Lesson 4 Solutions

Stefanie Molin

April 17, 2017

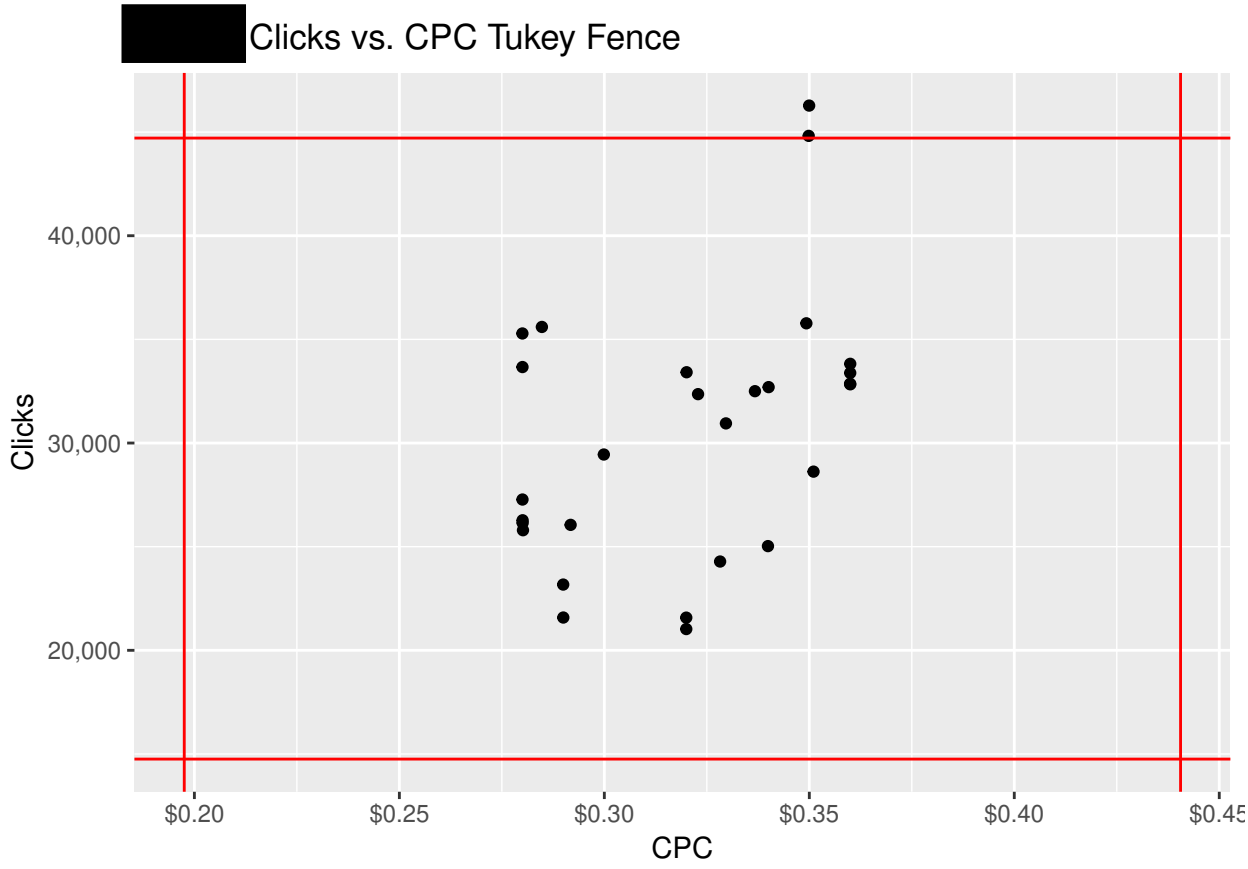
Let's do some practice problems to challenge your understanding of `ggplot2` and review the material from the prior lessons.

1. Using `ggplot2` create a scatterplot of CPC and clicks for the client of your choice daily for the last 30 days. Add red crosshairs on the graph to indicate points outside the Tukey fence (outliers), which has bounds $Q1 - 1.5 * IQR$ and $Q3 + 1.5 * IQR$ where interquartile range (IQR) is defined as $IQR = Q3 - Q1$ and Q stands for quartile. Note you will need the `quantile()` and `IQR()` functions. Try to write a simple query and use `dplyr` to manipulate your data.

```
# load libraries
library(dplyr)
library(ggplot2)
library(RJDBC)
```

```
# QueryVertica() is already sourced and username/password defined

# query for last 30 days of client stats for [redacted]
query <- "
SELECT
  *
FROM
 [redacted]
WHERE
  day >= '%s'
  AND client_id = 4624
"

[redacted] <- QueryVertica(username, sprintf(query, Sys.Date() - 30), password)

# select columns, group by day to summarize, add ctr and cpc and sort by day
[redacted]_pivot <- [redacted] %>%
  select(day, clicks, revenue) %>%
  group_by(day) %>%
  summarize(total_clicks = sum(clicks, na.rm = TRUE),
            spend = sum(revenue, na.rm = TRUE)) %>%
  mutate(cpc = spend/total_clicks) %>%
  select(total_clicks, cpc)
```

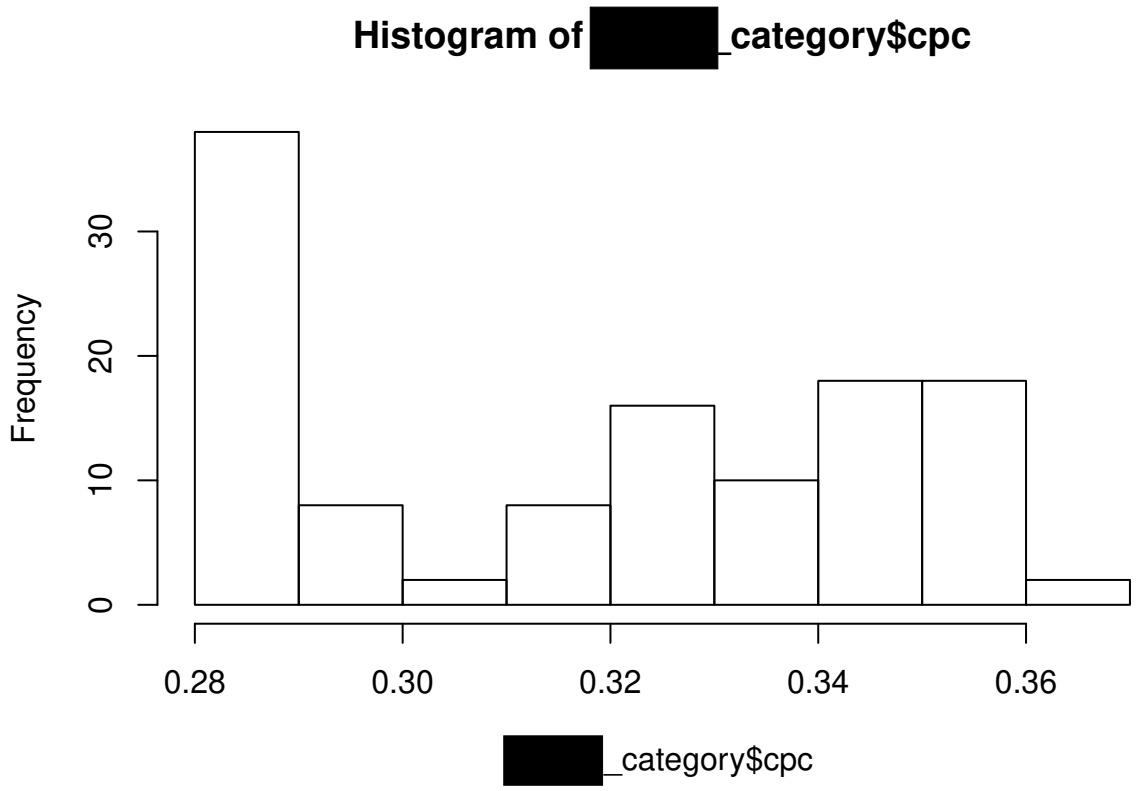
```r
# generate plot
[redacted]_pivot %>%
  ggplot(aes(x = cpc, y = total_clicks)) +
  geom_point() +
  geom_hline(yintercept = quantile([redacted]_pivot$total_clicks, 0.25) -
               1.5 * IQR([redacted]_pivot$total_clicks), col = "red") +
  geom_hline(yintercept = quantile([redacted]_pivot$total_clicks, 0.75) +
               1.5 * IQR([redacted]_pivot$total_clicks), col = "red") +
  geom_vline(xintercept = quantile([redacted]_pivot$cpc, 0.25) -
               1.5 * IQR([redacted]_pivot$cpc), col = "red") +
  geom_vline(xintercept = quantile([redacted]_pivot$cpc, 0.75) +
               1.5 * IQR([redacted]_pivot$cpc), col = "red") +
  ggtitle("[redacted] Clicks vs. CPC Tukey Fence") +
  labs(x = "CPC", y = "Clicks") +
  scale_x_continuous(labels = scales::dollar) +
  scale_y_continuous(labels = scales::comma)
```



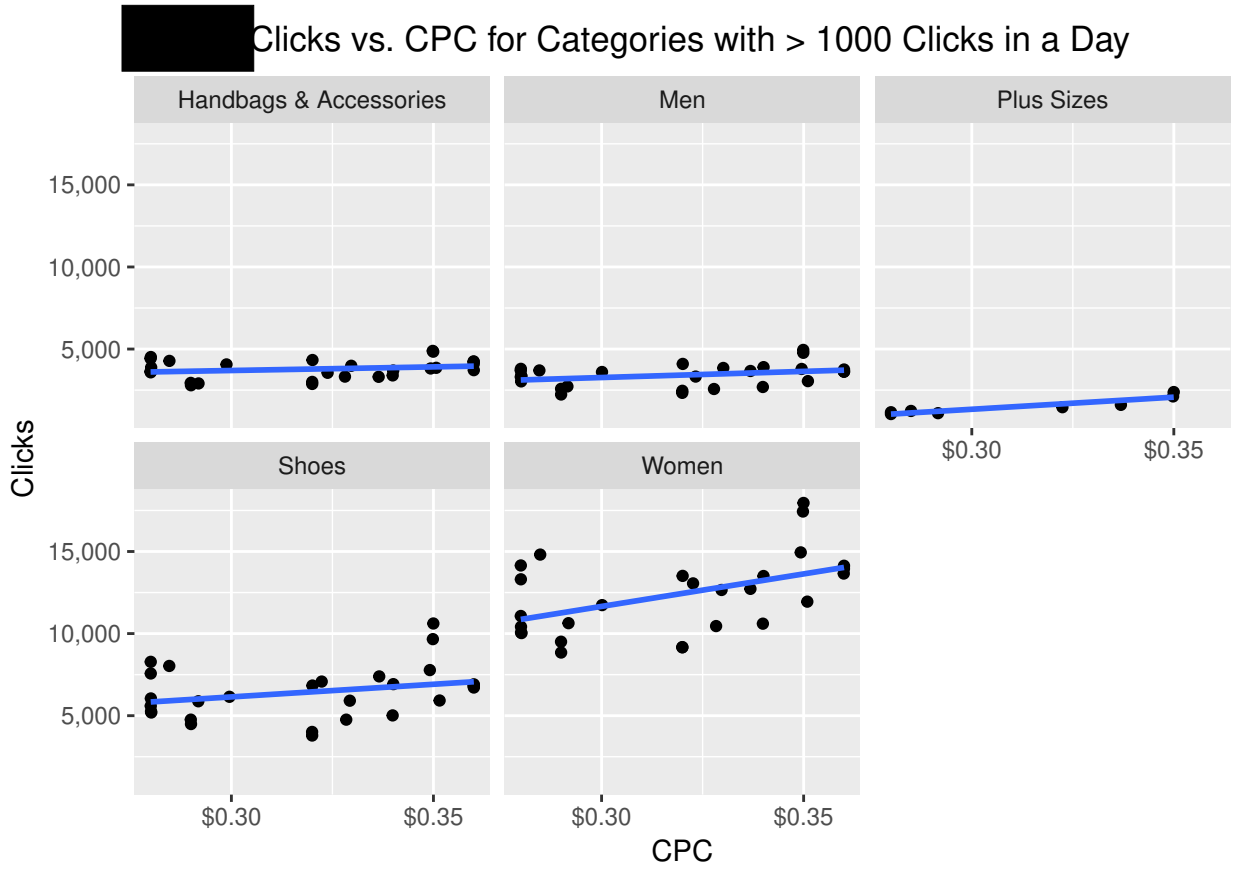
2. Pull clicks and CPC by category by day for the last 30 days on the client of your choice (choose a category level that makes sense). Create a quick base R histogram to see the distribution of CPC. Then, create a scatterplot of CPC and clicks faceted by category (be sure to try both `facet_wrap()` and `facet_grid()` to see which works best for your data). Limit to categories that have more than 1000 clicks on a given day (or a relevant threshold). Add in regression lines. Note that in this example `facet_wrap()` will most likely look the best, but you should still try `facet_grid()`.

```
# set up dynamic query parameters
category_level <- 2
merchant_id <- 5535

# here we are using %d because we are inserting numbers, but %s will still work
query <- "
SELECT
    name AS category
    , day
    , clicks
    , cpc
FROM
    (SELECT
        category_id
        , day
        , SUM(clicks) AS clicks
        , SUM(revenue)/SUM(clicks) AS cpc
    FROM
       [redacted]
    WHERE
        category_level = %d
        AND day >= CURRENT_DATE() - 30
        AND merchant_id = %d
    GROUP BY
        category_id
        , day) stats
JOIN
    (SELECT
        category_id
        , name
    FROM
       [redacted]
    WHERE
        merchant_id = %d
        AND category_level = %d
    GROUP BY
        category_id
        , name) cat
ON
    cat.category_id = stats.category_id
GROUP BY
    name
    , day
    , clicks
    , cpc
"
```

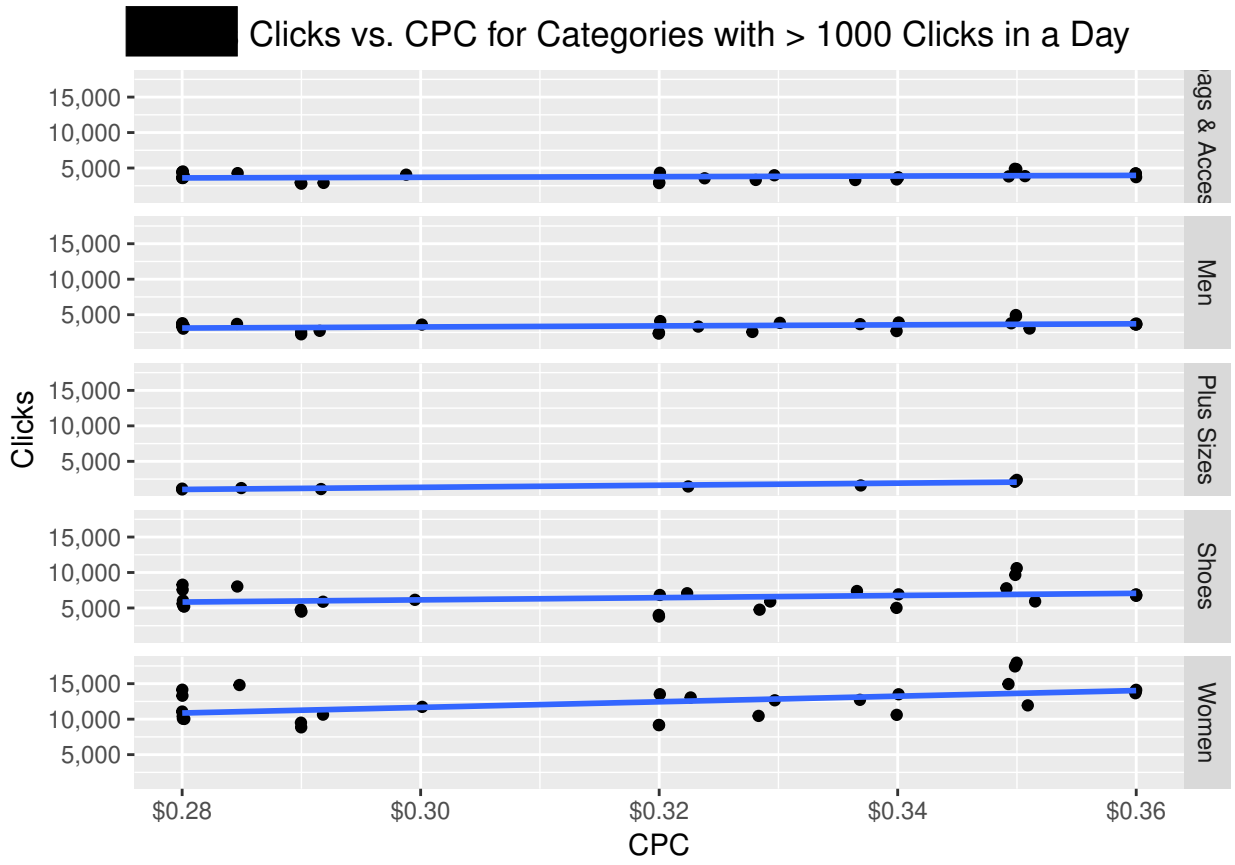
```
[redacted]_category <- QueryVertica(username,
                                    sprintf(query, category_level, merchant_id,
                                            merchant_id, category_level), password)

# use dplyr to filter to categories with more than 1000 clicks per day
[redacted]_category <- [redacted]_category %>% filter(clicks > 1000)

# histogram to see distribution of CPC
hist([redacted]_category$cpc)
```



```r
# make ggplot facet_wrap
[redacted]category %>%
  ggplot(aes(x = cpc, y = clicks)) +
  geom_point() +
  facet_wrap(~ category) +
  scale_y_continuous(name = "Clicks", labels = scales::comma) +
  scale_x_continuous(name = "CPC", labels = scales::dollar,
                     breaks = scales::pretty_breaks(n = 3)) +
  ggtitle("[redacted] Clicks vs. CPC for Categories with > 1000 Clicks in a Day") +
  stat_smooth(method = "lm", se = FALSE, alpha = 0.6)
```



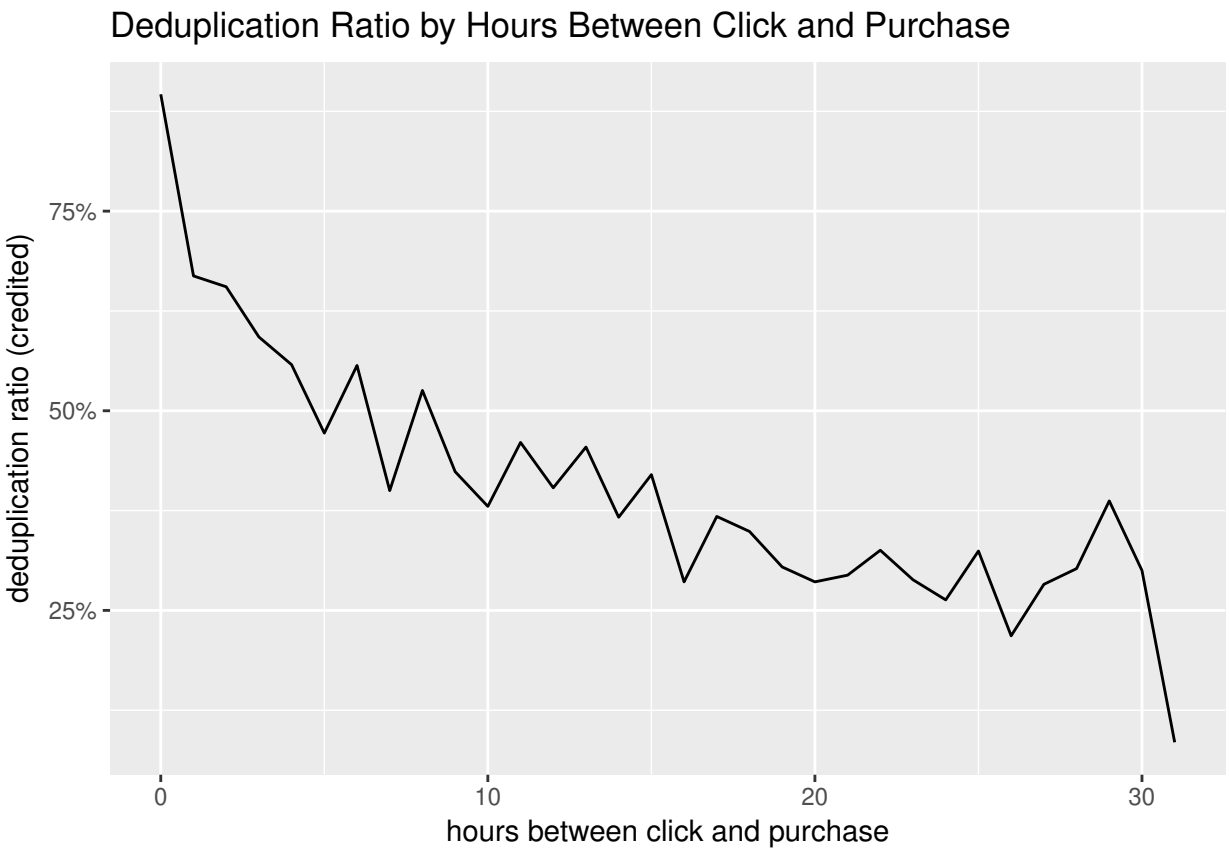
```r
# make ggplot facet_grid
[redacted]_category %>%
  ggplot(aes(x = cpc, y = clicks)) +
  geom_point() +
  facet_grid(category ~ .) +
  scale_y_continuous(name = "Clicks", labels = scales::comma) +
  scale_x_continuous(name = "CPC", labels = scales::dollar) +
  ggtitle("[redacted] Clicks vs. CPC for Categories with > 1000 Clicks in a Day") +
  stat_smooth(method = "lm", se = FALSE, alpha = 0.6)
```



3. Graph the percentage of deduplicated sales credited to [redacted] by hour between click and purchase for the last 30 days. If you can't think of a client using the deduplication parameter, use [redacted].
    - Write a Vertica query to pull in the amount of deduplicated sales credited to [redacted] by hour between click and purchase as well as the total sales for the last 30 days. Don't bucket the hours in your query–you will work with them in `dplyr`!
    - Using `dplyr` verbs, bucket the hours above 30 to the hour 31 bucket, remove any nonsense data, add a column for percentage of duplicated sales credited to [redacted] reduce the data into just the 2 columns needed for the graph, and sort the data by bucket. Note it is possible to use all 5 verbs and `group_by()` here and that you should use the pipe operator (`%>%`).
    - Use `ggplot2` to generate a line graph of the dedup ratio by hour.

    Extra credit: format the y-axis and add labels for the axes and the title.

```
query <- "
SELECT
    FLOOR((trans_timestamp - click_timestamp)/60/60) AS hours_between
    , COUNT(DISTINCT(CASE WHEN deduplication_matching = 1
  THEN transaction_id ELSE NULL END)) AS deduped_conv
  , COUNT(DISTINCT transaction_id) AS total_conv
FROM
  [redacted]
WHERE
    day >= CURRENT_DATE() - 31
    AND merchant_id = 1749
    AND attribution_type = 'pc'
GROUP BY
    FLOOR((trans_timestamp - click_timestamp)/60/60)"

[redacted] <- QueryVertica(username, query, password)

[redacted] %>%
  filter(hours_between >= 0) %>%
  mutate(hour_bucket = ifelse(hours_between > 30, 31, hours_between)) %>%
  group_by(hour_bucket) %>%
  summarize(deduped_conv = sum(deduped_conv, na.rm = TRUE),
            total_conv = sum(total_conv, na.rm = TRUE)) %>%
  mutate(dedup_ratio = deduped_conv/total_conv) %>%
  select(hour_bucket, dedup_ratio) %>%
  arrange(hour_bucket) %>%
  ggplot(aes(x = hour_bucket, y = dedup_ratio)) +
  geom_line() +
  ggtitle("Deduplication Ratio by Hours Between Click and Purchase") +
  labs(x = "hours between click and purchase", y = "deduplication ratio (credited)") +
  scale_y_continuous(labels = scales::percent)
```


Deduplication Ratio by Hours Between Click and Purchase
deduplication ratio (credited)
75%
50%
25%
0
10
20
30
hours between click and purchase