R Training

Lesson 4

Stefanie Molin

April 17, 2017

I. Data Visualization

Now, we are going to build on the knowledge from the last session and learn how to use the `ggplot2` package to visualize data in conjunction with the packages we covered in prior lessons. Let's continue to work with the dataframe [redacted]`_pivot` that we defined in the `dplyr` section as:

```
# load dplyr
library(dplyr)

# get date for 30 days ago
startDate <- Sys.Date() - 30

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

# query Vertica for data and store in dataframe [redacted]
[redacted] <- QueryVertica(username, sprintf(query, startDate), password)

# pivot dataframe
[redacted]pivot <- [redacted] %>%
  select(day, displays, clicks, revenue, pc_conv = post_click_conversions,
         pc_sales = post_click_sales) %>%
  filter(as.Date(day) >= Sys.Date() - 25) %>%
  group_by(day) %>%
  summarize(total_clicks = sum(clicks, na.rm = TRUE),
            total_imps = sum(displays, na.rm = TRUE),
            spend = sum(revenue, na.rm = TRUE),
            conv = sum(pc_conv, na.rm = TRUE)) %>%
  mutate(ctr = total_clicks/total_imps, cpc = spend/total_clicks) %>%
  arrange(day)
```

```
# see first few rows
head([redacted]_pivot)
```

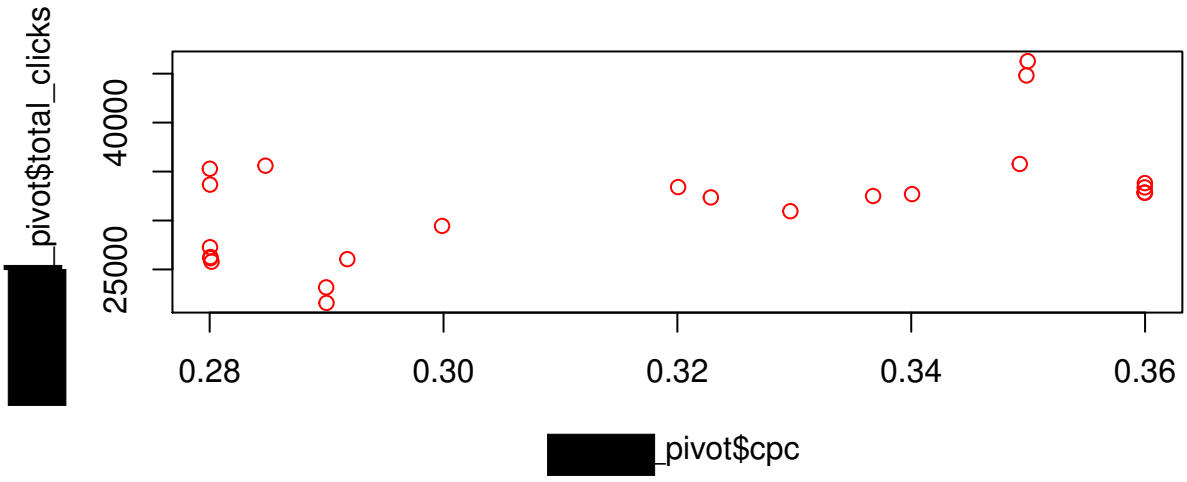
```
## # A tibble: 6 × 7
##          day total_clicks total_imps    spend  conv        ctr       cpc
##        <chr>        <dbl>      <dbl>    <dbl> <dbl>      <dbl>     <dbl>
## 1 2017-03-23 [redacted]
## 2 2017-03-24
## 3 2017-03-25
## 4 2017-03-26
## 5 2017-03-27
## 6 2017-03-28
```

A. Plotting with base R

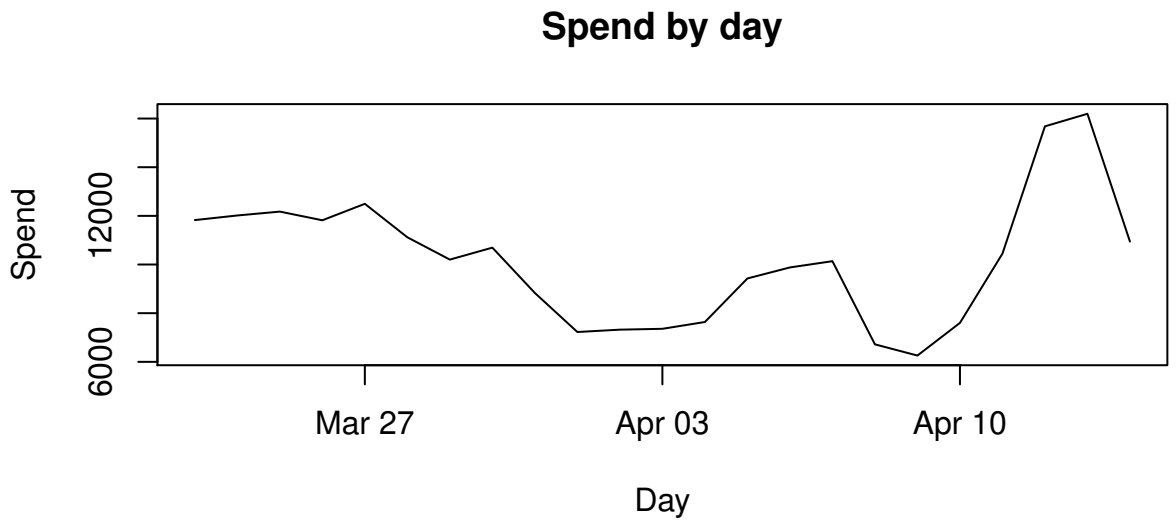
Before we get into using `ggplot2`, let's learn how to make some simple plots with base R.

`plot()`: can be used to make scatterplots, line graphs, and variations of the two.

```
# using the macys data, create a red scatterplot of clicks and cpc
plot([redacted]_pivot$cpc, [redacted]_pivot$total_clicks, col = "red")
```

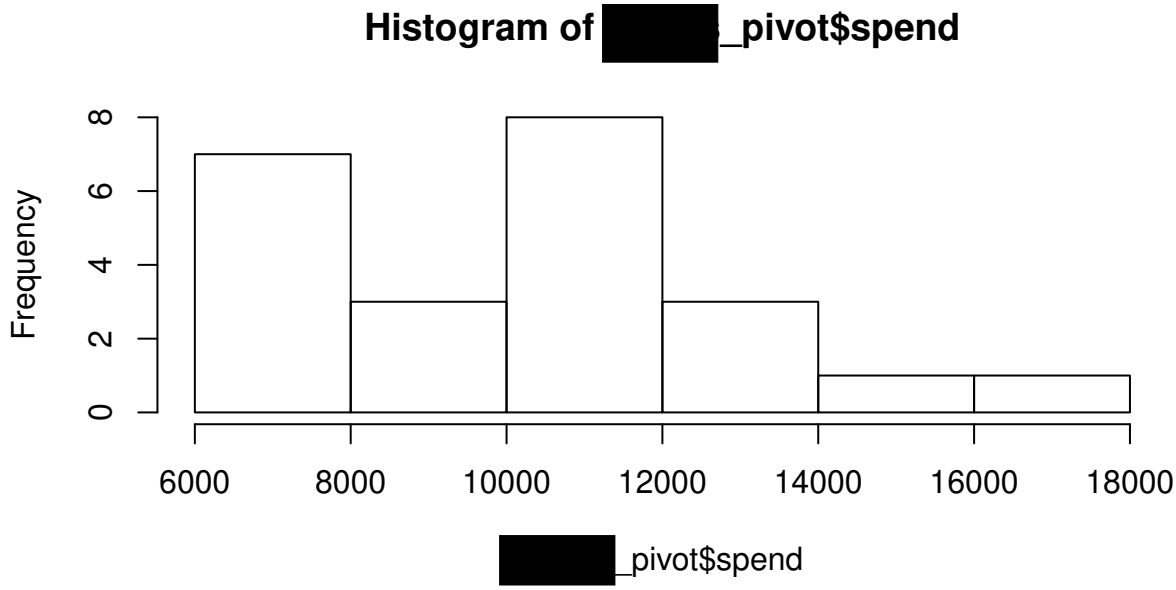


```
# plot spend by day with labels
plot(as.Date([redacted]pivot$day),[redacted]_pivot$spend, type = "l",
     main = "Spend by day", xlab = "Day", ylab = "Spend")
```



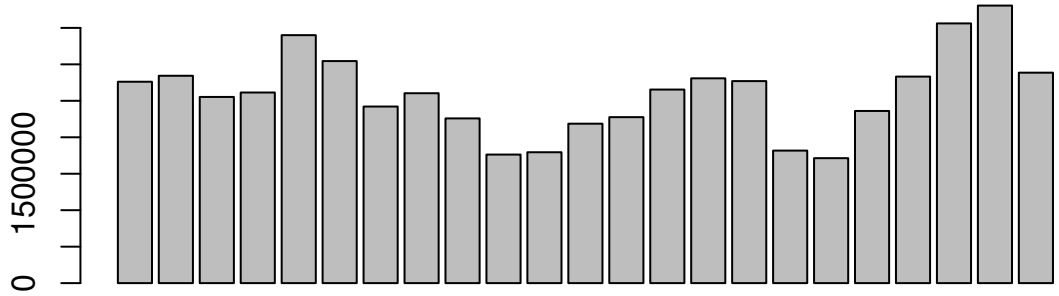
`hist()`: make a histogram

```
# histogram of spend the last 25 days
hist[redacted]_pivot$spend)
```



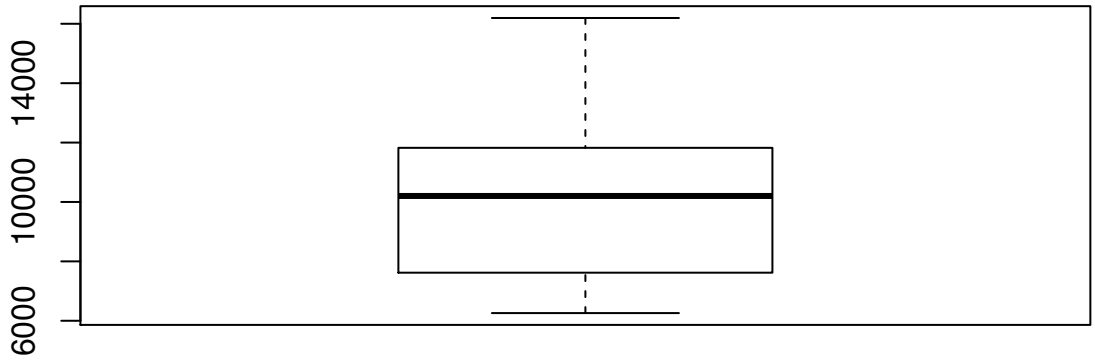
`barplot()`: draw a bar chart

```
# bar chart of impressions
barplot([redacted]_pivot$total_imps)
```



`boxplot()`: box-and-whisker plot

```
# box-and-whisker plot of spend
boxplot([redacted]_pivot$spend)
```



We won't go into how to change point shape, color, line type, etc. You can read up about that in the documentation for each function.

B. ggplot2

Base R plots are pretty basic and not that pretty, so they are more used to quickly get a sense of any patterns in the data you are dealing with but not for sharing with other parties. To make graphs fit for sharing, we will use the popular `ggplot2` package. This package is based on the grammar of graphics; you tell `ggplot2` how to map your variables and what options to use and `ggplot2` builds the plot. Note that is is just a sampling of what you can do with `ggplot2`, you can find plenty more plot types, customizations, etc. in the package documentation.

First, we load the packages:

```
# for plotting
library(ggplot2)
```

1. Basic syntax

```
# basic ggplot useage with geom and ... for additional aesthetics (col, fill, size, etc.)
df %>% ggplot(aes(x = col_name_1, y = col_name_2, ...)) +
  geom_*()
```

2. Aesthetics layer (`aes()`)

The aesthetics layer tells ggplot how to map our variables. We are going to cover the most common here.

x, y: map variables to the x and y axes

col: determine the colors based on unique values of this variable

fill: does the same as color but for bar charts

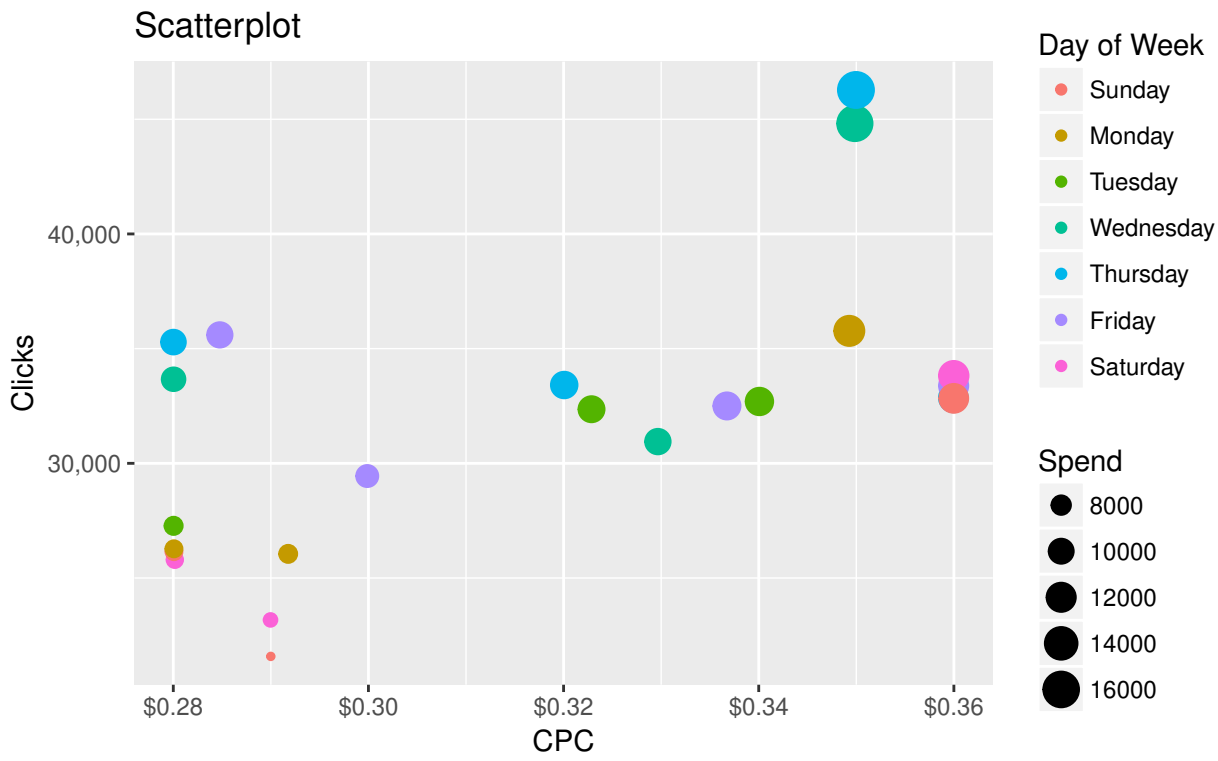
size: determine the size of points based on the value of this variable

3. Common geoms

The geom layer tells ggplot how to display the mapped variables.

`geom_point()` – scatterplot

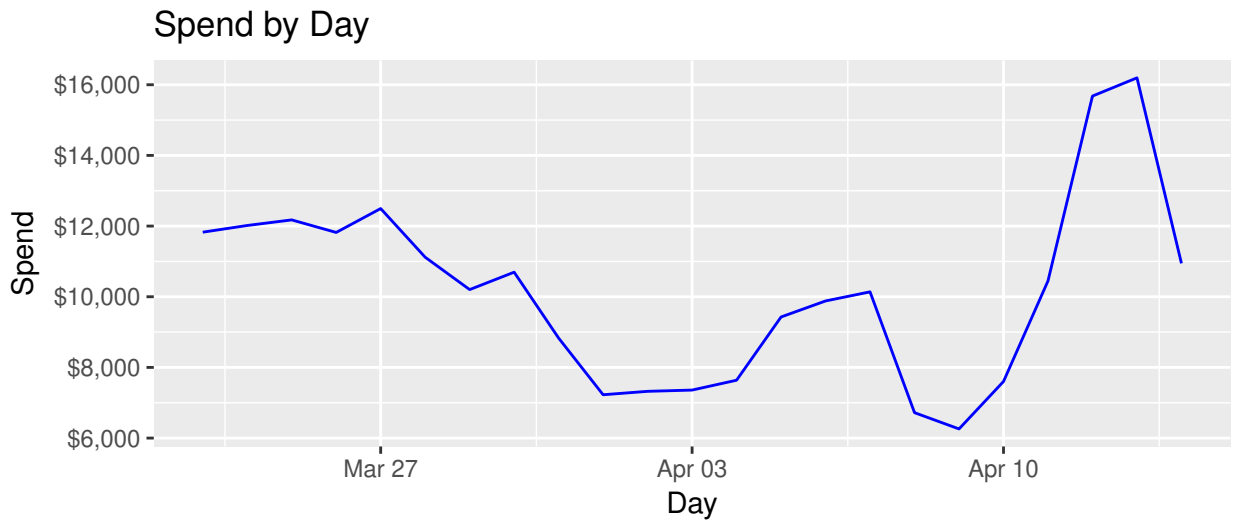
```
# scatterplot showing relation of CPC and clicks for [redacted]
# show days of the week as different colors sized by spend (just to illustrate use)

# define aesthetics
[redacted]_pivot %>% ggplot(aes(x = cpc, y = total_clicks,
                           col = factor(format(as.Date([redacted]_pivot$day), "%A"),
                                        levels = c("Sunday", "Monday", "Tuesday",
                                                   "Wednesday", "Thursday", "Friday",
                                                   "Saturday")),
                           size = spend)) +
  # add points
  geom_point() +

  # optional stuff to make it cleaner
  # format labels
  scale_x_continuous(labels = scales::dollar) +
  scale_y_continuous(labels = scales::comma) +

  # add title
  ggtitle("Scatterplot") +

  # fix axis labels
  xlab("CPC") +
  ylab("Clicks") +

  # fix legend titles (match to aes labels)
  labs(col = "Day of Week", size = "Spend")
```



That may look more complicated, but it is very readable and easy to add onto. Notice how we used `factors` here to show the days of the week in the order we wanted (using `levels`). We also cleaned up the axis labels and legend title since they get labeled exactly what you put into that value as the aesthetics, which can be confusing in the case of the `col` argument [`factor(format(as.Date([redacted]_pivot$day), "%A"), levels = c("Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"))`]. The "`::`" notation in `scale_x_continuous(labels = scales::dollar)` is not required, but it helps to see where the function the code is using came from.

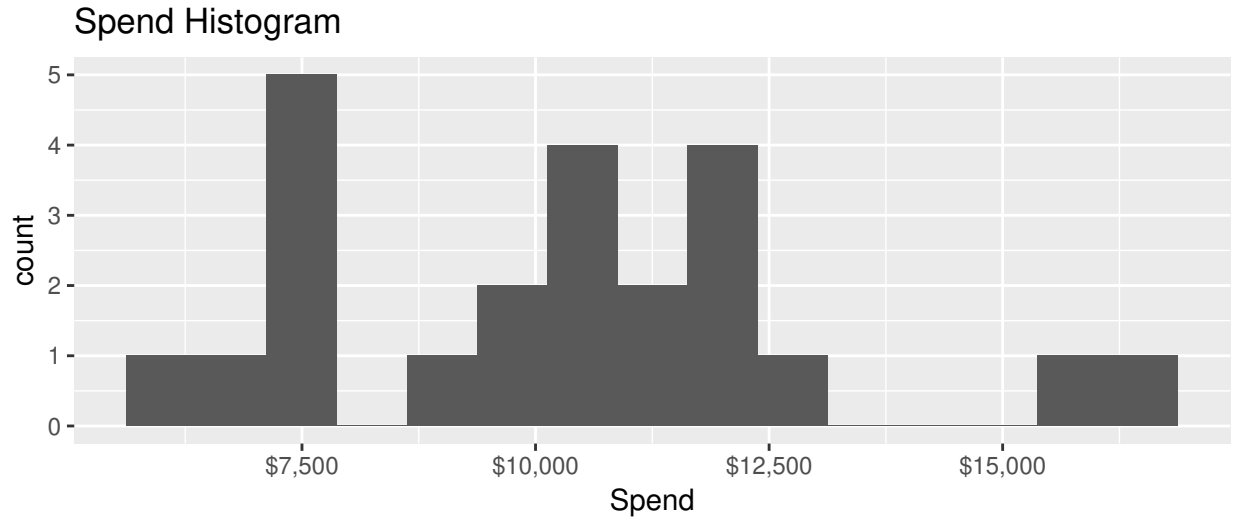
`geom_line()` – line graph

```
# spend by day for[redacted]

# define aesthetics
[redacted]_pivot %>% ggplot(aes(x = as.Date(day), y = spend)) +
  # add line
  geom_line(col = "blue") +

  # optional stuff to make it cleaner
  # format labels
  scale_y_continuous(labels = scales::dollar) +

  # add title
  ggtitle("Spend by Day") +

  # fix axis labels
  xlab("Day") +
  ylab("Spend")
```



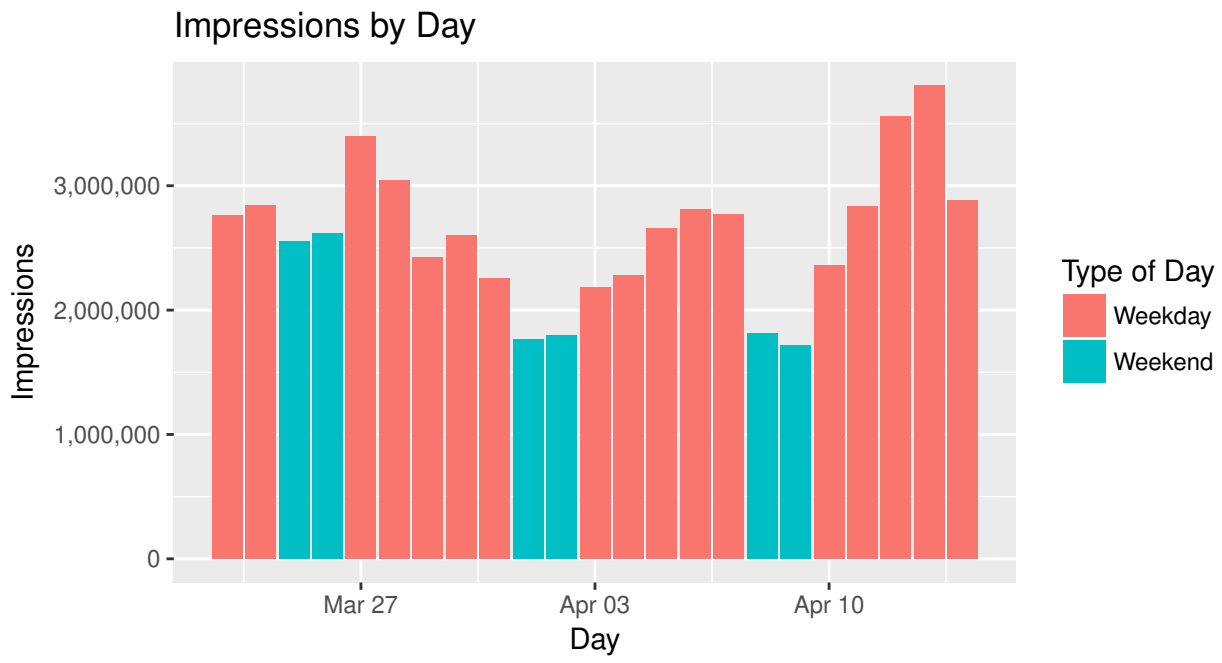
`geom_histogram()` – histogram

```
# histogram of [redacted] spend

# define aesthetics
[redacted]_pivot %>% ggplot(aes(x = spend)) +
  # add histogram
  geom_histogram(binwidth = 750) +

  # optional stuff to make it cleaner
  # format labels
  scale_x_continuous(labels = scales::dollar) +

  # add title
  ggtitle("Spend Histogram") +

  # fix axis labels
  xlab("Spend")
```



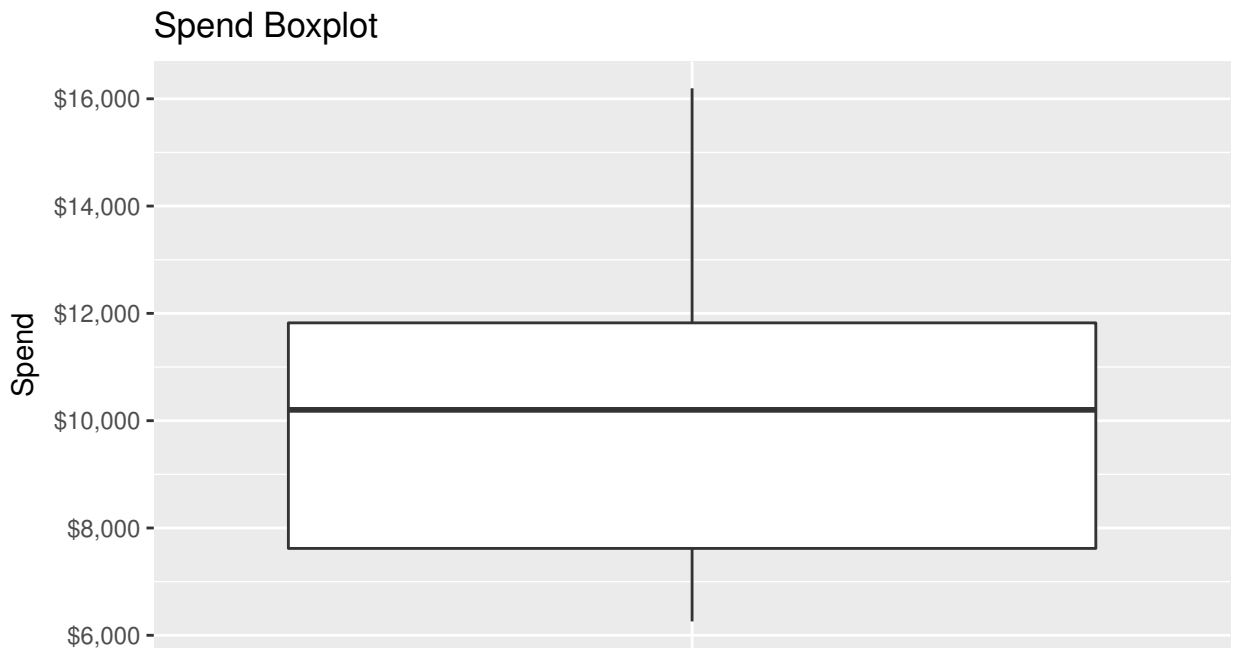
`geom_bar()` – bar chart of count `geom_col()` – bar chart of values

```
# bar chart counting days of week in this data set

# define aesthetics
[redacted]_pivot %>%
  ggplot(aes(x = as.Date(day), y = total_imps,
             fill = factor(
               ifelse(format(as.Date([redacted]_pivot$day), "%A") %in%
                   c("Sunday", "Saturday"),
                   "Weekend", "Weekday")))) +
  # add bars
  geom_col() +

  # optional stuff to make it cleaner
  # format labels
  scale_y_continuous(labels = scales::comma) +

  # add title
  ggtitle("Impressions by Day") +

  # fix axis labels
  xlab("Day") +
  ylab("Impressions")  +

  # fix legend title
  labs(fill = "Type of Day")
```



- `geom_boxplot()` – box and whisker plot

```
#[redacted]spend boxplot

# define aesthetics
[redacted]_pivot %>%
  ggplot(aes(x = factor(1L), y = spend)) +

  # add boxplot
  geom_boxplot() +

  # optional stuff to make it cleaner
  # format labels
  scale_y_continuous(labels = scales::dollar) +

  # add title
  ggtitle("Spend Boxplot") +

  # remove x-axis and fix y-axis labels
  theme(axis.title.x = element_blank(),
        axis.text.x = element_blank(),
        axis.ticks.x = element_blank()) +
  ylab("Spend")
```



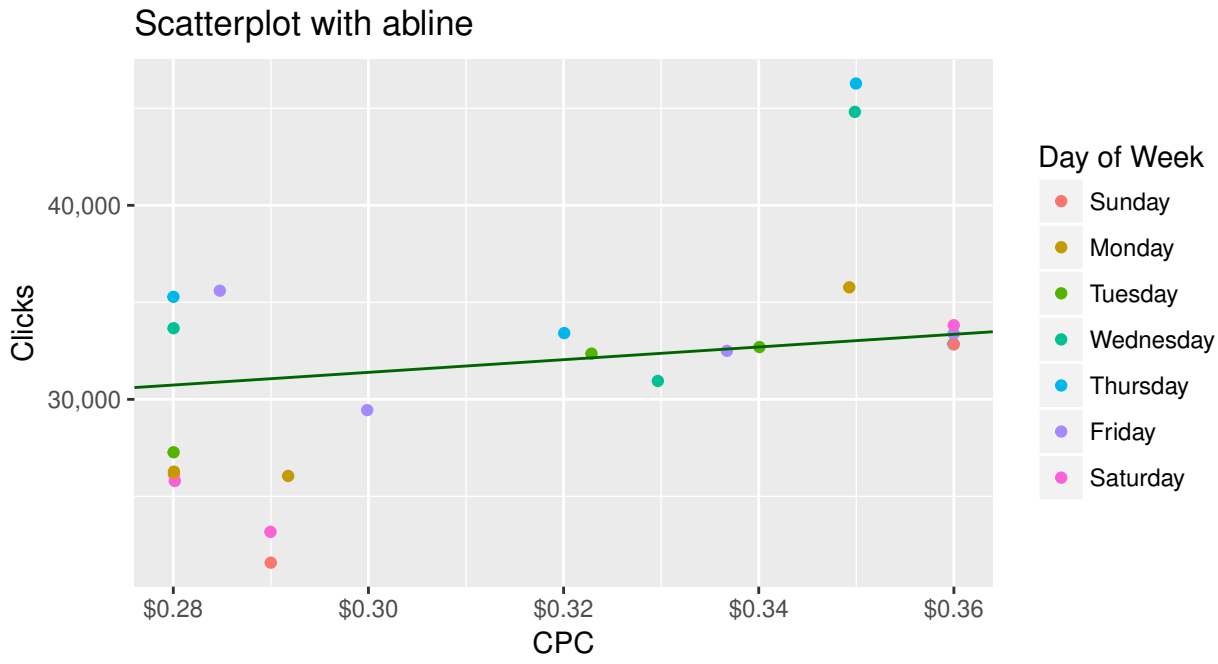
4. Layering on

i. `geom_abline()/geom_hline()/geom_vline()`

Add a line to your graph either by specifiying slope and intercept or the location for a vertical/horizontal line. These are the `ggplot2` versions of base R.

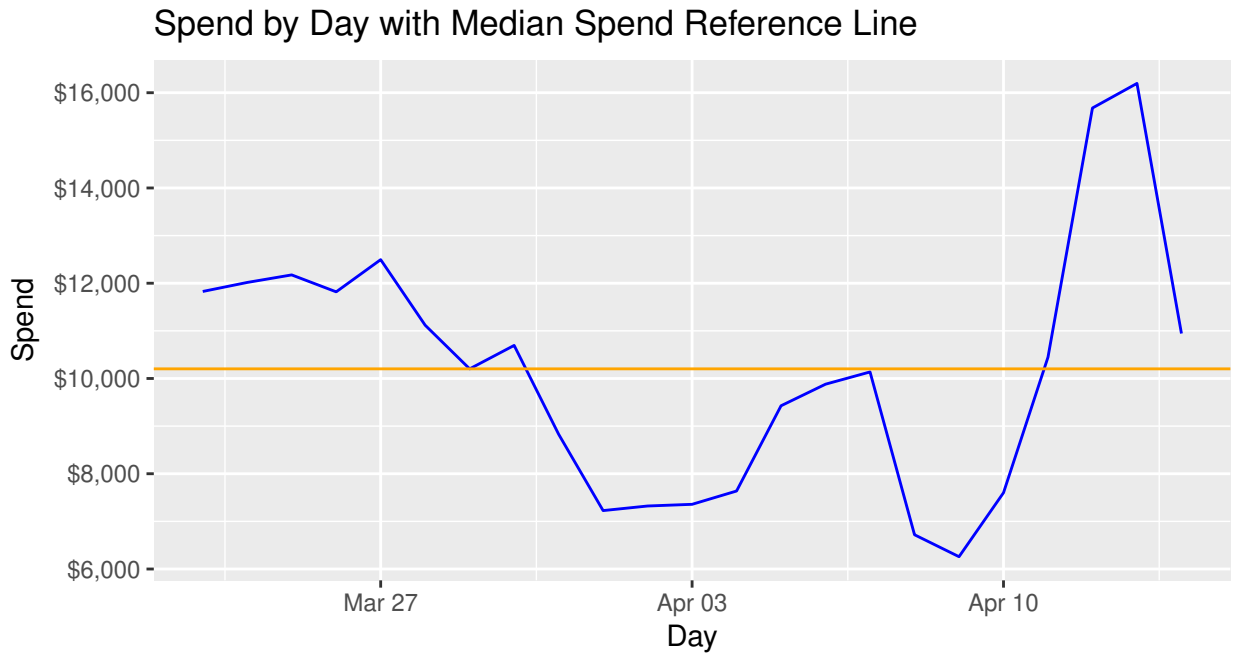
`geom_abline()` Add a darkgreen abline to the scatterplot.

```
# scatterplot showing relation of CPC and clicks for [redacted]
# show days of the week as different colors

# define aesthetics
[redacted]_pivot %>% ggplot(aes(x = cpc, y = total_clicks,
                           col = factor(format(as.Date([redacted]_pivot$day), "%A"),
                                        levels = c("Sunday", "Monday", "Tuesday",
                                                   "Wednesday", "Thursday", "Friday",
                                                   "Saturday")))) +

  # add points
  geom_point() +

  # optional stuff to make it cleaner
  # format labels
  scale_x_continuous(labels = scales::dollar) +
  scale_y_continuous(labels = scales::comma) +

  # add title
  ggtitle("Scatterplot with abline") +

  # fix axis labels
  xlab("CPC") +
  ylab("Clicks") +

  # fix legend title
  labs(col = "Day of Week") +

  # add line specifying slope and intercept
  geom_abline(intercept = min([redacted]_pivot$total_clicks),
              slope = median([redacted]_pivot$total_clicks), col = "darkgreen")
```



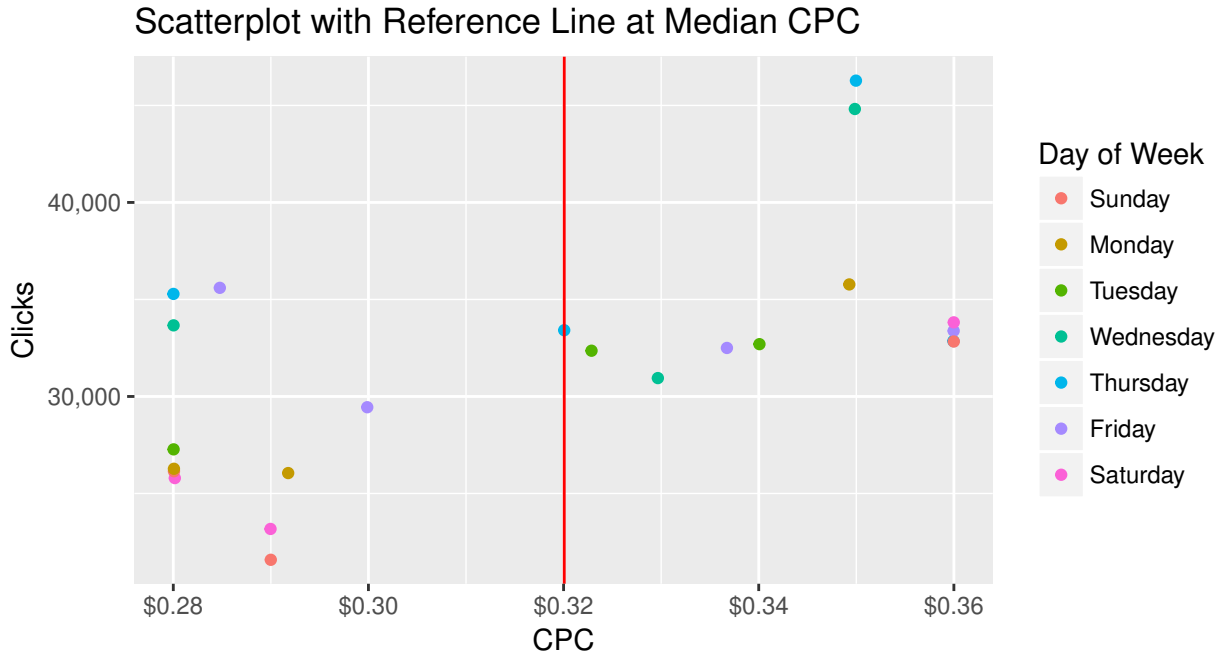
`geom_hline()` Add a orange horizontal line at median spend ($10,202.06).

```
# spend by day for [redacted]

# define aesthetics
[redacted]pivot %>%
  ggplot(aes(x = as.Date(day), y = spend)) +

  # add line
  geom_line(col = "blue") +

  # optional stuff to make it cleaner
  # format labels
  scale_y_continuous(labels = scales::dollar) +

  # add title
  ggtitle("Spend by Day with Median Spend Reference Line") +

  # fix axis labels
  xlab("Day") +
  ylab("Spend")  +

  # add reference line
  geom_hline(yintercept = median[redacted]pivot$spend), col = "orange")
```



`geom_vline()` Add a red vertical line at the median CPC ($0.32).

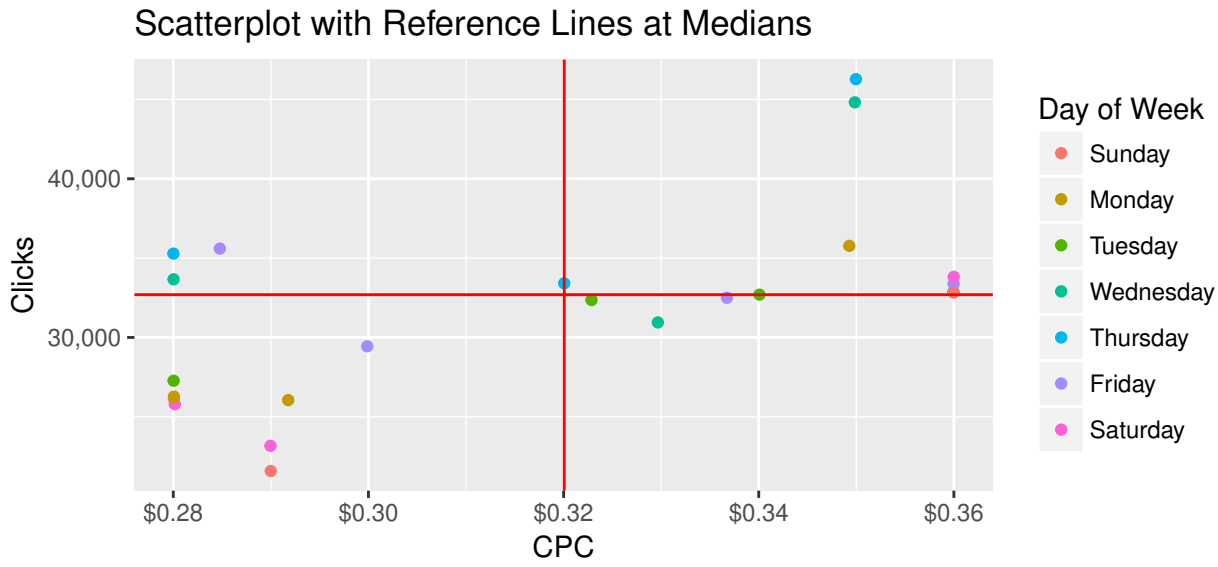
```
# scatterplot showing relation of CPC and clicks for [redacted]
# show days of the week as different colors

# define aesthetics
[redacted]_pivot %>% ggplot(aes(x = cpc, y = total_clicks,
                           col = factor(format(as.Date[redacted]_pivot$day), "%A"),
                                        levels = c("Sunday", "Monday", "Tuesday",
                                                   "Wednesday", "Thursday", "Friday",
                                                   "Saturday")))) +

  # add points
  geom_point() +

  # optional stuff to make it cleaner
  # format labels
  scale_x_continuous(labels = scales::dollar) +
  scale_y_continuous(labels = scales::comma) +

  # add title
  ggtitle("Scatterplot with Reference Line at Median CPC") +

  # fix axis labels
  xlab("CPC") +
  ylab("Clicks") +

  # fix legend title
  labs(col = "Day of Week") +

  # add reference line
  geom_vline(xintercept = median[redacted]_pivot$cpc), col = "red")
```



Note that these can also be combined (you can keep adding layers). Let's see what the median CPC and

median clicks look like on the scatterplot.

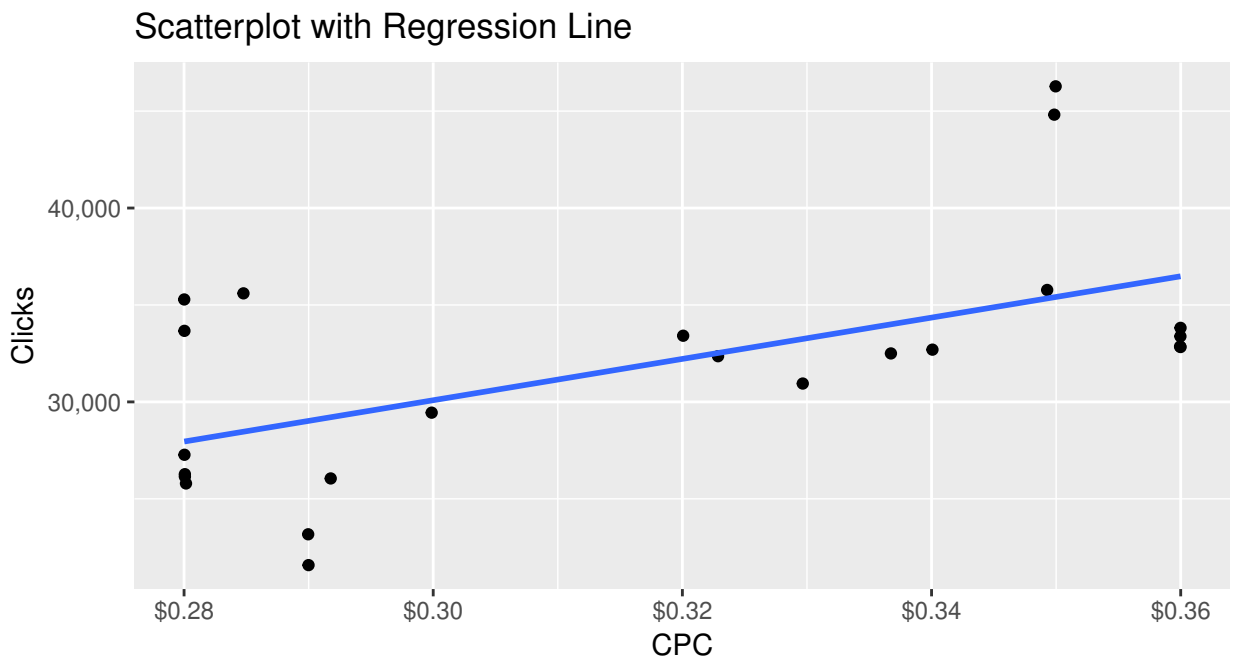
```
# scatterplot showing relation of CPC and clicks for [redacted]
# show days of the week as different colors

# define aesthetics
[redacted]_pivot %>% ggplot(aes(x = cpc, y = total_clicks,
                           col = factor(format(as.Date([redacted]_pivot$day), "%A"),
                                        levels = c("Sunday", "Monday", "Tuesday",
                                                   "Wednesday", "Thursday", "Friday",
                                                   "Saturday")))) +

  # add points
  geom_point() +

  # optional stuff to make it cleaner
  # format labels
  scale_x_continuous(labels = scales::dollar) +
  scale_y_continuous(labels = scales::comma) +

  # add title
  ggtitle("Scatterplot with Reference Lines at Medians") +

  # fix axis labels
  xlab("CPC") +
  ylab("Clicks") +

  # fix legend title
  labs(col = "Day of Week") +

  # add crosshairs for medians
  geom_vline(xintercept = median([redacted]_pivot$cpc), col = "red") +
  geom_hline(yintercept = median([redacted]_pivot$total_clicks), col = "red")
```



ii. `stat_smooth(method = "lm", se = FALSE, alpha = 0.6)`

Add a regression line with transparency 0.6 (`alpha`). Note that when plotting multiple items on a graph, it may be useful to adjust the transparency of the series so that they don't cover each other up. This can be done by specifying the `alpha` parameter with a value less than 1 (1 = opaque, 0 = transparent).

```
# scatterplot showing relation of CPC and clicks for [redacted]

# define aesthetics
[redacted]_pivot %>%
  ggplot(aes(x = cpc, y = total_clicks)) +

  # add points
  geom_point() +

  # optional stuff to make it cleaner
  # format labels
  scale_x_continuous(labels = scales::dollar) +
  scale_y_continuous(labels = scales::comma) +

  # add title
  ggtitle("Scatterplot with Regression Line") +

  # fix axis labels
  xlab("CPC") +
  ylab("Clicks") +

  # add regression line
  stat_smooth(method = "lm", se = FALSE, alpha = 0.6)
```

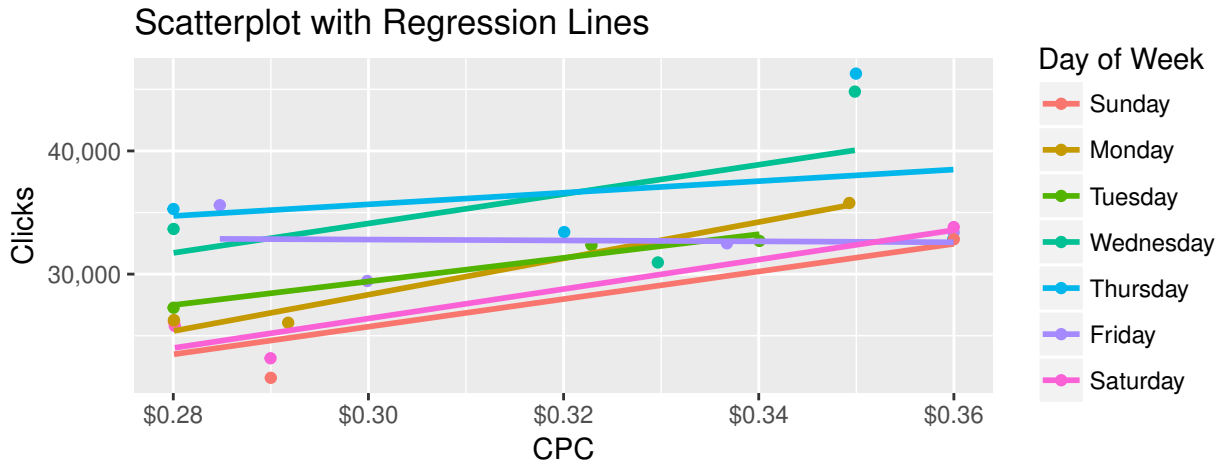


5. Faceting

What if we want to see metrics over categories, like events by site type or tag name? We could put these on the color aesthetic, but things get messy very quickly, and it gets hard to distinguish colors after 5 or 6 of them get added to the graph. Faceting helps us by breaking them out into separate graphs either wrapped or in a grid with all possible combinations, so we can see everything in one plot.

Look at the linear regression of the scatterplot with the day of week added to see the motivation behind this.

```
# scatterplot showing relation of CPC and clicks for [redacted]
# show days of the week as different colors

# define aesthetics
[redacted]_pivot %>% ggplot(aes(x = cpc, y = total_clicks,
                           col = factor(format(as.Date[redacted]_pivot$day), "%A"),
                                        levels = c("Sunday", "Monday", "Tuesday",
                                                   "Wednesday", "Thursday", "Friday",
                                                   "Saturday")))) +

  # add points
  geom_point() +

  # optional stuff to make it cleaner
  # format labels
  scale_x_continuous(labels = scales::dollar) +
  scale_y_continuous(labels = scales::comma) +

  # add title
  ggtitle("Scatterplot with Regression Lines") +

  # fix axis labels
  xlab("CPC") +
  ylab("Clicks") +

  # fix legend title
  labs(col = "Day of Week") +

  # add regression lines
  stat_smooth(method = "lm", se = FALSE, alpha = 0.6)
```

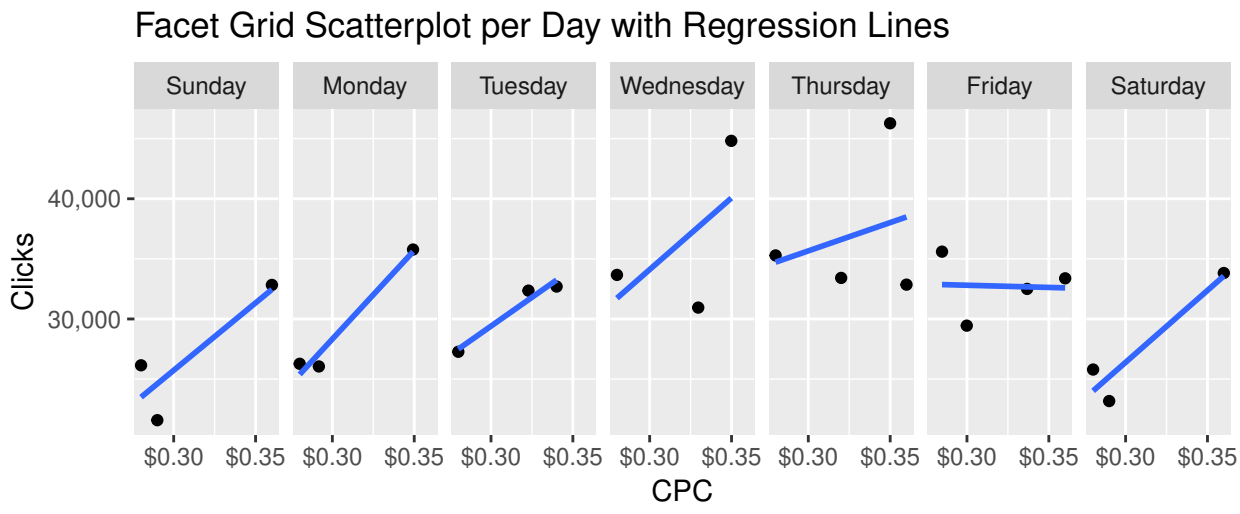


Wow, that's hard to read! Let's clean this up by faceting.

i. facet_grid()

Facet grids contain all possible combinations of the variables you want to facet on. Note that the "." here means you aren't faceting on anything to be printed along the y-axis. The "~" separates the rows facet from the columns facet. Here we only facet by columns with the day of week.

```
# scatterplot showing relation of CPC and clicks for [redacted]

# define aesthetics
[redacted]_pivot %>%
  ggplot(aes(x = cpc, y = total_clicks)) +

  # add points
  geom_point() +

  # format labels
  scale_x_continuous(labels = scales::dollar, breaks = scales::pretty_breaks(n = 2)) +
  scale_y_continuous(labels = scales::comma) +

  # add title
  ggtitle("Facet Grid Scatterplot per Day with Regression Lines") +

  # fix legend title and axis labels
  labs(col = "Day of Week", x = "CPC", y = "Clicks") +

  # add regression lines
  stat_smooth(method = "lm", se = FALSE, alpha = 0.6) +

  # add facet grid
  facet_grid(. ~ factor(format(as.Date([redacted]_pivot$day), "%A"),
                        levels = c("Sunday", "Monday", "Tuesday", "Wednesday",
                                   "Thursday", "Friday", "Saturday")))
```

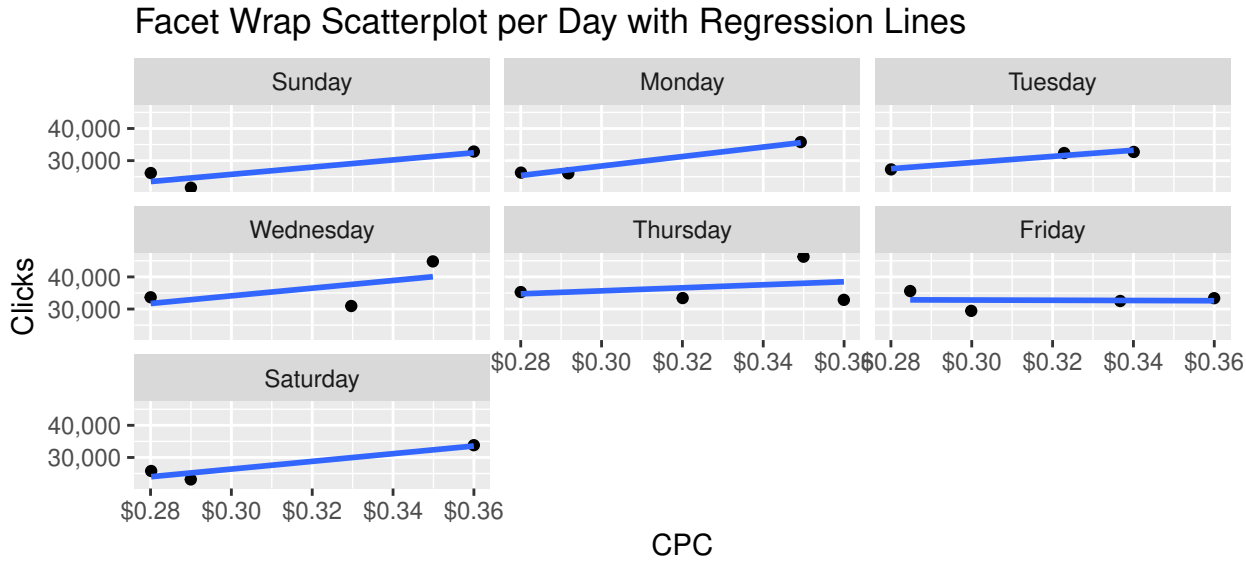


ii. `facet_wrap()`

Facet wrap only contains all possible combinations for which there is data to fill the intersection of a given row and column whereas facet grids will have a blank section if there is no data. This is why facet grids will not have whitespace and facet wraps will. Facet wraps, just as the name implies, create a series of plots from left to right and wrap to the next rows until they have plotted all combinations with data.

Note that here we don't need to add "." if we aren't using that faceting element as we did with ***`facet_grid()`***

```
# scatterplot showing relation of CPC and clicks for [redacted]

# define aesthetics
[redacted]pivot %>% ggplot(aes(x = cpc, y = total_clicks)) +
  # add points
  geom_point() +

  # format labels
  scale_x_continuous(labels = scales::dollar) +
  scale_y_continuous(labels = scales::comma) +

  # add title
  ggtitle("Facet Wrap Scatterplot per Day with Regression Lines") +

  # fix legend title and axis labels
  labs(col = "Day of Week", x = "CPC", y = "Clicks") +

  # add regression lines
  stat_smooth(method = "lm", se = FALSE, alpha = 0.6) +

  # add facet wrap
  facet_wrap(~ factor(format(as.Date([redacted]pivot$day), "%A"),
                      levels = c("Sunday", "Monday", "Tuesday",
                                 "Wednesday", "Thursday", "Friday",
                                 "Saturday")))
```



Notice how we have empty space the size of 2 plots next to Saturday in this graph, but we didn't have that with `facet_grid()`. Which you use depends on your data and what you are looking to show.

II. Exercises

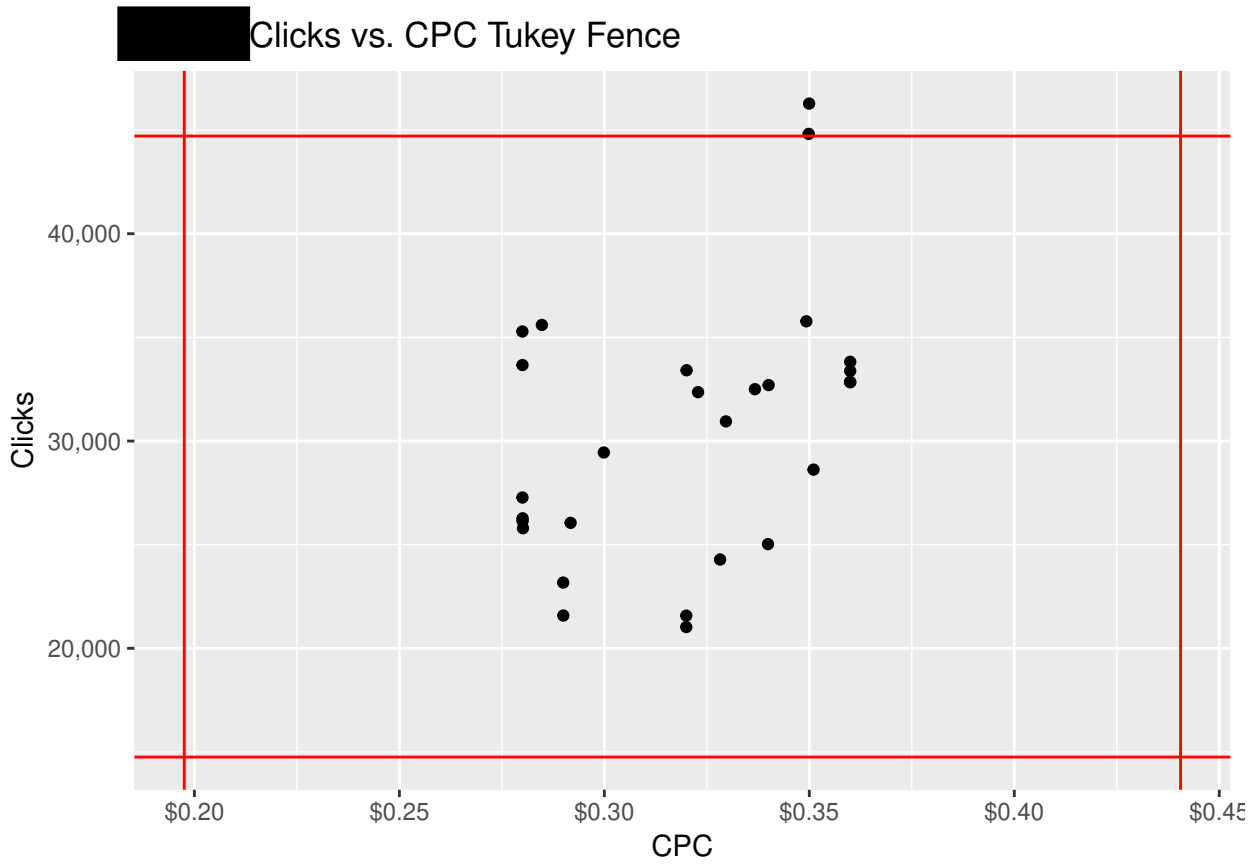
Let's do some practice problems to challenge your understanding of `ggplot2` and review the material from the prior lessons.

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)
```

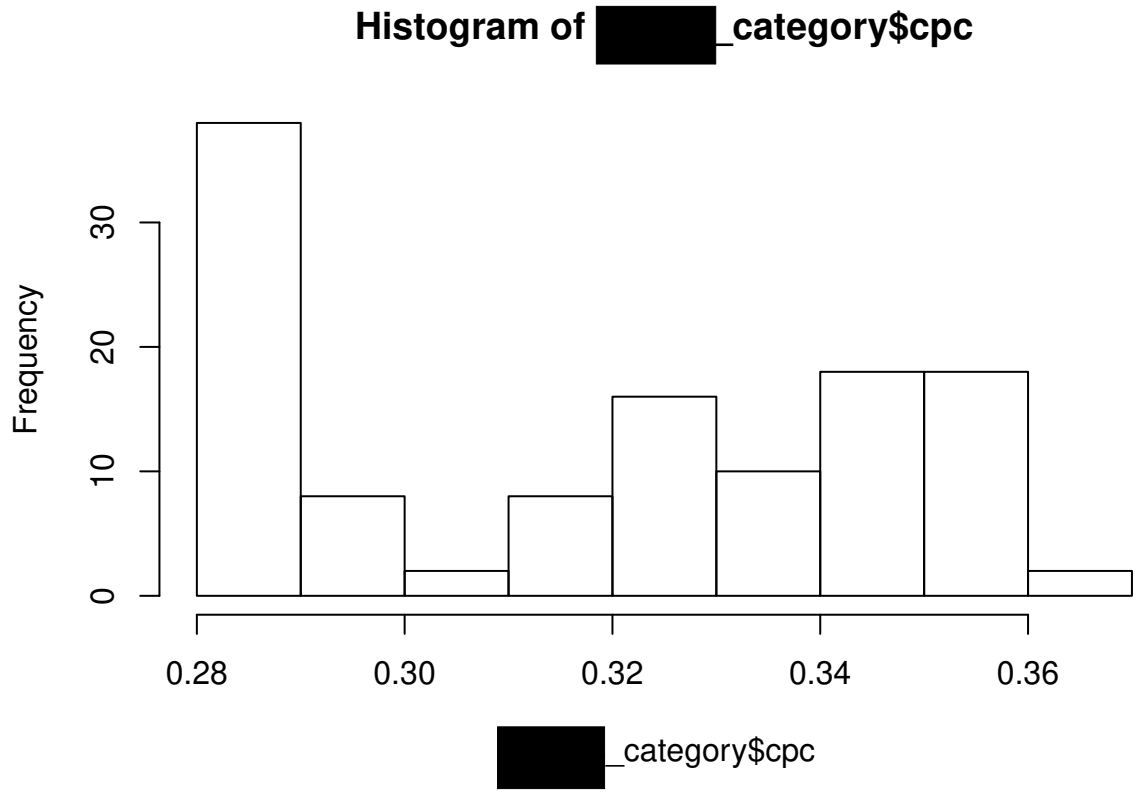
```r
# generate plot
[redacted]_pivot %>%
  ggplot(aes(x = cpc, y = total_clicks)) +
  geom_point() +
  geom_hline(yintercept = quantile([redacted]_pivot$total_clicks, 0.25) -
               1.5 * IQR([redacted]_pivot$total_clicks), col = "red") +
  geom_hline(yintercept = quantile([redacted]_pivot$total_clicks, 0.75) +
               1.5 * IQR([redacted]_pivot$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)
```



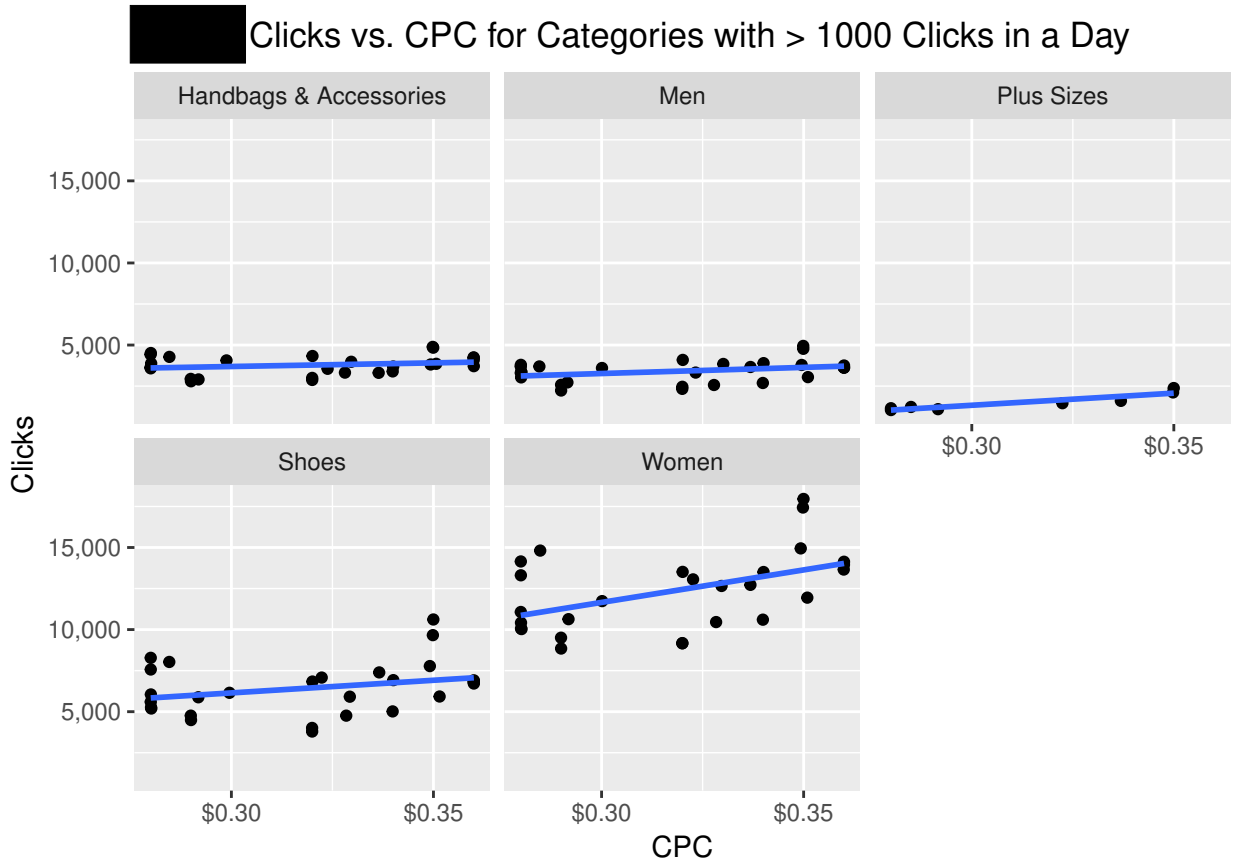
2. Pull clicks and CPC by category by day for the last 30 days on the client of your choice (choose a category level that makes sense). Create a quick base R histogram to see the distribution of CPC. Then, create a scatterplot of CPC and clicks faceted by category (be sure to try both `facet_wrap()` 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
"
```

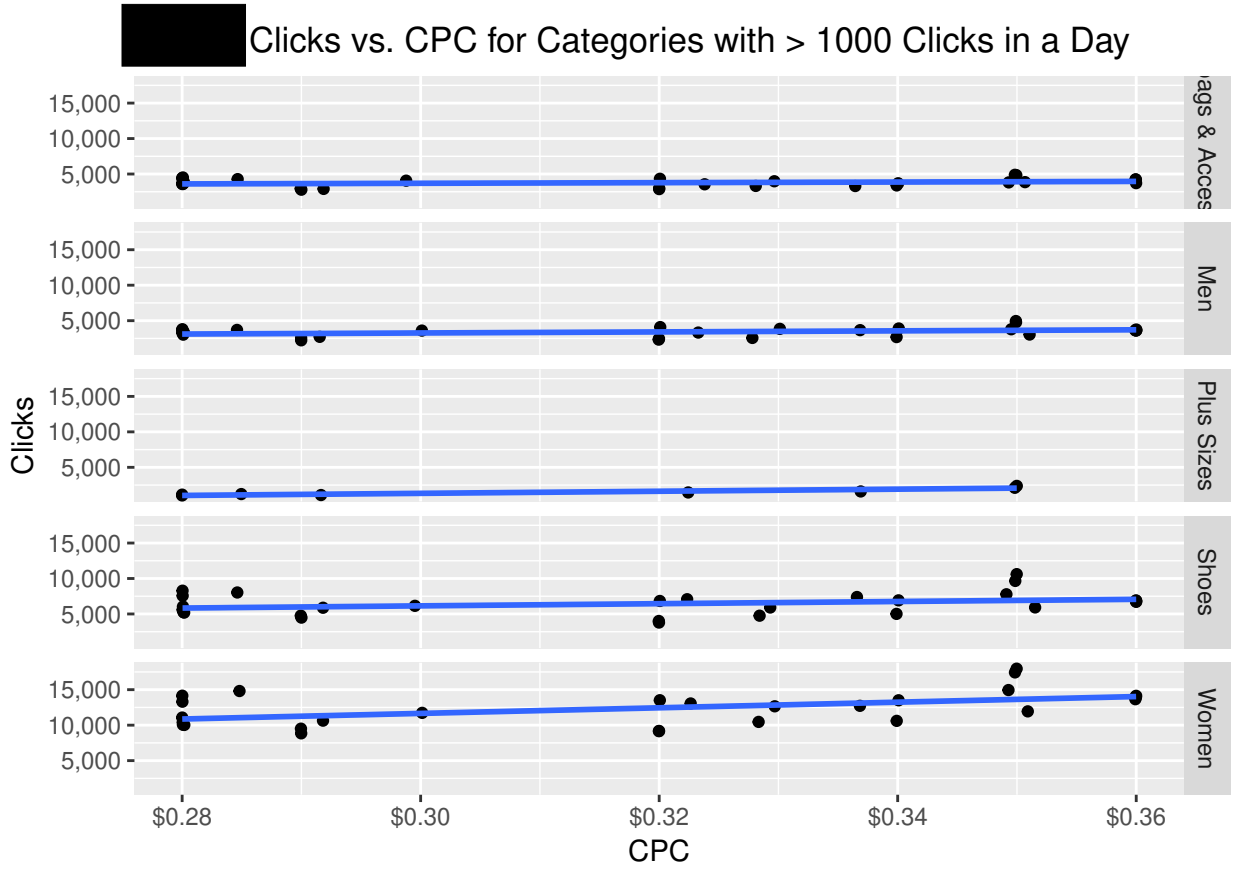
```
[redacted]_category <- QueryVertica(username,
                                    sprintf(query, category_level, merchant_id,
                                            merchant_id, category_level), password)

# 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)
```



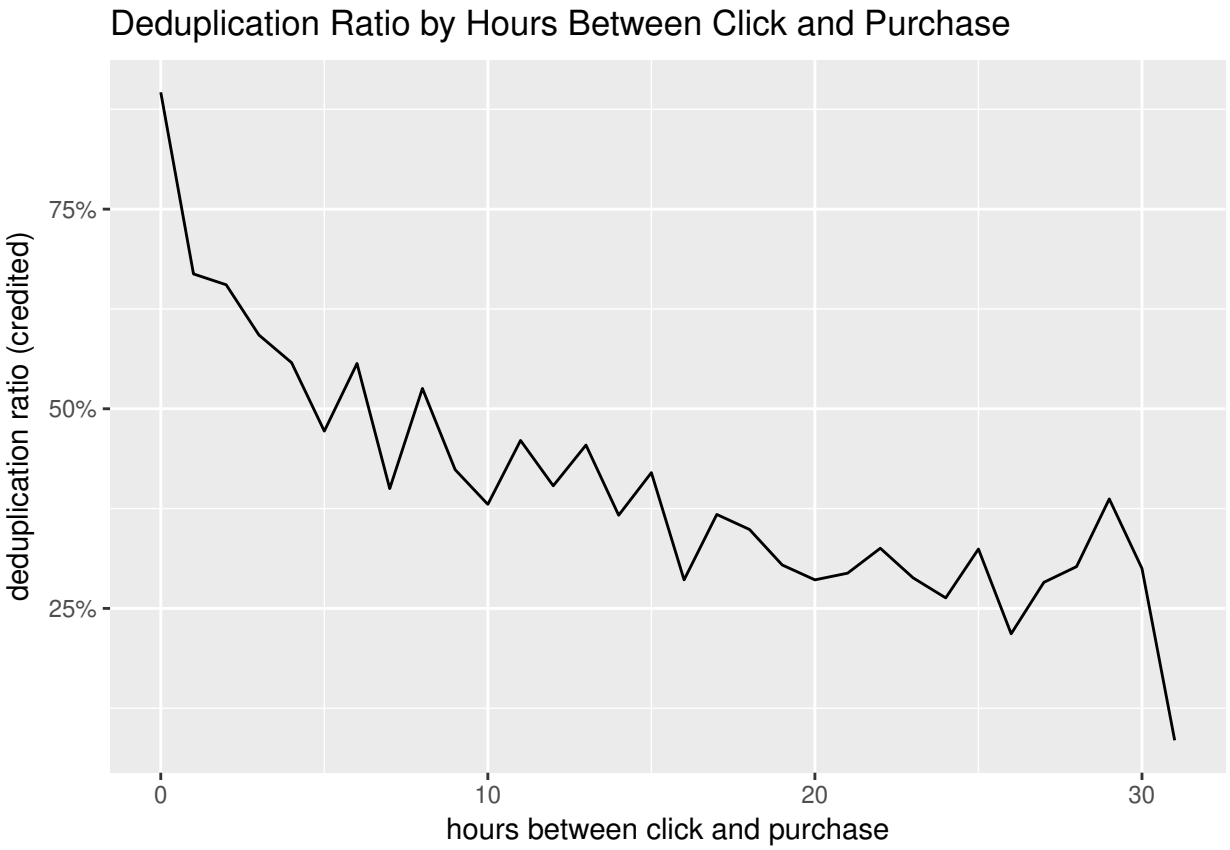
```
# make ggplot facet_grid
[redacted]_category %>%
  ggplot(aes(x = cpc, y = clicks)) +
  geom_point() +
  facet_grid(category ~ .) +
  scale_y_continuous(name = "Clicks", labels = scales::comma) +
  scale_x_continuous(name = "CPC", labels = scales::dollar) +
  ggtitle("[redacted] Clicks vs. CPC for 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