Predicting Revenue from Search Engine Advertising Data

MATH2319 - Machine Learning
Course Project

Ben Cole - s3412349

Print Date: 08/06/2019

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",
                          "mlr")))    ## 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
80057	43	68	1	12	Wednesday	0.00	0.00	0.0000	0.0001	1.00000
119763	43	43	1	18	Tuesday	0.01	0.18	0.3775	7.5000	0.83333
152300	43	77	5	22	Saturday	0.00	0.00	0.0000	0.0001	1.00000
192283	43	56	3	27	Thursday	0.00	0.00	0.0000	7.5000	0.83333
148537	95	234	1	21	Friday	0.01	0.40	0.7900	1.6000	0.15625
66777	159	98	1	10	Monday	0.00	0.00	0.0000	7.5000	0.83333
146704	95	234	1	21	Friday	1.86	3.97	7.9300	7.5000	0.83333
11676	159	75	1	3	Monday	0.10	0.19	0.3763	0.0001	1.00000
211793	43	13	3	30	Sunday	0.00	0.00	0.0000	7.5000	0.83333
116685	159	17	3	17	Monday	0.00	0.00	0.0000	0.0001	1.00000
213501	43	234	1	30	Sunday	0.06	0.11	0.3509	2.8080	0.12821
111622	95	77	2	17	Monday	0.05	0.05	0.0500	7.5000	0.83333
69833	43	166	2	11	Tuesday	5.85	5.85	5.8482	0.0001	1.00000
201110	159	190	3	28	Friday	0.00	0.00	0.0000	0.0001	1.00000
155551	43	57	3	22	Saturday	0.18	0.37	0.7602	6.5520	0.12363
8356	43	202	1	2	Sunday	0.00	0.00	0.0000	9.0000	1.00000
43836	43	75	3	7	Friday	1.49	2.80	5.6069	7.5000	0.83333
29062	95	234	2	5	Wednesday	0.05	0.63	1.2600	7.5000	0.83333
121480	43	13	3	18	Tuesday	0.71	0.92	1.8418	7.5000	0.83333
44627	95	234	3	7	Friday	0.05	2.08	4.1500	18.0000	2.00000

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
80057	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	3.4866667
119763	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	2.4682927
152300	74	74	0.3893	0.0270	1.0000	1.0000	0.1486	1.0000	0.0000	0.0000	8.9928571
192283	255	255	0.0454	0.0235	0.8765	1.0000	0.6196	0.0000	0.2078	0.7882	1.2566667
148537	6172	4387	0.7107	0.0014	0.3080	0.8632	0.5936	0.0424	0.5553	0.4023	0.5974495
66777	433156	198504	0.0564	0.0049	0.6498	1.0000	0.9231	0.0614	0.1179	0.8205	0.1120485
146704	15299	3141	0.4030	0.0060	0.2644	0.6469	0.6587	0.0688	0.2996	0.6316	0.5977643
11676	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0825431
211793	241	151	0.0258	0.0132	0.6096	1.0000	0.9801	0.0000	0.9338	0.0662	0.1099174
116685	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0273556
213501	4751	3434	0.0867	0.0090	0.2556	0.8427	0.6462	0.2199	0.4715	0.3087	0.5596208
111622	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.2681034
69833	8256	119	0.0222	0.2353	1.0000	1.0000	0.3193	1.0336	0.0000	0.0000	0.0985474
201110	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0640000
155551	710	538	0.1144	0.0056	0.7140	0.7546	0.7175	0.0167	0.4963	0.4870	0.4890323
8356	20261	20170	0.1521	0.0044	0.4138	1.0000	0.7132	0.0646	0.2429	0.6925	0.4245755
43836	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3886010
29062	0	0	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.3426662
121480	11793	4438	1.3703	0.0020	0.8515	0.8274	0.5865	0.0059	0.9051	0.0895	1.2953341
44627	9910	5211	0.4961	0.0035	0.1346	0.6813	0.7540	0.0115	0.6402	0.3485	0.7812626

```
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.

```
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)
```

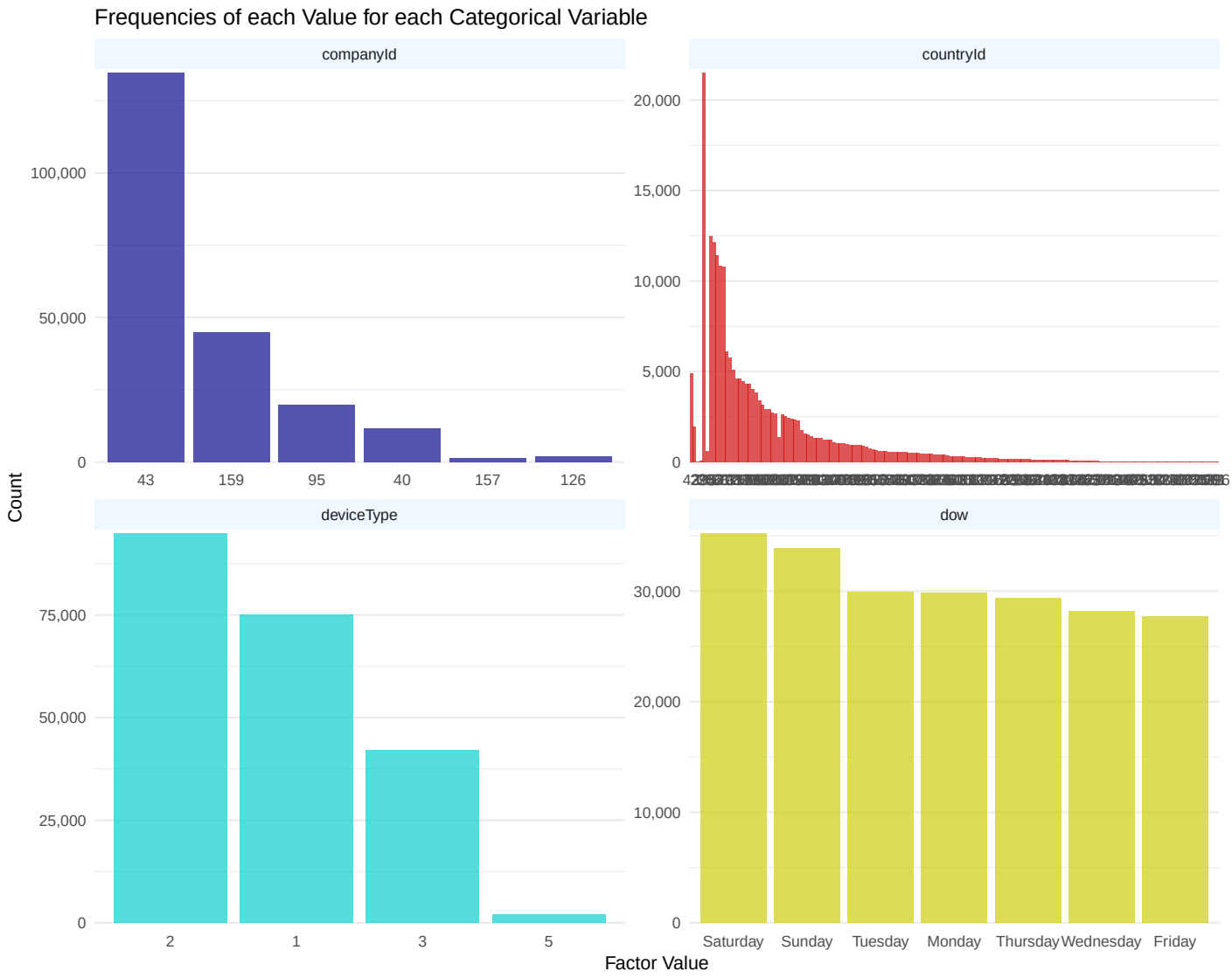
1.2.4.2 Categorical and Non-Numeric Features

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.

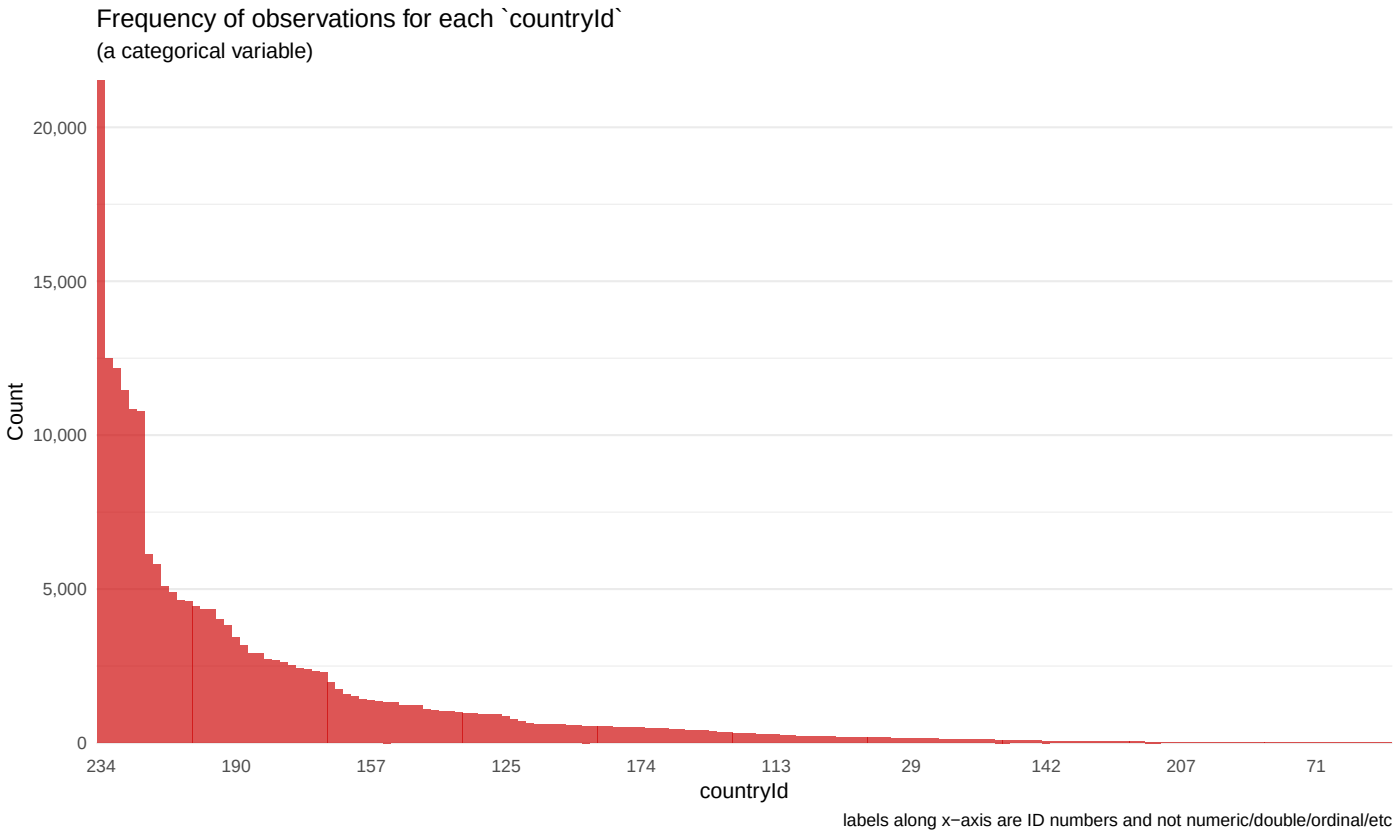
```
advertising_train_long_cat <- select(advertising_train,
                                     countryId,
                                     companyId, deviceType,
                                     dow)

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))
```



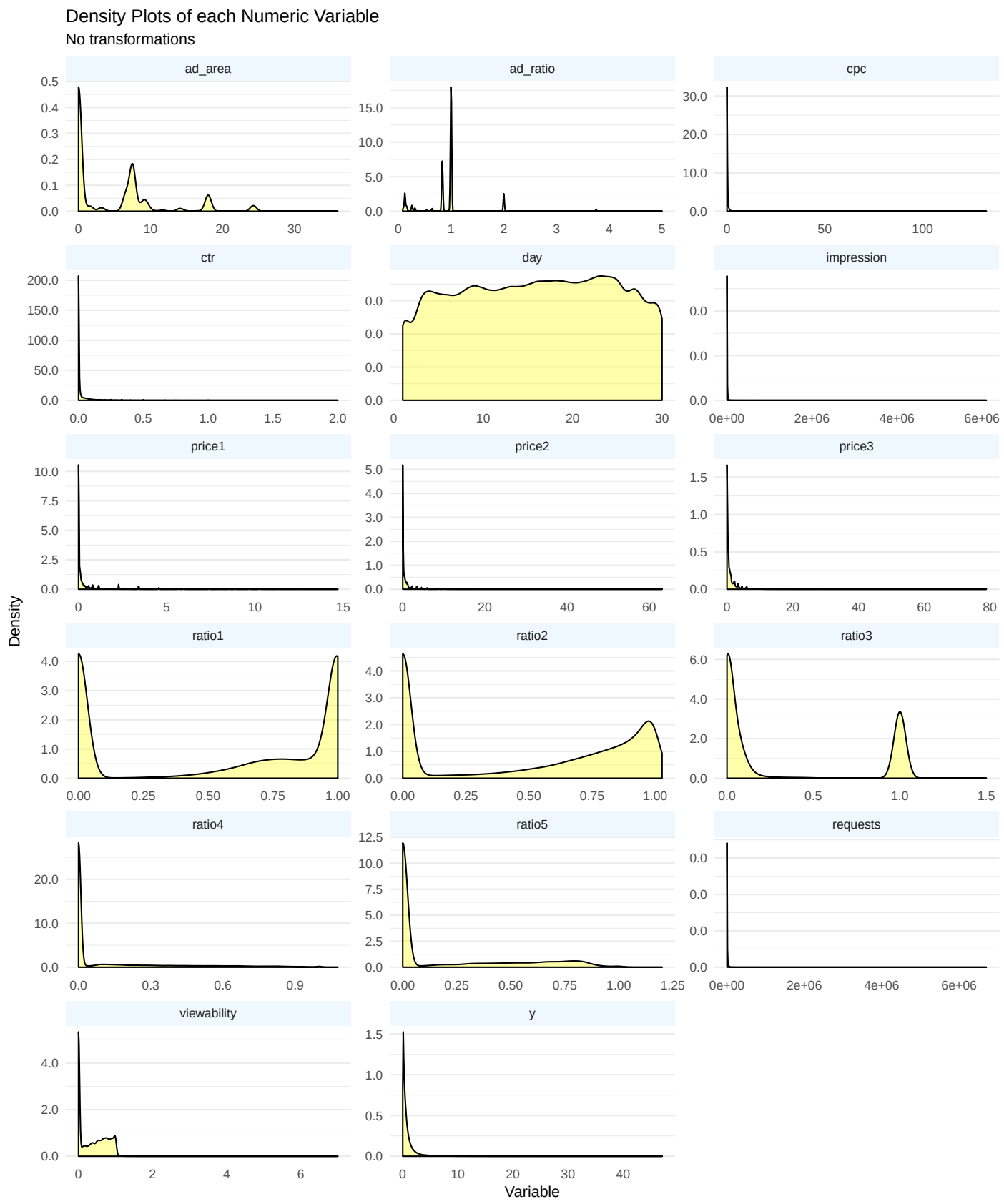

```
country_labels <- levels(fct_infreq(advertising_train$countryId))[c(seq(1,
                                                                        length(levels(fct_infreq(adv
                                                                        ceiling(length(levels(fct_ir


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

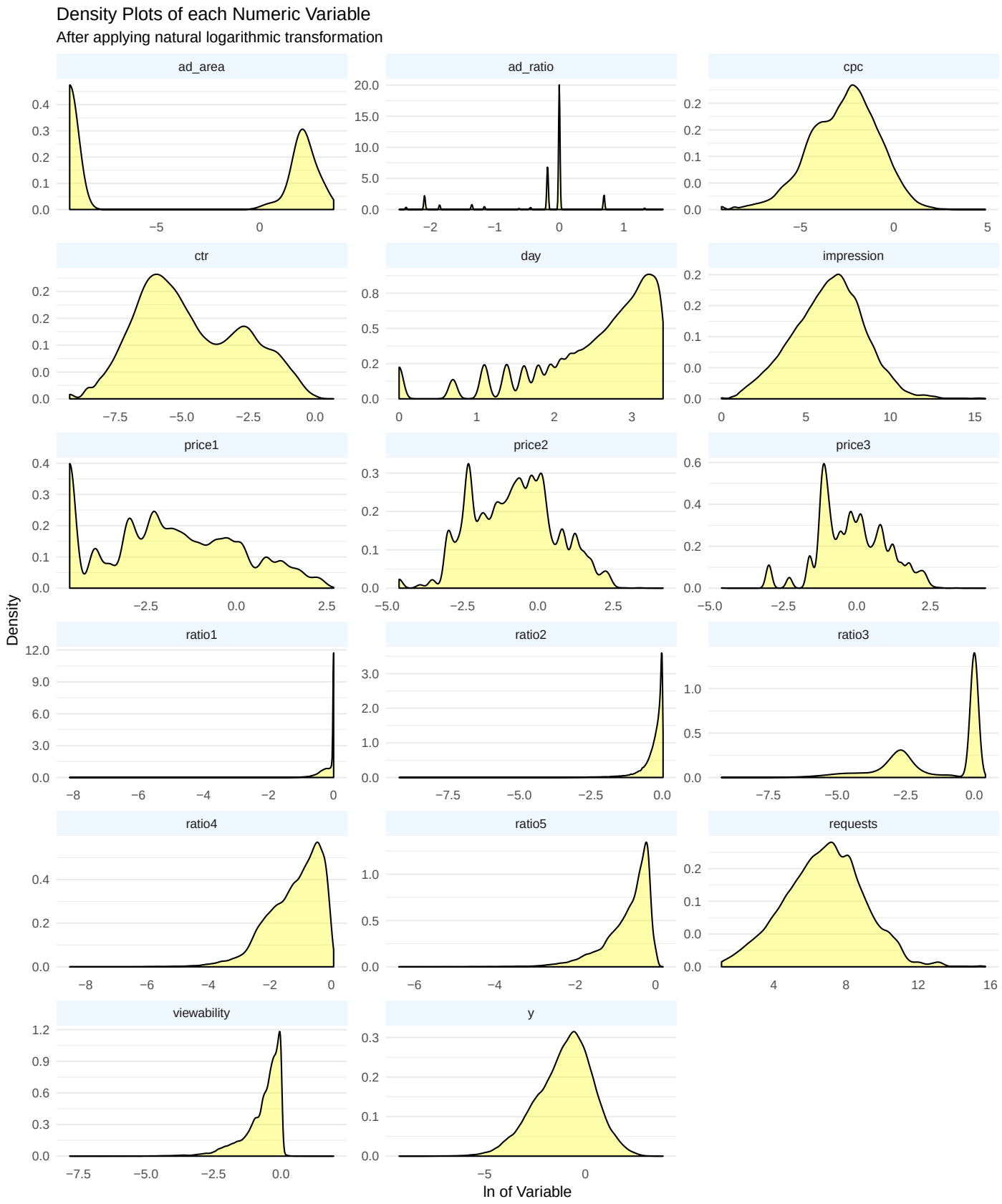
```
ggplot(advertising_train_long_num) +
   geom_density(aes(x = Value),
                fill = "yellow",
                alpha = 1/3) +
   facet_rep_wrap(~Variable,
                  repeat.tick.labels = T,
                  scales = "free",
                  ncol = 3) +
   scale_y_continuous(labels = 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))
```




```
ggplot(advertising_train_long_num) +
   geom_density(aes(x = log(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 = "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
4793	159	98	1	1	Saturday	0.00	0.00	0.000	0.0001	1.000	0	0
207441	159	57	2	29	Saturday	0.07	0.36	0.714	0.0001	1.000	7822	7016
27864	43	116	2	5	Wednesday	0.00	0.00	0.000	0.0001	1.000	0	0
103904	43	116	1	16	Sunday	0.06	0.27	0.547	0.0001	1.000	0	0
197928	43	22	2	28	Friday	0.00	0.00	0.000	0.0001	1.000	20	20
204185	159	200	2	29	Saturday	0.01	0.17	0.351	0.0001	1.000	1683	1644
46245	43	190	2	8	Saturday	0.79	0.79	0.794	0.0001	1.000	1318	72
34697	43	231	1	6	Thursday	0.00	0.00	0.000	0.0001	1.000	212	205
181205	40	95	2	26	Wednesday	0.00	0.00	0.000	0.0001	1.000	684	659
28964	40	107	1	5	Wednesday	0.00	0.00	0.000	0.0001	1.000	0	0
11896	43	226	1	3	Monday	0.03	0.08	0.296	7.5000	0.833	252	170
172059	95	234	2	24	Monday	0.01	0.10	0.290	7.5000	0.833	2641	1697
127999	159	200	2	19	Wednesday	0.00	0.00	0.000	0.0001	1.000	0	0
156782	40	22	2	22	Saturday	0.00	0.00	0.000	0.0001	1.000	1394	1140
211920	43	63	1	30	Sunday	0.00	0.00	0.000	0.0001	1.000	0	0
117027	43	56	3	17	Monday	0.00	0.00	0.000	7.5000	0.833	224	224
213836	43	56	2	30	Sunday	0.00	0.00	0.000	9.6000	3.750	1940	1925
75607	43	77	1	12	Wednesday	0.00	0.00	0.000	9.0000	1.000	0	0
58989	43	182	1	9	Sunday	0.00	0.00	0.000	7.5000	0.833	0	0
113848	159	190	3	17	Monday	0.02	0.14	0.410	0.0001	1.000	550	418

```
advertising_train <- mutate(advertising_train,
                            "ln_cpc" = log(cpc + 0.005),
                            "ln_ctr" = log(ctr + 0.005),
                            "ln_impr" = log(impression + 0.005),
                            "ln_req" = log(requests + 0.005),
                            "ln_y" = log(y + 0.005))

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) +
```

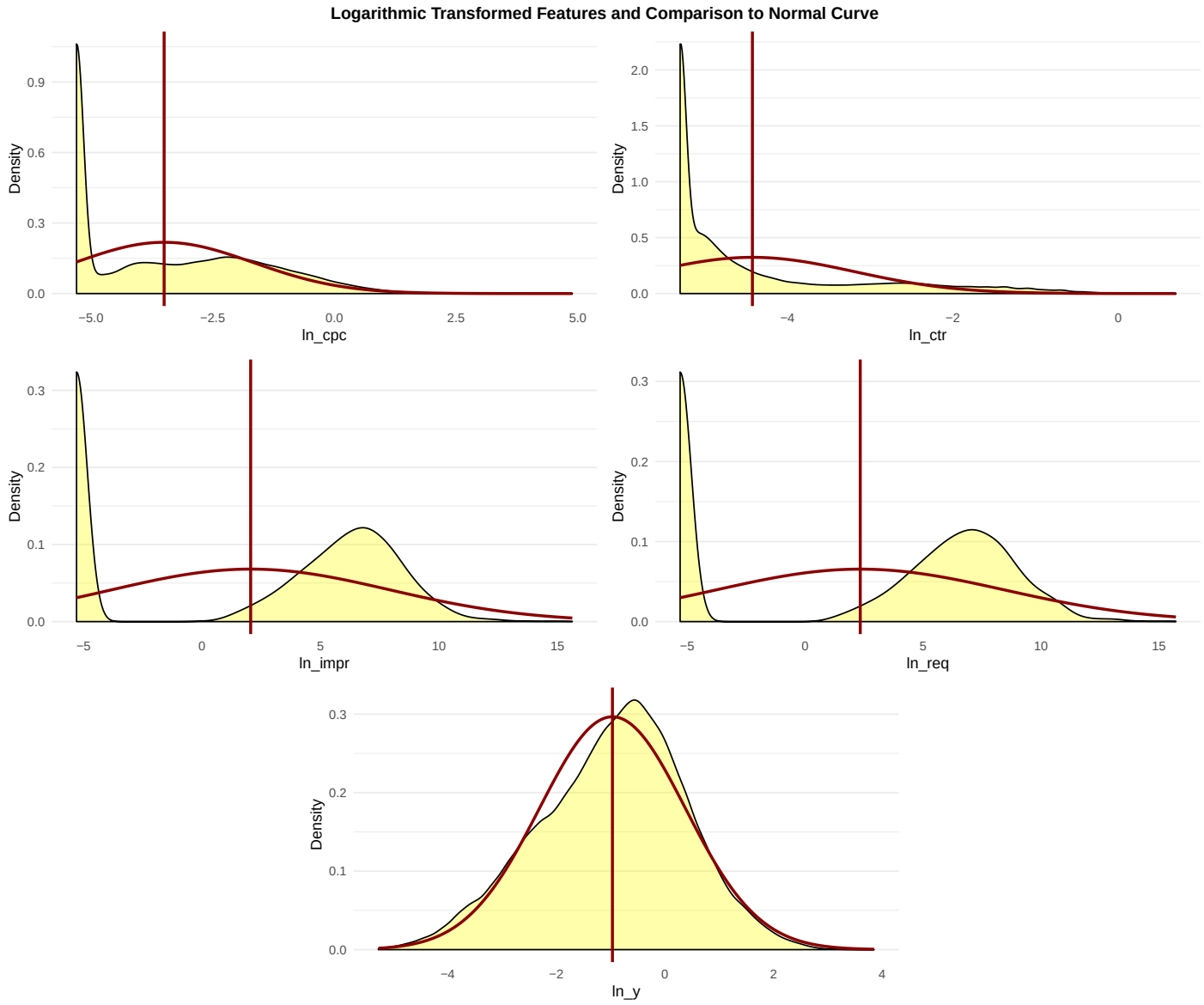
Table 5: Sample of advertising_train Data Frame After Logarithmic Transformations (cont)

case_id	cpc	ctr	viewability	ratio1	ratio2	ratio3	ratio4	ratio5	y	ln_cpc	ln_ctr	ln_impr	ln_req	ln_y
4793	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.04028	-5.298	-5.30	-5.30	-5.30	-3.095
207441	0.5009	0.0007	0.106	0.470	0.853	1.0000	0.0000	0.000	0.35515	-0.681	-5.17	8.86	8.96	-1.021
27864	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.02500	-5.298	-5.30	-5.30	-5.30	-3.507
103904	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.01705	-5.298	-5.30	-5.30	-5.30	-3.814
197928	0.0091	0.0500	0.909	1.000	1.000	1.0000	0.0000	0.000	0.33529	-4.262	-2.90	3.00	3.00	-1.078
204185	0.0692	0.0036	0.810	0.895	0.926	1.0000	0.0000	0.000	0.19189	-2.601	-4.76	7.40	7.43	-1.625
46245	0.0568	0.0139	1.000	0.986	0.931	1.0000	0.0000	0.000	0.08289	-2.784	-3.97	4.28	7.18	-2.432
34697	0.1156	0.0049	0.333	1.000	0.859	0.1902	0.0146	0.795	0.48889	-2.115	-4.62	5.32	5.36	-0.705
181205	0.4312	0.0015	0.675	1.000	0.680	1.0000	0.0000	0.000	0.35204	-0.830	-5.04	6.49	6.53	-1.030
28964	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.04413	-5.298	-5.30	-5.30	-5.30	-3.013
11896	0.0022	0.0059	0.423	1.000	0.306	0.0000	0.7412	0.259	0.00943	-4.934	-4.52	5.14	5.53	-4.238
172059	2.4878	0.0006	0.273	0.948	0.381	1.0000	0.0000	0.000	0.77020	0.913	-5.18	7.44	7.88	-0.255
127999	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.03396	-5.298	-5.30	-5.30	-5.30	-3.245
156782	0.0854	0.0009	0.782	1.000	0.976	1.0000	0.0000	0.000	0.04366	-2.404	-5.13	7.04	7.24	-3.023
211920	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.02605	-5.298	-5.30	-5.30	-5.30	-3.472
117027	0.0270	0.0268	0.904	1.000	0.714	0.0000	0.3571	0.643	0.75430	-3.442	-3.45	5.41	5.41	-0.275
213836	0.1148	0.0042	0.861	1.000	0.899	1.0005	0.0000	0.000	0.59947	-2.122	-4.69	7.56	7.57	-0.503
75607	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	0.16462	-5.298	-5.30	-5.30	-5.30	-1.774
58989	0.0000	0.0000	0.000	0.000	0.000	0.0000	0.0000	0.000	1.62222	-5.298	-5.30	-5.30	-5.30	0.487
113848	0.0182	0.0048	0.692	0.629	0.983	0.0383	0.3014	0.660	0.07742	-3.764	-4.63	6.04	6.31	-2.496

```
   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"))
```

```
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))
```

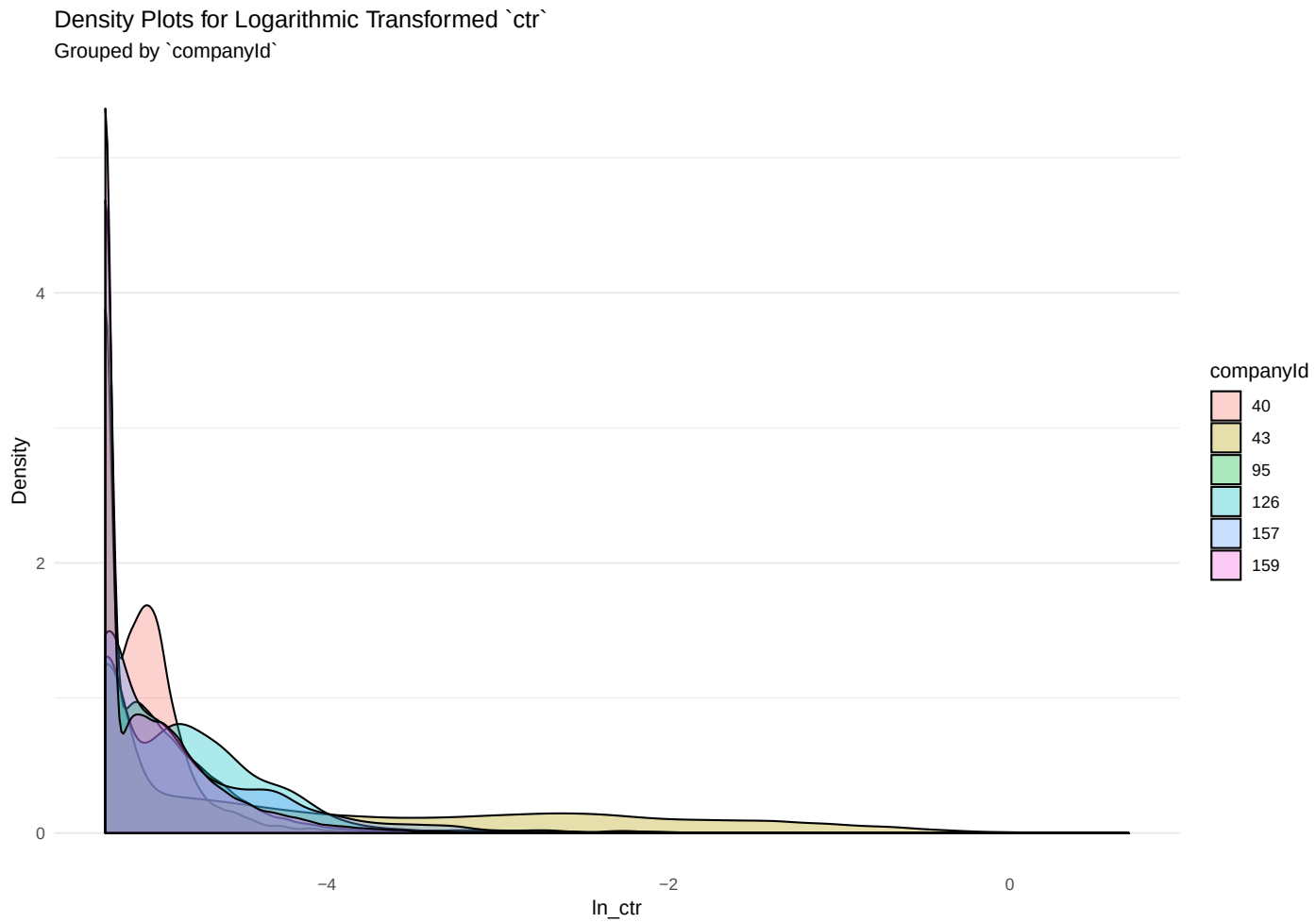


The natural logarithmic transformations of `impression` and `requests` clearly approached a normal distribution. The transformed `y` target feature somewhat resembled a normal distribution, albeit less closely as compared to `impression`. Both `cpc` and `ctr` appeared to be bimodal distributions after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

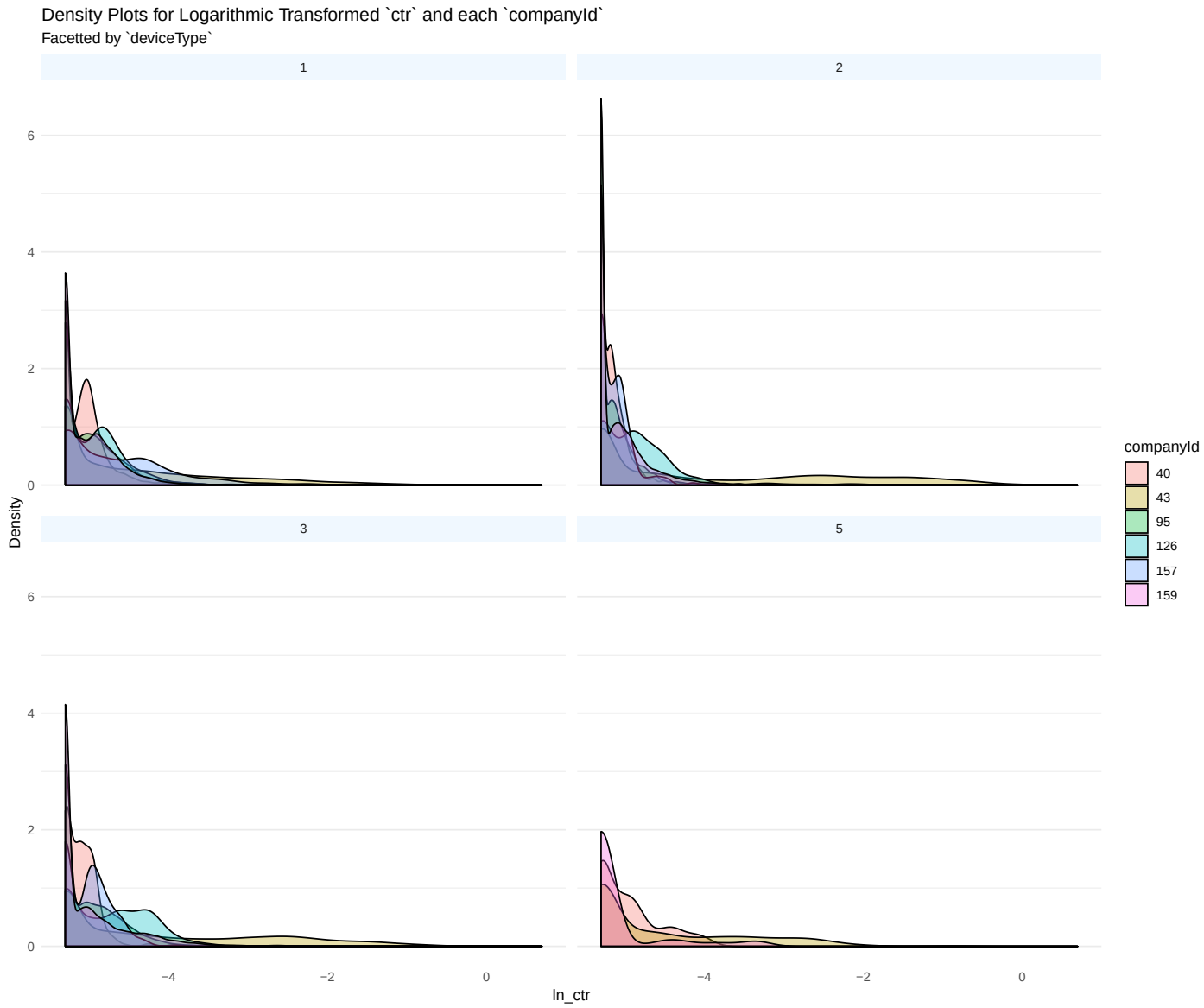
```
ggplot(advertising_train) +
   geom_density(aes(x = ln_ctr, fill = companyId),
                alpha = 1/3) +
   labs(title = "Density Plots for Logarithmic Transformed \`ctr\`",
        subtitle = "Grouped by \`companyId\`",
        y = "Density") +
   theme_minimal() +
   theme(panel.grid.major.x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



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.

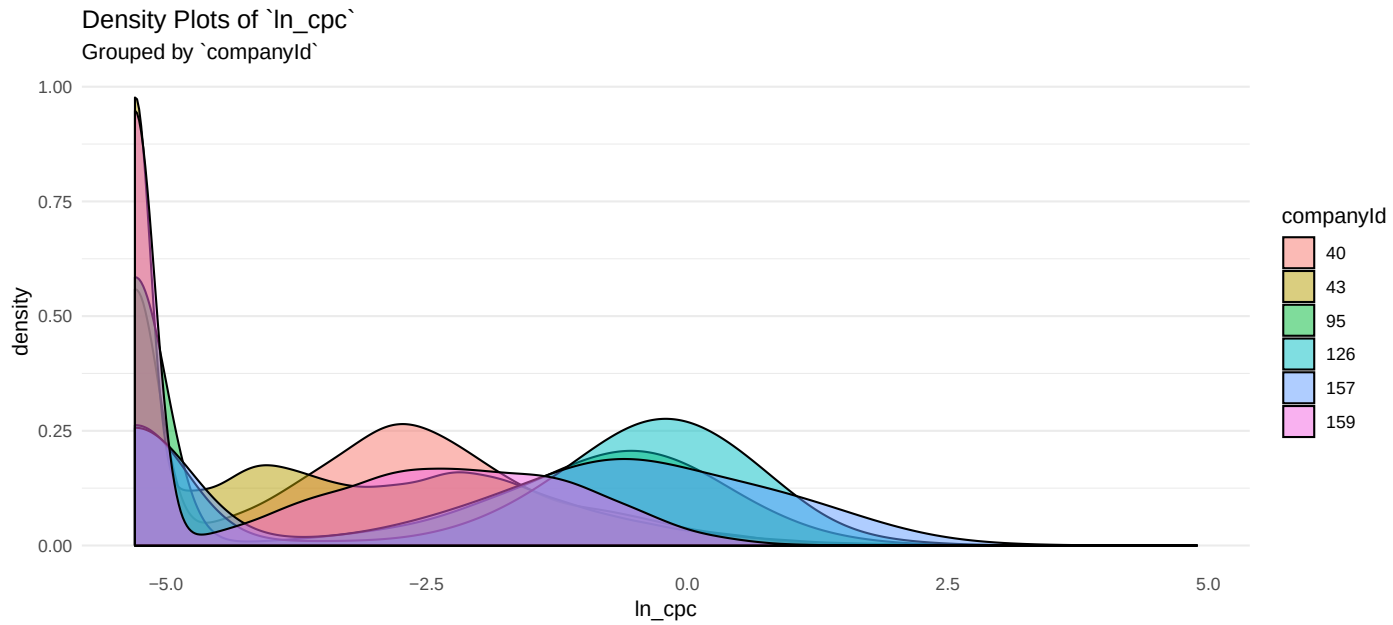
```
ggplot(advertising_train) +
   geom_density(aes(x = ln_ctr, fill = companyId),
                alpha = 1/3) +
   facet_rep_wrap(~deviceType) +
   labs(title = "Density Plots for Logarithmic Transformed \`ctr\` and each \`companyId\`",
        subtitle = "Facetted by \`deviceType\`",
        y = "Density")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

```
        strip.background = element_rect(fill = "aliceblue",
                                        colour = NA))
```

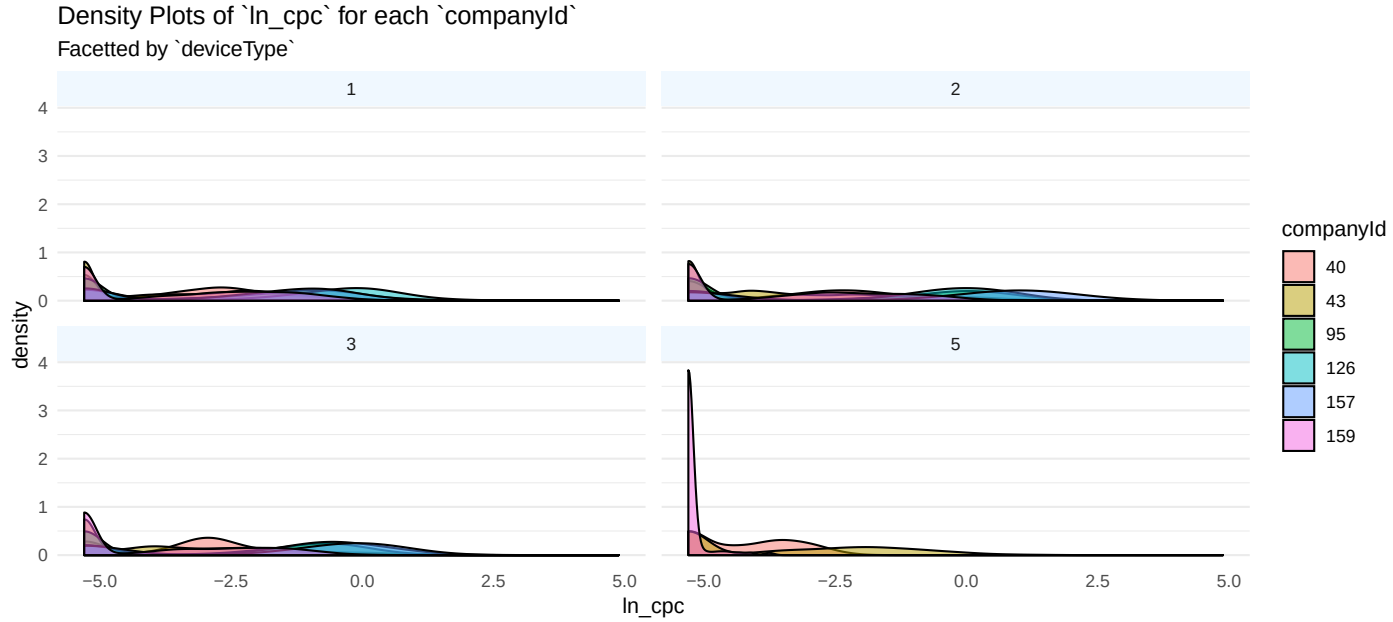


As above for `ln_ctr`, grouping by `companyId` and facetting by `deviceType` revealed a multivariate relationship between aforementioned descriptive features and the transformed `ln_cpc`.

```
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())
```




```
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))
```



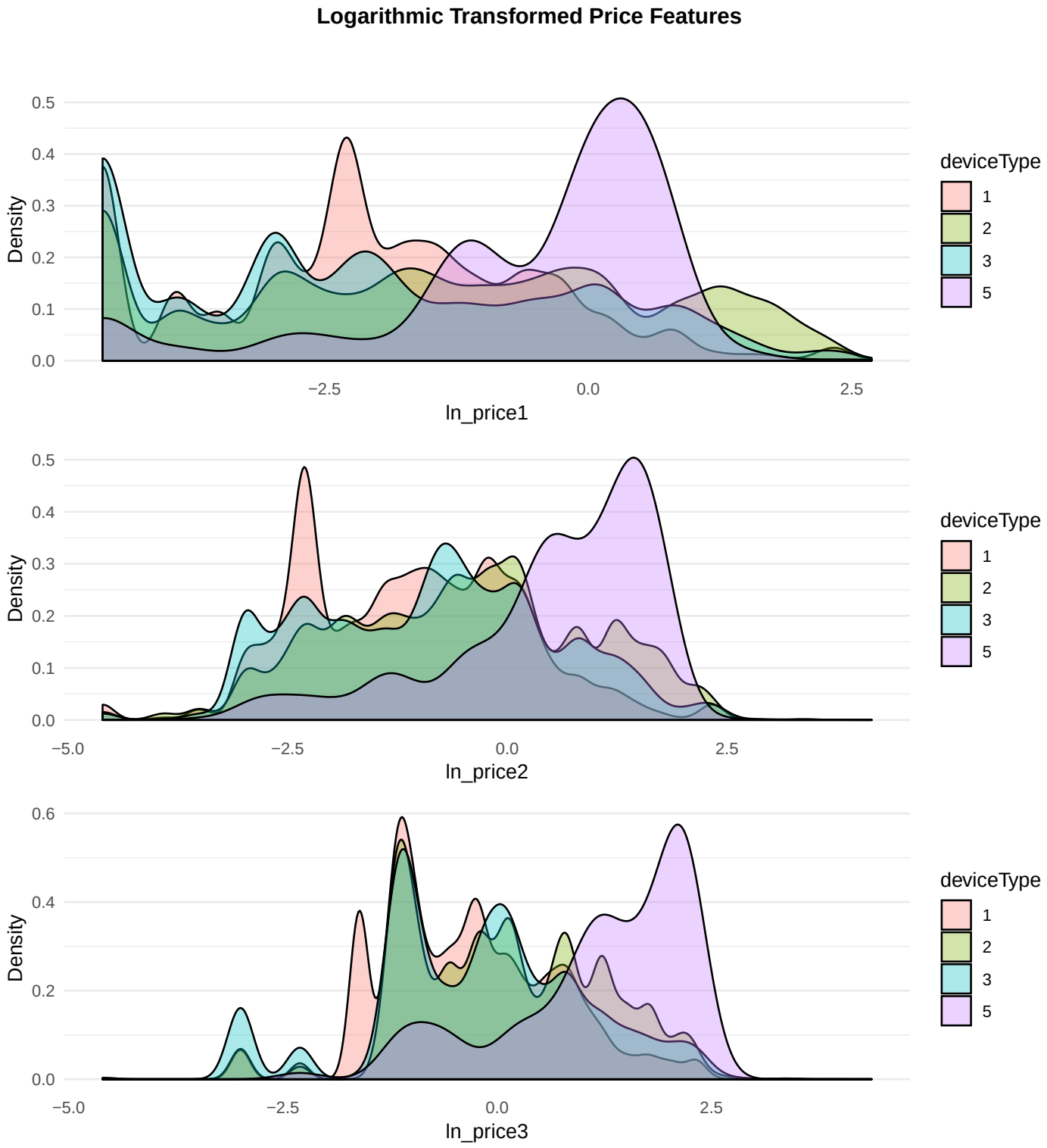
Each of the pricing features, (`price1`, `price2`, `price3`) were not suitably transformed by either logarithmic, square root, or cube root. Logarithmic transformations appeared to spread the data the most, but these transformations considerably diverged from a symmetrical normal distribution. Further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

```r
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).
```

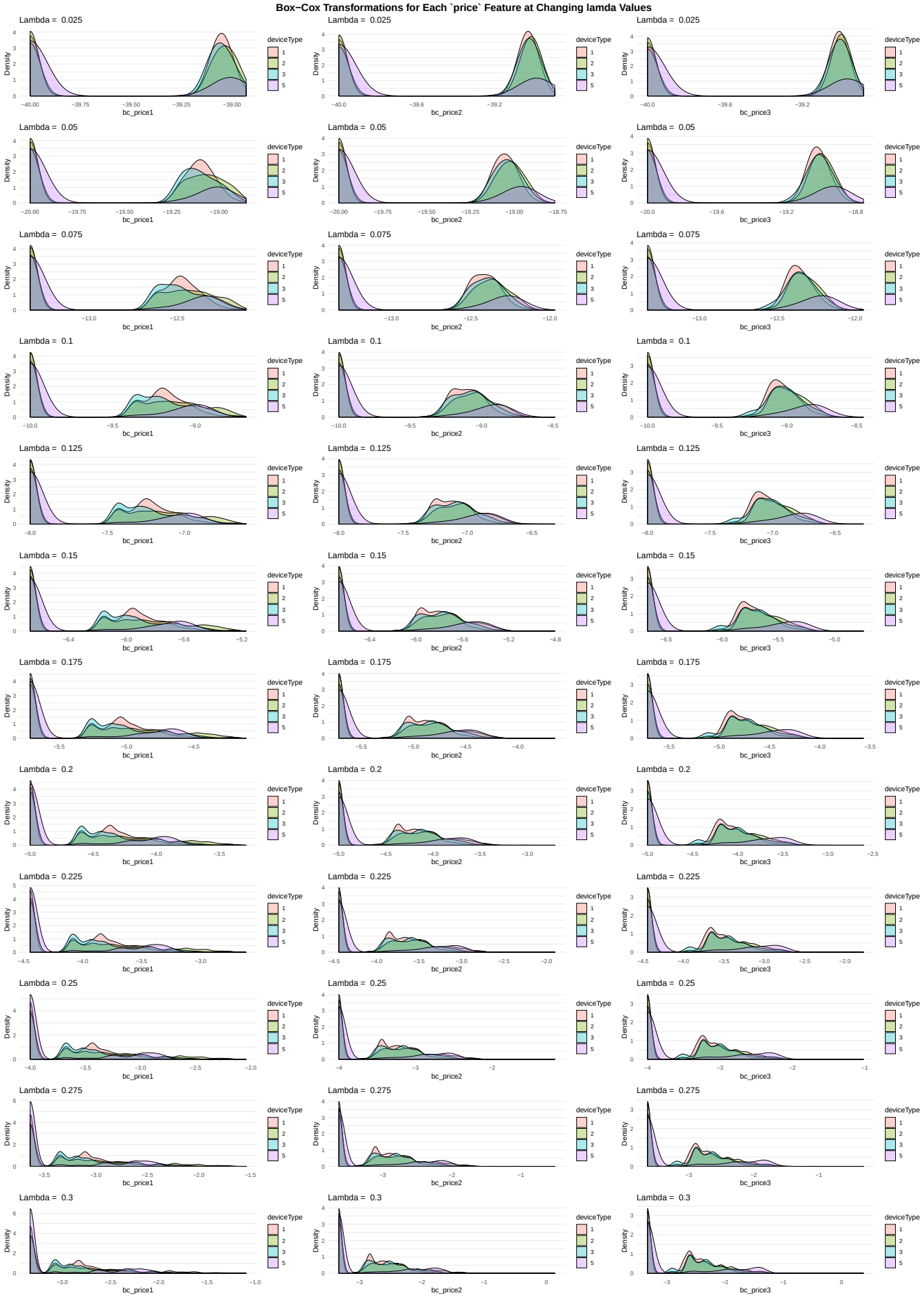


Box-Cox transformations with a range of lamda values also did not convert the price features into distributions that resembled a normal curve.

```
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))

}
```

```
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)] <- 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]]

}


sample_adv <- sample_n(advertising_train, 20)

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)
```

```
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 = ",")),
```

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
160,149	43	56	2	23	Sunday	0.00	0.00	0.00	18.0000	0.50	518	518	0.0665	0.0193	0.649
54,420	43	189	1	9	Sunday	0.00	0.00	0.00	6.5520	0.12	380	380	0.0844	0.0026	0.173
106,830	43	75	2	16	Sunday	0.74	1.81	3.62	6.5520	0.12	6,360	4,550	0.1737	0.0075	0.586
53,278	43	234	3	9	Sunday	0.00	0.00	0.00	7.5000	0.83	98	98	0.4482	0.0102	0.747
34,914	43	77	1	6	Thursday	0.00	0.00	0.00	7.5000	0.83	269	269	0.2343	0.0037	0.381
71,305	43	191	1	11	Tuesday	0.79	0.79	0.79	0.0001	1.00	0	0	0.0000	0.0000	0.000
204,435	43	77	1	29	Saturday	0.09	0.23	0.45	6.5520	0.12	0	0	0.0000	0.0000	0.000
211,856	43	202	1	30	Sunday	0.02	0.08	0.34	18.0000	2.00	9,425	9,186	0.1551	0.0016	0.050
213,195	43	57	2	30	Sunday	1.19	1.19	1.19	7.5000	0.83	0	0	0.0000	0.0000	0.000
33,652	43	38	2	6	Thursday	0.32	0.64	1.29	7.5000	0.83	0	0	0.0000	0.0000	0.000
87,091	157	38	3	13	Thursday	0.01	0.40	0.01	7.5000	0.83	486	415	0.7468	0.0024	0.689
174,810	95	234	2	25	Tuesday	1.28	2.48	4.95	18.0000	2.00	1,965	481	0.9849	0.0021	0.207
140,282	159	190	2	20	Thursday	0.00	0.00	0.00	0.0001	1.00	0	0	0.0000	0.0000	0.000
2,565	43	110	1	1	Saturday	0.06	0.10	0.32	1.6000	0.16	0	0	0.0000	0.0000	0.000
199,296	95	38	3	28	Friday	0.02	0.19	0.37	7.5000	0.83	0	0	0.0000	0.0000	0.000
61,394	40	38	1	10	Monday	0.10	0.10	0.20	0.0001	1.00	8,627	6,439	1.0349	0.0006	0.534
113,497	43	202	1	17	Monday	0.00	0.00	0.00	18.0000	2.00	11,784	11,683	0.0798	0.0058	0.058
94,535	159	191	3	14	Friday	0.02	0.11	0.34	0.0001	1.00	53	52	0.0102	0.0192	0.882
8,791	43	139	2	2	Sunday	0.80	0.80	0.80	0.0001	1.00	164	5	0.0022	0.4000	1.000
89,673	43	179	2	14	Friday	2.26	2.26	2.26	0.0001	1.00	0	0	0.0000	0.0000	0.000

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
160,149	1.00	0.95	1.002	0.000	0.000	1.314	-2.638	-3.7	6.2	6.2	0.28	0.748	0.759	0.00000	0.0000	0.00000
54,420	1.00	0.89	0.495	0.018	0.487	0.273	-2.415	-4.9	5.9	5.9	-1.28	0.254	0.276	0.00000	0.0000	0.00000
106,830	0.61	0.97	1.000	0.000	0.000	0.886	-1.722	-4.4	8.4	8.8	-0.12	0.499	0.517	0.05037	0.0287	0.04592
53,278	1.00	0.56	0.204	0.214	0.582	4.646	-0.791	-4.2	4.6	4.6	1.54	0.249	0.276	0.00000	0.0000	0.00000
34,914	1.00	0.67	0.067	0.305	0.628	0.734	-1.430	-4.7	5.6	5.6	-0.30	0.163	0.172	0.00000	0.0000	0.00000
71,305	0.00	0.00	0.000	0.000	0.000	0.211	-5.298	-5.3	-5.3	-5.3	-1.53	0.333	0.345	0.05378	0.0125	0.00998
204,435	0.00	0.00	0.000	0.000	0.000	1.655	-5.298	-5.3	-5.3	-5.3	0.51	0.955	0.966	0.00613	0.0036	0.00575
211,856	0.93	0.75	0.100	0.124	0.777	0.182	-1.832	-5.0	9.1	9.2	-1.68	0.989	1.000	0.00136	0.0013	0.00430
213,195	0.00	0.00	0.000	0.000	0.000	2.038	-5.298	-5.3	-5.3	-5.3	0.71	0.996	1.000	0.08101	0.0189	0.01506
33,652	0.00	0.00	0.000	0.000	0.000	1.303	-5.298	-5.3	-5.3	-5.3	0.27	0.157	0.172	0.02178	0.0101	0.01641
87,091	0.93	0.69	0.000	0.906	0.094	1.295	-0.285	-4.9	6.0	6.2	0.26	0.407	0.414	0.00068	0.0063	0.00013
174,810	0.67	0.86	1.000	0.000	0.000	0.584	-0.010	-4.9	6.2	7.6	-0.53	0.816	0.828	0.08713	0.0393	0.06274
140,282	0.00	0.00	0.000	0.000	0.000	0.049	-5.298	-5.3	-5.3	-5.3	-2.91	0.655	0.655	0.00000	0.0000	0.00000
2,565	0.00	0.00	0.000	0.000	0.000	0.034	-5.298	-5.3	-5.3	-5.3	-3.25	0.012	0.000	0.00408	0.0016	0.00405
199,296	0.00	0.00	0.000	0.000	0.000	0.412	-5.298	-5.3	-5.3	-5.3	-0.87	0.931	0.931	0.00136	0.0030	0.00469
61,394	0.99	0.70	0.050	0.576	0.374	0.490	0.039	-5.2	8.8	9.1	-0.70	0.287	0.310	0.00681	0.0016	0.00253
113,497	1.00	0.88	0.041	0.159	0.799	0.458	-2.467	-4.5	9.4	9.4	-0.77	0.530	0.552	0.00000	0.0000	0.00000
94,535	1.00	1.00	0.058	0.942	0.000	0.276	-4.186	-3.7	4.0	4.0	-1.27	0.441	0.448	0.00136	0.0017	0.00430
8,791	1.00	0.60	1.000	0.000	0.000	0.011	-4.934	-0.9	1.6	5.1	-4.12	0.041	0.034	0.05446	0.0127	0.01012
89,673	0.00	0.00	0.000	0.000	0.000	1.443	-5.298	-5.3	-5.3	-5.3	0.37	0.419	0.448	0.15385	0.0358	0.02869

```
              font_size = 8, latex_options = c("striped"),
              full_width = T)
```

```
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 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	ratio1	norm_ratio2	norm_ratio3	norm_ratio4	norm_ratio5	norm_y	norm_ln_cpc	norm_ln_ctr	norm_ln_impr
160,149	0.500	0.0847	0.0000773	0.0000849	0.000502	0.0097	0.0927	1.00	0.92	0.668	0.000	0.000	0.02792	0.261	0.264	0.55
54,420	0.182	0.0082	0.0000567	0.0000623	0.000637	0.0013	0.0248	1.00	0.87	0.330	0.017	0.406	0.00581	0.283	0.070	0.54
106,830	0.182	0.0082	0.0009490	0.0007459	0.001311	0.0037	0.0837	0.61	0.94	0.667	0.000	0.000	0.01882	0.351	0.153	0.66
53,278	0.208	0.1525	0.0000146	0.0000161	0.003382	0.0051	0.1068	1.00	0.55	0.136	0.199	0.485	0.09872	0.442	0.185	0.47
34,914	0.208	0.1525	0.0000401	0.0000441	0.001768	0.0018	0.0545	1.00	0.66	0.045	0.283	0.524	0.01560	0.380	0.092	0.52
71,305	0.000	0.1864	0.0000000	0.0000000	0.000000	0.0000	0.0000	0.00	0.00	0.000	0.000	0.000	0.00448	0.000	0.000	0.00
204,435	0.182	0.0082	0.0000000	0.0000000	0.000000	0.0000	0.0000	0.00	0.00	0.000	0.000	0.000	0.03517	0.000	0.000	0.00
211,856	0.500	0.3898	0.0014063	0.0015058	0.001170	0.0008	0.0072	0.93	0.73	0.066	0.115	0.647	0.00386	0.340	0.046	0.69
213,195	0.208	0.1525	0.0000000	0.0000000	0.000000	0.0000	0.0000	0.00	0.00	0.000	0.000	0.000	0.04330	0.000	0.000	0.00
33,652	0.208	0.1525	0.0000000	0.0000000	0.000000	0.0000	0.0000	0.00	0.00	0.000	0.000	0.000	0.02768	0.000	0.000	0.00
87,091	0.208	0.1525	0.0000725	0.0000680	0.005635	0.0012	0.0984	0.93	0.67	0.000	0.841	0.078	0.02752	0.492	0.065	0.54
174,810	0.500	0.3898	0.0002932	0.0000788	0.007431	0.0010	0.0296	0.67	0.84	0.667	0.000	0.000	0.01242	0.519	0.059	0.55
140,282	0.000	0.1864	0.0000000	0.0000000	0.000000	0.0000	0.0000	0.00	0.00	0.000	0.000	0.000	0.00105	0.000	0.000	0.00
2,565	0.044	0.0148	0.0000000	0.0000000	0.000000	0.0000	0.0000	0.00	0.00	0.000	0.000	0.000	0.00072	0.000	0.000	0.00
199,296	0.208	0.1525	0.0000000	0.0000000	0.000000	0.0000	0.0000	0.00	0.00	0.000	0.000	0.000	0.00876	0.000	0.000	0.00
61,394	0.000	0.1864	0.0012872	0.0010555	0.007809	0.0003	0.0763	0.99	0.68	0.033	0.535	0.312	0.01041	0.524	0.019	0.67
113,497	0.500	0.3898	0.0017583	0.0019151	0.000602	0.0029	0.0083	1.00	0.86	0.028	0.148	0.666	0.00974	0.278	0.128	0.70
94,535	0.000	0.1864	0.0000079	0.0000085	0.000077	0.0096	0.1261	1.00	0.97	0.038	0.875	0.000	0.00587	0.109	0.263	0.44
8,791	0.000	0.1864	0.0000245	0.0000008	0.000017	0.2000	0.1429	1.00	0.58	0.667	0.000	0.000	0.00024	0.036	0.733	0.33
89,673	0.000	0.1864	0.0000000	0.0000000	0.000000	0.0000	0.0000	0.00	0.00	0.000	0.000	0.000	0.03066	0.000	0.000	0.00

2 Phase 2: Algorithm Implementation

2.1 Overview

2.1.1 Data Normalisation

The data normalisation above led to infinite values when the untransformed value was 0. Imputing `-inf` and `inf` for `NA` is not strictly correct. To prevent this normalisation of

```
for (i in colnames(advertising_train)) {

  print(paste(i, paste( paste(
              "Number of Finite:",
              sum(is.finite(advertising_train[[i]]))),
              paste(
              "Number of Finite:",
              sum(is.finite(advertising_train[[i]]))),
              sep = "\n")))

}
```

```
## [1] "case_id Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "companyId Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "countryId Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "deviceType Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "day Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "dow Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "price1 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "price2 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "price3 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ad_area Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ad_ratio Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "requests Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "impression Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "cpc Number of Finite: 214128\nNumber of Finite: 214128"
```

```
## [1] "ctr Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "viewability Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ratio1 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ratio2 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ratio3 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ratio4 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ratio5 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "y Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ln_cpc Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ln_ctr Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ln_impr Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ln_req Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "ln_y Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_case_id Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_day Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_price1 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_price2 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_price3 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ad_area Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ad_ratio Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_requests Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_impression Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_cpc Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ctr Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_viewability Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ratio1 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ratio2 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ratio3 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ratio4 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ratio5 Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_y Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ln_cpc Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ln_ctr Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ln_impr Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ln_req Number of Finite: 214128\nNumber of Finite: 214128"
## [1] "norm_ln_y Number of Finite: 214128\nNumber of Finite: 214128"
```

2.2 Feature Selection

2.3 Performance Comparison

2.4 Results

2.4.1 Discussion

2.4.1.1 Limitations and Improvements

2.5 Conclusions

2.6 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