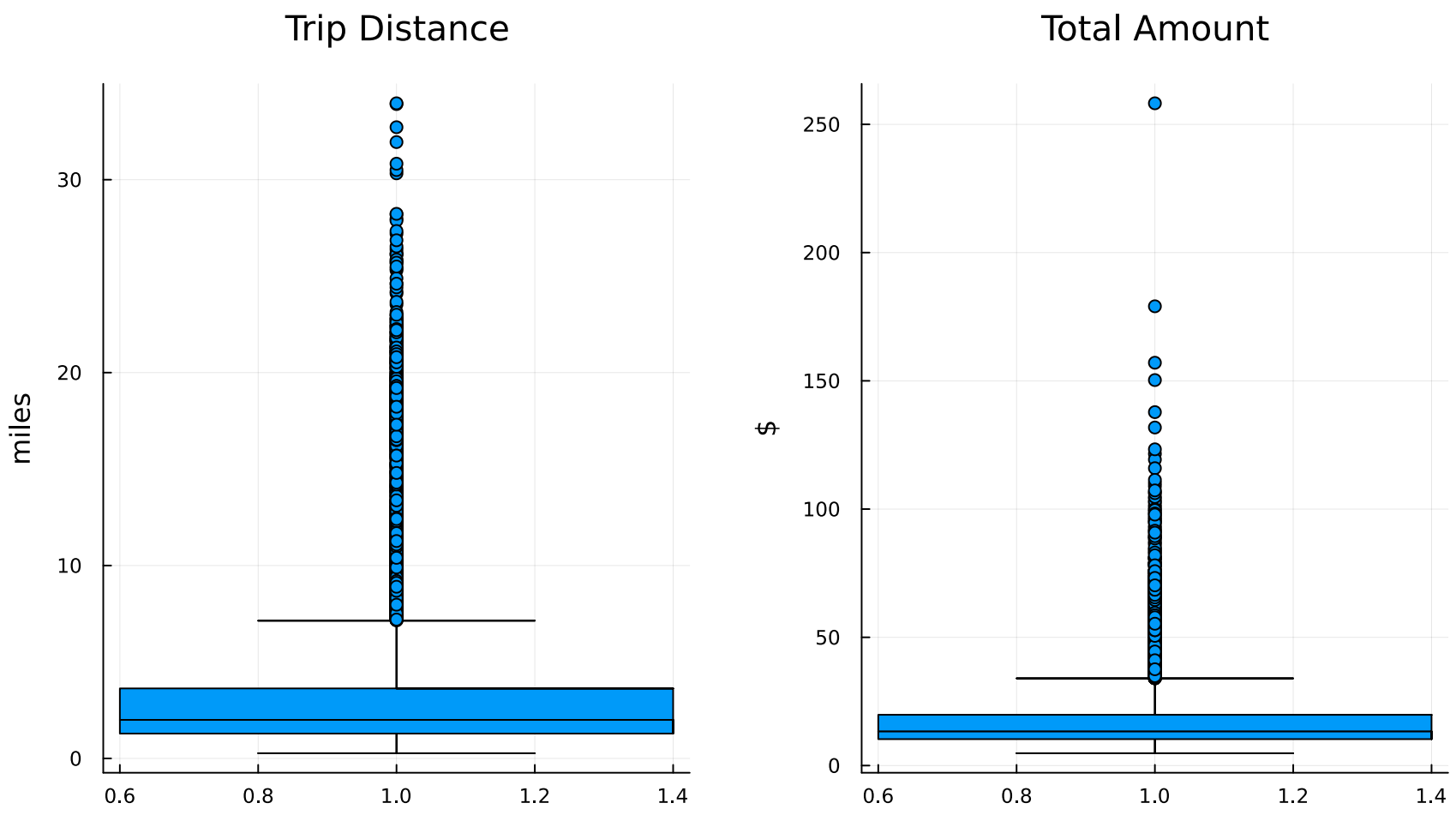
Graph 4:

Selection deleted

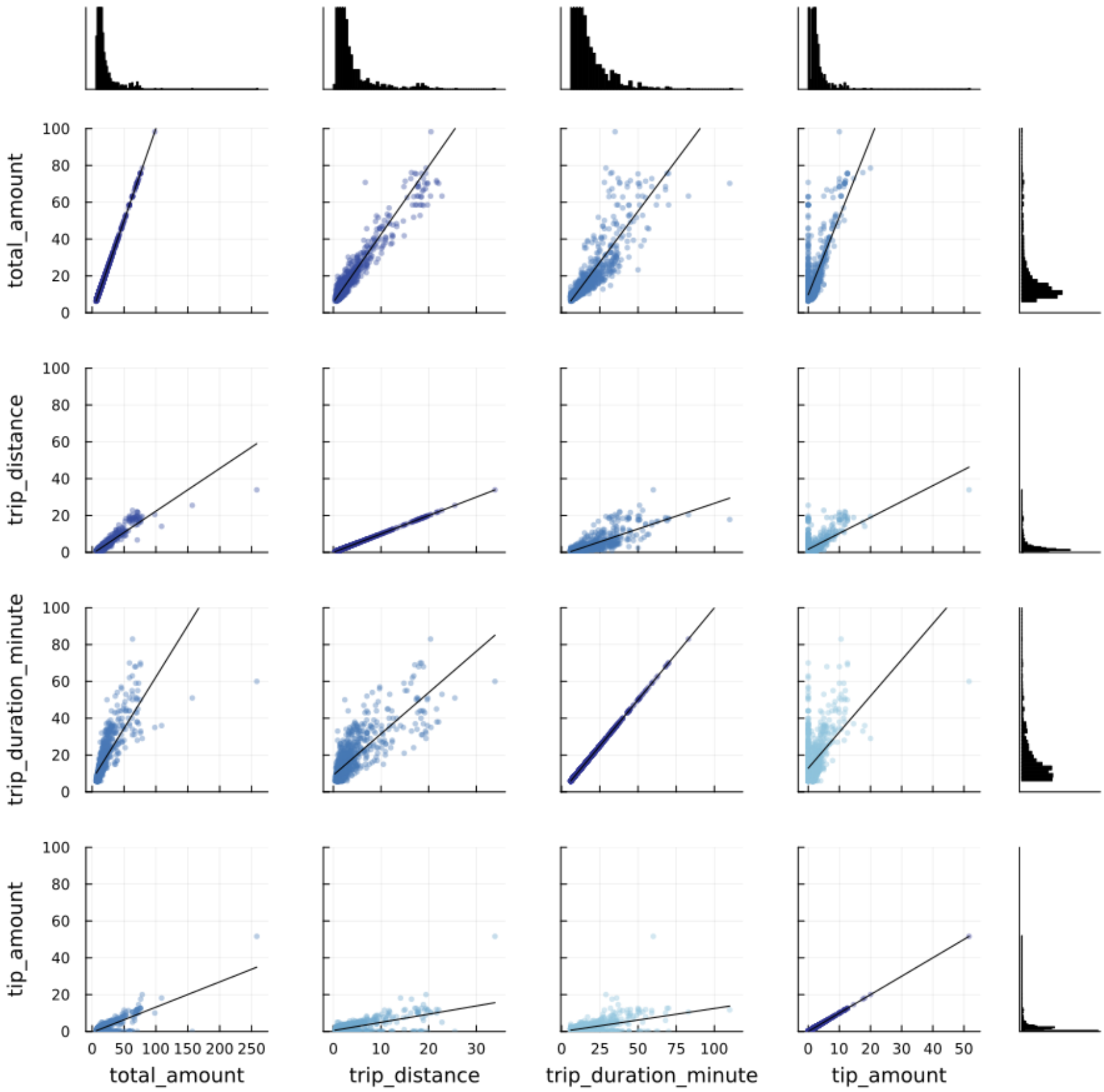


Longer trips are expected in the early morning hours (2-5 miles), with shorter trips (1-3 miles) during business hours.

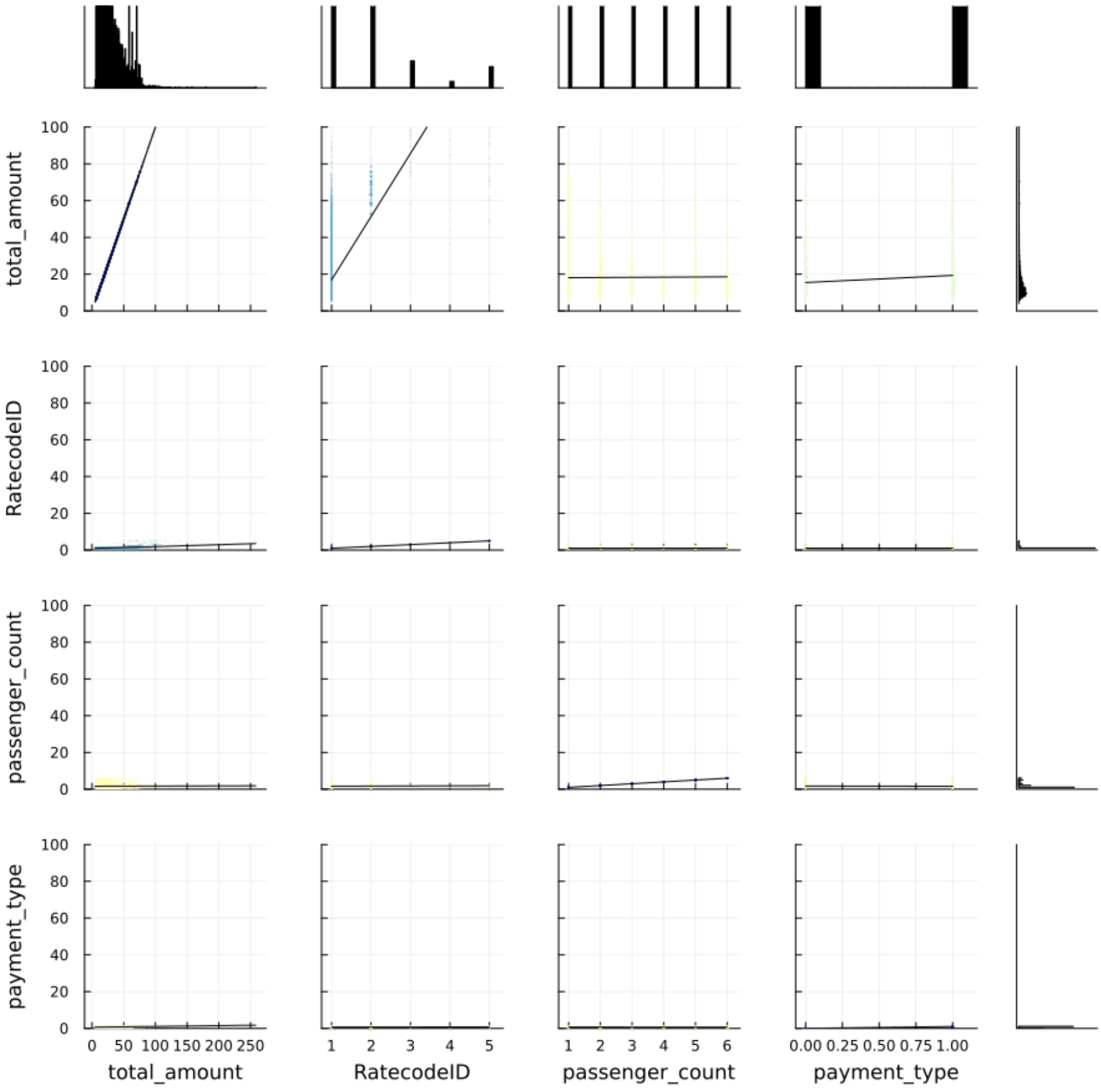
Graph 5:



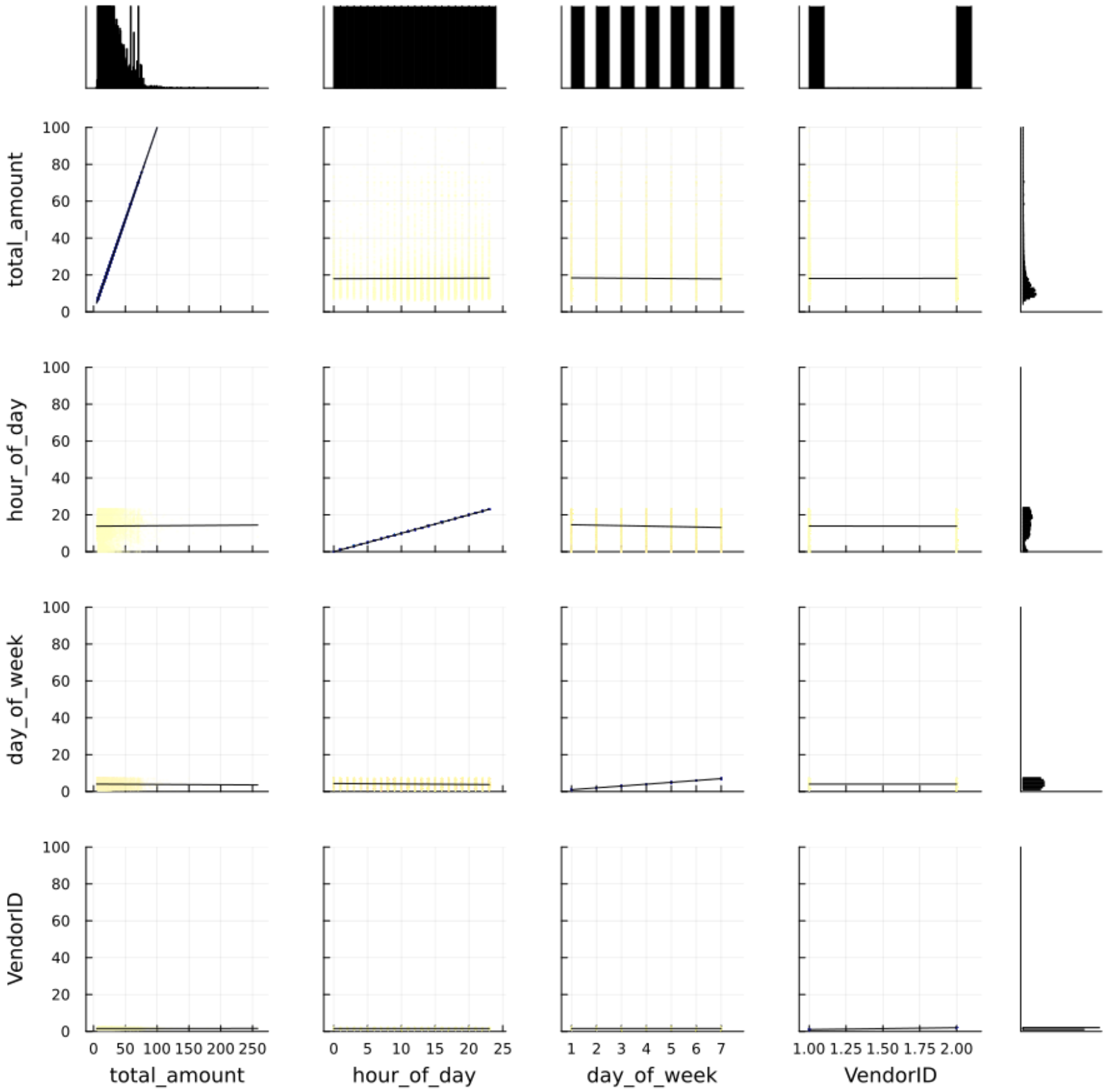
Graph 6:


total_amount
trip_distance
trip_duration_minute
tip_amount
0
20
40
60
80
100
0 50 100 150 200 250
0 10 20 30
0 25 50 75 100
0 10 20 30 40 50
total_amount
trip_distance
trip_duration_minute
tip_amount

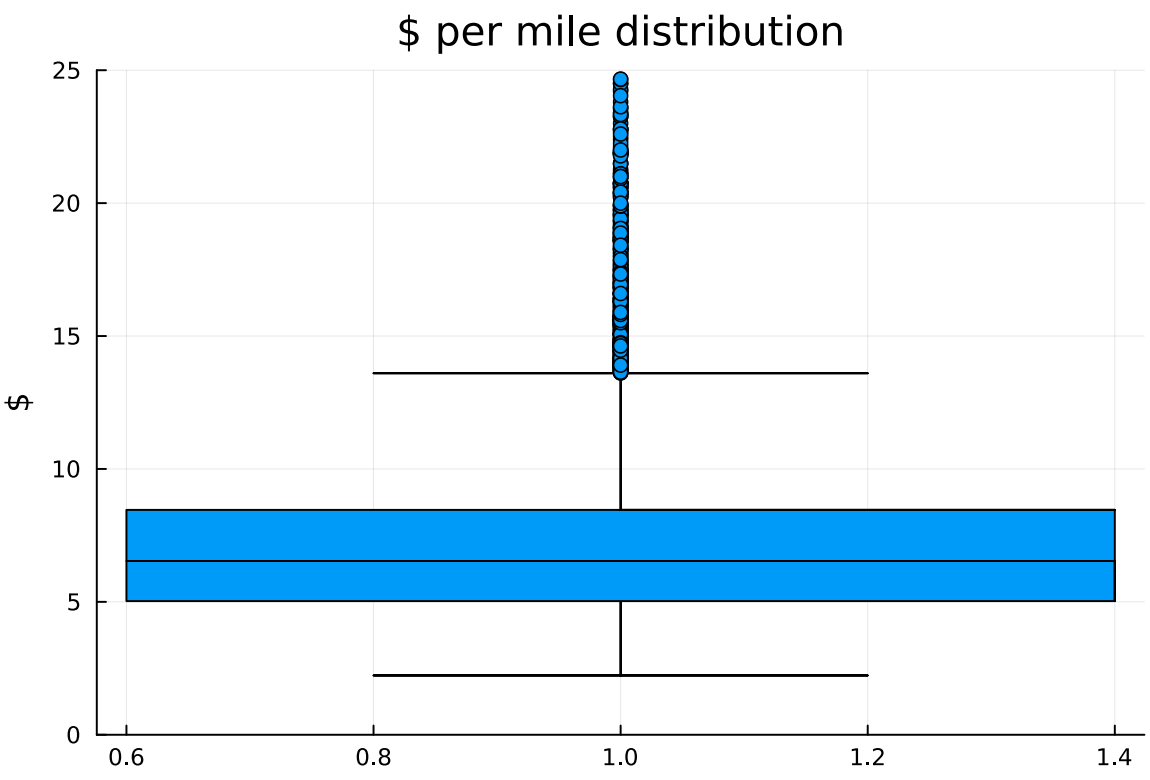
Graph 7:


total_amount
RatecodeID
passenger_count
payment_type
0
20
40
60
80
100
0 50 100 150 200 250
1 2 3 4 5
1 2 3 4 5 6
0.00 0.25 0.50 0.75 1.00
total_amount
RatecodeID
passenger_count
payment_type

Graph 8:


total_amount
hour_of_day
day_of_week
VendorID
0
20
40
60
80
100
0 50 100 150 200 250
0 5 10 15 20 25
1 2 3 4 5 6 7
1.00 1.25 1.50 1.75 2.00

Graph 9:


$ per mile distribution
$
0
5
10
15
20
25
0.6
0.8
1.0
1.2
1.4

raw_data =

	Column1	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store
1	24870114	2	2017-03-25T08:55:43	2017-03-25T09:09:47	6	3.34	1	"N"
2	35634249	1	2017-04-11T14:53:28	2017-04-11T15:19:58	1	1.8	1	"N"
3	106203690	1	2017-12-15T07:26:56	2017-12-15T07:34:08	1	1.0	1	"N"
4	38942136	2	2017-05-07T13:17:59	2017-05-07T13:48:14	1	3.7	1	"N"
5	30841670	2	2017-04-15T23:32:20	2017-04-15T23:49:03	1	4.37	1	"N"
6	23345809	2	2017-03-25T20:34:11	2017-03-25T20:42:11	6	2.3	1	"N"
7	37660487	2	2017-05-03T19:04:09	2017-05-03T20:03:47	1	12.83	1	"N"
8	69059411	2	2017-08-15T17:41:06	2017-08-15T18:03:05	1	2.98	1	"N"
9	8433159	2	2017-02-04T16:17:07	2017-02-04T16:29:14	1	1.2	1	"N"
10	95294817	1	2017-11-10T15:20:29	2017-11-10T15:40:55	1	1.6	1	"N"
more								
22699	17208911	1	2017-03-02T13:02:49	2017-03-02T13:16:09	1	2.1	1	"N"

	variable	mean	std	min	max	median	nunique
1	:Column1	5.67585e7	3.27449e7	12127	113486300	5.67315e7	nothing
2	:VendorID	1.55624	0.496838	1	2	2.0	nothing
3	:tpep_pickup_datetime	nothing	nothing	2017-01-01T00:08:25	2017-12-31T23:45:30	2017-06-23T12:35:57	22687
4	:tpep_dropoff_datetime	nothing	nothing	2017-01-01T00:17:20	2017-12-31T23:49:24	2017-06-23T12:55:11	22688
5	:passenger_count	1.64232	1.28523	0	6	1.0	nothing
6	:trip_distance	2.91331	3.65317	0.0	33.96	1.61	nothing
7	:RatecodeID	1.04339	0.708391	1	99	1.0	nothing
8	:store_and_fwd_flag	nothing	nothing	"N"	"Y"	nothing	2
9	:PULocationID	162.412	66.6334	1	265	162.0	nothing
10	:DOLocationID	161.528	70.1397	1	265	162.0	nothing
more							
18	:total_amount	16.3105	16.0973	-120.3	1200.29	11.8	nothing

```
describe(raw_data, :mean, :std, :min, :max, :median, :nunique, :nmissing)
```

variable_countmap =

```
Dict("tolls_amount" => Dict(5.44 => 1, 15.58 => 1, 16.0 => 2, 2.64 => 10, 5.49 => 1, 5.54 => 234, 8.0 => 1, 5.16 => 1, 2.7
```

Data cleaning record

1. No duplicate records found.
2. No records with missing variables found.
3. Removed fares over $400 and less than $0.
4. Renamed Column1 to ID
5. Removed records with a trip_duration_minute value less than 5 and over 800

Data restructuring Record

1. Converted `store_and_fwd_flag` to boolean.
2. Created `trip_duration` variable from pickup and drop-off time (in minutes).

cleaned_df =

	ID	VendorID	tpep_pickup_datetime	tpep_dropoff_datetime	passenger_count	trip_distance	RatecodeID	store
1	111133227	2	2017-01-01T00:08:25	2017-01-01T00:17:20	1	0.52	1	"N"
2	111139506	1	2017-01-01T00:26:35	2017-01-01T00:36:12	1	0.9	1	"N"
3	111145951	1	2017-01-01T00:43:10	2017-01-01T00:57:21	4	4.9	1	"N"
4	111055119	1	2017-01-01T01:21:59	2017-01-01T01:34:12	3	2.1	1	"N"
5	111059310	1	2017-01-01T01:32:44	2017-01-01T01:47:41	2	5.4	1	"N"
6	111059488	1	2017-01-01T01:33:12	2017-01-01T01:44:09	1	2.2	1	"N"
7	111060763	2	2017-01-01T01:36:30	2017-01-01T01:42:17	1	0.88	1	"N"
8	111064442	2	2017-01-01T01:46:05	2017-01-01T01:57:33	5	2.12	1	"N"
9	111069315	2	2017-01-01T01:59:22	2017-01-01T02:12:59	5	5.57	1	"N"
10	111075273	2	2017-01-01T02:15:23	2017-01-01T02:22:51	6	0.75	1	"N"
more								
18478	110935471	1	2017-12-31T22:28:54	2017-12-31T22:37:29	1	1.5	1	"N"

variable_desciptions =

	Column name	Description
1	"ID"	"Trip identification number"
2	"VendorID"	"A code indicating the TPEP provider t
3	"tpep_pickup_datetime"	"The date and time when the meter was
4	"tpep_dropoff_datetime"	"The date and time when the meter was
5	"Passenger_count"	"The number of passengers in the vehic
6	"Trip_distance"	"The elapsed trip distance in miles re
7	"PULocationID"	"TLC Taxi Zone in which the taximeter
8	"DOLocationID"	"TLC Taxi Zone in which the taximeter
9	"RateCodeID"	"The final rate code in effect at the
10	"Store_and_fwd_flag"	"This flag indicates whether the trip
more		

pmnt_types = Dict(5 ⇒ "Unknown", 4 ⇒ "Dispute", 6 ⇒ "Voided Trip", 2 ⇒ "Cash", 3 ⇒ "No Charge", 1 ⇒ "Credit Card")

Appendix B

space =

Libraries

```
# libraries
begin
    using CSV, DataFramesMeta, Dates
    using GLM, StatsBase, Metrics, HypothesisTests
    using MLJ, DecisionTree, MLJDecisionTreeInterface, XGBoost, MLJXGBoostInterface
    using StatsPlots, ColorSchemes, Measures
    using PlutoUI
end
```

```
WARNING: using GLM.r2 in module workspace#3 conflicts with an existing identifier.
WARNING: using StatsBase.r2 in module workspace#3 conflicts with an existing identifier.
```

Table of Contents


```
PlutoUI.TableOfContents()
```

```
# page/cell width control
"""
<style>
    main {
    max-width: 1100px;
    align-left: flex-start;
    margin-left: 50px;
    }
""" |> HTML
```