Predicting Revenue from Search Engine Advertising Data

MATH2319 - Machine Learning
Course Project

Ben Cole - s3412349

Print Date: 31/05/2019

Contents

1 Phase 1 - Introduction, Cleaning, and Exploration

1.1 Outline

The prescribed data set contained advertising metrics provided by a prominent search engine. The data contained several descriptive features pertaining to a range of information. Finally, the target feature was a measure of revenue associated with each of the observations.
The dataset was used to create a supervised machine learning model to predict values for the target feature. Phase 1 of this report contains the introduction, cleaning, and exploration of the dataset. Phase 2 contains the creation, training, and deployment of the machine learning algorithm.

1.1.1 Nature of the Data

The below is an exerpt from accompanying documentation about the dataset.

Features in this data set are as follows:

- companyId: Company ID of record (categorical)
- countryId: Country ID of record (categorical)
- deviceType: Device type of record (categorical corresponding to desktop, mobile, tablet)
- day: Day of record (integer between 1 (oldest) and 30 for train, 31 and 35 (most recent) for test)
- dow: Day of week of the record (categorical)
- price1, price2, price3: Price combination for the record set by the company (numeric)
- ad_area: area of advertisement (numeric)
- ad_ratio: ratio of advertisement's length to its width (numeric)
- requests, impression, cpc, ctr, viewability: Various metrics related to the record (numeric)
- ratio1, …, ratio5: Ratio characteristics related to the record (numeric)
- y (target feature): revenue-related metric (numeric)

1.1.1.1 Target Feature

The column/variable **y** was selected as the target feature in the dataset.

1.1.1.2 Descriptive Features

All other columns/variables in the dataset, as outlined above, were chosen as descriptive features.

1.2 Data Processing

1.2.1 Libraries

The following libraries were used in the below data processing and exploration.

```
library(pacman)                             ## for loading multiple packages

suppressMessages(p_load(character.only = T,
                        install = F,
                        c("tidyverse",   ## thanks Hadley
                          "lubridate",   ## for handling dates
                          "forcats",     ## for categorial variables, not for felines
                          "zoo",         ## some data cleaning capabilities
                          "lemon",       ## add ons for ggplot
                          "rvest",       ## scraping web pages
                          "knitr",       ## knitting to RMarkdown
                          "kableExtra",  ## add ons for knitr tables
                          "scales",      ## quick and easy formatting prettynums
                          "grid",        ## for stacking ggplots
                          "gridExtra",   ## also for stacking ggplots
                          "e1071",       ## for skew and kurtosis
                          "janitor",     ## cleaning colnames
                          "beepr")))     ## plays a beep tone
```

Table 1: Sample of Advertising Data Frame

case_id	companyId	countryId	deviceType	day	dow	price1	price2	price3	ad_area	ad_ratio
20862	40	226	2	4	Tuesday	0.00	0.00	0.0000	0.0001	1.00000
49367	43	231	3	8	Saturday	0.00	0.00	0.0000	0.0001	1.00000
146248	43	56	3	21	Friday	0.00	0.00	0.0000	0.0001	1.00000
204523	159	200	2	29	Saturday	0.17	0.42	0.8503	24.2500	0.25773
37251	40	2	1	6	Thursday	0.00	0.00	0.0000	0.0001	1.00000
170232	43	56	1	24	Monday	1.36	1.36	1.3614	24.2500	0.25773
71818	95	38	2	11	Tuesday	0.02	0.52	1.0400	7.5000	0.83333
2360	43	198	3	1	Saturday	0.00	0.00	0.0000	0.0001	1.00000
97620	43	105	1	15	Saturday	0.00	0.00	0.0000	9.4080	0.83333
18866	43	57	1	4	Tuesday	0.01	0.42	0.8320	0.0001	1.00000
37056	159	57	3	6	Thursday	0.14	0.52	1.0427	0.0001	1.00000
89659	43	172	3	14	Friday	0.00	0.00	0.0000	0.0001	1.00000
156889	159	38	3	23	Sunday	0.10	0.37	0.7601	0.0001	1.00000
48107	95	234	2	8	Saturday	0.05	0.05	0.0500	18.0000	2.00000
32092	43	202	1	6	Thursday	0.00	0.00	0.0000	7.0920	0.11421
147928	159	102	2	21	Friday	0.00	0.00	0.0000	0.0001	1.00000
100808	95	105	1	15	Saturday	0.05	0.05	0.0500	7.5000	0.83333
151748	95	77	1	22	Saturday	0.05	0.25	0.2500	1.6000	0.15625
108671	43	12	2	16	Sunday	8.55	8.55	8.5455	0.0001	1.00000
150281	43	153	2	22	Saturday	5.96	5.96	5.9569	0.0001	1.00000

1.2.2 Loading Data

The prescribed data was made available in comma separated value file format.

```
advertising_train <- read_csv("advertising_train.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   dow = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

```
sample_adv <- sample_n(advertising_train, 20)

kable_styling(kable(sample_adv[ , 1:(ncol(sample_adv)/2)],
                    caption = "Sample of Advertising Data Frame"),
              font_size = 8.5, latex_options = c("striped"),
              full_width = F)
```

```
kable_styling(kable(sample_adv[ , c(1, ((ncol(sample_adv)/2)+1):ncol(sample_adv))],
                    caption = "Sample of Advertising Data Frame (cont)"),
              font_size = 8.5, latex_options = c("striped"),
              full_width = F)
```

1.2.3 Classifying Data

R and `dplyr` parse data files to guessed data types when loaded. Typically, columns with text are parsed as `character` type, columns with digits are parsed as `numeric`, and boolean columns are parsed as `logical`. Per the above feature definitions, the categorical data was re-classified as `factors`.

```
advertising_train$companyId <- as.factor(advertising_train$companyId)

advertising_train$countryId <- as.factor(advertising_train$countryId)
```

Table 2: Sample of Advertising Data Frame (cont)

case_id	requests	impression	cpc	ctr	viewability	ratio1	ratio2	ratio3	ratio4	ratio5	y
20862	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1145126
49367	143	142	0.0709	0.0070	0.4050	1.0000	0.8310	0.0000	0.2394	0.7606	0.1778947
146248	98	98	0.0128	0.3776	0.9014	1.0000	0.9592	0.0000	0.0000	1.0000	5.7234043
204523	1276	1117	0.3062	0.0018	0.8668	0.6177	0.9615	1.0000	0.0000	0.0000	0.4872699
37251	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3665974
170232	4535	1154	0.1975	0.0087	0.8526	0.9853	0.7686	0.1092	0.1170	0.7721	0.3167115
71818	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2971585
2360	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	2.0717172
97620	485	301	0.0534	0.0033	0.5636	1.0000	0.4551	0.4319	0.0000	0.5681	0.0441844
18866	176	151	0.0292	0.0132	0.8095	0.3113	0.8344	0.0464	0.6887	0.2649	0.4083871
37056	3466	1695	0.2026	0.0041	0.7067	0.7186	0.7475	0.0313	0.3923	0.5770	0.5136212
89659	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.4000000
156889	934	931	0.2486	0.0054	0.8058	0.9882	0.9452	0.0043	0.8625	0.1332	1.0596859
48107	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.9110368
32092	990	984	0.1155	0.0030	0.7813	1.0000	0.9675	0.1159	0.0000	0.8841	0.3008351
147928	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1267606
100808	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0391304
151748	1168	859	0.6423	0.0012	0.6776	0.9127	0.5681	0.0512	0.3912	0.5576	0.4951348
108671	27	5	0.0257	0.2000	0.6667	0.8000	0.8000	1.0000	0.0000	0.0000	1.3761905
150281	785	35	0.0105	0.7143	0.8182	0.9429	1.0000	1.0000	0.0000	0.0000	0.2478475

```r
advertising_train$deviceType <- as.factor(advertising_train$deviceType)

advertising_train$dow <- as.factor(advertising_train$dow)

sapply(advertising_train, class)
```

```
##      case_id    companyId    countryId   deviceType          day          dow
##    "numeric"     "factor"     "factor"     "factor"    "numeric"     "factor"
##       price1       price2       price3      ad_area     ad_ratio     requests
##    "numeric"    "numeric"    "numeric"    "numeric"    "numeric"    "numeric"
##   impression          cpc          ctr  viewability       ratio1       ratio2
##    "numeric"    "numeric"    "numeric"    "numeric"    "numeric"    "numeric"
##       ratio3       ratio4       ratio5            y
##    "numeric"    "numeric"    "numeric"    "numeric"
```

1.2.4 Descriptive Statistics

1.2.4.1 Numeric Features

The below table outlines basic descriptive statistics about the centre and spread of the data for each of the numeric descriptive features, and numeric target feature. This table indicates that the numeric features each had distributions of different shapes and locations.

```r
advertising_train_long_num <- select(advertising_train,
                                     colnames(advertising_train),
                                     -case_id, -countryId,
                                     -companyId, -deviceType,
                                     -dow)

advertising_train_long_num <- gather(advertising_train_long_num,
                                     key = "Variable",
                                     value = "Value")

summary_adv_num <- summarise(group_by(advertising_train_long_num,
```

Table 3: Summary Statistics of Numeric Variables

Variable	Mean	Std Dev	Min	Q1	Median	Q3	Max	Number of NA
ad_area	4.724	6.273	0.000	0.000	0.000	7.500	36.000	0.000
ad_ratio	0.923	0.482	0.083	0.833	1.000	1.000	5.000	0.000
cpc	0.178	0.707	0.000	0.000	0.016	0.125	132.534	0.000
ctr	0.033	0.093	0.000	0.000	0.002	0.012	2.000	0.000
day	15.791	8.386	1.000	9.000	16.000	23.000	30.000	0.000
impression	5,585.714	98,713.340	0.000	0.000	99.000	1,058.000	6,100,324.000	0.000
price1	0.438	1.281	0.000	0.000	0.010	0.190	14.690	0.000
price2	0.630	1.482	0.000	0.000	0.090	0.570	63.120	0.000
price3	0.932	1.840	0.000	0.000	0.295	0.986	78.900	0.000
ratio1	0.558	0.447	0.000	0.000	0.750	1.000	1.000	0.000
ratio2	0.491	0.414	0.000	0.000	0.627	0.896	1.027	0.000
ratio3	0.312	0.444	0.000	0.000	0.028	1.000	1.500	0.000
ratio4	0.131	0.240	0.000	0.000	0.000	0.164	1.077	0.000
ratio5	0.188	0.297	0.000	0.000	0.000	0.385	1.200	0.000
requests	8,678.997	122,347.229	0.000	0.000	147.000	1,633.000	6,701,924.000	0.000
viewability	0.378	0.366	0.000	0.000	0.332	0.716	7.000	0.000
y	0.847	1.391	0.000	0.150	0.419	0.959	47.060	0.000

```
                                              Variable),
                                     "Mean" = mean(Value, na.rm = T),
                                     "Std Dev" = sd(Value, na.rm = T),
                                     "Min" = min(Value, na.rm = T),
                                     "Q1" = quantile(Value, 0.25, na.rm = T),
                                     "Median" = median(Value, na.rm = T),
                                     "Q3" = quantile(Value, 0.75, na.rm = T),
                                     "Max" = max(Value, na.rm = T),
                                     "Number of NA" = sum(is.na(Value)))

kable_styling(kable(summary_adv_num,
                    digits = 3, format.args = list(nsmall = 3,
                                                   scientific = F,
                                                   big.mark = ","),
                    caption = "Summary Statistics of Numeric Variables"),
              font_size = 8.5, latex_options = c("striped"),
              full_width = F)
```

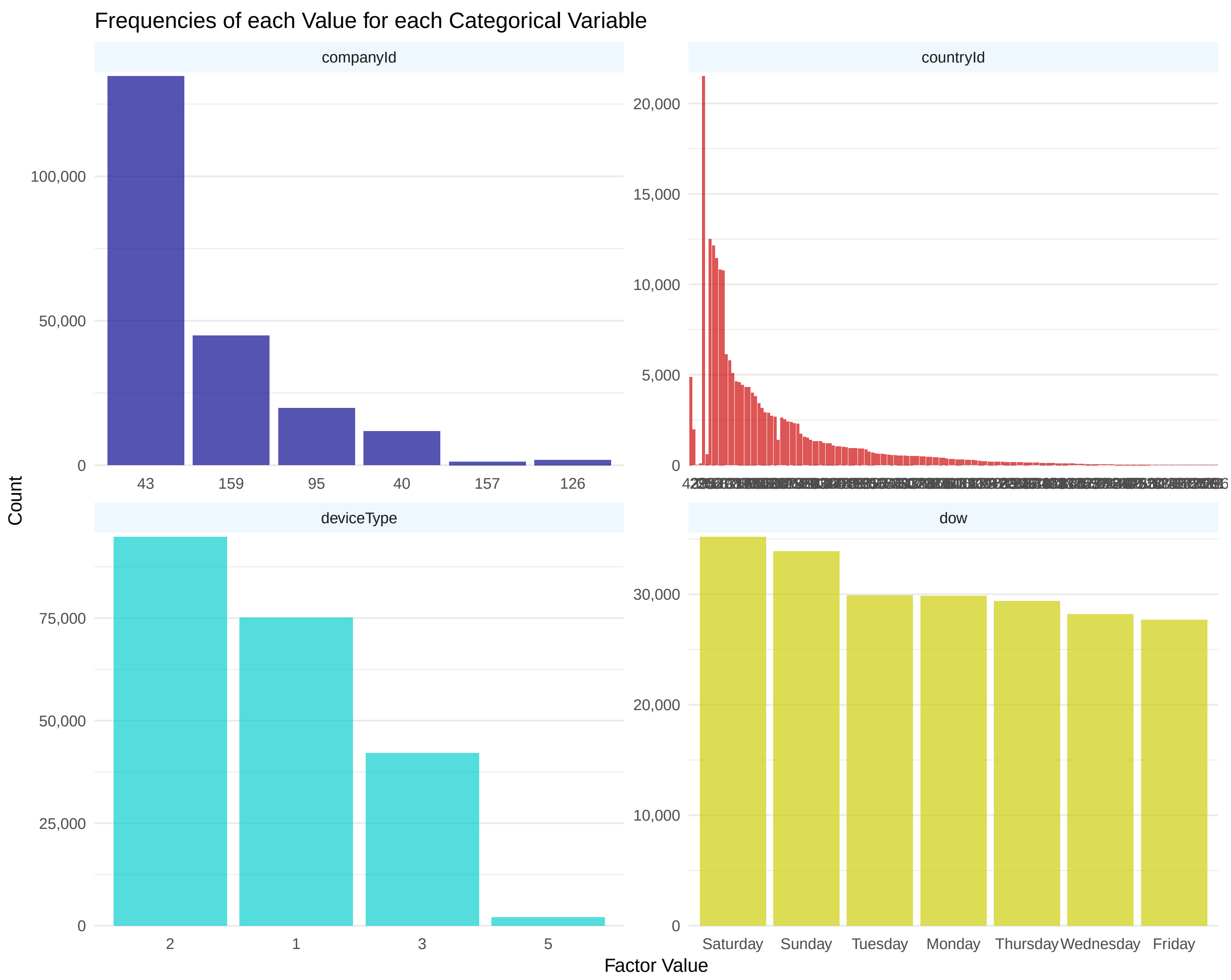
1.2.4.2 Categorical and Non-Numeric Features

When examining the frequencies of individual levels of each Categorical (non-numeric) descriptive feature, variability was observed in `companyId`, `countryId`, and `deviceType`. Far less variability in frequencies was observed in `dow`, with Sunday being the only day of the week to return a markedly lower frequency.

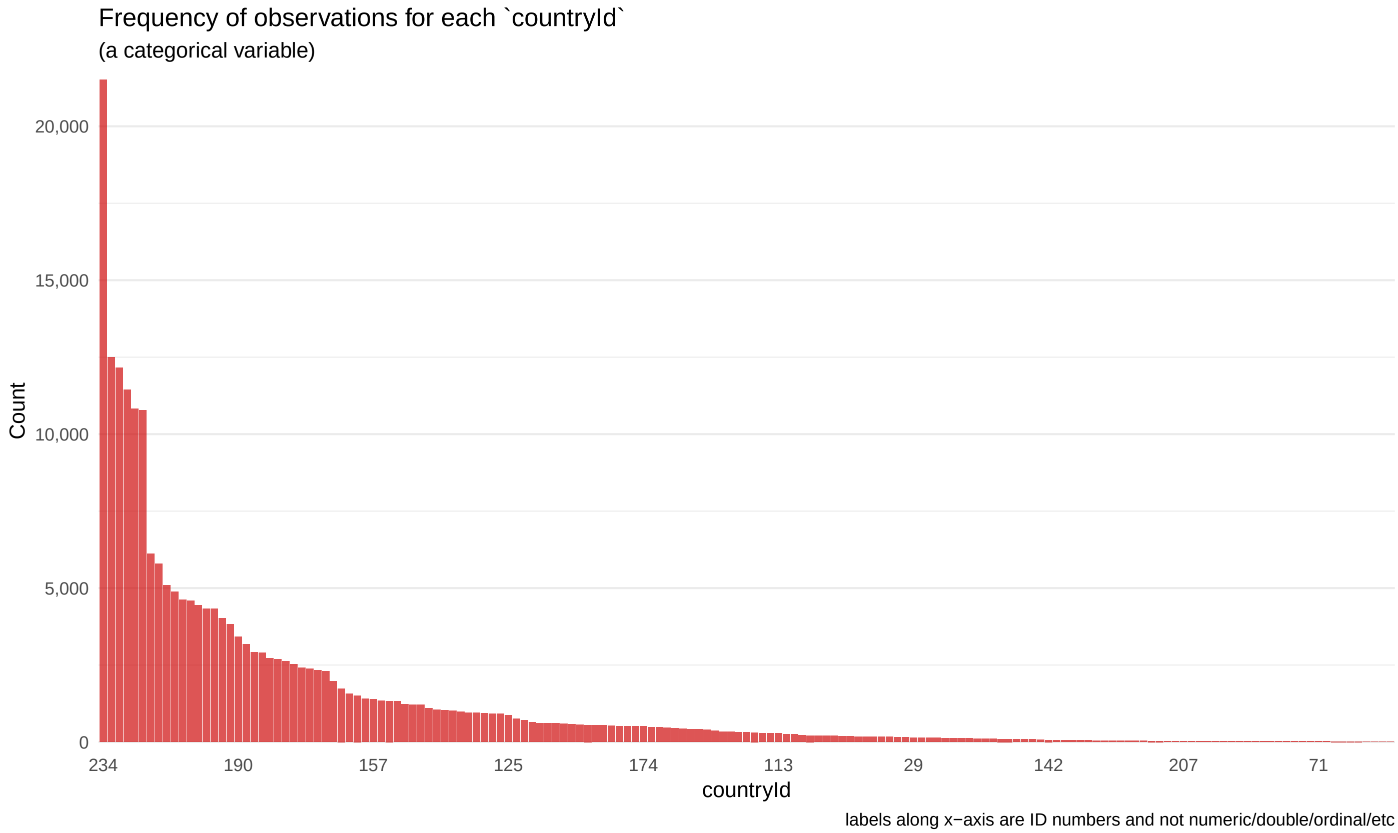
```
advertising_train_long_cat <- select(advertising_train,
                                     countryId,
                                     companyId, deviceType,
                                     dow)

advertising_train_long_cat <- gather(advertising_train_long_cat,
                                     key = "Variable",
                                     value = "Value")
```

```
## Warning: attributes are not identical across measure variables;
## they will be dropped
```

```r
advertising_train_long_cat$Variable <- as.factor(advertising_train_long_cat$Variable)

advertising_train_long_cat$Value <- as.factor(advertising_train_long_cat$Value)

ggplot(advertising_train_long_cat) +
   geom_bar(aes(x = fct_infreq(Value),
                fill = Variable),
            show.legend = F, alpha = 2/3) +
   facet_rep_wrap(~Variable,
                  repeat.tick.labels = T,
                  scales = "free") +
   scale_y_continuous(labels = comma,
                      expand = c(0.01, 0),
                      "Count") +
   scale_x_discrete("Factor Value") +
   scale_fill_manual(values = c("blue4", "red3", "cyan3", "yellow3")) +
   labs(title = "Frequencies of each Value for each Categorical Variable") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank(),
         strip.background = element_rect(fill = "aliceblue",
                                         colour = NA))
```




```r
country_labels <- levels(fct_infreq(advertising_train$countryId))[c(seq(1,
                                                                         length(levels(fct_infreq(adv
                                                                         ceiling(length(levels(fct_in


ggplot(advertising_train) +
   geom_bar(aes(x = fct_infreq(countryId)),
            fill = "red3", alpha = 2/3) +
   scale_y_continuous(labels = comma,
                      expand = c(0.01, 0),
                      "Count") +
   scale_x_discrete(breaks = country_labels,
                    "countryId") +
   labs(title = "Frequency of observations for each \`countryId\`",
        subtitle = "(a categorical variable)",
        caption = "labels along x-axis are ID numbers and not numeric/double/ordinal/etc") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank())
```

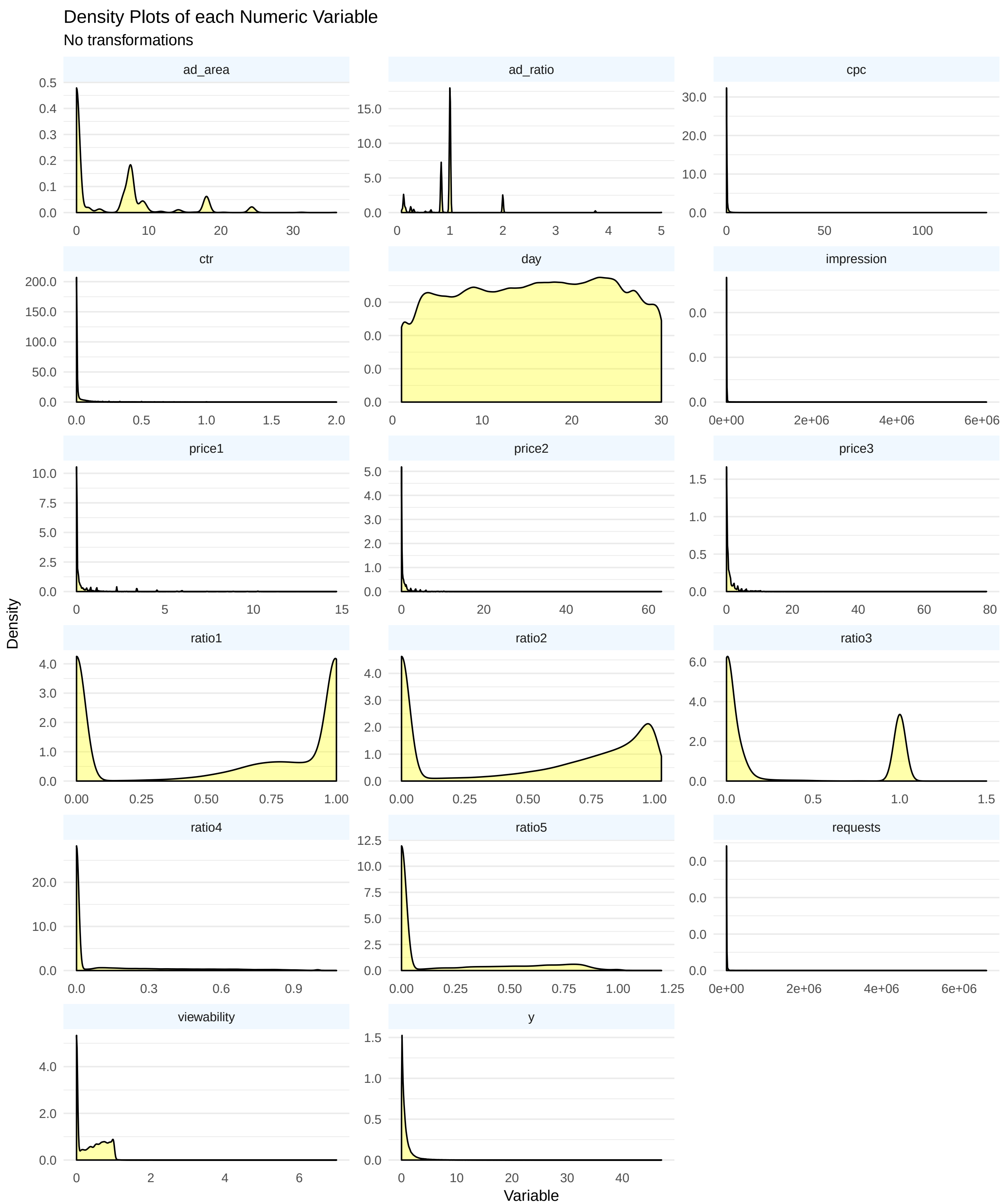


Frequency of observations for each `countryId`
(a categorical variable)
Count
20,000
15,000
10,000
5,000
0
234
190
157
125
174
113
29
142
207
71
countryId
labels along x−axis are ID numbers and not numeric/double/ordinal/etc

1.2.5 Univariate Plots

1.2.5.1 Numeric Variables

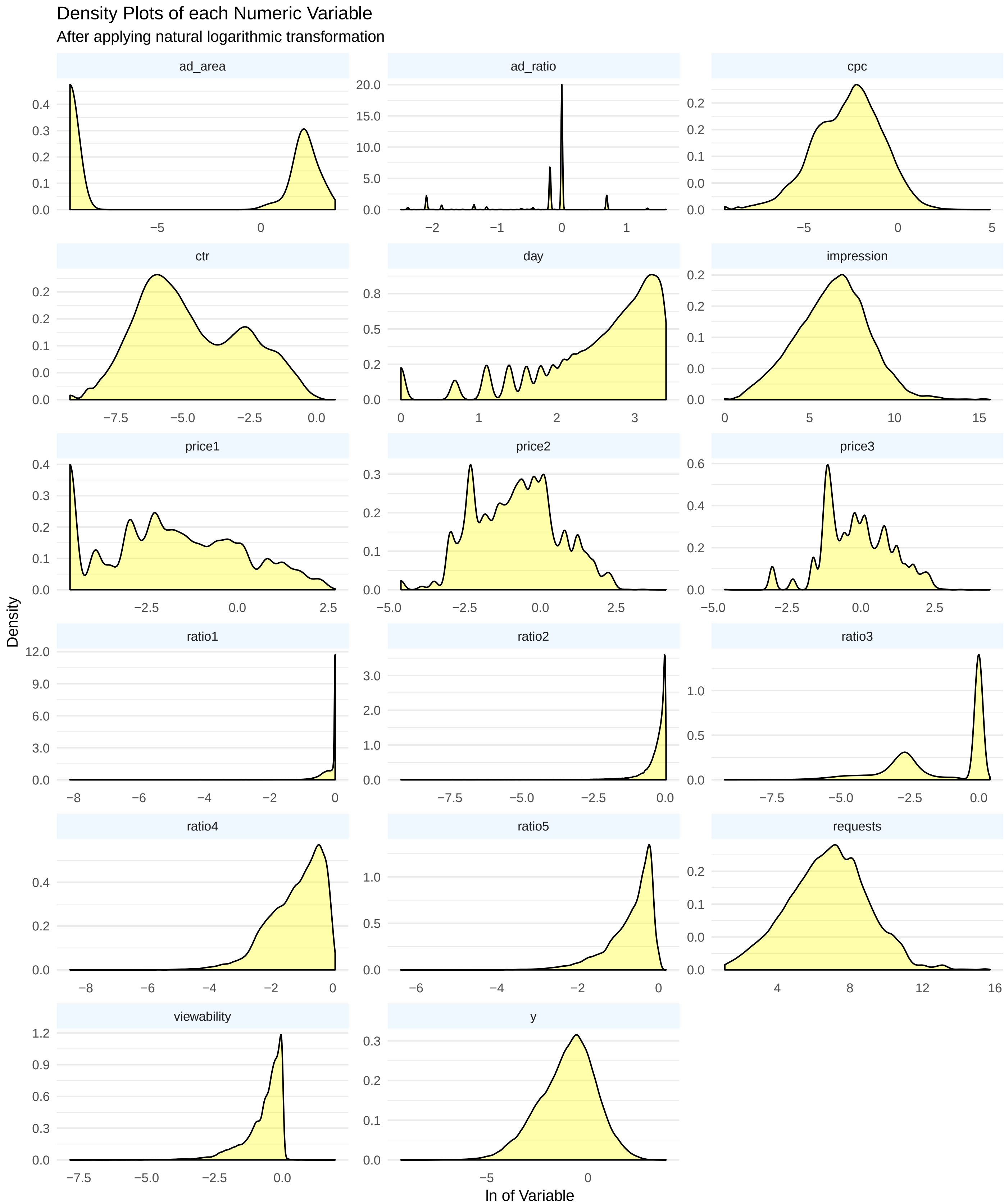
```r
ggplot(advertising_train_long_num) +
   geom_density(aes(x = Value),
                fill = "yellow",
                alpha = 1/3) +
   facet_rep_wrap(~Variable,
                  repeat.tick.labels = T,
                  scales = "free",
                  ncol = 3) +
   scale_y_continuous(labels = comma_format(accuracy = 0.1)) +
   labs(title = "Density Plots of each Numeric Variable",
        subtitle = "No transformations",
        x = "Variable",
        y = "Density")+
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank(),
         strip.background = element_rect(fill = "aliceblue",
                                         colour = NA))
```




```r
ggplot(advertising_train_long_num) +
  geom_density(aes(x = log(Value)),
               fill = "yellow",
               alpha = 1/3) +
```

```r
  facet_rep_wrap(~Variable,
                 repeat.tick.labels = T,
                 scales = "free",
                 ncol = 3) +
  scale_y_continuous(labels = comma_format(accuracy = 0.1)) +
  labs(title = "Density Plots of each Numeric Variable",
       subtitle = "After applying natural logarithmic transformation",
       x = "ln of Variable",
       y = "Density") +
  theme_minimal() +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        strip.background = element_rect(fill = "aliceblue",
                                        colour = NA))
```

```
## Warning: Removed 1213004 rows containing non-finite values (stat_density).
```



1.2.5.2 Logarithmic Transformations

It was observed from the plots above that natural logarithmic transformations were applicable for descriptive features `cpc`, `impression`, and potentially `ctr`. Target feature `y` was also suitable for a logarithmic transformation.

Table 4: Sample of advertising_train Data Frame After Logarithmic Transformations

case_id	companyId	countryId	deviceType	day	dow	price1	price2	price3	ad_area	ad_ratio	requests	impression
36845	159	100	1	6	Thursday	0.09	0.29	0.573	7.5000	0.8333	0	0
159944	43	116	1	23	Sunday	0.07	0.20	0.408	0.0001	1.0000	0	0
189734	159	43	3	27	Thursday	0.00	0.00	0.000	0.0001	1.0000	78	78
108368	43	19	2	16	Sunday	0.00	0.00	0.000	0.0001	1.0000	0	0
53202	43	56	2	9	Sunday	0.00	0.00	0.000	18.0000	2.0000	391	391
183052	43	56	3	26	Wednesday	0.00	0.00	0.000	0.0001	1.0000	407	407
199059	159	56	3	28	Friday	0.01	0.14	0.398	0.0001	1.0000	0	0
207900	43	100	3	30	Sunday	0.00	0.00	0.000	7.0920	0.1142	289	287
61883	43	226	2	10	Monday	0.22	0.83	1.644	9.4080	0.8333	753	634
13509	95	234	3	3	Monday	0.92	1.51	3.030	7.5000	0.8333	0	0
86364	43	227	2	13	Thursday	0.00	0.00	0.000	0.0001	1.0000	3	3
5493	95	38	2	1	Saturday	0.46	1.39	2.780	7.5000	0.8333	1183	850
108512	43	56	2	16	Sunday	0.41	0.83	1.663	8.7300	0.0928	1055	810
19399	43	171	1	4	Tuesday	0.00	0.00	0.000	0.0001	1.0000	13	9
62929	43	89	1	10	Monday	0.00	0.00	0.000	7.5000	0.8333	1252	1252
179089	43	38	2	25	Tuesday	1.19	1.19	1.191	7.5000	0.8333	0	0
141365	43	20	1	20	Thursday	0.00	0.00	0.000	0.0001	1.0000	0	0
134988	43	234	1	20	Thursday	0.00	0.00	0.000	0.0001	1.0000	14	11
177027	159	102	1	25	Tuesday	0.00	0.00	0.000	7.5000	0.8333	0	0
131526	43	68	2	19	Wednesday	0.00	0.00	0.000	0.0001	1.0000	28	23

```
advertising_train <- mutate(advertising_train,
                            "ln_cpc" = log(cpc),
                            "ln_ctr" = log(ctr),
                            "ln_impr" = log(impression),
                            "ln_req" = log(requests),
                            "ln_y" = log(y))

sample_adv <- sample_n(advertising_train, 20)

kable_styling(kable(sample_adv[ , 1 : floor(ncol(sample_adv)/2) ],
                    format.args = list(digits = 3),
                    caption = "Sample of advertising\\_train Data Frame After Logarithmic Transforma
              font_size = 8.5, latex_options = c("striped"),
              full_width = F)

kable_styling(kable(sample_adv[ , c(1, seq(from = floor(ncol(sample_adv)/2)+1,
                                           to = ncol(sample_adv),
                                           by = 1))],
                    format.args = list(digits = 3),
                    caption = "Sample of advertising\\_train Data Frame After Logarithmic Transforma
              font_size = 8.5, latex_options = c("striped"),
              full_width = F)
```

1.2.5.3 Comparison of Transformed Features to Normal Curve

As the logarithmic transformation resulted in infinite values, the data frame was trimmed to only include finite values. The finite data frame was then used to calculate the centre and spread of `ln_cpc`, `ln_ctr`, `ln_impr`, `ln_req`, and `ln_y`.

```
finite_cpc <- filter(advertising_train,
                     is.finite(ln_cpc))

p_cpc <- ggplot(finite_cpc) +
```

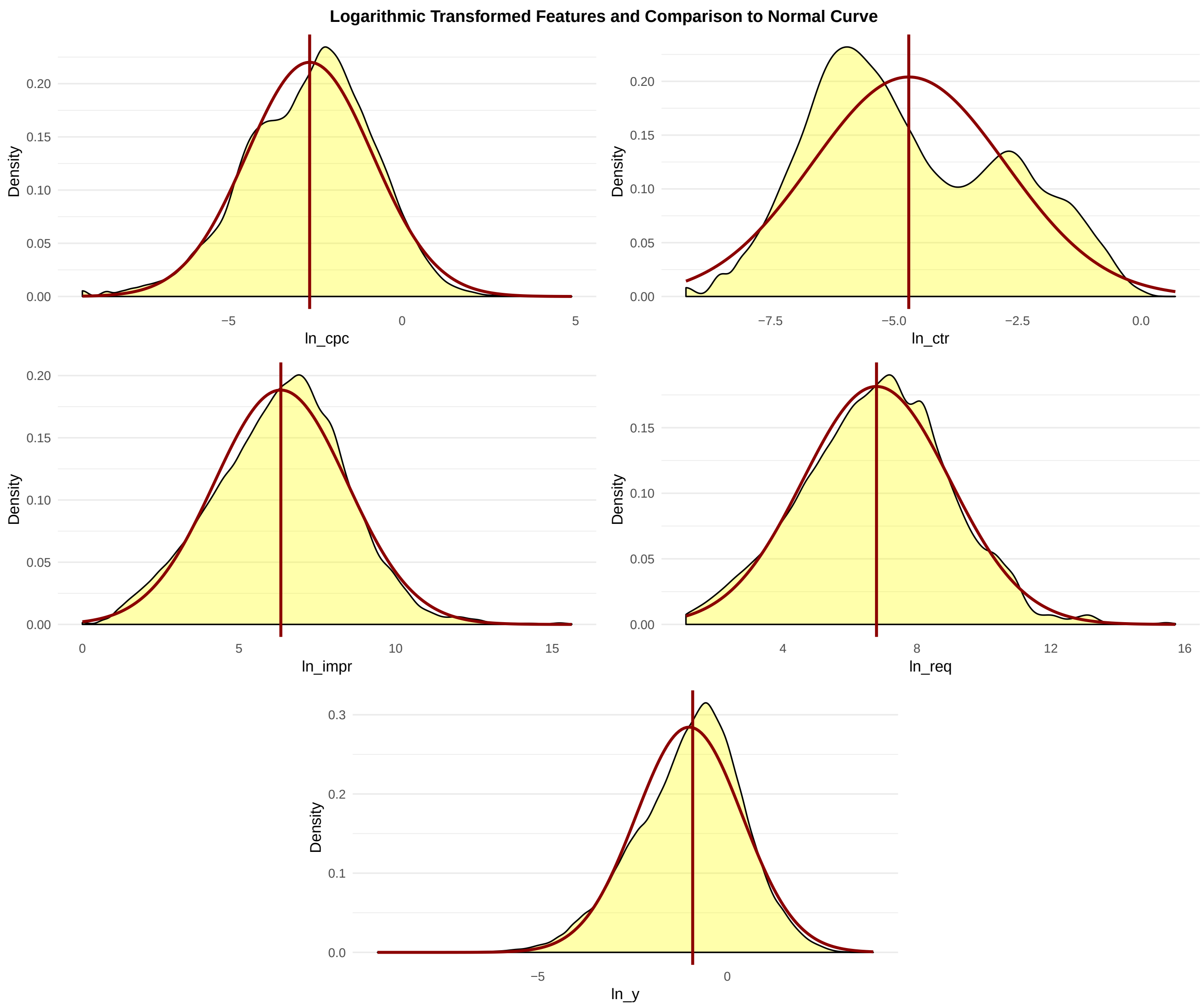
Table 5: Sample of advertising_train Data Frame After Logarithmic Transformations (cont)

case_id	cpc	ctr	viewability	ratio1	ratio2	ratio3	ratio4	ratio5	y	ln_cpc	ln_ctr	ln_impr	ln_req	ln_y
36845	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.2659	-Inf	-Inf	-Inf	-Inf	-1.3245
159944	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.0667	-Inf	-Inf	-Inf	-Inf	-2.7081
189734	0.0725	0.0128	0.811	1.000	0.923	0.0000	0.6923	0.308	1.1714	-2.6242	-4.36	4.36	4.36	0.1582
108368	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.5333	-Inf	-Inf	-Inf	-Inf	-0.6286
53202	0.0592	0.0051	0.244	1.000	0.557	1.0026	0.0000	0.000	0.2377	-2.8268	-5.28	5.97	5.97	-1.4367
183052	0.0106	0.0835	0.890	1.000	0.975	0.0074	0.2801	0.717	1.1341	-4.5469	-2.48	6.01	6.01	0.1258
199059	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.1610	-Inf	-Inf	-Inf	-Inf	-1.8264
207900	0.0881	0.0035	0.556	1.000	0.951	0.0836	0.1568	0.760	0.4604	-2.4293	-5.65	5.66	5.67	-0.7757
61883	0.1078	0.0063	0.650	0.639	0.953	1.0000	0.0000	0.000	0.4511	-2.2275	-5.07	6.45	6.62	-0.7961
13509	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	1.5468	-Inf	-Inf	-Inf	-Inf	0.4362
86364	0.0006	0.3333	1.000	1.000	0.667	1.0000	0.0000	0.000	0.2200	-7.4186	-1.10	1.10	1.10	-1.5141
5493	0.9710	0.0012	0.330	0.482	0.829	1.0000	0.0000	0.000	1.0158	-0.0294	-6.73	6.75	7.08	0.0157
108512	0.3086	0.0025	0.545	0.675	0.967	1.0000	0.0000	0.000	0.5512	-1.1757	-5.99	6.70	6.96	-0.5956
19399	0.0129	0.1111	1.000	1.000	1.000	0.0000	1.0000	0.000	0.9815	-4.3505	-2.20	2.20	2.56	-0.0187
62929	0.3065	0.0032	0.750	1.000	0.844	0.4481	0.0248	0.527	1.1109	-1.1825	-5.74	7.13	7.13	0.1051
179089	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.9166	-Inf	-Inf	-Inf	-Inf	-0.0871
141365	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.9444	-Inf	-Inf	-Inf	-Inf	-0.0572
134988	0.0820	0.0909	1.000	1.000	0.909	0.0000	0.4545	0.545	7.7429	-2.5010	-2.40	2.40	2.64	2.0468
177027	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.3806	-Inf	-Inf	-Inf	-Inf	-0.9660
131526	0.0071	0.1304	0.765	1.000	0.870	1.0435	0.0000	0.000	0.4125	-4.9477	-2.04	3.14	3.33	-0.8855

```r
  geom_density(aes(x = ln_cpc),
               fill = "yellow", alpha = 1/3) +
  stat_function(geom = "path", fun = dnorm,
                n = 200, col = "red4", size = 1,
                args = list(mean(finite_cpc$ln_cpc),
                            sd(finite_cpc$ln_cpc))) +
  geom_vline(xintercept = mean(finite_cpc$ln_cpc),
             col = "red4", size = 1) +
  ylab("Density") +
  theme_minimal() +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank())

finite_ctr <- filter(advertising_train,
                     is.finite(ln_ctr))

p_ctr <- ggplot(finite_ctr) +
  geom_density(aes(x = ln_ctr),
               fill = "yellow", alpha = 1/3) +
  stat_function(geom = "path", fun = dnorm,
                n = 200, col = "red4", size = 1,
                args = list(mean(finite_ctr$ln_ctr),
                            sd(finite_ctr$ln_ctr))) +
  geom_vline(xintercept = mean(finite_ctr$ln_ctr),
             col = "red4", size = 1) +
  ylab("Density") +
  theme_minimal() +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank())

finite_impr <- filter(advertising_train,
                      is.finite(ln_impr))
```

```
p_impr <- ggplot(finite_impr) +
   geom_density(aes(x = ln_impr),
                fill = "yellow", alpha = 1/3) +
   stat_function(geom = "path", fun = dnorm,
                 n = 200, col = "red4", size = 1,
                 args = list(mean(finite_impr$ln_impr),
                             sd(finite_impr$ln_impr))) +
   geom_vline(xintercept = mean(finite_cpc$ln_impr),
              col = "red4", size = 1) +
   ylab("Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank())

finite_req <- filter(advertising_train,
                     is.finite(ln_req))

p_req <- ggplot(finite_req) +
   geom_density(aes(x = ln_req),
                fill = "yellow", alpha = 1/3) +
   stat_function(geom = "path", fun = dnorm,
                 n = 200, col = "red4", size = 1,
                 args = list(mean(finite_req$ln_req),
                             sd(finite_req$ln_req))) +
   geom_vline(xintercept = mean(finite_cpc$ln_req),
              col = "red4", size = 1) +
   ylab("Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank())

finite_y <- filter(advertising_train,
                   is.finite(ln_y))

p_y <- ggplot(finite_y) +
   geom_density(aes(x = ln_y),
                fill = "yellow", alpha = 1/3) +
   stat_function(geom = "path", fun = dnorm,
                 n = 200, col = "red4", size = 1,
                 args = list(mean(finite_y$ln_y),
                             sd(finite_y$ln_y))) +
   geom_vline(xintercept = mean(finite_cpc$ln_y),
              col = "red4", size = 1) +
   ylab("Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank())

ln_vars_title <- textGrob("Logarithmic Transformed Features and Comparison to Normal Curve",
                          gp = gpar(fontface = "bold"))
```

```r
grid.arrange(top = ln_vars_title,
             p_cpc, p_ctr,
             p_impr, p_req,
             p_y,
             layout_matrix = matrix(c(1,1,2,2,
                                      3,3,4,4,
                                      NA,5,5,NA),
                                    ncol = 4,
                                    byrow = T))
```



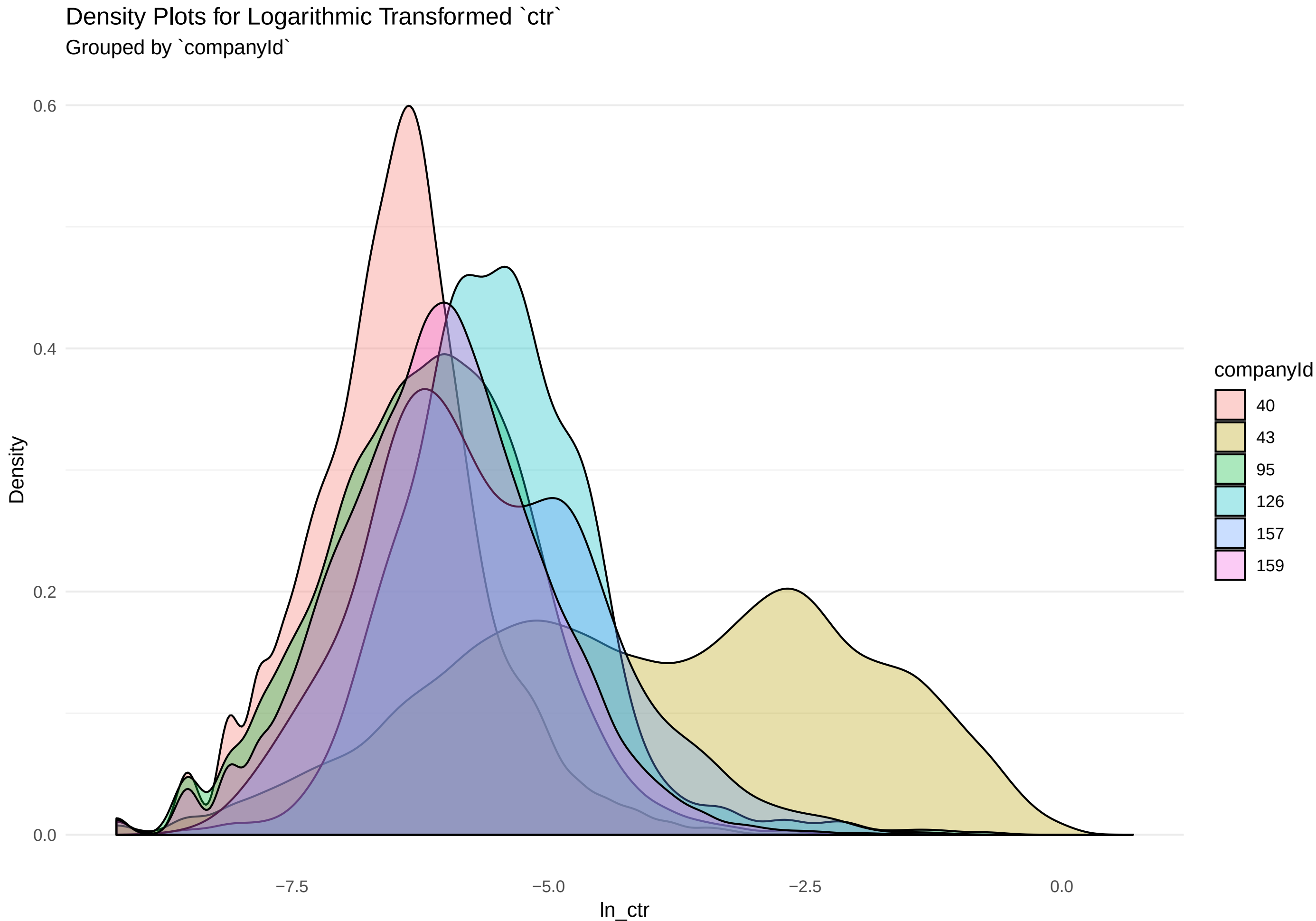
The natural logarithmic transformations of `impression` and `requests` clearly approached a normal distribution. The transformed `y` target feature somewhat resembled a normal distribution, albeit less closely as compared to `impression`. Both `cpc` and `ctr` appeared to be bimodal distributions after logarithmic transformation, with `ln_ctr` inarguably so.

1.2.6 Multivariate Plots

After transformation, grouping the `ln_ctr` distribution by level within the `companyId` factor revealed several distinct distributions. The distribution for `companyId` == 43 still appeared bimodal, which possibly indicated a further dimension of the multivariate relationship.

```r
ggplot(advertising_train) +
   geom_density(aes(x = ln_ctr, fill = companyId),
                alpha = 1/3) +
   labs(title = "Density Plots for Logarithmic Transformed \`ctr\`",
        subtitle = "Grouped by \`companyId\`",
        y = "Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank())
```

```
## Warning: Removed 78957 rows containing non-finite values (stat_density).
```

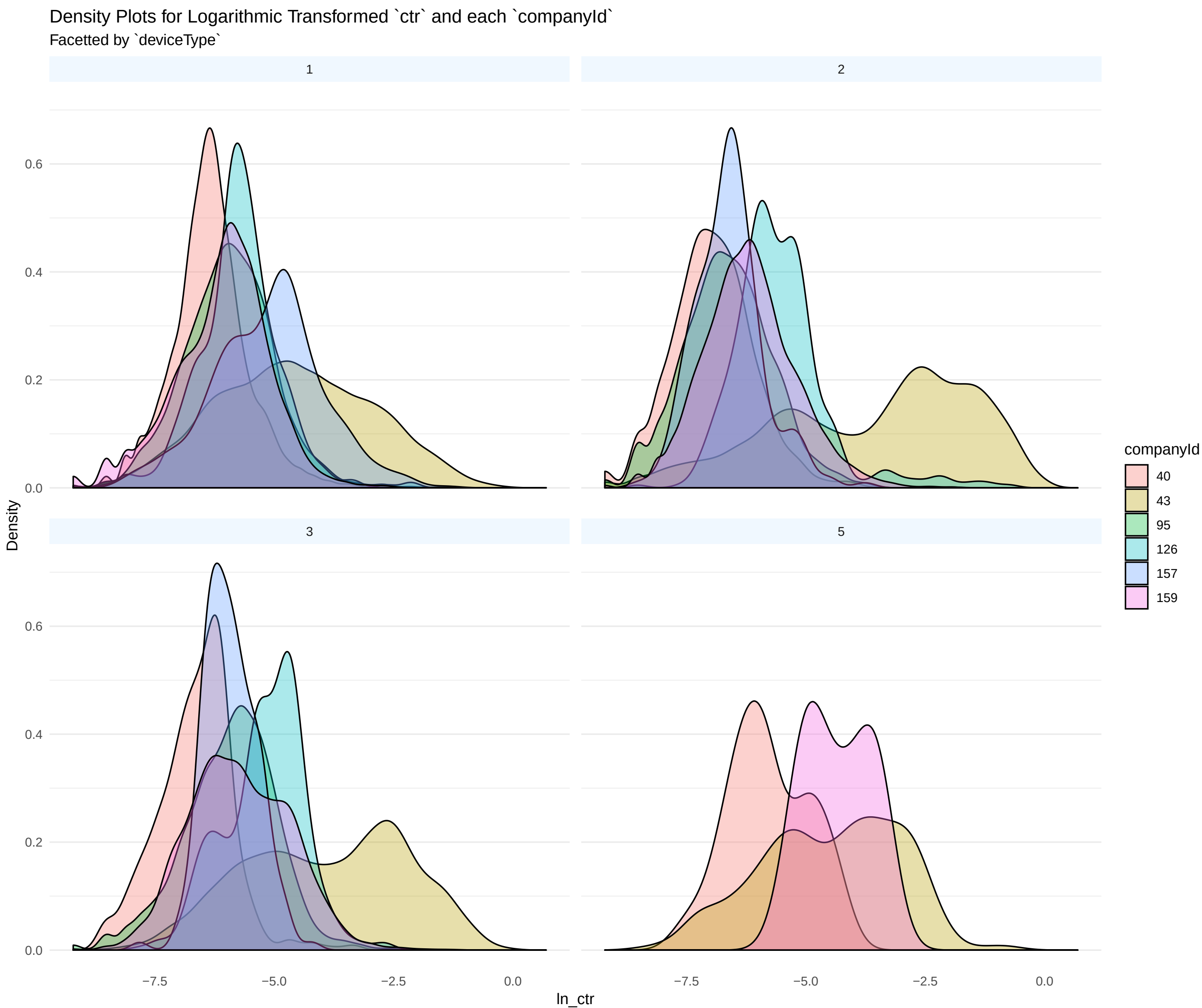


Producing separate density plots for each level within `deviceType` suggested some trivariate relationship between `ln_ctr`, `companyId`, and `deviceType`. The effect of facetting by `deviceType` was particularly apparent when examining `companyId` == 43, yet it still did not yield Gaussian distributions.

```r
ggplot(advertising_train) +
   geom_density(aes(x = ln_ctr, fill = companyId),
                alpha = 1/3) +
   facet_rep_wrap(~deviceType) +
   labs(title = "Density Plots for Logarithmic Transformed \`ctr\` and each \`companyId\`",
        subtitle = "Facetted by \`deviceType\`",
        y = "Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
```

```
        panel.grid.minor.x = element_blank(),
        strip.background = element_rect(fill = "aliceblue",
                                        colour = NA))
```

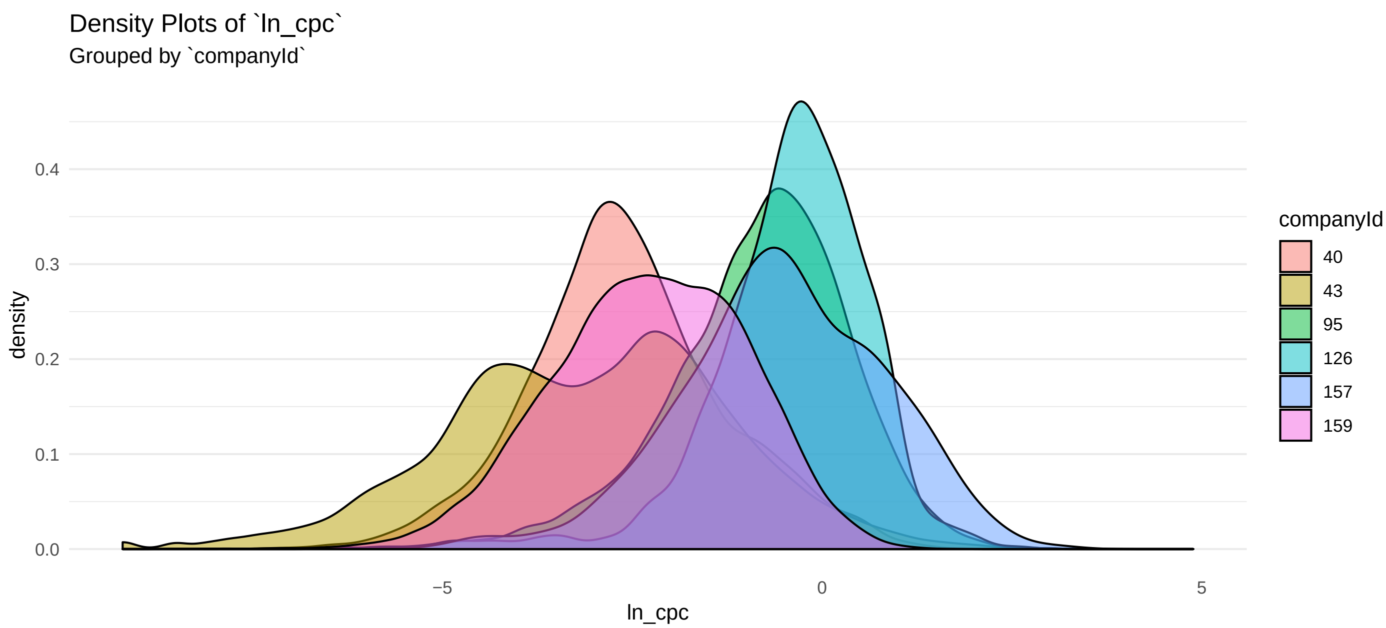
```
## Warning: Removed 78957 rows containing non-finite values (stat_density).
```



As above for `ln_ctr`, grouping by `companyId` and facetting by `deviceType` revealed a multivariate relationship between aforementioned descriptive features and the transformed `ln_cpc`.

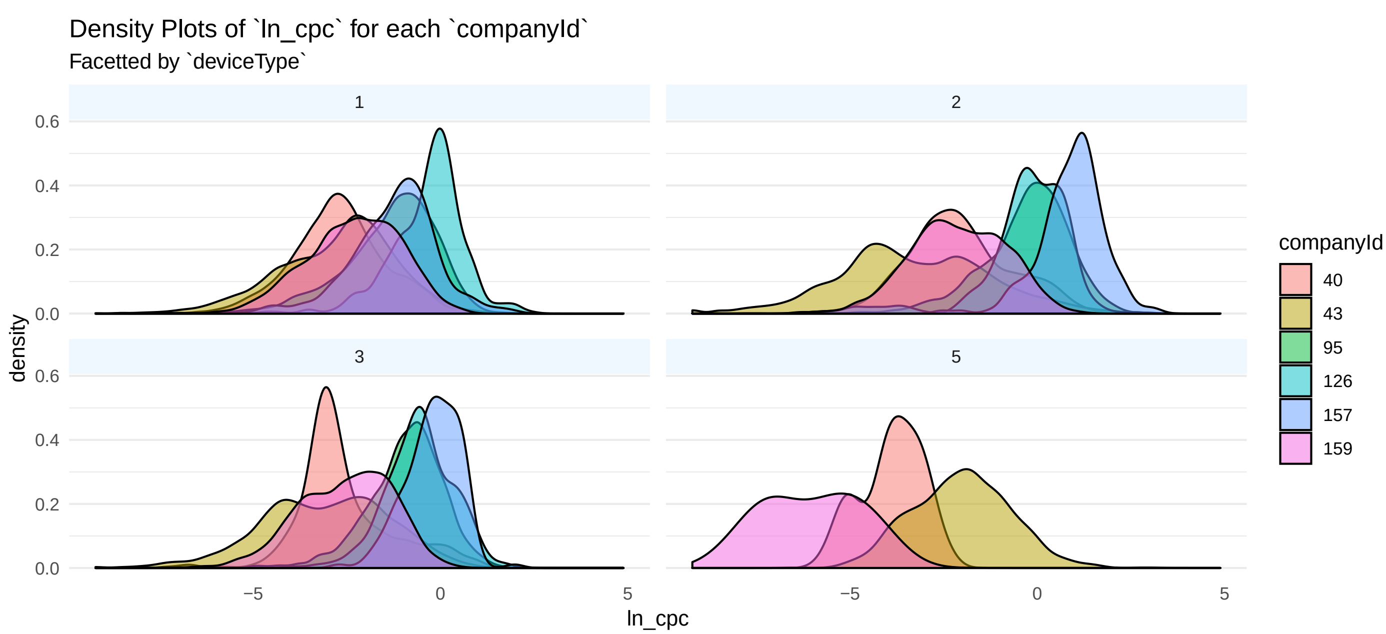
```
ggplot(advertising_train) +
   geom_density(aes(x = ln_cpc, fill = companyId),
                alpha = 1/2) +
   labs(title = "Density Plots of \`ln_cpc\`",
        subtitle = "Grouped by \`companyId\`",
        ylab = "Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank())
```

```
## Warning: Removed 78913 rows containing non-finite values (stat_density).
```




```
ggplot(advertising_train) +
  geom_density(aes(x = ln_cpc, fill = companyId),
               alpha = 1/2) +
  facet_rep_wrap(~deviceType) +
  labs(title = "Density Plots of \`ln_cpc\` for each \`companyId\`",
       subtitle = "Facetted by \`deviceType\`",
       ylab = "Density") +
  theme_minimal() +
  theme(panel.grid.major.x = element_blank(),
        panel.grid.minor.x = element_blank(),
        strip.background = element_rect(fill = "aliceblue",
                                        colour = NA))
```

```
## Warning: Removed 78913 rows containing non-finite values (stat_density).
```

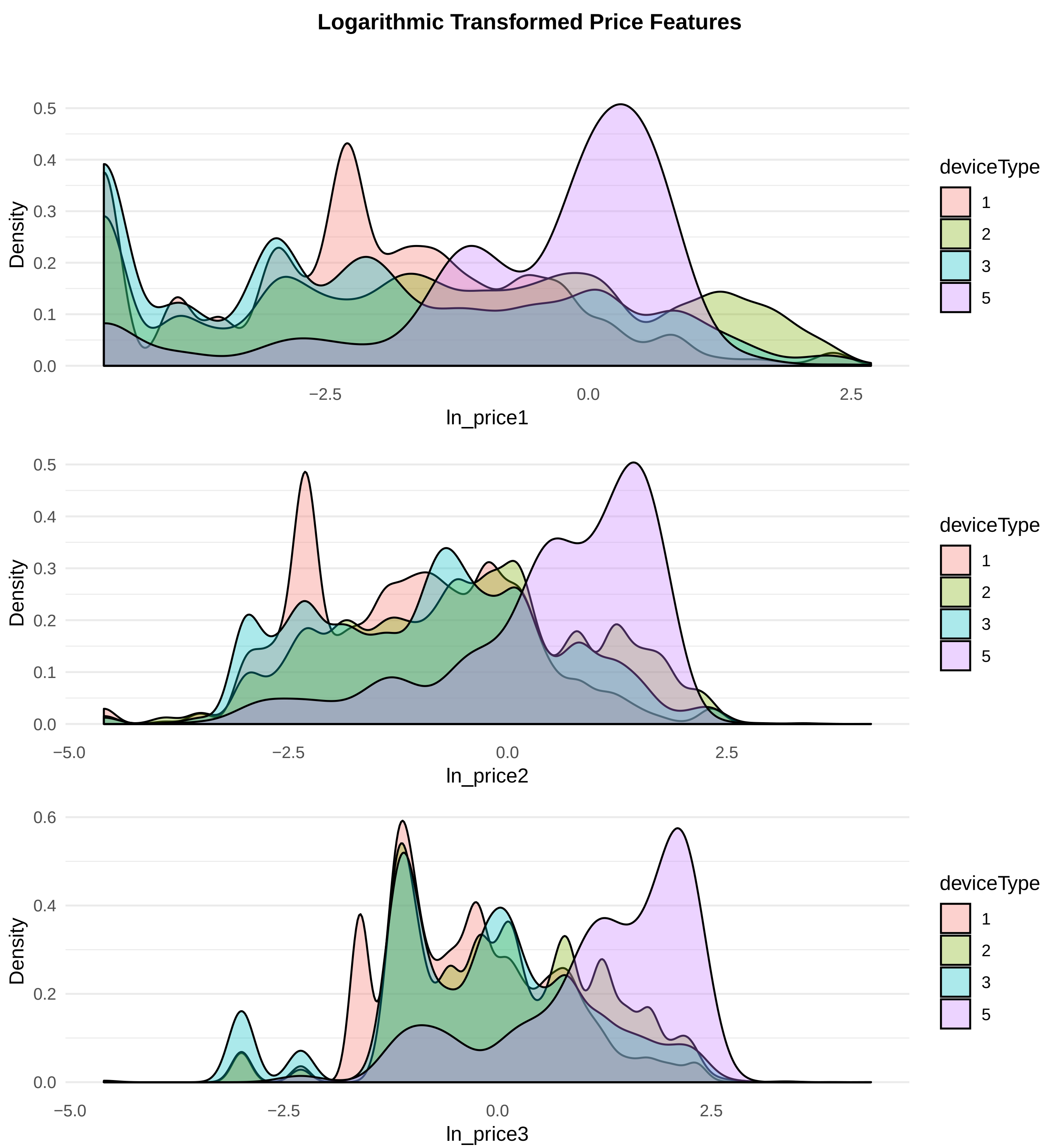


Each of the pricing features, (`price1`, `price2`, `price3`) were not suitably transformed by either logarithmic, square root, or cube root. Logarithmic transformations appeared to spread the data the most, but these transformations considerably diverged from a symmetrical normal distribution. Further grouping by `deviceType` did not reveal Gaussian distributions.

```
price_trans <- mutate(advertising_train,
                      "ln_price1" = log(price1),
                      "ln_price2" = log(price2),
                      "ln_price3" = log(price3))

p_price1_trans <- ggplot(price_trans) +
   geom_density(aes(x = ln_price1, fill = deviceType),
                alpha = 1/3) +
   labs(y = "Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank())

p_price2_trans <- ggplot(price_trans) +
   geom_density(aes(x = ln_price2, fill = deviceType),
                alpha = 1/3) +
   labs(y = "Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank())

p_price3_trans <- ggplot(price_trans) +
   geom_density(aes(x = ln_price3, fill = deviceType),
                alpha = 1/3) +
   labs(y = "Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),
         panel.grid.minor.x = element_blank())


price_vars_title <- textGrob("Logarithmic Transformed Price Features",
                             gp = gpar(fontface = "bold"))

grid.arrange(price_vars_title,
             p_price1_trans, p_price2_trans,
             p_price3_trans,
             layout_matrix = matrix(c(1,
                                      2,
                                      2,
                                      2,
                                      3,
                                      3,
                                      3,
                                      4,
                                      4,
                                      4),
                                    ncol = 1,
                                    byrow = T))
```

```
## Warning: Removed 92892 rows containing non-finite values (stat_density).
```

```
## Warning: Removed 92804 rows containing non-finite values (stat_density).

## Warning: Removed 92804 rows containing non-finite values (stat_density).
```

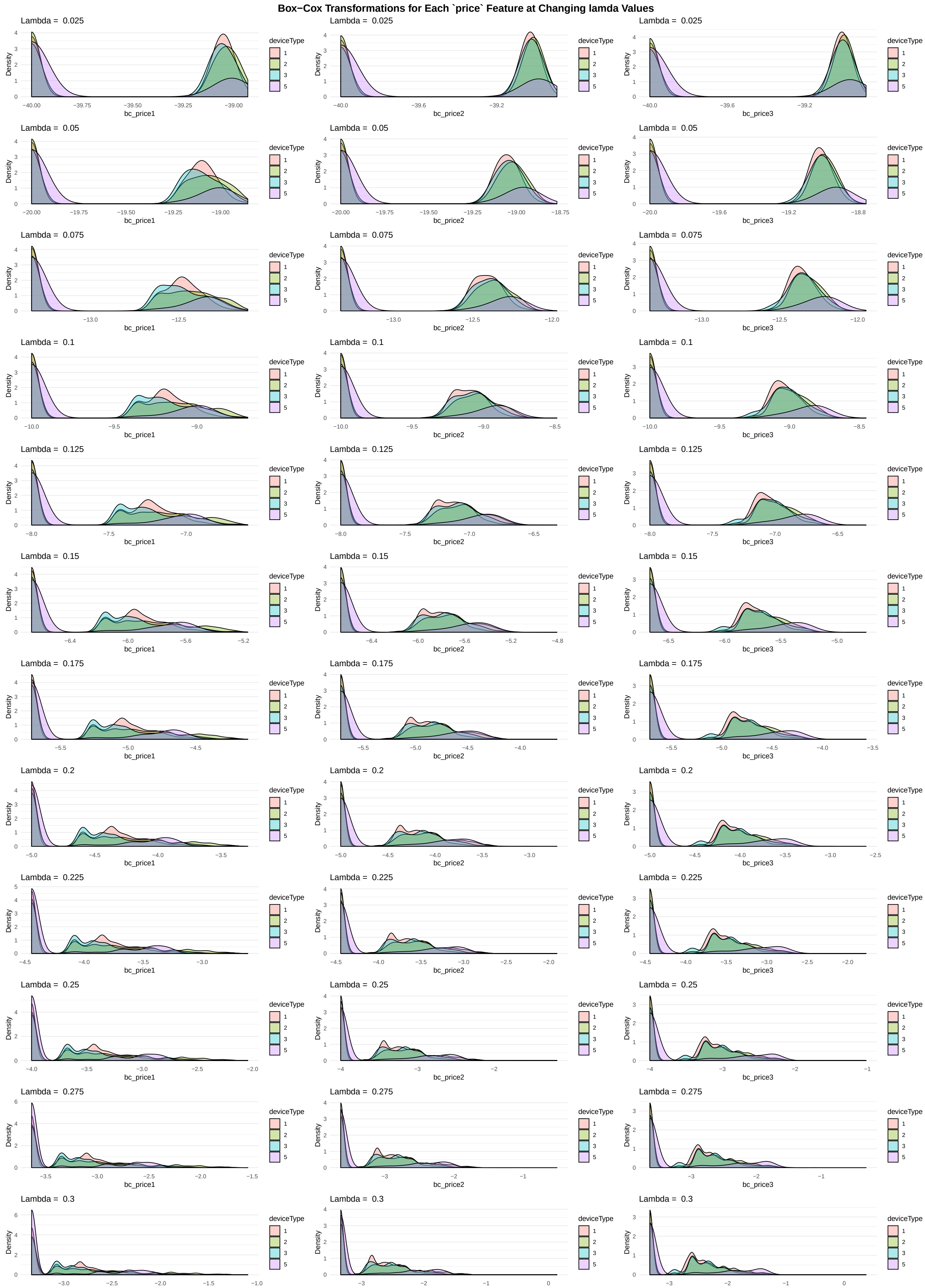


Box-Cox transformations with a range of lamda values also did not convert the price features into distributions that resembled a normal curve.

```r
boxcox <- function(x, lambda = 1) {

   (x^(lambda) - 1 /
       (lambda))

}

box_grobs_2 <- list()
box_grobs_higher <- list()

for (i in 1:length(seq(0.025, 0.3, 0.025))) {

   j <- seq(0.025, 0.3, 0.025)[i]

   boxcox_price <- mutate(advertising_train,
                          "bc_price1" = boxcox(x = price1,
                                               lambda = j),
                          "bc_price2" = boxcox(x = price2,
                                               lambda = j),
                          "bc_price3" = boxcox(x = price3,
                                               lambda = j))

   bc_colnames <- colnames(boxcox_price)[str_detect(colnames(boxcox_price), "bc_price")]

   for (k in bc_colnames) {

      m <- which(bc_colnames %in% k)

      box_grobs_2[[m]] <- ggplot(select(boxcox_price,
                                        k, deviceType)) +
         geom_density(aes(x = .data[[k]], fill = deviceType),
                      alpha = 1/3) +
         labs(title = paste("Lambda = ", j)) +
         ylab("Density") + xlab(k) +
         theme_minimal() +
         theme(panel.grid.major.x = element_blank(),
               panel.grid.minor.x = element_blank())

   }

   box_grobs_higher[[i]] <- box_grobs_2

}

density_by_lambda <- list()

for (i in 1:12) {

   density_by_lambda[[i]] <-  do.call(what = grid.arrange,
                                      args = list(grobs = box_grobs_higher[[i]],
                                                  nrow = 1))
```

```
}
```

```
do.call(what = grid.arrange,
        args = list(grobs = density_by_lambda,
                    top = textGrob("Box-Cox Transformations for Each \`price\` Feature at Changing l
                                   gp = gpar(fontsize=16,
                                             fontface = "bold")),
                    ncol = 1))
```


Box−Cox Transformations for Each `price` Feature at Changing lamda Values
Lambda = 0.025
Lambda = 0.05
Lambda = 0.075
Lambda = 0.1
Lambda = 0.125
Lambda = 0.15
Lambda = 0.175
Lambda = 0.2
Lambda = 0.225
Lambda = 0.25
Lambda = 0.275
Lambda = 0.3
Density
bc_price1
bc_price2
bc_price3
deviceType
1
2
3
5

The remaining numeric features (`ad_area`, `ad_ratio`, `day`, `ratio1`, `ratio2`, `ratio3`, `ratio4`, `ratio5`, and `viewability`) were not able to be transformed to distributions that approached normal curves via root or logarithmic methods. Despite the accompanying documentation for the prescribed dataset, the `ad_area` and `day` may not strictly be classed as numeric/double variables. Considering the low range, `ad_area` could be intepreted as an identifier, and so categorical. The feature `day`, values 1 - 30, is better interpreted as an ordinal or time value. However, time series forecasting is outside the scope of this project, and so the `day` feature will be largely ignored from the model and only used for partitioning.

1.2.6.1 Data Normalisation

Considering each of the features span differing ranges, both in their raw and transformed applications, it was deemed necessary to normalise each. Normalising the data allowed for more

As outlined in **Fundamentals of Machine Learning**, the below formula was used for normalising the data:

$$a_i^{'} = \left(\frac{a_i - min(a)}{max(a) - min(a)} \right) \times (high - low) + low$$

Where a is the feature, whether descriptive or target, $high$ is the highest value in the normalised data range, and low is the lowest value in the normalised data range. A range of 0 - 1 was chosen, so these values were used for low and $high$ respectively.

```
normalise <- function(x) {

   x[is.infinite(x) == T] <- NA

   (((x - min(x, na.rm = T)) /
        (max(x, na.rm = T) - min(x, na.rm = T))) * (1 - 0) + 0)

}

num_feats <- select(advertising_train,
                    case_id,
                    which(sapply(advertising_train, class)=="numeric"))

for ( i in colnames(num_feats)) {

   newfeat <- paste0("norm_", i)

   advertising_train[[newfeat]] <- normalise(num_feats[[i]])

   advertising_train[[newfeat]][is.na(advertising_train[[newfeat]])] <- advertising_train[[i]]

}
```

```
## Warning in advertising_train[[newfeat]]
## [is.na(advertising_train[[newfeat]])] <- advertising_train[[i]]: number of
## items to replace is not a multiple of replacement length

## Warning in advertising_train[[newfeat]]
## [is.na(advertising_train[[newfeat]])] <- advertising_train[[i]]: number of
## items to replace is not a multiple of replacement length

## Warning in advertising_train[[newfeat]]
## [is.na(advertising_train[[newfeat]])] <- advertising_train[[i]]: number of
## items to replace is not a multiple of replacement length

## Warning in advertising_train[[newfeat]]
## [is.na(advertising_train[[newfeat]])] <- advertising_train[[i]]: number of
## items to replace is not a multiple of replacement length
```

```
sample_adv <- sample_n(advertising_train, 20)
```

Table 6: Sample of advertising_train Data Frame with Normalised Numeric Features (1/3)

case_id	companyId	countryId	deviceType	day	dow	price1	price2	price3	ad_area	ad_ratio	requests	impression	cpc	ctr	viewability
128,648	40	113	1	19	Wednesday	0.10	0.10	0.20	0.0001	1.000	295	136	0.0297	0.0074	0.516
91,462	43	103	1	14	Friday	0.18	0.53	1.05	7.5000	0.833	0	0	0.0000	0.0000	0.000
1,872	43	56	2	1	Saturday	0.00	0.00	0.00	3.9600	0.091	2,162	2,151	0.0543	0.0033	0.704
57,676	40	116	1	9	Sunday	0.10	0.10	0.20	0.0001	1.000	0	0	0.0000	0.0000	0.000
117,805	43	112	2	17	Monday	5.70	5.70	5.70	0.0001	1.000	194	43	0.0302	0.0930	0.609
44,642	95	234	1	7	Friday	0.05	0.05	0.05	7.5000	0.833	0	0	0.0000	0.0000	0.000
24,282	159	17	2	4	Tuesday	0.00	0.00	0.00	0.0001	1.000	682	610	0.0202	0.0033	0.862
106,357	43	98	2	16	Sunday	0.00	0.00	0.00	0.0001	1.000	0	0	0.0000	0.0000	0.000
129,661	43	12	2	19	Wednesday	0.00	0.00	0.00	0.0001	1.000	18	18	0.0093	0.0556	1.000
178,825	43	234	5	25	Tuesday	0.31	1.38	2.76	7.5000	0.833	3,984	1,843	4.7411	0.0005	0.630
48,986	95	234	3	8	Saturday	0.17	1.24	2.48	18.0000	2.000	1,246	972	0.5858	0.0021	0.158
173,674	95	234	1	25	Tuesday	1.12	2.54	5.07	7.5000	0.833	3,504	2,176	2.3288	0.0009	0.180
202,874	95	13	3	29	Saturday	0.01	0.45	0.89	18.0000	2.000	0	0	0.0000	0.0000	0.000
124,197	43	12	3	18	Tuesday	0.00	0.00	0.00	0.0001	1.000	20	20	0.0056	0.1000	0.250
30,209	43	12	2	5	Wednesday	0.57	0.57	0.57	7.5000	0.833	0	0	0.0000	0.0000	0.000
97,721	43	56	3	15	Saturday	0.48	0.78	1.56	14.0400	0.641	837	699	0.0601	0.0086	0.630
29,316	95	234	1	5	Wednesday	0.54	2.29	4.57	7.5000	0.833	5,580	1,463	2.2940	0.0007	0.083
127,966	159	98	1	19	Wednesday	0.00	0.00	0.00	0.0001	1.000	52,276	19,271	0.0508	0.0053	0.563
29,618	43	227	1	5	Wednesday	0.11	0.31	0.61	7.5000	0.833	0	0	0.0000	0.0000	0.000
141,192	43	38	2	20	Thursday	0.00	0.00	0.00	0.0001	1.000	0	0	0.0000	0.0000	0.000

```r
kable_styling(kable(sample_adv[, 1:floor(ncol(sample_adv)/3)],
                    caption = "Sample of advertising\\_train Data Frame with Normalised Numeric Feat
                    format.args = list(digits = 2, scientific = F,
                                       big.mark = ",")),
              font_size = 8, latex_options = c("striped"),
              full_width = T)
```

```r
kable_styling(kable(sample_adv[, c(1,
                                   seq(from = floor(ncol(sample_adv)/3)*1+1,
                                       to = floor(ncol(sample_adv)/3)*2,
                                       by = 1))],
                    caption = "Sample of advertising\\_train Data Frame with Normalised Numeric Feat
                    format.args = list(digits = 2, scientific = F,
                                       big.mark = ",")),
              font_size = 8, latex_options = c("striped"),
              full_width = T)
```

```r
kable_styling(kable(sample_adv[, c(1,
                                   seq(from = floor(ncol(sample_adv)/3)*2+1,
                                       to = floor(ncol(sample_adv)/3)*3,
                                       by = 1))],
                    caption = "Sample of advertising\\_train Data Frame with Normalised Numeric Feat
                    format.args = list(digits = 2, scientific = F,
                                       big.mark = ",")),
              font_size = 8, latex_options = c("striped"),
              full_width = T)
```

Table 7: Sample of advertising_train Data Frame with Normalised Numeric Features (2/3)

case_id	ratio1	ratio2	ratio3	ratio4	ratio5	y	ln_cpc	ln_ctr	ln_impr	ln_req	ln_y	norm_case_id	norm_day	norm_price1	norm_price2	norm_price3
128,648	0.99	0.85	0.0294	0.250	0.72	0.057	-3.52	-4.9	4.9	5.7	-2.86	0.6008	0.62	0.00681	0.00158	0.00253
91,462	0.00	0.00	0.0000	0.000	0.00	1.614	-Inf	-Inf	-Inf	-Inf	0.48	0.4271	0.45	0.01225	0.00840	0.01334
1,872	1.00	0.96	1.0000	0.000	0.00	0.181	-2.91	-5.7	7.7	7.7	-1.71	0.0087	0.00	0.00000	0.00000	0.00000
57,676	0.00	0.00	0.0000	0.000	0.00	0.040	-Inf	-Inf	-Inf	-Inf	-3.22	0.2693	0.28	0.00681	0.00158	0.00253
117,805	1.00	0.51	1.0000	0.000	0.00	0.325	-3.50	-2.4	3.8	5.3	-1.12	0.5502	0.55	0.38802	0.09030	0.07221
44,642	0.00	0.00	0.0000	0.000	0.00	0.406	-Inf	-Inf	-Inf	-Inf	-0.90	0.2085	0.21	0.00340	0.00079	0.00063
24,282	1.00	0.94	1.0000	0.000	0.00	0.051	-3.90	-5.7	6.4	6.5	-2.98	0.1134	0.10	0.00000	0.00000	0.00000
106,357	0.00	0.00	0.0000	0.000	0.00	0.040	-Inf	-Inf	-Inf	-Inf	-3.22	0.4967	0.52	0.00000	0.00000	0.00000
129,661	1.00	0.94	1.0556	0.000	0.00	0.263	-4.68	-2.9	2.9	2.9	-1.34	0.6055	0.62	0.00000	0.00000	0.00000
178,825	0.78	0.30	1.0000	0.000	0.00	0.901	1.56	-7.6	7.5	8.3	-0.10	0.8351	0.83	0.02110	0.02186	0.03494
48,986	0.75	0.99	0.0031	0.836	0.16	0.735	-0.53	-6.2	6.9	7.1	-0.31	0.2288	0.24	0.01157	0.01965	0.03143
173,674	0.73	0.96	0.1002	0.467	0.43	1.204	0.85	-7.0	7.7	8.2	0.19	0.8111	0.83	0.07624	0.04024	0.06426
202,874	0.00	0.00	0.0000	0.000	0.00	0.790	-Inf	-Inf	-Inf	-Inf	-0.24	0.9474	0.97	0.00068	0.00713	0.01128
124,197	1.00	0.65	0.0000	0.200	0.80	0.756	-5.18	-2.3	3.0	3.0	-0.28	0.5800	0.59	0.00000	0.00000	0.00000
30,209	0.00	0.00	0.0000	0.000	0.00	1.161	-Inf	-Inf	-Inf	-Inf	0.15	0.1411	0.14	0.03880	0.00903	0.00724
97,721	0.83	0.99	0.0129	0.195	0.79	0.529	-2.81	-4.8	6.5	6.7	-0.64	0.4564	0.48	0.03268	0.01236	0.01976
29,316	0.63	0.66	0.0478	0.430	0.52	0.335	0.83	-7.3	7.3	8.6	-1.09	0.1369	0.14	0.03676	0.03628	0.05792
127,966	1.00	1.00	0.0606	0.073	0.87	0.090	-2.98	-5.2	9.9	10.9	-2.40	0.5976	0.62	0.00000	0.00000	0.00000
29,618	0.00	0.00	0.0000	0.000	0.00	0.865	-Inf	-Inf	-Inf	-Inf	-0.15	0.1383	0.14	0.00749	0.00491	0.00767
141,192	0.00	0.00	0.0000	0.000	0.00	2.457	-Inf	-Inf	-Inf	-Inf	0.90	0.6594	0.66	0.00000	0.00000	0.00000

Table 8: Sample of advertising_train Data Frame with Normalised Numeric Features (3/3)

case_id	norm_ad_area	norm_ad_ratio	norm_requests	norm_impression	norm_cpc	norm_ctr	norm_viewability	norm_ratio1	norm_ratio2	norm_ratio3	norm_ratio4	norm_ratio5	norm_y	norm_ln_cpc	norm_ln_ctr	norm_ln_impr
128,648	0.00	0.1864	0.0000440	0.0000223	0.000224	0.00370	0.074	0.99	0.83	0.0196	0.232	0.60	0.00122	0.40	0.43	0.31
91,462	0.21	0.1525	0.0000000	0.0000000	0.000000	0.00000	0.000	0.00	0.00	0.0000	0.000	0.00	0.03430	-3.51	-Inf	7.76
1,872	0.11	0.0015	0.0003226	0.0003526	0.000410	0.00165	0.101	1.00	0.94	0.6667	0.000	0.00	0.00385	0.45	0.35	0.49
57,676	0.00	0.1864	0.0000000	0.0000000	0.000000	0.00000	0.000	0.00	0.00	0.0000	0.000	0.00	0.00085	-3.79	-Inf	-Inf
117,805	0.00	0.1864	0.0000289	0.0000070	0.000228	0.04650	0.087	1.00	0.50	0.6667	0.000	0.00	0.00691	0.41	0.69	0.24
44,642	0.21	0.1525	0.0000000	0.0000000	0.000000	0.00000	0.000	0.00	0.00	0.0000	0.000	0.00	0.00863	-5.65	-3.32	3.78
24,282	0.00	0.1864	0.0001018	0.0001000	0.000152	0.00165	0.123	1.00	0.91	0.6667	0.000	0.00	0.00107	0.38	0.35	0.41
106,357	0.00	0.1864	0.0000000	0.0000000	0.000000	0.00000	0.000	0.00	0.00	0.0000	0.000	0.00	0.00085	-3.32	-Inf	-Inf
129,661	0.00	0.1864	0.0000027	0.0000030	0.000070	0.02780	0.143	1.00	0.92	0.7037	0.000	0.00	0.00558	0.32	0.64	0.18
178,825	0.21	0.1525	0.0005945	0.0003021	0.035773	0.00025	0.090	0.78	0.29	0.6667	0.000	0.00	0.01914	0.76	0.16	0.48
48,986	0.50	0.3898	0.0001859	0.0001593	0.004420	0.00105	0.023	0.75	0.96	0.0021	0.777	0.13	0.01562	0.62	0.31	0.44
173,674	0.21	0.1525	0.0005228	0.0003567	0.017571	0.00045	0.026	0.73	0.94	0.0668	0.434	0.36	0.02559	0.71	0.22	0.49
202,874	0.50	0.3898	0.0000000	0.0000000	0.000000	0.00000	0.000	0.00	0.00	0.0000	0.000	0.00	0.01678	-Inf	-Inf	9.04
124,197	0.00	0.1864	0.0000030	0.0000033	0.000042	0.05000	0.036	1.00	0.63	0.0000	0.186	0.67	0.01605	0.29	0.70	0.19
30,209	0.21	0.1525	0.0000000	0.0000000	0.000000	0.00000	0.000	0.00	0.00	0.0000	0.000	0.00	0.02467	-Inf	-Inf	-Inf
97,721	0.39	0.1134	0.0001249	0.0001146	0.000454	0.00430	0.090	0.83	0.97	0.0086	0.181	0.66	0.01124	0.45	0.45	0.42
29,316	0.21	0.1525	0.0008326	0.0002398	0.017309	0.00035	0.012	0.63	0.65	0.0319	0.399	0.44	0.00711	0.71	0.20	0.47
127,966	0.00	0.1864	0.0078001	0.0031590	0.000383	0.00265	0.080	1.00	0.97	0.0404	0.067	0.72	0.00192	0.44	0.40	0.63
29,618	0.21	0.1525	0.0000000	0.0000000	0.000000	0.00000	0.000	0.00	0.00	0.0000	0.000	0.00	0.01837	-1.97	-4.89	-Inf
141,192	0.00	0.1864	0.0000000	0.0000000	0.000000	0.00000	0.000	0.00	0.00	0.0000	0.000	0.00	0.05221	-3.76	-3.15	-Inf

1.3 References

- Kelleher J.D., Namee B.M., D'Arcy A., 2015, *Fundamentals of Machine Learning for Predictive Data Analytics*, Massachusetts Institute of Technology, USA.
- Osborne J.W., 2010, *Improving your data transformations: Applying the Box-Cox transformation*, Practical Assessment, Research & Evaluation, V.05 No.12, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.470.7417&rep=rep1&type=pdfhttp://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.470.7417&rep=rep1&type=pdf