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.

```r
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
194899	159	190	2	27	Thursday	0.12	0.27	0.5437	0.0001	1.00000
93357	159	57	1	14	Friday	0.00	0.00	0.0000	0.0001	1.00000
107878	43	38	2	16	Sunday	0.00	0.00	0.0000	0.0001	1.00000
182788	159	57	1	26	Wednesday	0.00	0.00	0.0000	7.5000	0.83333
87515	95	102	2	13	Thursday	0.01	0.10	0.3100	18.0000	2.00000
92109	43	231	2	14	Friday	0.00	0.00	0.0000	0.0001	1.00000
1323	43	50	2	1	Saturday	4.56	4.56	4.5626	0.0001	1.00000
87219	43	57	1	13	Thursday	0.42	1.14	2.2947	9.4080	0.83333
108001	95	234	3	16	Sunday	0.34	3.08	6.1600	0.0001	1.00000
45449	159	102	1	8	Saturday	0.00	0.00	0.0000	0.0001	1.00000
160091	43	166	1	23	Sunday	0.57	1.26	2.5072	7.5000	0.83333
159255	95	102	2	23	Sunday	0.06	0.36	0.7200	7.5000	0.83333
142184	43	234	3	21	Friday	0.00	0.00	0.0000	0.0001	1.00000
78179	43	38	3	12	Wednesday	0.62	1.17	2.3562	18.0000	2.00000
157641	159	57	2	23	Sunday	0.00	0.00	0.0000	0.0001	1.00000
24273	40	229	1	4	Tuesday	0.00	0.00	0.0000	0.0001	1.00000
204275	43	137	2	29	Saturday	0.00	0.00	0.0000	8.4000	0.74405
182350	43	202	2	26	Wednesday	0.35	0.49	0.9831	24.2500	0.25773
27779	159	57	2	5	Wednesday	0.00	0.00	0.0000	24.2500	0.25773
63785	159	13	2	10	Monday	0.01	0.09	0.3401	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
194899	7341	3174	0.0391	0.0050	0.7322	0.6043	0.9726	1.0000	0.0000	0.0000	0.1007012
93357	1386	1342	0.4081	0.0007	0.2297	1.0000	0.6587	0.0559	0.2765	0.6677	0.2413623
107878	3479	3338	0.0161	0.1747	0.9847	1.0000	0.8841	1.0000	0.0000	0.0000	2.4645122
182788	4980	4636	0.1030	0.0022	0.1458	1.0000	0.5360	0.0770	0.1555	0.7673	0.1945722
87515	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2444444
92109	141	130	0.0014	0.2077	0.7604	1.0000	0.4462	1.0000	0.0000	0.0000	0.2677419
1323	2493	465	0.0204	0.1140	0.7860	1.0000	0.2473	1.0000	0.0000	0.0000	0.3271824
87219	934	828	0.2527	0.0085	0.3122	0.7283	0.9396	0.0761	0.2488	0.6751	1.9321094
108001	10807	9974	0.7545	0.0049	0.4125	0.8686	0.7951	0.0066	0.8091	0.1843	3.5766827
45449	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2391304
160091	691	350	0.6633	0.0029	0.9563	0.6571	0.9571	0.0543	0.3629	0.5829	0.7271283
159255	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2242857
142184	7137	5045	0.0284	0.0934	0.7590	1.0000	0.8936	0.0056	0.8942	0.1005	1.8050425
78179	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3123636
157641	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.5865874
24273	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1765060
204275	3254	2785	0.1000	0.0011	0.3368	1.0000	0.3088	1.0000	0.0000	0.0000	0.0628732
182350	3881	3169	0.2267	0.0032	0.6088	0.8252	0.9350	1.0104	0.0000	0.0000	0.5461247
27779	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6422222
63785	1666	1559	0.6317	0.0013	0.4865	0.9981	0.7774	1.0000	0.0000	0.0000	0.7579822

```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

```r
                                            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)
```

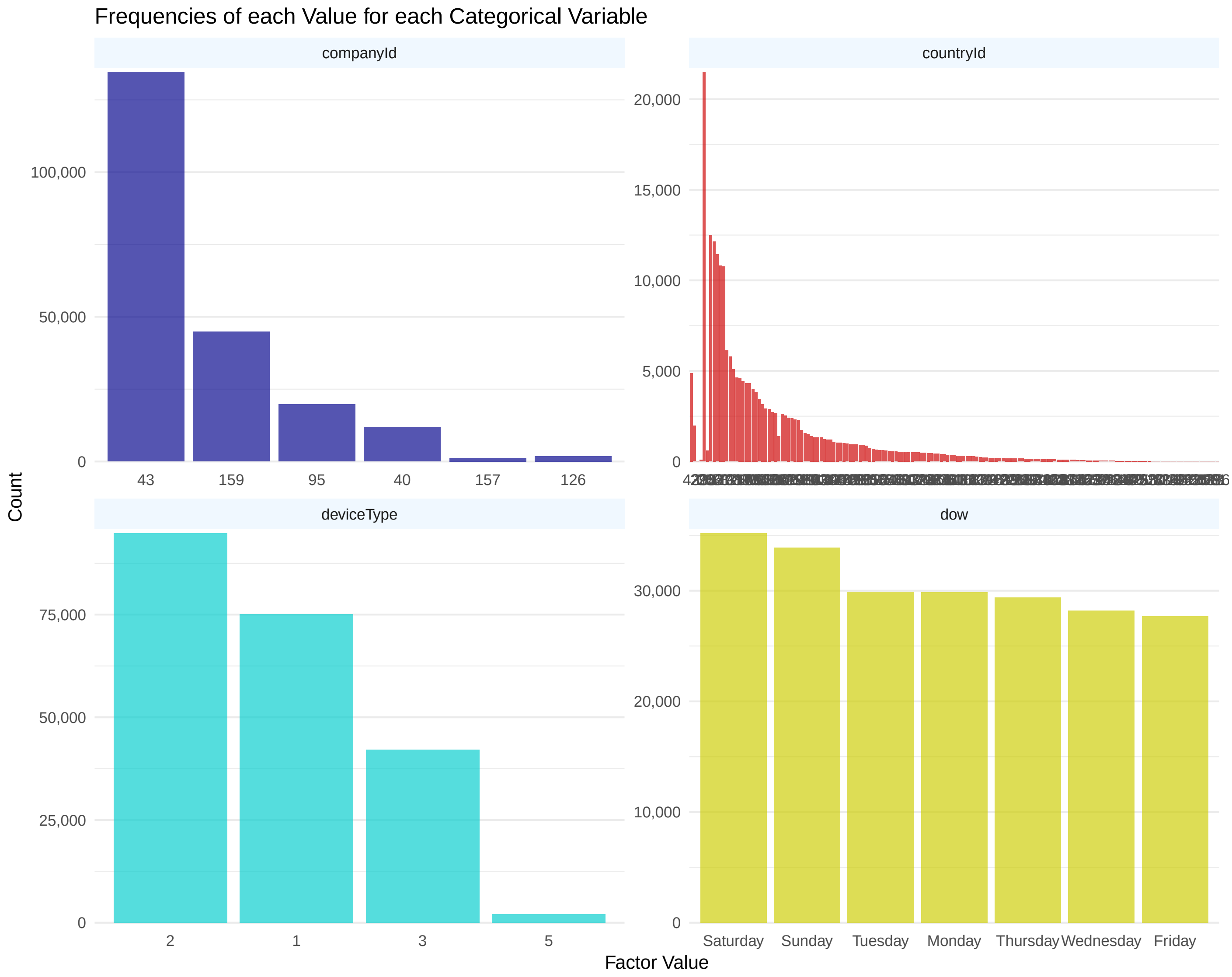
1.2.4.2 Categorical and Non-Numeric Features

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.

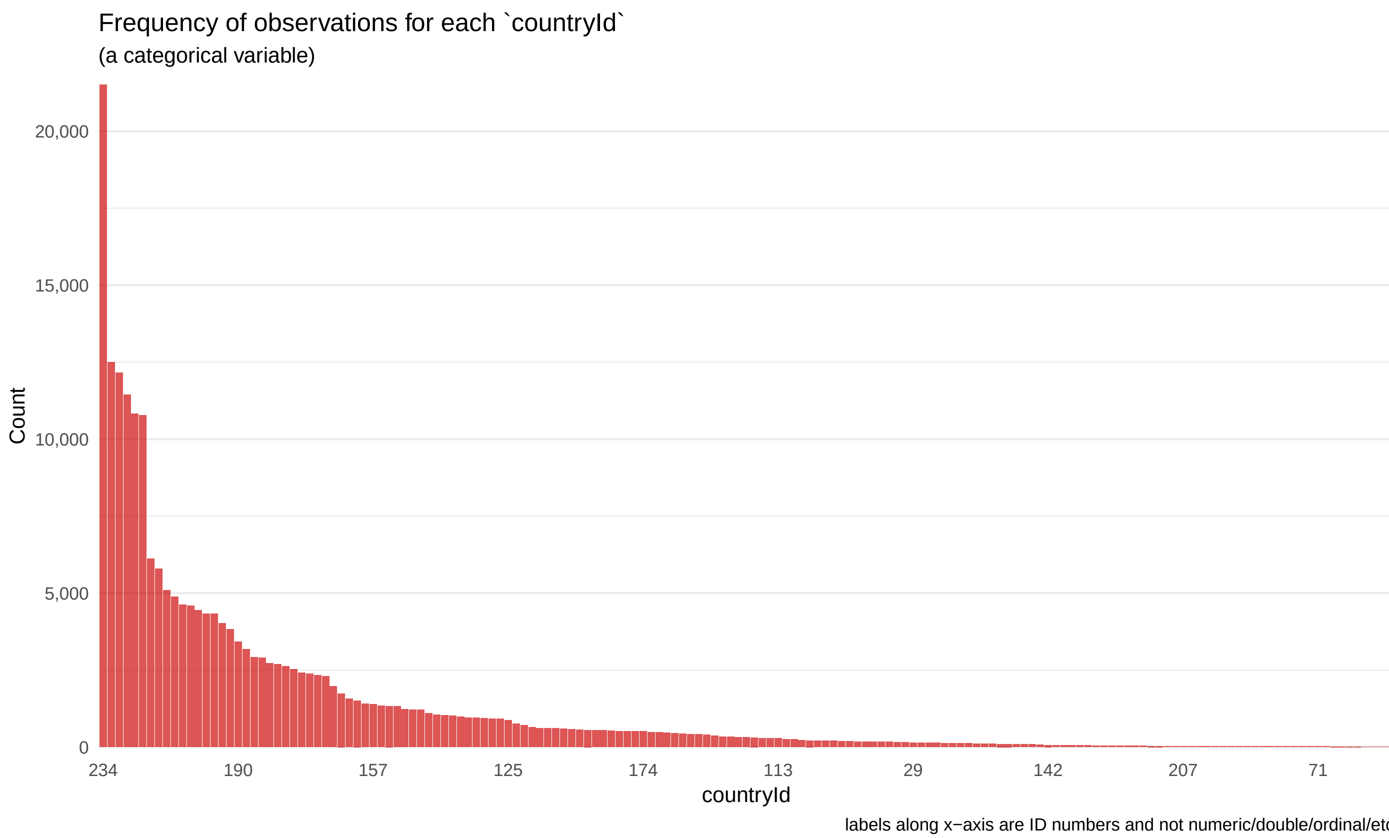
```r
advertising_train_long_cat <- select(advertising_train,
                                     countryId,
                                     companyId, deviceType,
                                     dow)

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))
```




```
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())
```

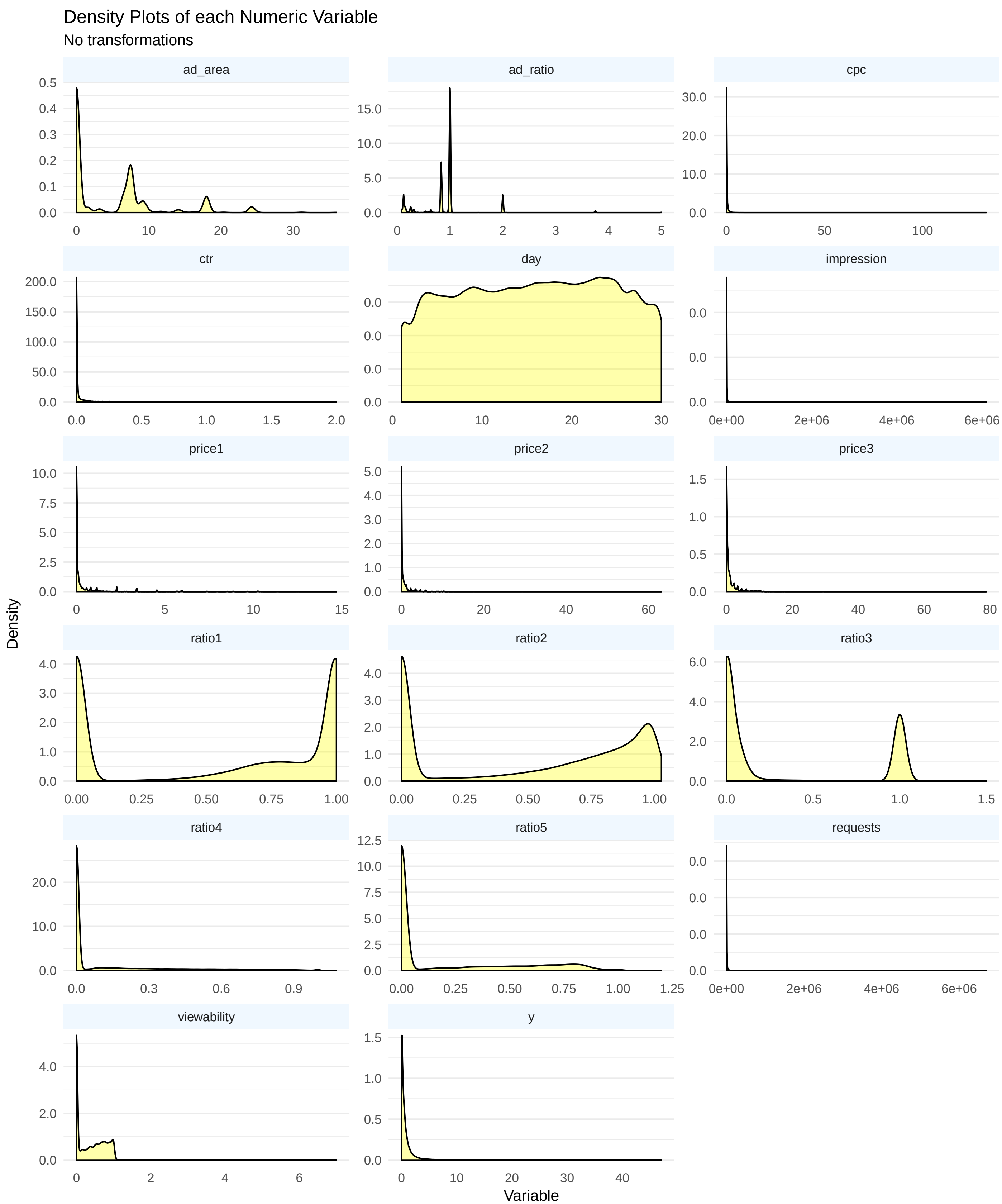


Frequency of observations for each `countryId`
(a categorical variable)
20,000
15,000
10,000
5,000
0
Count
234
190
157
125
174
113
29
142
207
71
countryId
labels along x−axis are ID numbers and not numeric/double/ordinal/etc

1.2.5 Univariate Plots

1.2.5.1 Numeric Variables

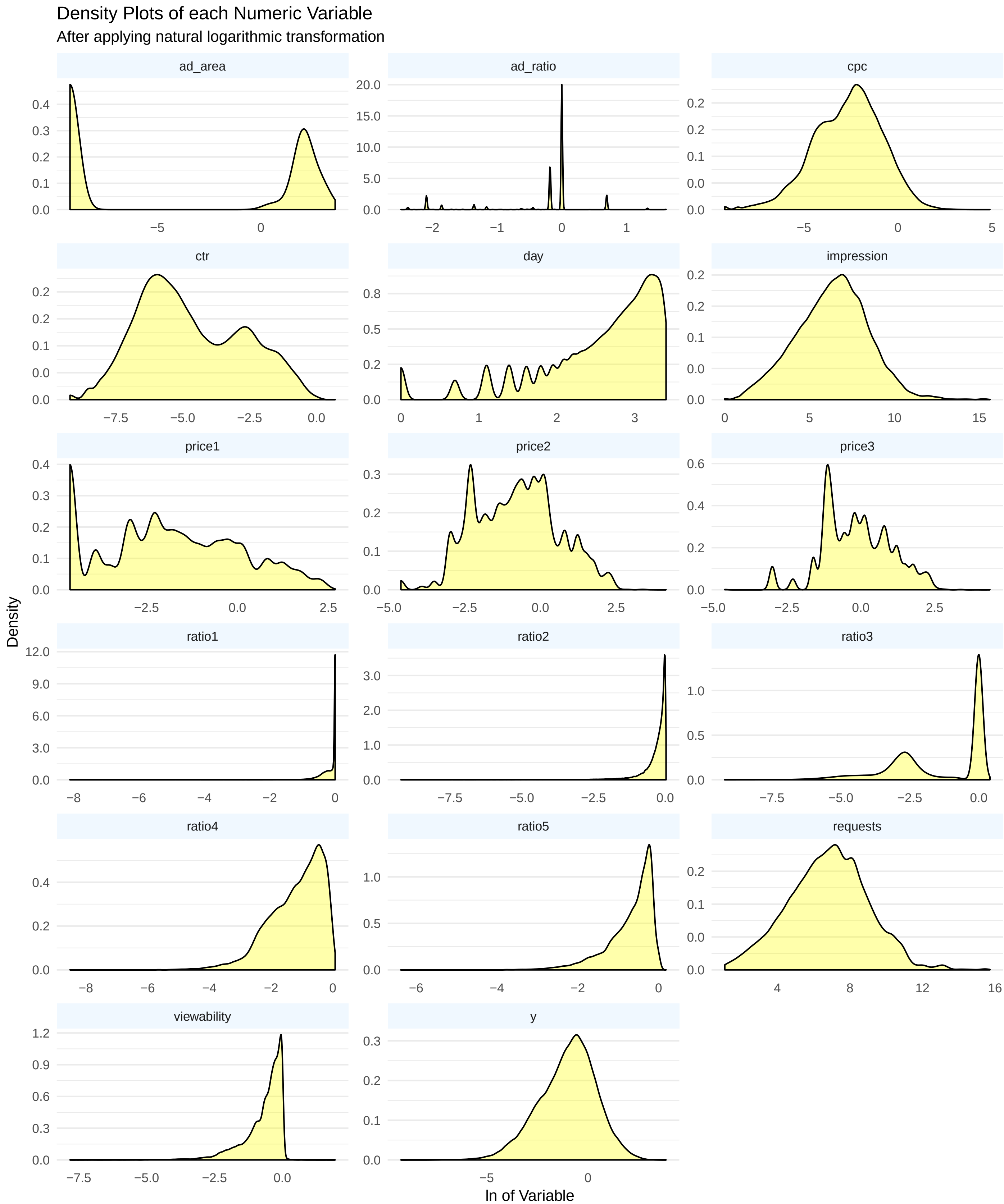
```
ggplot(advertising_train_long_num) +
  geom_density(aes(x = Value),
               fill = "yellow",
               alpha = 1/3) +
  facet_rep_wrap(~Variable,
                 repeat.tick.labels = T,
                 scales = "free",
                 ncol = 3) +
  scale_y_continuous(labels = comma_format(accuracy = 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
152090	43	189	3	22	Saturday	0.00	0.00	0.000	0.0001	1.000	0	0
195261	43	178	2	28	Friday	0.00	0.00	0.000	7.5000	0.833	1662	1396
22121	43	56	1	4	Tuesday	0.00	0.00	0.000	7.5000	0.833	0	0
48865	43	57	3	8	Saturday	0.00	0.00	0.000	7.5000	0.833	0	0
191639	43	166	1	27	Thursday	0.00	0.00	0.000	7.5000	0.833	431	431
67409	43	224	3	11	Tuesday	0.00	0.00	0.000	0.0001	1.000	33	33
169774	159	59	2	24	Monday	0.00	0.00	0.000	24.2500	0.258	0	0
3282	43	57	1	1	Saturday	0.01	0.06	0.297	7.5000	0.833	61977	17820
43813	159	17	2	7	Friday	0.06	0.15	0.296	0.0001	1.000	3479	3054
142978	43	10	3	21	Friday	0.00	0.00	0.000	0.0001	1.000	19	18
154832	159	200	3	22	Saturday	0.00	0.00	0.000	0.0001	1.000	0	0
193779	43	234	1	27	Thursday	0.05	0.20	0.408	7.5000	0.833	458636	167771
51741	40	153	1	8	Saturday	0.10	0.10	0.200	0.0001	1.000	0	0
114172	43	234	2	17	Monday	0.00	0.00	0.000	0.0001	1.000	206	206
134887	43	43	1	20	Thursday	0.01	0.18	0.376	7.5000	0.833	0	0
122349	43	57	2	18	Tuesday	3.43	3.43	3.432	0.0001	1.000	0	0
8111	43	202	2	2	Sunday	0.00	0.00	0.000	31.1850	0.318	643	643
82317	95	171	2	13	Thursday	0.44	1.34	2.670	7.5000	0.833	0	0
143867	159	167	1	21	Friday	0.00	0.00	0.000	0.0001	1.000	0	0
103279	95	38	3	16	Sunday	0.10	0.36	0.720	7.5000	0.833	1043	967

```
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) +
   geom_density(aes(x = ln_cpc),
```

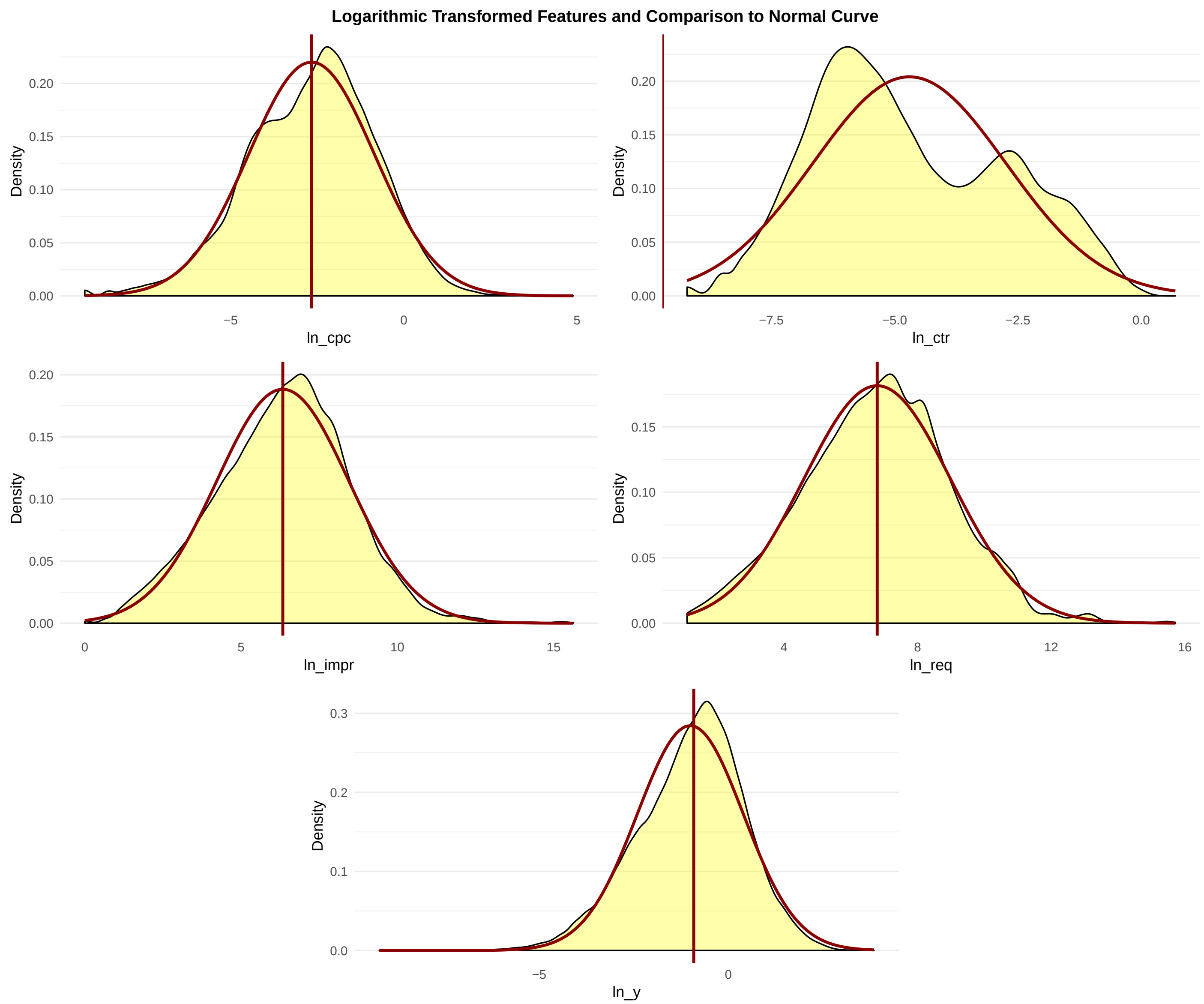
Table 5: Sample of advertising_train Data Frame After Logarithmic Transformations (cont)

case_id	cpc	ctr	viewability	ratio1	ratio2	ratio3	ratio4	ratio5	y	ln_cpc	ln_ctr	ln_impr	ln_req	ln_y
152090	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	0.5378	-Inf	-Inf	-Inf	-Inf	-0.6203
195261	0.0510	0.0064	0.6348	1.000	0.950	1.0000	0.0000	0.000	0.5869	-2.9759	-5.05	7.24	7.42	-0.5329
22121	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	0.1740	-Inf	-Inf	-Inf	-Inf	-1.7488
48865	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	2.6056	-Inf	-Inf	-Inf	-Inf	0.9576
191639	1.0575	0.0023	0.6969	1.000	0.513	0.5360	0.0209	0.443	2.4318	0.0559	-6.07	6.07	6.07	0.8886
67409	0.0170	0.0606	0.8333	1.000	0.909	0.0000	0.5455	0.455	1.2442	-4.0745	-2.80	3.50	3.50	0.2185
169774	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	2.0500	-Inf	-Inf	-Inf	-Inf	0.7178
3282	0.2631	0.0011	0.1300	0.849	0.537	0.0457	0.1703	0.784	0.0811	-1.3352	-6.81	9.79	11.03	-2.5127
43813	0.0563	0.0036	0.7796	0.899	0.993	1.0000	0.0000	0.000	0.1417	-2.8771	-5.63	8.02	8.15	-1.9537
142978	0.0038	0.1667	1.0000	1.000	0.889	0.0000	0.0000	0.944	1.0500	-5.5728	-1.79	2.89	2.94	0.0488
154832	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	0.1504	-Inf	-Inf	-Inf	-Inf	-1.8943
193779	0.3929	0.0011	0.1829	0.864	0.334	0.0492	0.6442	0.307	0.1523	-0.9342	-6.81	12.03	13.04	-1.8817
51741	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	0.0882	-Inf	-Inf	-Inf	-Inf	-2.4282
114172	0.0773	0.1602	0.9207	1.000	0.403	1.0000	0.0000	0.000	13.8333	-2.5601	-1.83	5.33	5.33	2.6271
134887	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	1.5000	-Inf	-Inf	-Inf	-Inf	0.4055
122349	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	4.4143	-Inf	-Inf	-Inf	-Inf	1.4848
8111	0.1052	0.0062	0.0998	1.000	0.863	1.0000	0.0000	0.000	0.7780	-2.2519	-5.08	6.47	6.47	-0.2510
82317	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	0.8291	-Inf	-Inf	-Inf	-Inf	-0.1874
143867	0.0000	0.0000	0.0000	0.000	0.000	0.0000	0.0000	0.000	0.2250	-Inf	-Inf	-Inf	-Inf	-1.4917
103279	0.9821	0.0010	0.3670	0.904	0.750	0.0000	0.8283	0.172	0.8022	-0.0181	-6.91	6.87	6.95	-0.2205

```r
                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_cpc$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))
```



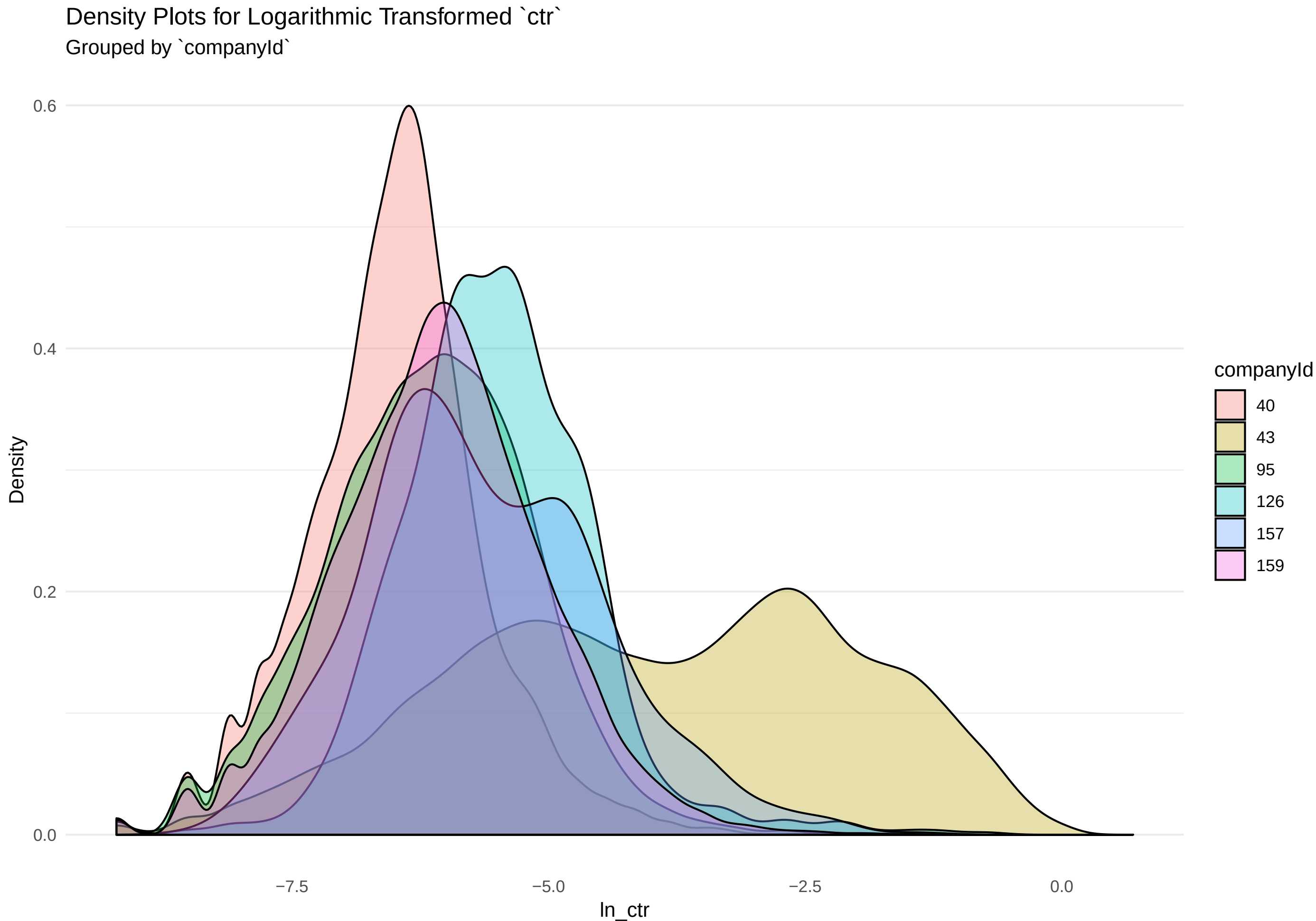
The natural logarithmic transformations of `impression` and `requests` clearly approached a normal distribution. The transformed `y` target feature somewhat resembled a normal distribution, albeit less closely as compared to `impression`. Both `cpc` and `ctr` appeared to be bimodal distributions after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

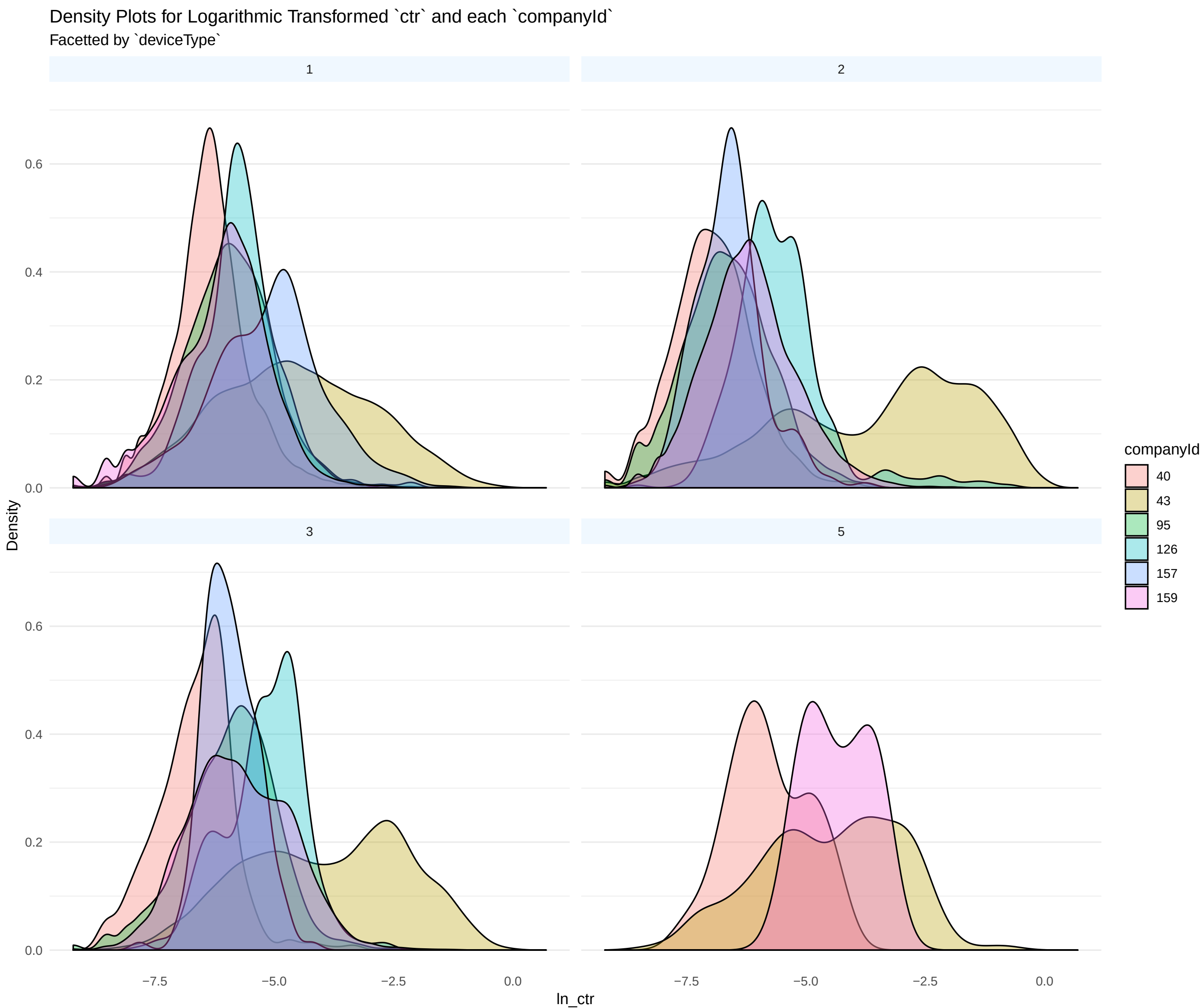


Producing separate density plots for each level within `deviceType` suggested some trivariate relationship between `ln_ctr`, `companyId`, and `deviceType`. The effect of facetting by `deviceType` was particularly apparent when examining `companyId` == 43, yet it still did not yield Gaussian distributions.

```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))
```

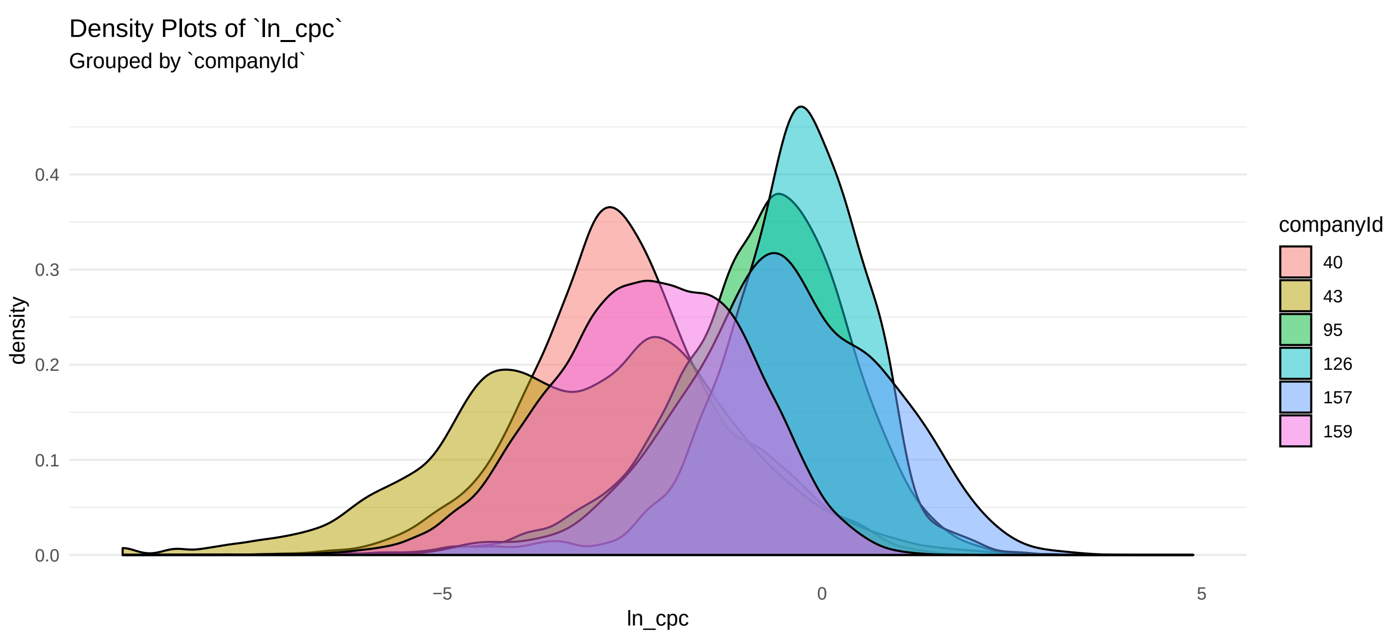
```
## Warning: Removed 78957 rows containing non-finite values (stat_density).
```



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`.

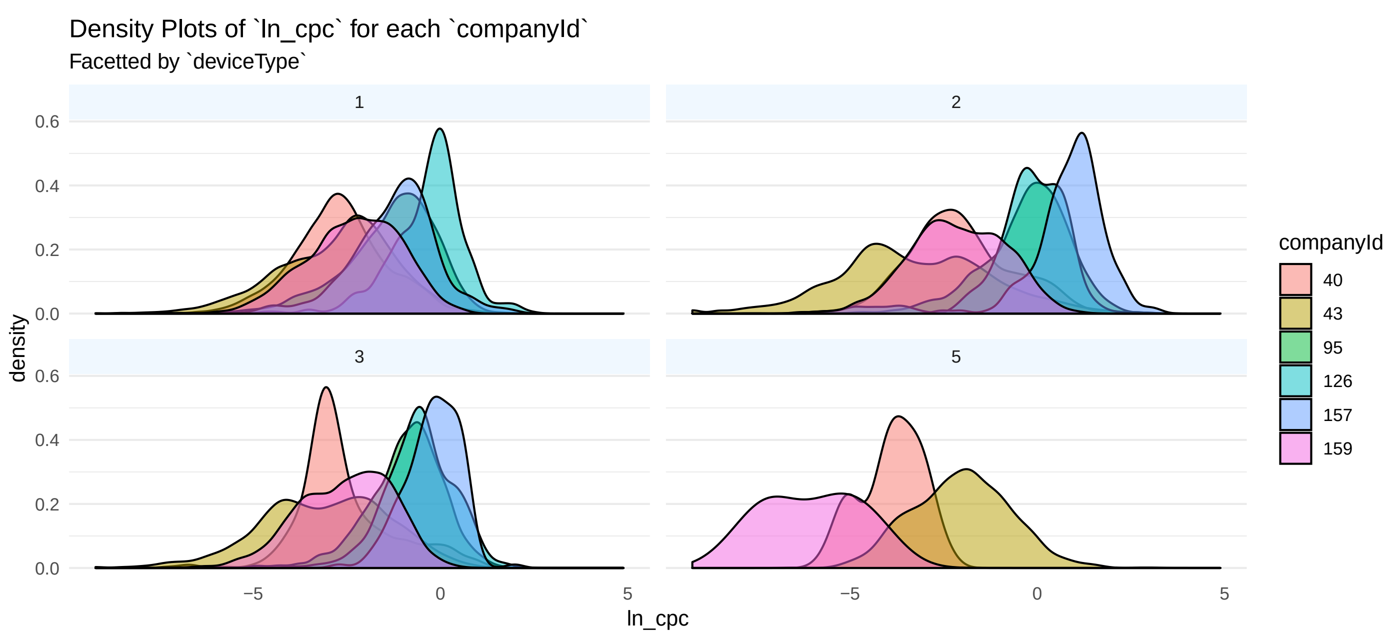
```
ggplot(advertising_train) +
   geom_density(aes(x = ln_cpc, fill = companyId),
                alpha = 1/2) +
   labs(title = "Density Plots of \`ln_cpc\`",
        subtitle = "Grouped by \`companyId\`",
        ylab = "Density") +
   theme_minimal() +
   theme(panel.grid.major.x = element_blank(),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

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




```r
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).
```

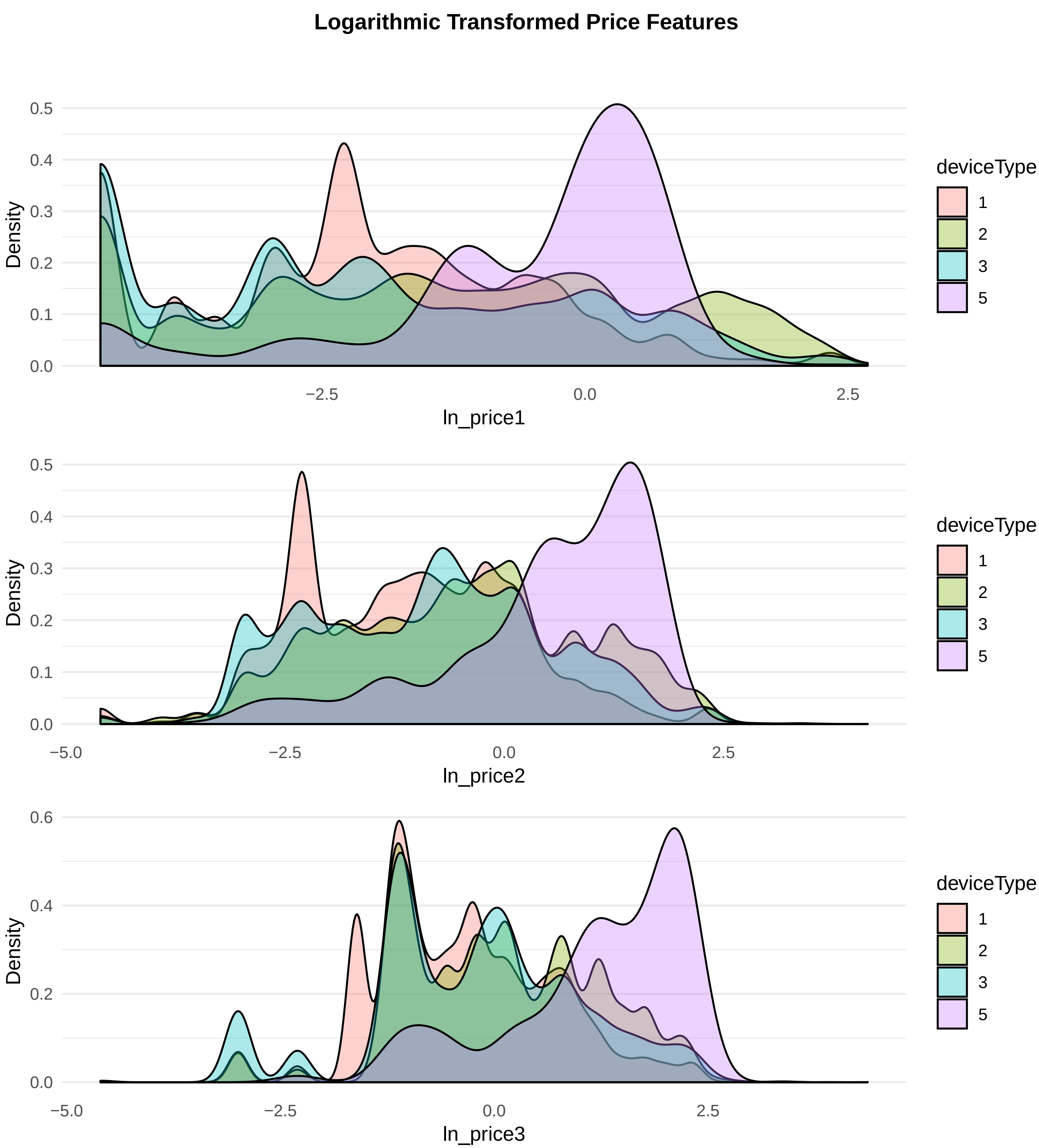


Each of the pricing features, (`price1`, `price2`, `price3`) were not suitably transformed by either logarithmic, square root, or cube root. Logarithmic transformations appeared to spread the data the most, but these transformations considerably diverged from a symmetrical normal distribution. Further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

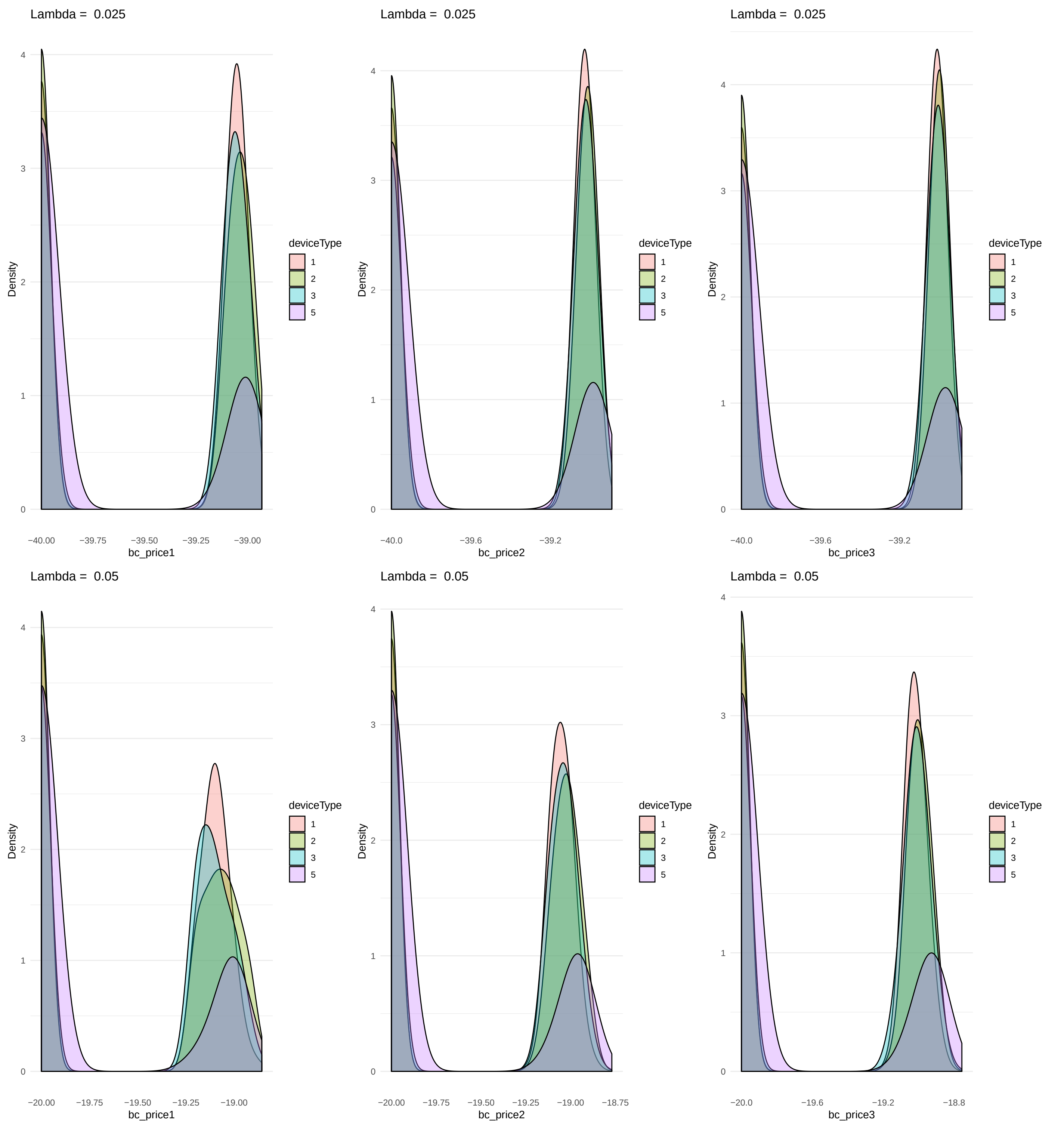
```
## Warning: Removed 92804 rows containing non-finite values (stat_density).

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

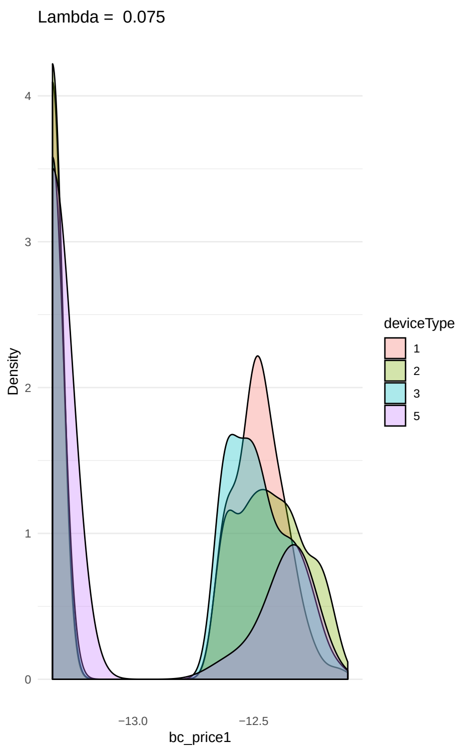


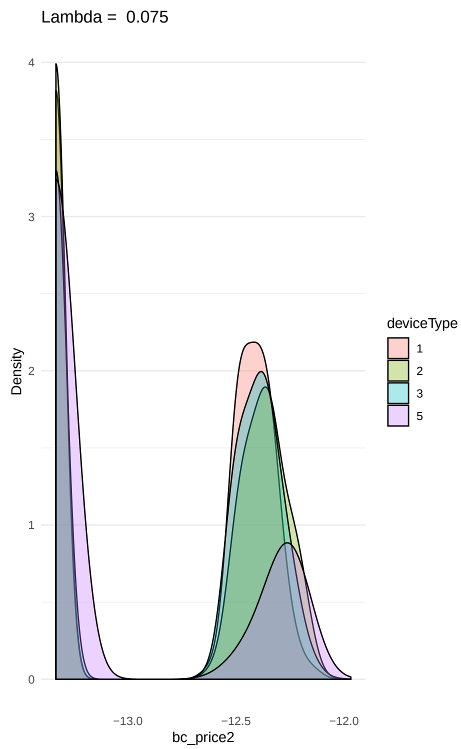
Box-Cox transformations with a range of lamda values also did not convert the price features into distributions that resembled a normal curve.

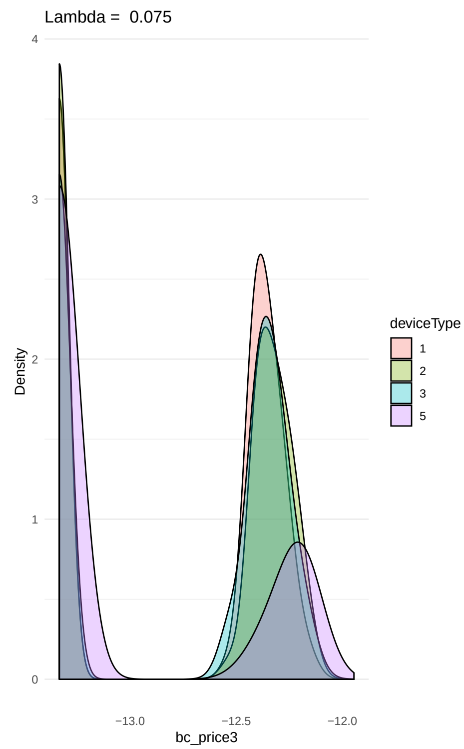
```
boxcox <- function(x, lambda = 1) {

   (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))
```

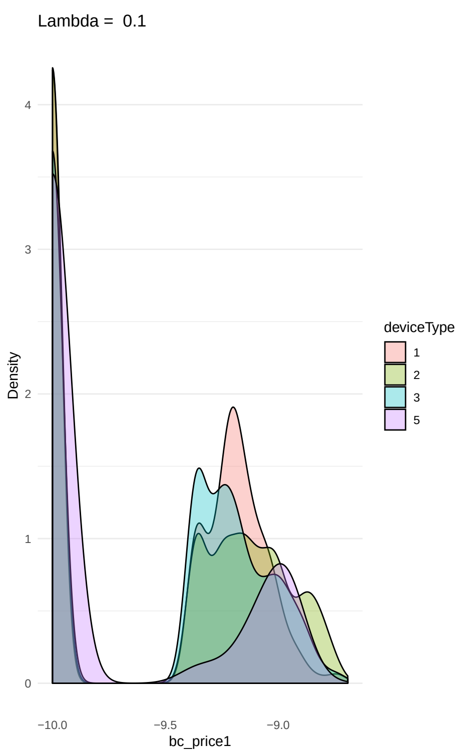
```
}
```

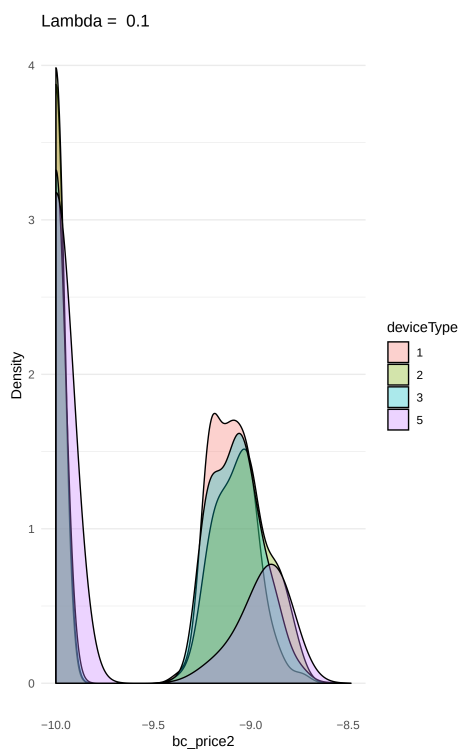

Lambda = 0.025
Lambda = 0.025
Lambda = 0.025
Density
bc_price1
bc_price2
bc_price3
deviceType
1
2
3
5
−40.00
−39.75
−39.50
−39.25
−39.00
−40.0
−39.6
−39.2
Lambda = 0.05
Lambda = 0.05
Lambda = 0.05
−20.00
−19.75
−19.50
−19.25
−19.00
−18.75
−20.0
−19.6
−19.2
−18.8
0
1
2
3
4

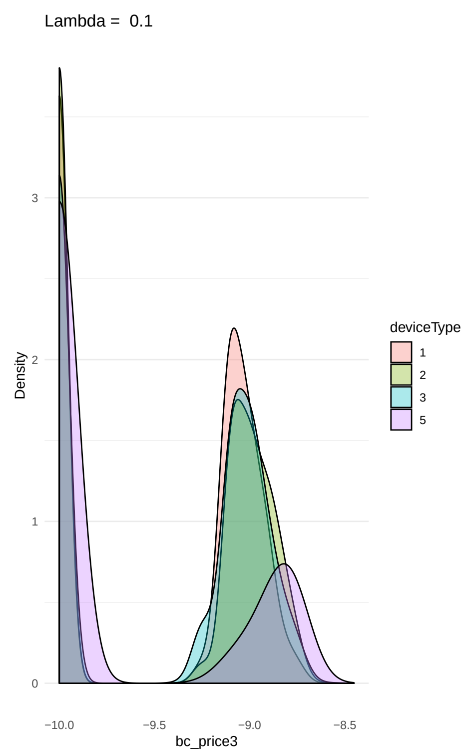

Lambda = 0.075
Density
bc_price1
−13.0
−12.5
deviceType
1
2
3
5

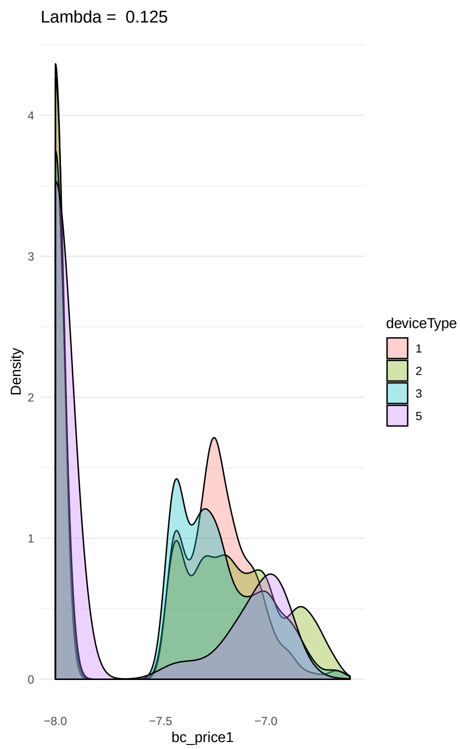

Lambda = 0.075
Density
bc_price2
−13.0
−12.5
−12.0
deviceType
1
2
3
5

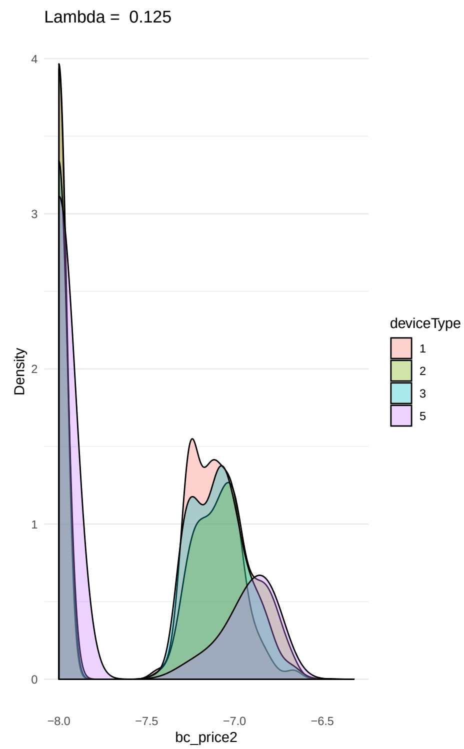

Lambda = 0.075
Density
bc_price3
−13.0
−12.5
−12.0
deviceType
1
2
3
5

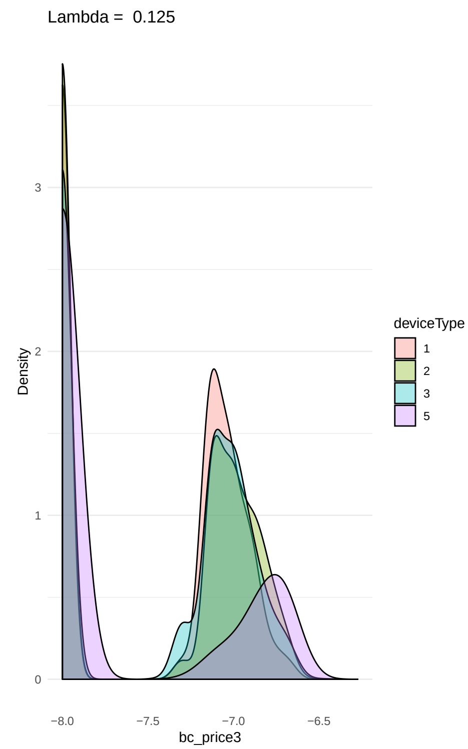

Lambda = 0.1
Density
bc_price1
−10.0
−9.5
−9.0
deviceType
1
2
3
5

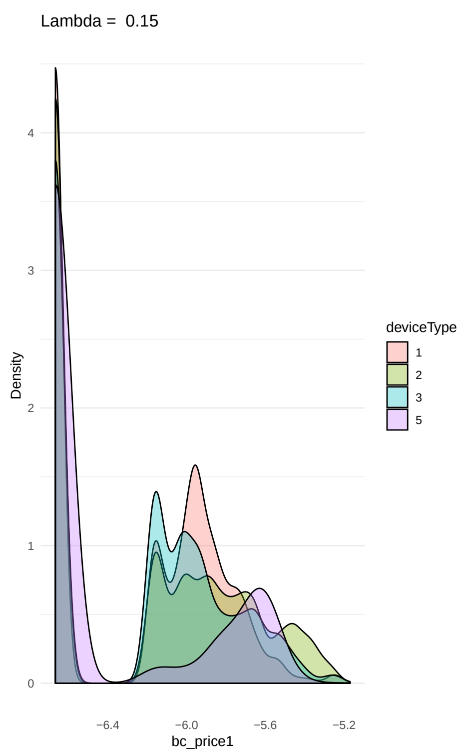

Lambda = 0.1
Density
bc_price2
−10.0
−9.5
−9.0
−8.5
deviceType
1
2
3
5

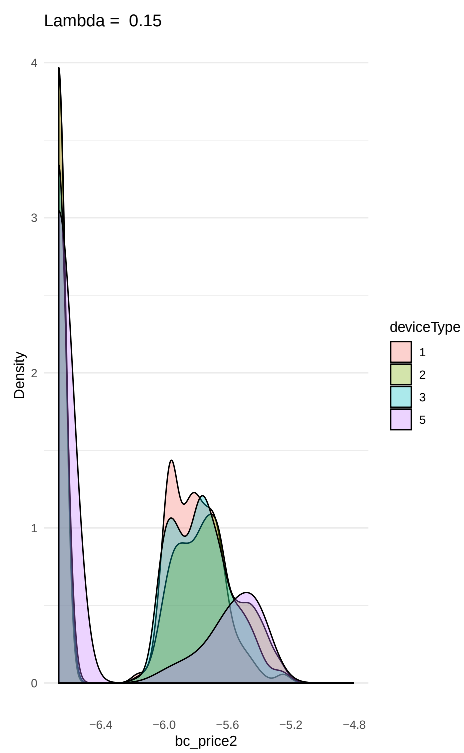

Lambda = 0.1
Density
bc_price3
−10.0
−9.5
−9.0
−8.5
deviceType
1
2
3
5

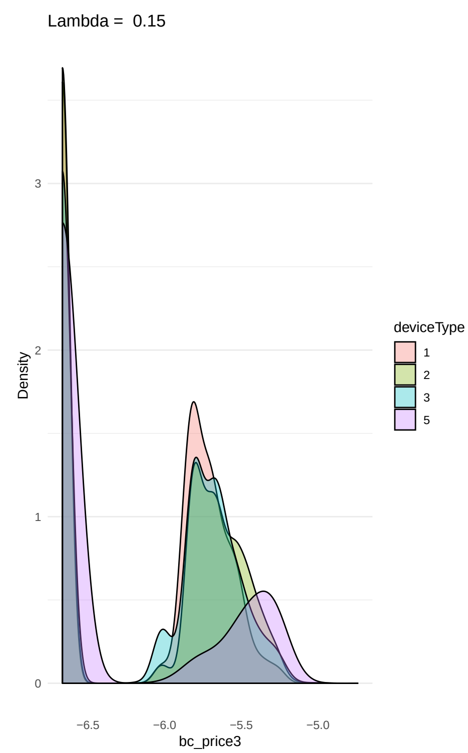

Lambda = 0.125
Density
4
3
2
1
0
−8.0
−7.5
−7.0
bc_price1
deviceType
1
2
3
5

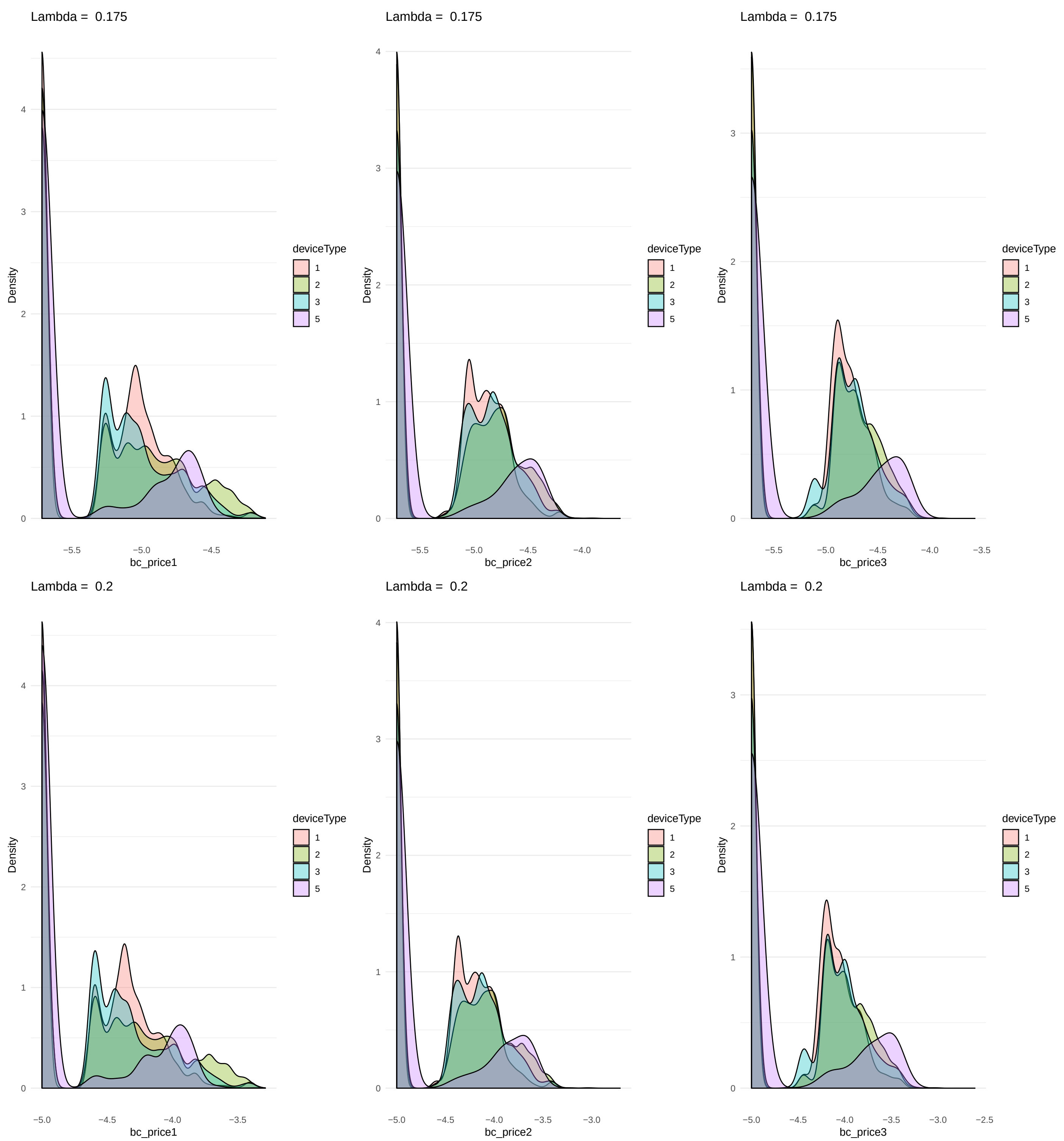

Lambda = 0.125
Density
4
3
2
1
0
−8.0
−7.5
−7.0
−6.5
bc_price2
deviceType
1
2
3
5

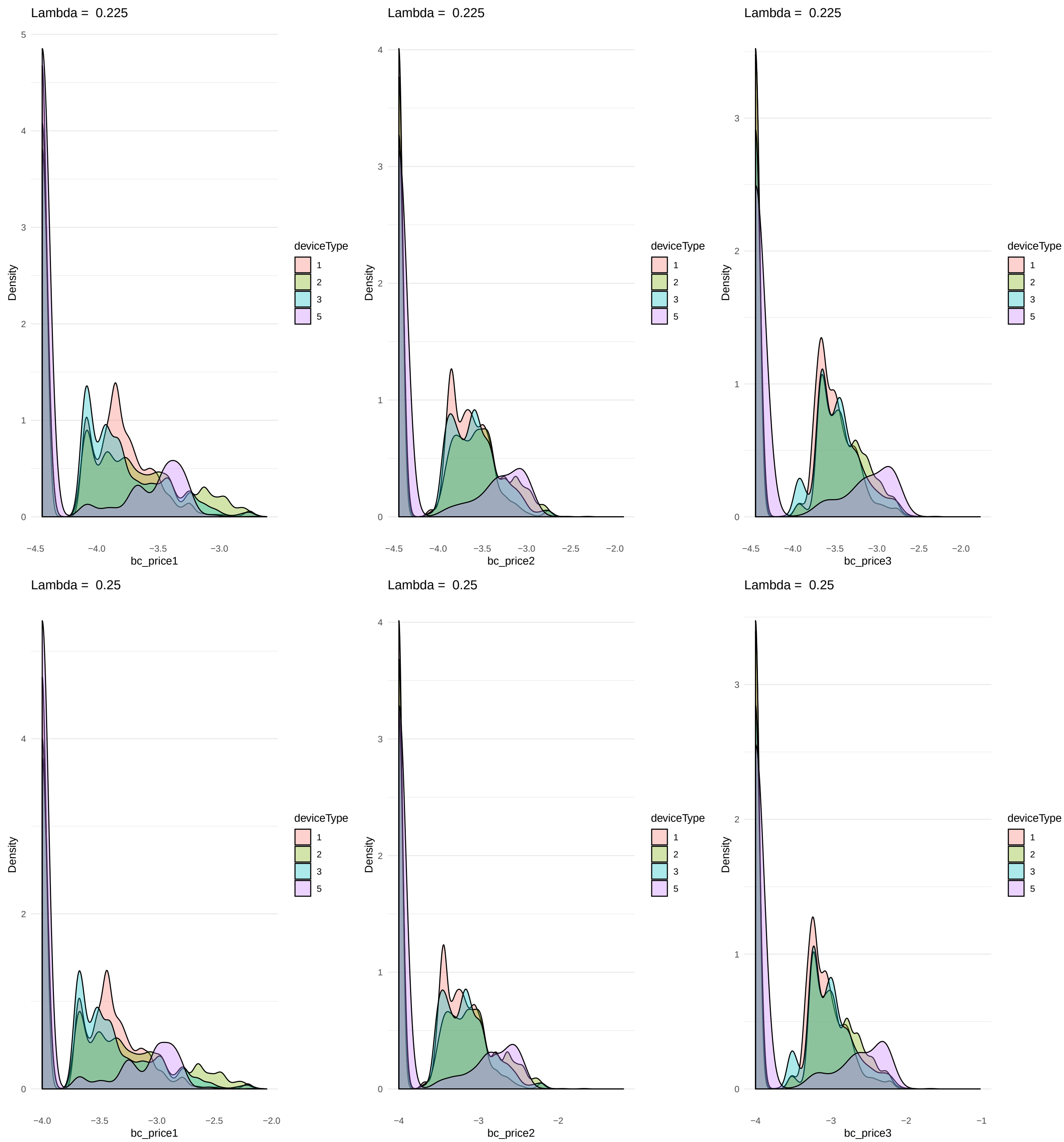

Lambda = 0.125
Density
3
2
1
0
−8.0
−7.5
−7.0
−6.5
bc_price3
deviceType
1
2
3
5

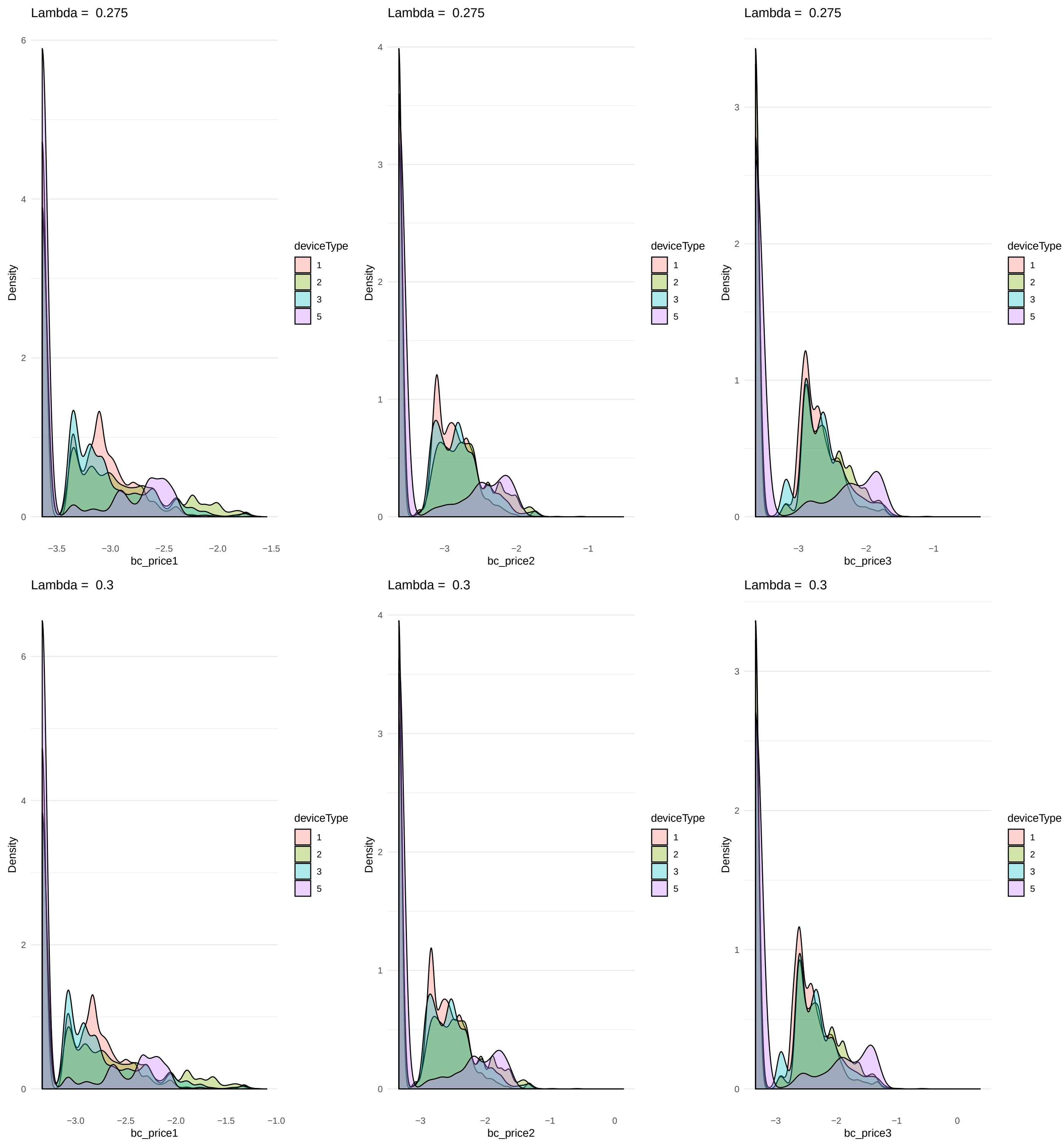

Lambda = 0.15
Density
4
3
2
1
0
−6.4
−6.0
−5.6
−5.2
bc_price1
deviceType
1
2
3
5


Lambda = 0.15
Density
4
3
2
1
0
−6.4
−6.0
−5.6
−5.2
−4.8
bc_price2
deviceType
1
2
3
5


Lambda = 0.15
Density
3
2
1
0
−6.5
−6.0
−5.5
−5.0
bc_price3
deviceType
1
2
3
5

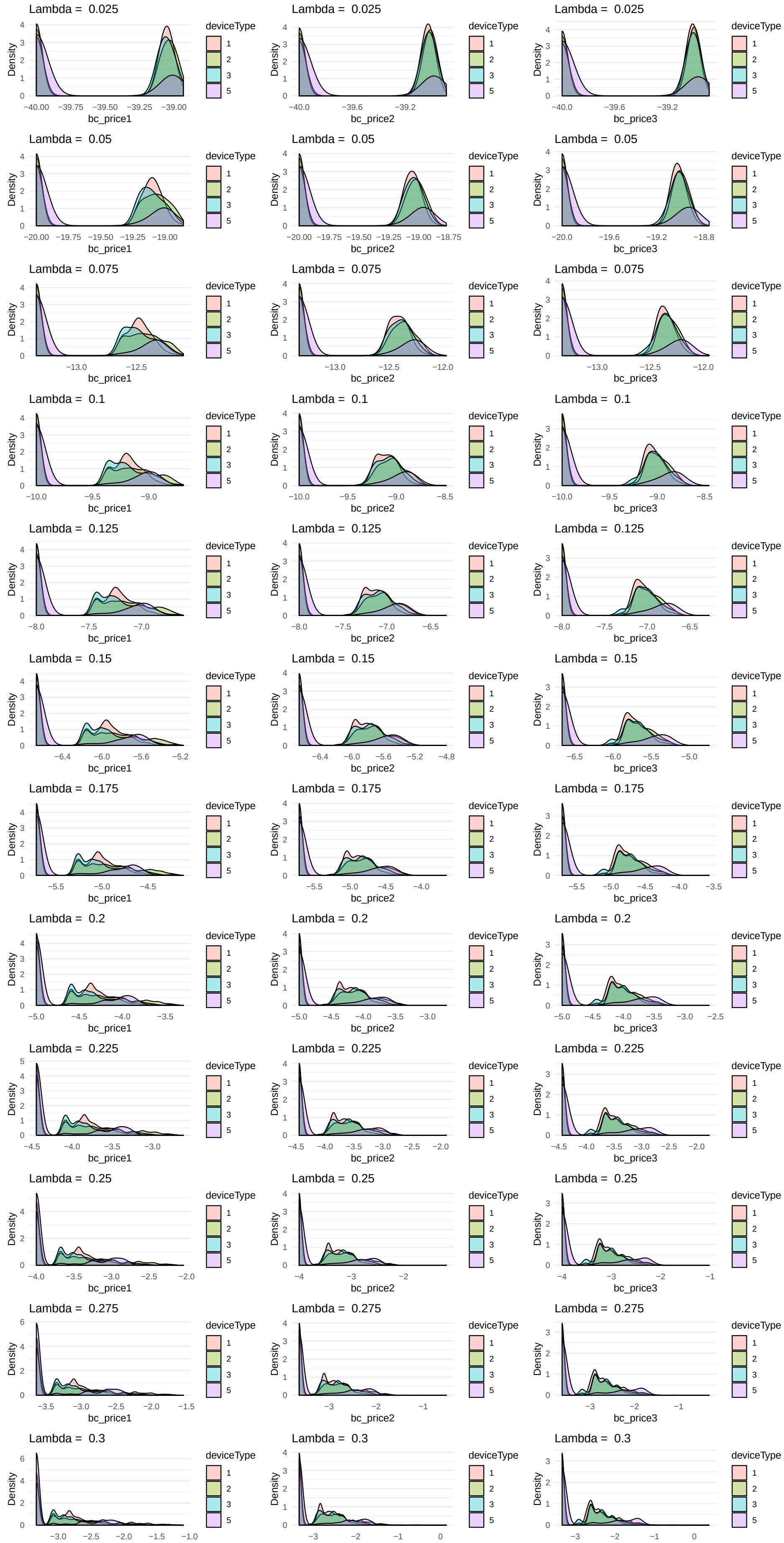

Lambda = 0.175
Lambda = 0.175
Lambda = 0.175
Lambda = 0.2
Lambda = 0.2
Lambda = 0.2
Density
deviceType
1
2
3
5
bc_price1
bc_price2
bc_price3


Lambda = 0.225
Density
bc_price1
deviceType
1
2
3
5
Lambda = 0.225
Density
bc_price2
deviceType
1
2
3
5
Lambda = 0.225
Density
bc_price3
deviceType
1
2
3
5
Lambda = 0.25
Density
bc_price1
deviceType
1
2
3
5
Lambda = 0.25
Density
bc_price2
deviceType
1
2
3
5
Lambda = 0.25
Density
bc_price3
deviceType
1
2
3
5




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

Box−Cox Transformations for Each `price` Feature at Changing lamda Values



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 should 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
192239	43	135	1	27	Thursday	0.79	0.79	0.7928	0.0001	1.00000	38	11	0.0101	0.0909
133464	43	139	3	19	Wednesday	0.57	0.57	0.5698	0.0001	1.00000	0	0	0.0000	0.0000
147536	43	103	2	21	Friday	5.93	5.93	5.9311	0.0001	1.00000	1360	6	0.0084	0.3333
69112	43	114	2	11	Tuesday	0.11	0.19	0.3824	7.5000	0.83333	51176	14871	3.0370	0.0002
22538	43	191	1	4	Tuesday	0.11	0.31	0.6040	7.5000	0.83333	9095	2114	0.1473	0.0014
26691	43	202	3	5	Wednesday	0.00	0.00	0.0000	11.7000	0.53419	20	20	0.0122	0.1000
74954	43	167	3	12	Wednesday	0.00	0.00	0.0000	0.0001	1.00000	267	247	0.0137	0.2024
85807	43	171	3	13	Thursday	0.00	0.00	0.0000	0.0001	1.00000	151	33	0.0710	0.0303
160703	43	198	1	23	Sunday	0.00	0.00	0.0000	7.5000	0.83333	0	0	0.0000	0.0000
20179	95	38	2	4	Tuesday	0.05	0.05	0.0500	7.5000	0.83333	215	181	0.1338	0.0055
164138	159	12	3	23	Sunday	0.10	0.37	0.7601	0.0001	1.00000	672	670	0.2905	0.0090
200153	43	234	2	28	Friday	2.25	3.37	6.7380	8.7300	0.09278	6234	3918	0.4528	0.0097
81597	43	191	1	13	Thursday	0.00	0.00	0.0000	0.0001	1.00000	84	45	0.0041	0.0667
41190	43	43	1	7	Friday	0.73	1.46	2.9288	9.4080	0.83333	192	174	0.1516	0.0287
131614	43	77	2	19	Wednesday	0.00	0.00	0.0000	0.0001	1.00000	715	659	0.0738	0.0501
48918	43	56	3	8	Saturday	0.01	0.03	0.3061	9.6000	3.75000	0	0	0.0000	0.0000
76350	95	13	2	12	Wednesday	0.05	0.05	0.0500	18.0000	2.00000	0	0	0.0000	0.0000
177756	43	56	2	25	Tuesday	0.79	0.79	0.7942	0.0001	1.00000	6320	1724	0.0691	0.0464
121767	43	157	3	18	Tuesday	0.00	0.00	0.0000	0.0001	1.00000	0	0	0.0000	0.0000
164380	43	114	2	23	Sunday	8.85	8.85	8.8490	0.0001	1.00000	16872	68	0.0633	0.2059

```r
kable_styling(kable(sample_adv[, 1:floor(ncol(sample_adv)/3)],
                    caption = "Sample of advertising\\_train Data Frame with Normalised Numeric Feat
              font_size = 8.5, latex_options = c("striped"),
              full_width = F)
```

```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
              font_size = 8.5, latex_options = c("striped"),
              full_width = F)
```

```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
              font_size = 8.5, latex_options = c("striped"),
              full_width = F)
```

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
192239	0.9091	0.7273	0.0000	0.0000	1.0000	0.3555556	-4.5952199	-2.397995	2.397895	3.637586	-1.0340738	0.8977756	0.8965517
133464	0.0000	0.0000	0.0000	0.0000	0.0000	0.1333333	-Inf	-Inf	-Inf	-Inf	-2.0149030	0.6232890	0.6206897
147536	1.0000	1.0000	1.0000	0.0000	0.0000	0.0075472	-4.7795236	-1.098712	1.791759	7.215240	-4.8865826	0.6890070	0.6896552
69112	0.7051	0.8040	1.0025	0.0000	0.0000	0.1948742	1.1108702	-8.517193	9.607168	10.843026	-1.6354010	0.3227571	0.3448276
22538	0.5038	0.8917	0.1216	0.2398	0.6400	0.0609355	-1.9152840	-6.571283	7.656337	9.115480	-2.7979389	0.1052506	0.1034483
26691	1.0000	0.8500	0.1500	0.1500	0.7000	1.3000000	-4.4063193	-2.302585	2.995732	2.995732	0.2623643	0.1246457	0.1379310
74954	1.0000	1.0000	0.0000	0.9514	0.0486	3.6725581	-4.2903594	-1.597509	5.509388	5.587249	1.3008885	0.3500399	0.3793103
85807	1.0000	0.7273	0.0000	0.5152	0.4242	0.3176768	-2.6450754	-3.496608	3.496508	5.017280	-1.1467209	0.4007248	0.4137931
160703	0.0000	0.0000	0.0000	0.0000	0.0000	2.5248503	-Inf	-Inf	-Inf	-Inf	0.9261818	0.7504985	0.7586207
20179	1.0000	0.7072	1.0000	0.0000	0.0000	0.6302083	-2.0114091	-5.203007	5.198497	5.370638	-0.4617048	0.0942338	0.1034483
164138	0.9716	0.5522	0.0075	0.6104	0.3821	2.4675277	-1.2361517	-4.710531	6.507278	6.510258	0.9032167	0.7665404	0.7586207
200153	0.7940	0.9589	1.0000	0.0000	0.0000	2.6941992	-0.7923048	-4.635629	8.273337	8.737773	0.9911010	0.9347350	0.9310345
81597	1.0000	0.9556	0.0444	0.5333	0.4222	0.2803279	-5.4967683	-2.707550	3.806662	4.430817	-1.2717954	0.3810636	0.4137931
41190	0.8621	0.6724	0.0575	0.3793	0.5632	4.7503030	-1.8865098	-3.550858	5.159055	5.257495	1.5582084	0.1923578	0.2068966
131614	1.0000	0.5463	1.0030	0.0000	0.0000	4.2641654	-2.6063965	-2.993734	6.490723	6.572283	1.4502465	0.6146493	0.6206897
48918	0.0000	0.0000	0.0000	0.0000	0.0000	0.0993576	-Inf	-Inf	-Inf	-Inf	-2.3090298	0.2284485	0.2413793
76350	0.0000	0.0000	0.0000	0.0000	0.0000	0.7616822	-Inf	-Inf	-Inf	-Inf	-0.2722258	0.3565594	0.3793103
177756	0.9675	0.6984	1.0000	0.0000	0.0000	0.7572886	-2.6722005	-3.070456	7.452402	8.751475	-0.2780109	0.8301382	0.8275862
121767	0.0000	0.0000	0.0000	0.0000	0.0000	0.2177419	-Inf	-Inf	-Inf	-Inf	-1.5244447	0.5686625	0.5862069
164380	1.0000	0.1176	1.0000	0.0000	0.0000	0.0507430	-2.7598699	-1.580365	4.219508	9.733411	-2.9809825	0.7676706	0.7586207

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

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
192239	0.0000000	0.1864412	0.0000057	0.0000018	0.0000762	0.04545	0.1428571	0.9091	0.7081792	0.0000000
133464	0.0000000	0.1864412	0.0000000	0.0000000	0.0000000	0.00000	0.0000000	0.0000	0.0000000	0.0000000
147536	0.0000000	0.1864412	0.0002029	0.0000010	0.0000634	0.16665	0.1428571	1.0000	0.9737098	0.6666667
69112	0.2083311	0.1525423	0.0076360	0.0024377	0.0229149	0.00010	0.0648286	0.7051	0.7828627	0.6683333
22538	0.2083311	0.1525423	0.0013571	0.0003465	0.0011114	0.00070	0.0790429	0.5038	0.8682571	0.0810667
26691	0.3249981	0.0917003	0.0000030	0.0000033	0.0000921	0.05000	0.0827000	1.0000	0.8276534	0.1000000
74954	0.0000000	0.1864412	0.0000398	0.0000405	0.0001034	0.10120	0.1396857	1.0000	0.9737098	0.0000000
85807	0.0000000	0.1864412	0.0000225	0.0000054	0.0005357	0.01515	0.0756286	1.0000	0.7081792	0.0000000
160703	0.2083311	0.1525423	0.0000000	0.0000000	0.0000000	0.00000	0.0000000	0.0000	0.0000000	0.0000000
20179	0.2083311	0.1525423	0.0000321	0.0000297	0.0010096	0.00275	0.0811857	1.0000	0.6886076	0.6666667
164138	0.0000000	0.1864412	0.0001003	0.0001098	0.0021919	0.00450	0.1205857	0.9716	0.5376826	0.0050000
200153	0.2424979	0.0019220	0.0009302	0.0006423	0.0034165	0.00485	0.0700857	0.7940	0.9336904	0.6666667
81597	0.0000000	0.1864412	0.0000125	0.0000074	0.0000309	0.03335	0.0909143	1.0000	0.9304771	0.0296000
41190	0.2613313	0.1525423	0.0000286	0.0000285	0.0011439	0.01435	0.0348571	0.8621	0.6547225	0.0383333
131614	0.0000000	0.1864412	0.0001067	0.0001080	0.0005568	0.02505	0.1089571	1.0000	0.5319377	0.6686667
48918	0.2666646	0.7457629	0.0000000	0.0000000	0.0000000	0.00000	0.0000000	0.0000	0.0000000	0.0000000
76350	0.4999986	0.3898309	0.0000000	0.0000000	0.0000000	0.00000	0.0000000	0.0000	0.0000000	0.0000000
177756	0.0000000	0.1864412	0.0009430	0.0002826	0.0005214	0.02320	0.1196143	0.9675	0.6800389	0.6666667
121767	0.0000000	0.1864412	0.0000000	0.0000000	0.0000000	0.00000	0.0000000	0.0000	0.0000000	0.0000000
164380	0.0000000	0.1864412	0.0025175	0.0000111	0.0004776	0.10295	0.1428571	1.0000	0.1145083	0.6666667