SSAC_2023

Lorenzo Dube and Xander Schwartz

2023-03-05

Data Cleaning Output of the below cell is two dataframes: 'continuous_df' which is all of the continuous data from the dataset and 'categorical_df' which is all of the categorical data. Those with two categories have been 'one hot' encoded.

```
cat_data_read <- function(behavior, customer) {
customer = read_csv(customer)
behavior = read_csv(behavior)

customer <- customer[, colSums(!is.na(customer)) > 0] #remove all NA only columns
customer <- customer[, !grepl("input_indv", names(customer))] #remove input_indv columns
behavior <- behavior[, colSums(!is.na(behavior)) > 0] #remove all NA only columns

df <- merge(customer, behavior, by = "acct_id")

one_hot_column_list <- sapply(df, function(x) all(is.na(x) | x %in% c(0,1)))
cont_value_column_list = !one_hot_column_list
one_hot_column_list[1] = TRUE

one_hot_df <- df[, one_hot_column_list]
one_hot_df[is.na(one_hot_df)] <- 0

cont_val_df = df[, cont_value_column_list]
cont_val_list2 <- sapply(cont_val_df, function(col) all(is.numeric(col) | is.na(col)))
cont_val_list2[1] = TRUE
continuous_df <- cont_val_df[, cont_val_list2]

other_cat_list = !cont_val_list2
other_cat_list[1] = TRUE
other_cat_df <- cont_val_df[, other_cat_list]

categorical_df = merge(one_hot_df, other_cat_df, by = 'acct_id')}


cont_data_read <- function(behavior, customer) {
customer = read_csv(customer)
behavior = read_csv(behavior)

customer <- customer[, colSums(!is.na(customer)) > 0] #remove all NA only columns
customer <- customer[, !grepl("input_indv", names(customer))] #remove input_indv columns
behavior <- behavior[, colSums(!is.na(behavior)) > 0] #remove all NA only columns

df <- merge(customer, behavior, by = "acct_id")
```

```
one_hot_column_list <- sapply(df, function(x) all(is.na(x) | x %in% c(0,1)))
cont_value_column_list = !one_hot_column_list
one_hot_column_list[1] = TRUE

one_hot_df <- df[, one_hot_column_list]
one_hot_df[is.na(one_hot_df)] <- 0

cont_val_df = df[, cont_value_column_list]
cont_val_list2 <- sapply(cont_val_df, function(col) all(is.numeric(col) | is.na(col)))
cont_val_list2[1] = TRUE
continuous_df <- cont_val_df[, cont_val_list2]}

kc_categorical= cat_data_read('KC_Behavior.csv', 'KC_Cust.csv')
```

```
## Warning: One or more parsing issues, see `problems()` for details

## Rows: 25000 Columns: 264
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (37): cust_source_cd, la_id, cust_city_nm, cust_state_nm, cust_ctry_nm,...
## dbl (188): ult_party_id, acct_id, cust_postal_cd, age_two_yr_incr_input_indv...
## lgl  (39): cust_first_nm, cust_middle_nm, cust_last_nm, cust_addr_line_1, cu...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 25000 Columns: 65
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): cust_source_cd, la_id
## dbl (35): ult_party_id, acct_id, behavior_score_05, behavior_score_06, behav...
## lgl (28): dim_08_ind, dim_10_ind, dim_25_ind, dim_26_ind, dim_27_ind, dim_28...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
kc_continuous = cont_data_read('KC_Behavior.csv', 'KC_Cust.csv')
```

```
## Warning: One or more parsing issues, see `problems()` for details

## Rows: 25000 Columns: 264
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (37): cust_source_cd, la_id, cust_city_nm, cust_state_nm, cust_ctry_nm,...
## dbl (188): ult_party_id, acct_id, cust_postal_cd, age_two_yr_incr_input_indv...
## lgl  (39): cust_first_nm, cust_middle_nm, cust_last_nm, cust_addr_line_1, cu...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 25000 Columns: 65
## -- Column specification --------------------------------------------------------
## Delimiter: ","
```

```
## chr  (2): cust_source_cd, la_id
## dbl (35): ult_party_id, acct_id, behavior_score_05, behavior_score_06, behav...
## lgl (28): dim_08_ind, dim_10_ind, dim_25_ind, dim_26_ind, dim_27_ind, dim_28...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
lv_categorical = cat_data_read('Lville_Behavior.csv', 'LVille_Cust.csv')
```

```
## New names:
## * `` -> `...265`
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

```
## Rows: 25000 Columns: 265
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (37): cust_source_cd, la_id, cust_city_nm, cust_state_nm, cust_ctry_nm,...
## dbl (189): ult_party_id, acct_id, cust_postal_cd, age_two_yr_incr_input_indv...
## lgl  (39): cust_first_nm, cust_middle_nm, cust_last_nm, cust_addr_line_1, cu...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 25000 Columns: 65
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): cust_source_cd, la_id
## dbl (35): ult_party_id, acct_id, behavior_score_05, behavior_score_06, behav...
## lgl (28): dim_08_ind, dim_10_ind, dim_25_ind, dim_26_ind, dim_27_ind, dim_28...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
lv_continuous = cont_data_read('Lville_Behavior.csv', 'LVille_Cust.csv')
```

```
## New names:
## * `` -> `...265`
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

```
## Rows: 25000 Columns: 265
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (37): cust_source_cd, la_id, cust_city_nm, cust_state_nm, cust_ctry_nm,...
## dbl (189): ult_party_id, acct_id, cust_postal_cd, age_two_yr_incr_input_indv...
## lgl  (39): cust_first_nm, cust_middle_nm, cust_last_nm, cust_addr_line_1, cu...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 25000 Columns: 65
## -- Column specification --------------------------------------------------------
```

```
## Delimiter: ","
## chr  (2): cust_source_cd, la_id
## dbl (35): ult_party_id, acct_id, behavior_score_05, behavior_score_06, behav...
## lgl (28): dim_08_ind, dim_10_ind, dim_25_ind, dim_26_ind, dim_27_ind, dim_28...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
vb_categorical = cat_data_read('VB_Behavior2.csv', 'VB_Cust.csv')
```

```
## New names:
## * `` -> `...265`
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

```
## Rows: 25000 Columns: 265
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (37): cust_source_cd, la_id, cust_city_nm, cust_state_nm, cust_ctry_nm,...
## dbl (188): ult_party_id, acct_id, cust_postal_cd, age_two_yr_incr_input_indv...
## lgl  (40): cust_first_nm, cust_middle_nm, cust_last_nm, cust_addr_line_1, cu...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 25000 Columns: 65
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): cust_source_cd, la_id
## dbl (35): ult_party_id, acct_id, behavior_score_05, behavior_score_06, behav...
## lgl (28): dim_08_ind, dim_10_ind, dim_25_ind, dim_26_ind, dim_27_ind, dim_28...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
vb_continuous = cont_data_read('VB_Behavior2.csv', 'VB_Cust.csv')
```

```
## New names:
## * `` -> `...265`
```

```
## Warning: One or more parsing issues, see `problems()` for details
```

```
## Rows: 25000 Columns: 265
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (37): cust_source_cd, la_id, cust_city_nm, cust_state_nm, cust_ctry_nm,...
## dbl (188): ult_party_id, acct_id, cust_postal_cd, age_two_yr_incr_input_indv...
## lgl  (40): cust_first_nm, cust_middle_nm, cust_last_nm, cust_addr_line_1, cu...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 25000 Columns: 65
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): cust_source_cd, la_id
## dbl (35): ult_party_id, acct_id, behavior_score_05, behavior_score_06, behav...
## lgl (28): dim_08_ind, dim_10_ind, dim_25_ind, dim_26_ind, dim_27_ind, dim_28...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
categorical <- lv_categorical %>% mutate(CityName = "Lousiville") %>%
  bind_rows(vb_categorical %>%  mutate(CityName = "Virginia Beach")) %>%
  bind_rows(kc_categorical %>% mutate(CityName = "Kansas City"))

continuous <- lv_continuous %>% mutate(CityName = "Lousiville") %>%
  bind_rows(vb_continuous %>%  mutate(CityName = "Virginia Beach")) %>%
  bind_rows(kc_continuous %>% mutate(CityName = "Kansas City")) %>%
  mutate(personicx_group  = factor(case_when(psx_group_id <= 2 ~ "Youth",
                                     psx_group_id <= 8 ~ "CareerBuilding",
                                     psx_group_id <= 14 ~ "Earning",
                                     psx_group_id <= 20~ "LateCareer",
                                     psx_group_id > 20 ~ "Retired",
                                     T ~ "NA"
                                     ),ordered = T))

allData <- left_join(continuous, categorical) %>%
  filter(brkr_ind == 0|is.na(brkr_ind))
```

```
## Joining, by = c("acct_id", "CityName", "brkr_ind")
```

Removing all accounts for which there are multiple lines for whatever reason. ~1000 observations removed.

```
counts_cat <-
  categorical %>%
  group_by(acct_id) %>%
  summarise(ct = n()) %>%
  arrange(ct)

table(counts_cat$ct)
```

```
##
##     1    16    81
## 72972   993    14
```

```
counts_cont <-
  categorical %>%
  group_by(acct_id) %>%
  summarise(ct = n()) %>%
  arrange(ct)

table(counts_cont$ct)
```

```
##
##     1    16    81
## 72972   993    14
```

```
categorical_minus_duplicates <-
  categorical %>%
  left_join(counts_cat) %>%
  filter(ct == 1) %>%
  select(-ct)
```

```
## Joining, by = "acct_id"
```

```
continuous_minus_duplicates <-
  continuous %>%
  left_join(counts_cont) %>%
  filter(ct == 1) %>%
  select(-ct)
```

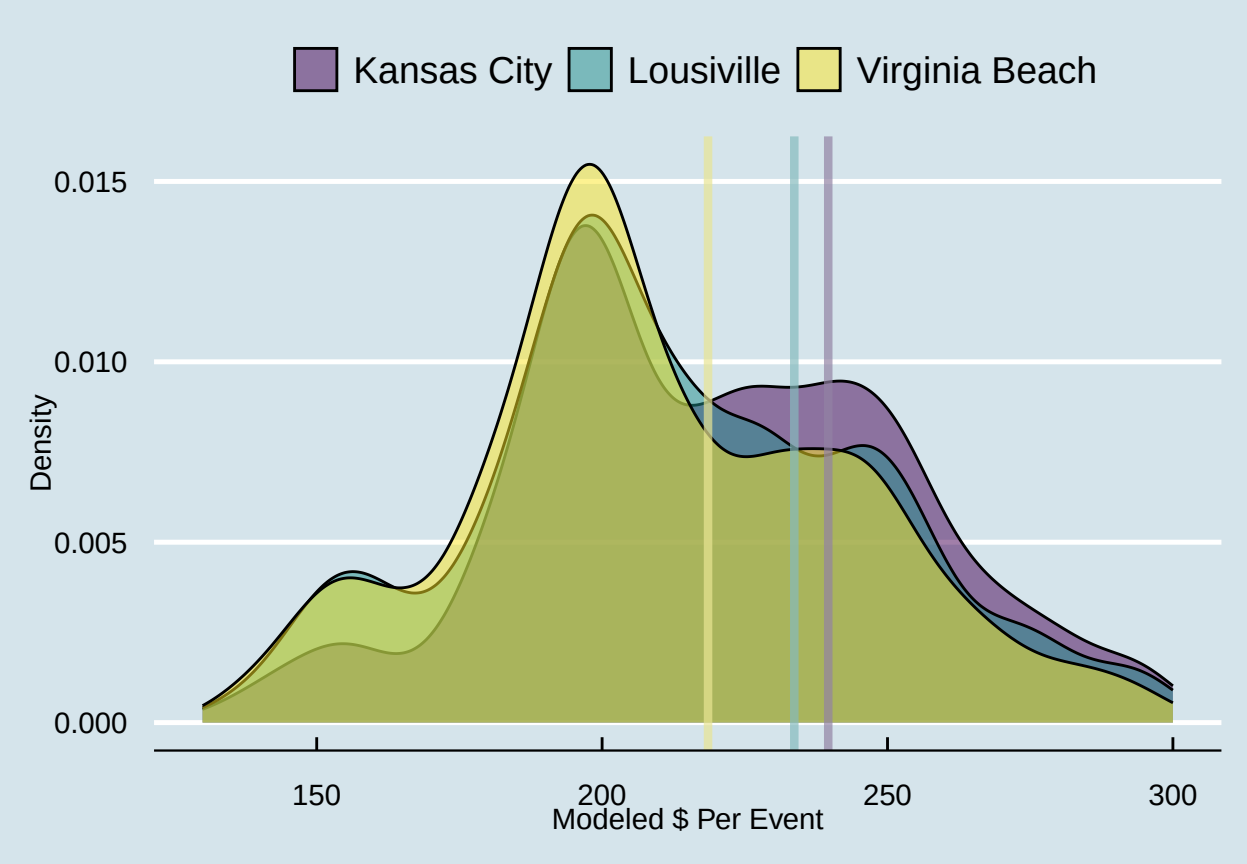
```
## Joining, by = "acct_id"
```

```
data_train <- allData %>% filter(!is.na(e3_spend_pe_m_sports) & !is.na(financial_pct_score) & !is.na(ps

    model <- randomForest::randomForest(e3_spend_pe_m_sports ~ financial_pct_score + personicx_group + 

save(file = "model.rda", model)

data_train$predicted_dollars <- predict(object = model, data = data_train)

data_train <- data_train%>%mutate(PredictedTotalSpend = predicted_dollars * e3_events_cnt)

spend_by_city <- data_train %>%
  group_by(CityName) %>%
  summarise(City_Spend = mean(predicted_dollars),
            City_Spend_sd = sd(predicted_dollars),
            total_spend = sum(predicted_dollars),
            sample_size = n())
```

```
ggplot(data_train, aes(x= predicted_dollars, fill = CityName)) +
  geom_density(alpha= .5) +
  xlim(130,300) +
  labs( x = "Modeled $ Per Event", fill = "", y = "Density") +
  scale_fill_viridis_d() +
  geom_vline(xintercept = spend_by_city$City_Spend, color = c("#9283a4", "#85bbbd", "#e9e593"), size =1
  ggthemes::theme_economist()
```

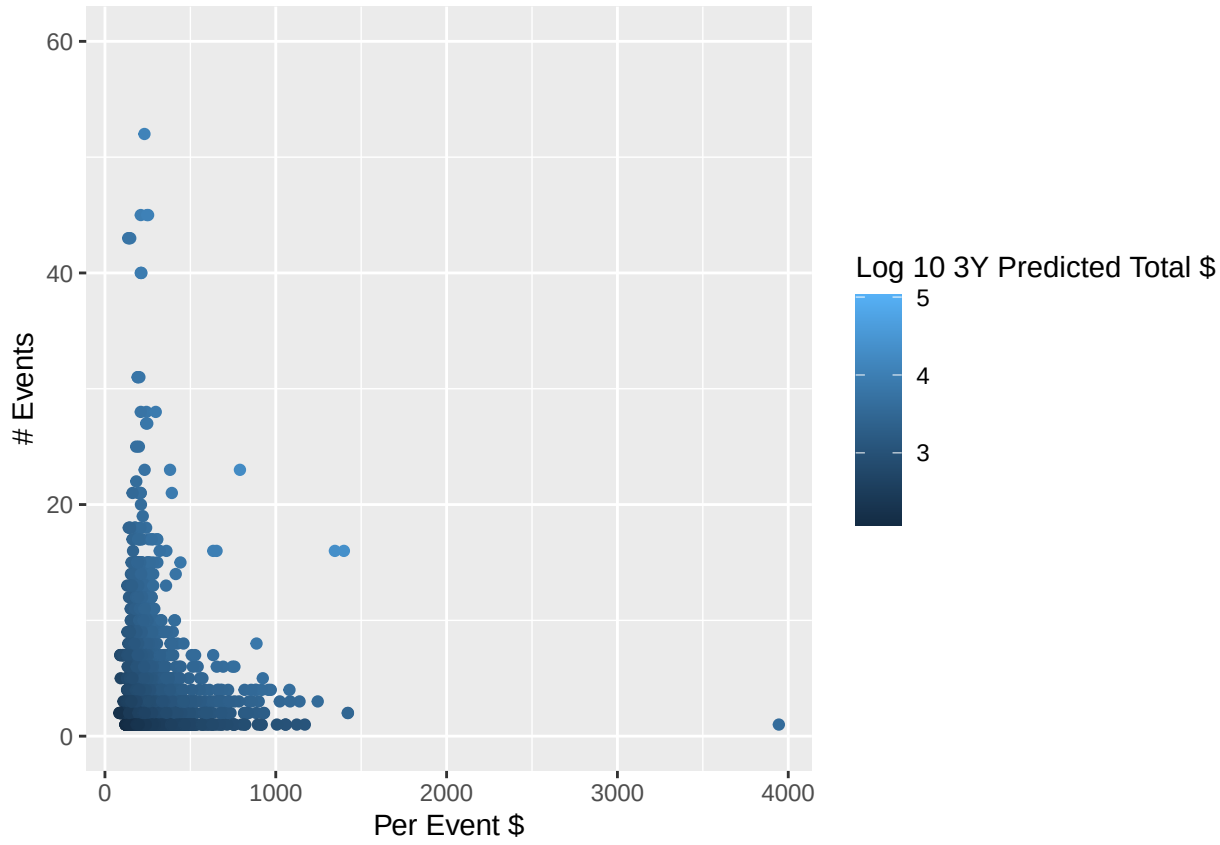
```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
```

```
## Warning: Removed 804 rows containing non-finite values (‘stat_density()‘).
```



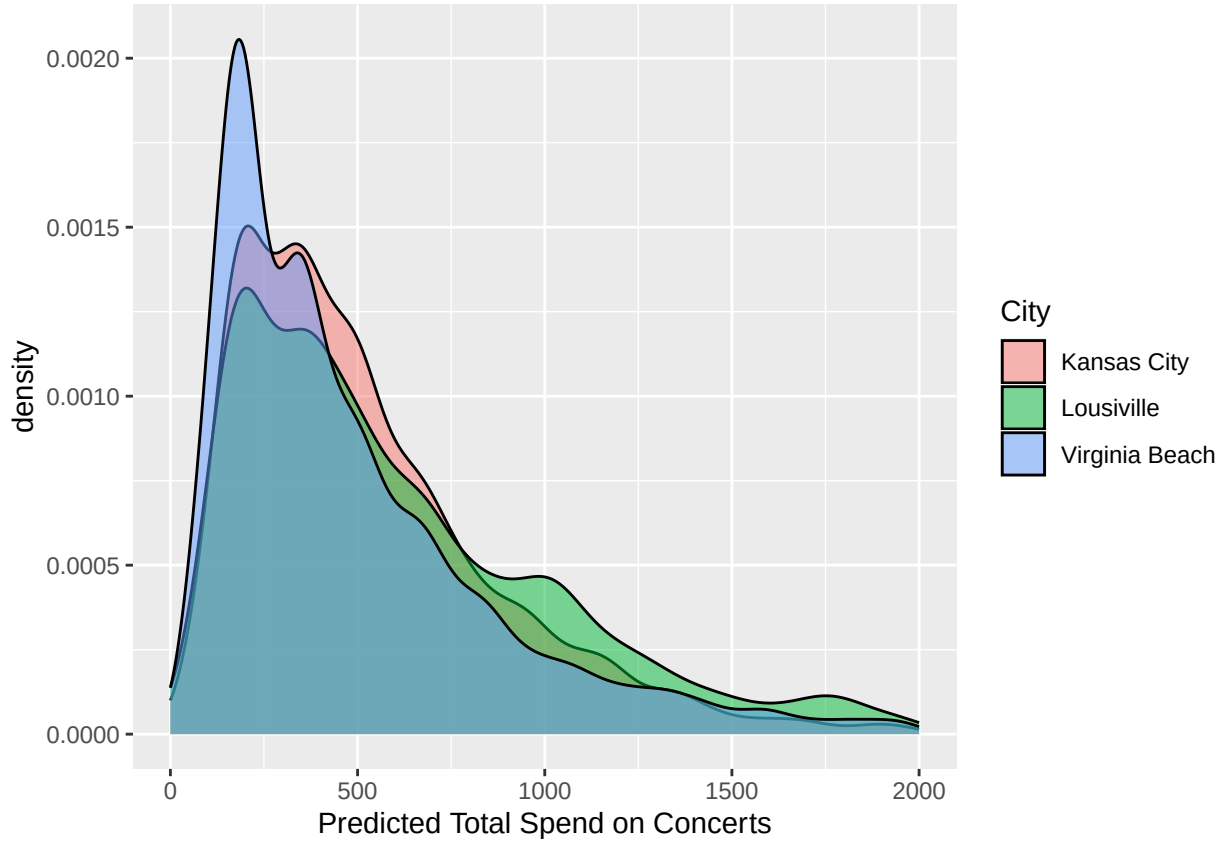

```
ggplot(data_train, aes(x = predicted_dollars, col = log10(PredictedTotalSpend), y = e3_events_cnt)) +
  geom_point() +
  labs(x = "Per Event $", y = "# Events", col = "Log 10 3Y Predicted Total $") +
  ylim(0,60)
```

```
## Warning: Removed 3 rows containing missing values (‘geom_point()‘).
```




```
data_train_concert <- allData %>% filter(!is.na(e3_spend_pe_m_concerts) & !is.na(financial_pct_score) &


model_concert <- randomForest::randomForest(e3_spend_pe_m_concerts ~ financial_pct_score + personicx_gr
data_train_concert$predicted_dollars_concert <- predict(object = model_concert, data = data_train_conce
data_train_concert <- data_train_concert%>%mutate(PredictedTotalSpendConcert = predicted_dollars_concer
ggplot(data_train_concert, aes(x= PredictedTotalSpendConcert, fill = CityName)) + geom_density(alpha= .
```

```
## Warning: Removed 358 rows containing non-finite values (‘stat_density()‘).
```




```
spend_by_city_concert <- data_train_concert %>%
  group_by(CityName) %>%
  summarise(City_Spend_concert = mean(predicted_dollars_concert),
            City_Spend_sd_concert = sd(predicted_dollars_concert),
            total_spend_concert = sum(predicted_dollars_concert),
            sample_size_concert = n())
```

Zip Codes

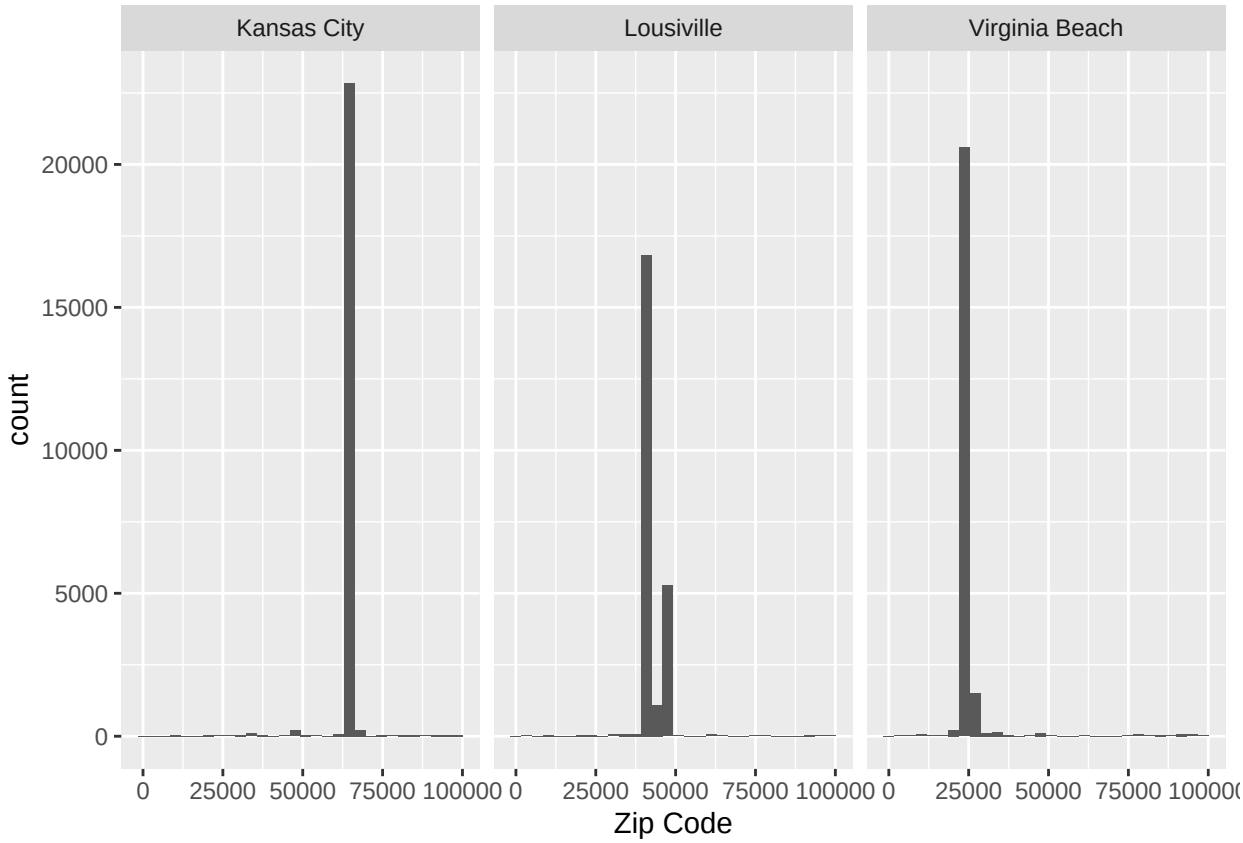
Hardcoding roughly where we expect the stadiums to be built

```
kc_zip <- 64129 #arrowhead
vb_zip <- 23456 #vb sports complex
lville_zip <- 40208 #cardinal stadium
```

```
ggplot(continuous, aes(x= cust_postal_cd )) +
  geom_histogram() +
  facet_wrap(~CityName) +
  labs(x = "Zip Code")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 5651 rows containing non-finite values (`stat_bin()`).
```



Where are different types of fans

```
population <- data.frame (city  = c("Virginia Beach", "Lousiville", "Kansas City"),
                  population = c(1800000, 1400000, 2200000))

fan_types <-
  continuous %>%
  select(contains("propn_score_minor_4"), CityName) %>%
  pivot_longer(values_to = "score", cols = c(contains("propn")), names_to = "Prop") %>%
  left_join(population, by = c("CityName" = "city"))%>%
  left_join(read_csv("prop_join.csv")) %>%
  mutate(prop_number = as.numeric(substring(Prop,nchar(Prop)-3+1), nchar(Prop))) %>%
  filter(between(prop_number, 410, 450)) %>%
  mutate(super_fan = ifelse(score >= 650, 1, 0),
         regular_fan= ifelse(score >= mean(score, na.rm=T), 1, 0),
         Sport = sub(".*:", "", Sport))
```

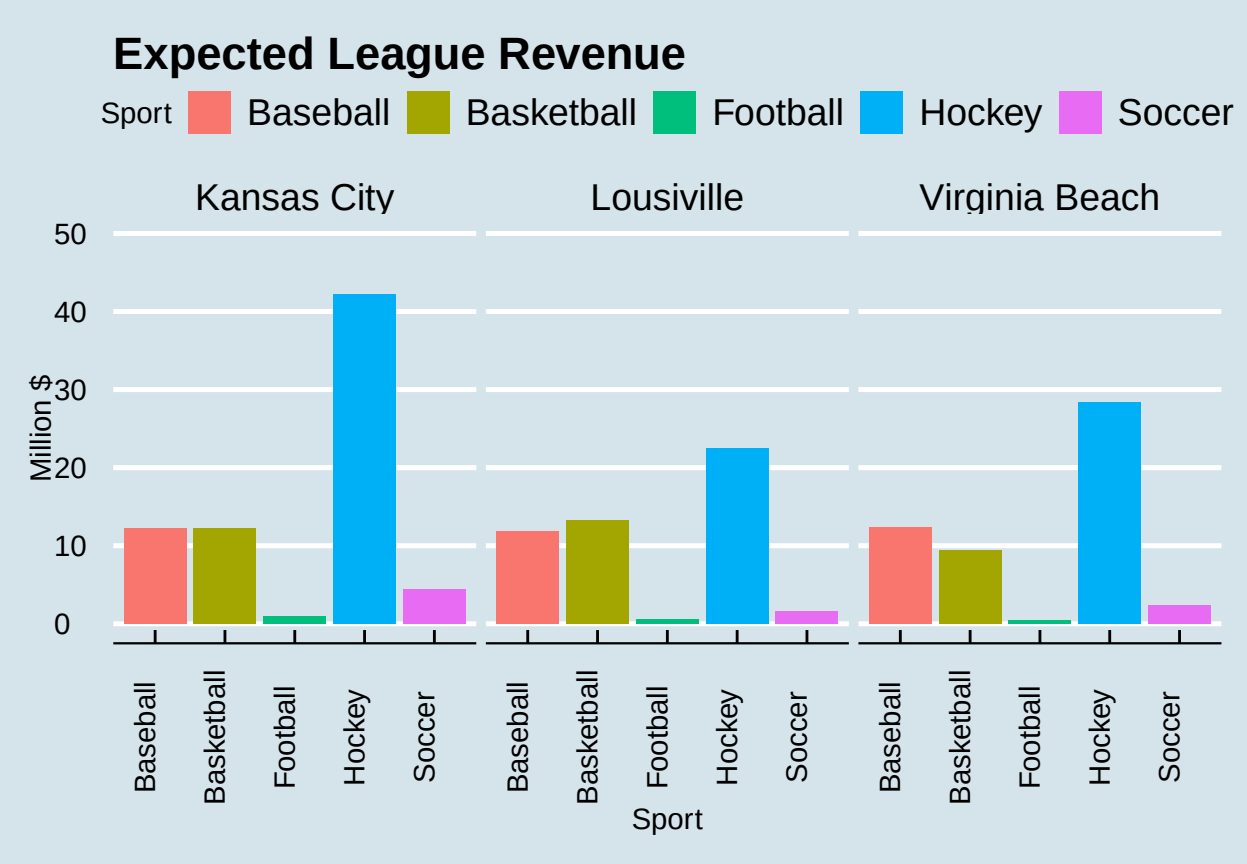
```
## Rows: 29 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Prop, Sport
##
## i Use `spec()` to retrieve the full column specification for this data.
```

```
## i Specify the column types or set ‘show_col_types = FALSE‘ to quiet this message.
## Joining, by = "Prop"
```

```
fan_totals <-
  fan_types %>%
  filter(!str_detect(Sport, "College"))%>%
  group_by(CityName) %>%
  summarise(city_total_regular_fans = sum(regular_fan, na.rm= T))

fan_types_by_city <-
  fan_types %>%
  filter(!str_detect(Sport, "College")) %>%
  group_by(CityName, Sport) %>%
  summarise(population = mean(population),
            avg_score = mean(score, na.rm=T),
            regular_fans =  sum(regular_fan, na.rm=T),
            super_fans = sum(super_fan, na.rm=T),
            ct = n(),
            expected_super_fans = super_fans*population/ct,
            expected_regular_fans = regular_fans*population/ct,
            ) %>%
  left_join(fan_totals) %>%
  rowwise() %>%
  mutate(perc_by_sport = regular_fans/city_total_regular_fans) %>%
  left_join(spend_by_city) %>%
  mutate(FINAL_SPEND = total_spend * expected_regular_fans/sample_size * perc_by_sport)%>%
  left_join(spend_by_city_concert) %>%
   mutate(FINAL_SPEND_concert = total_spend_concert * expected_regular_fans/sample_size)
```

```
## ‘summarise()‘ has grouped output by ’CityName’. You can override using the
## ‘.groups‘ argument.
## Joining, by = "CityName"
## Joining, by = "CityName"
## Joining, by = "CityName"
```

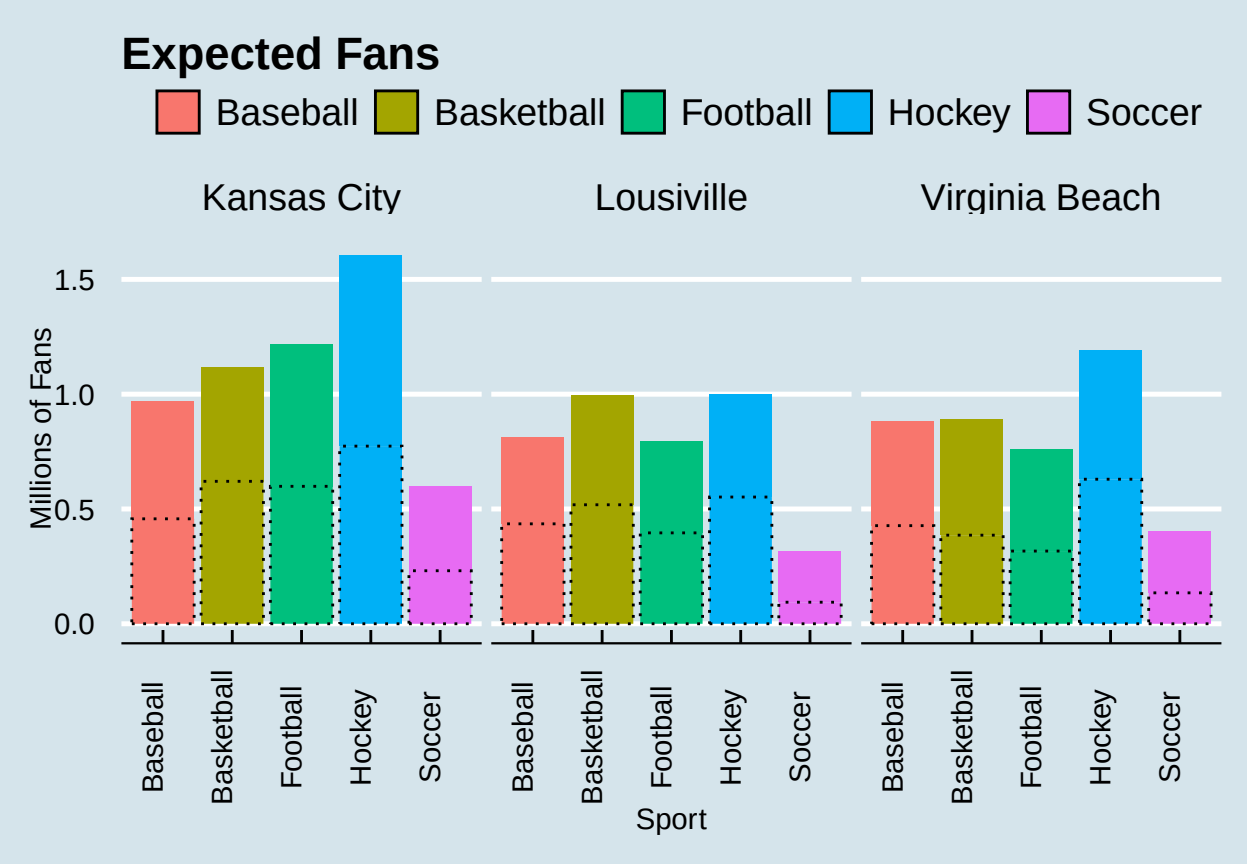
```
league_percent <- c("Football" = 0.015, "Hockey" = 0.35, "Basketball" = 0.2238, "Baseball" = .3, 'Socce

fan_types_by_city = fan_types_by_city %>%
    mutate(league_revenue = case_when(Sport == "Baseball" ~ FINAL_SPEND * 0.3,
                                Sport == "Football" ~ FINAL_SPEND *0.015,
                                Sport == "Soccer" ~ FINAL_SPEND *0.28,
                                Sport == "Hockey" ~ FINAL_SPEND * 0.377,
                                Sport == "Basketball" ~ FINAL_SPEND * 0.2238))

ggplot(fan_types_by_city, aes(x= Sport, y = league_revenue/1000000, fill = Sport)) +
  geom_col() +
  facet_wrap(~CityName) +
  labs(x = "Sport", y = "Million $", title = "Expected League Revenue") +
  ggthemes::theme_economist()+
  theme(axis.text.x = element_text(angle = 90)) + ylim(0,50)
```



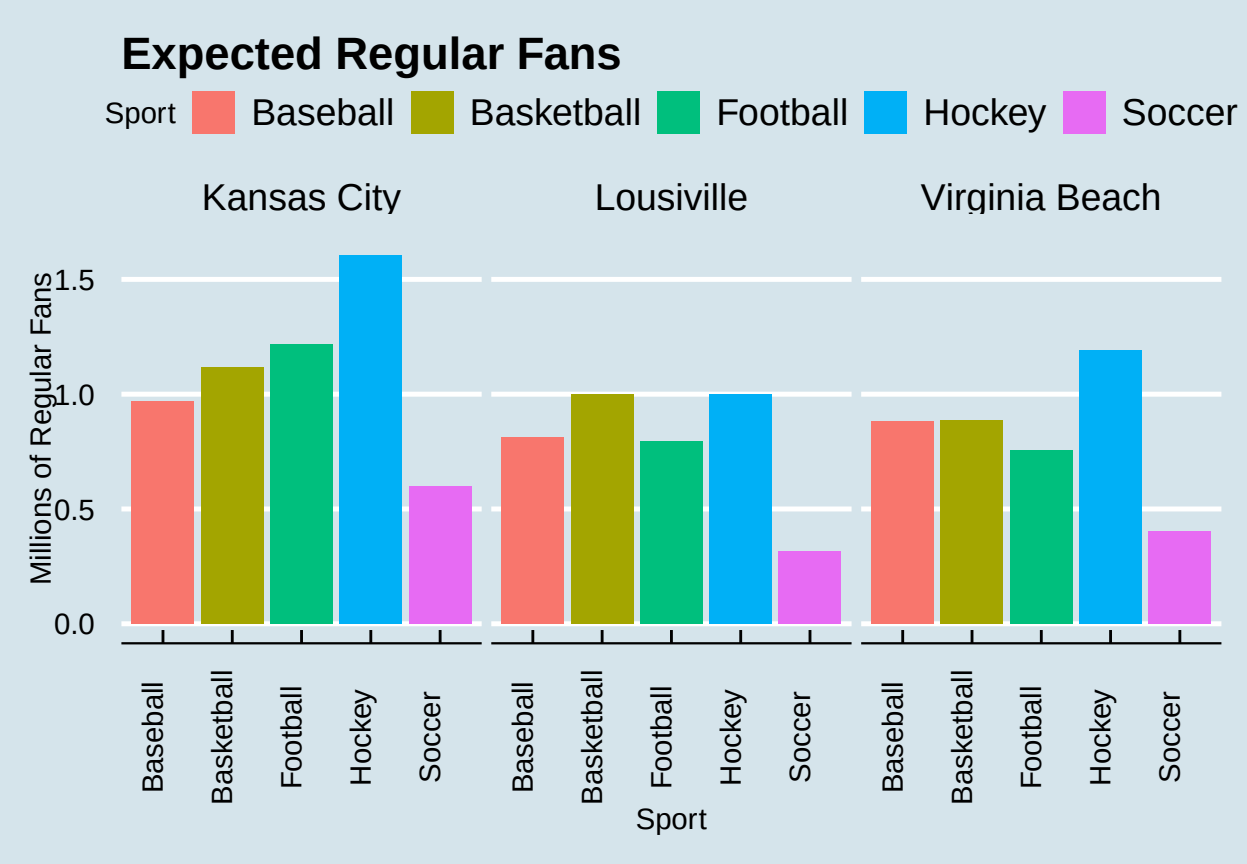

```
ggData <- fan_types_by_city %>%
  mutate(reg2 = expected_regular_fans-expected_super_fans) %>%select(CityName, Sport, reg2, expected_su
  rename(type = name) %>% mutate(type2 = factor(type, ordered =T,levels = c("reg2","expected_super_fans
```

```
ggplot(ggData, aes(x= Sport, y = value/1000000, fill = Sport,linetype=  type2)) +
  geom_col(color ="black") +
  facet_wrap(~CityName)+
  labs(x = "Sport", y = "Millions of Fans", title = "Expected Fans", fill = "")+
  ggthemes::theme_economist()+
  theme(axis.text.x = element_text(angle = 90))+ylim(0,1.7) +
  scale_linetype_manual(values=c("blank", "dotted"))+
  guides(linetype =F)
```

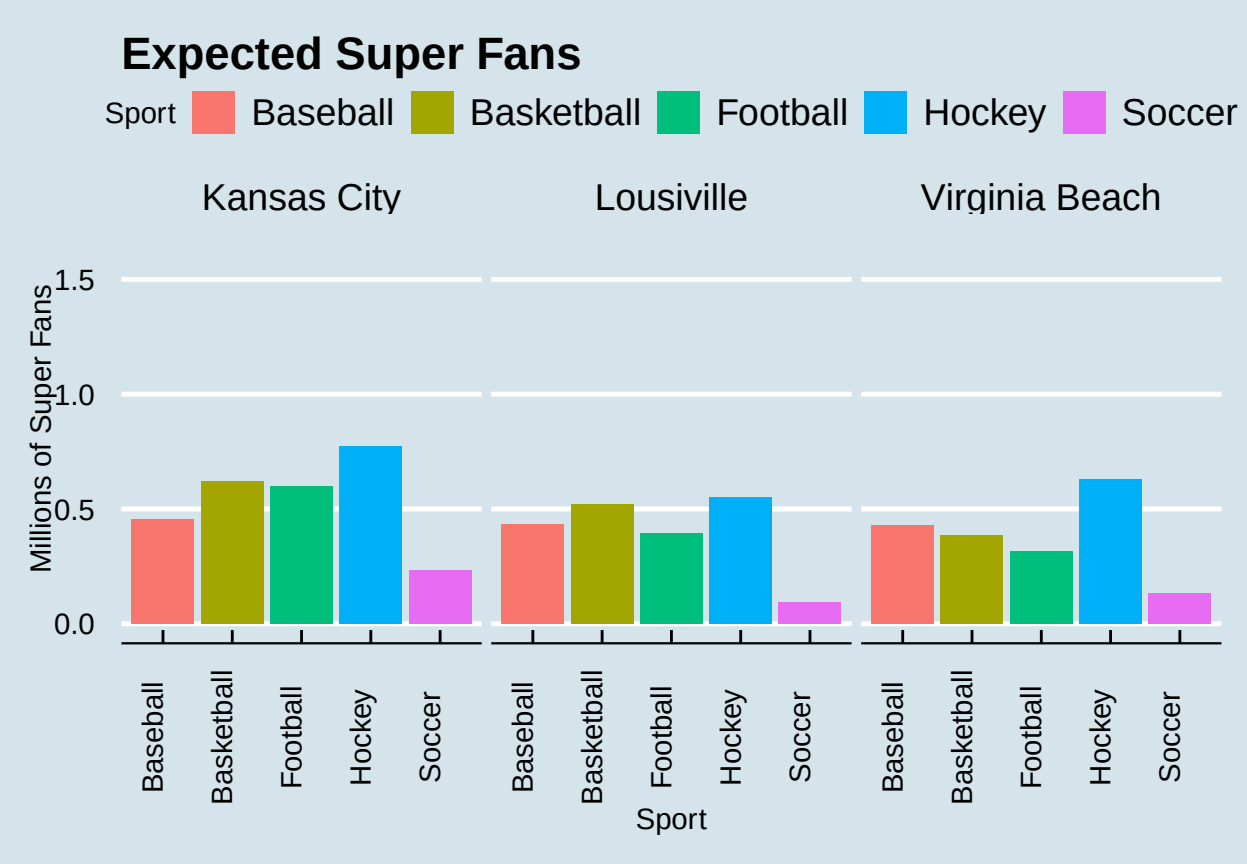
```
## Warning: The ‘<scale>‘ argument of ‘guides()‘ cannot be ‘FALSE‘. Use "none" instead as
## of ggplot2 3.3.4.
```



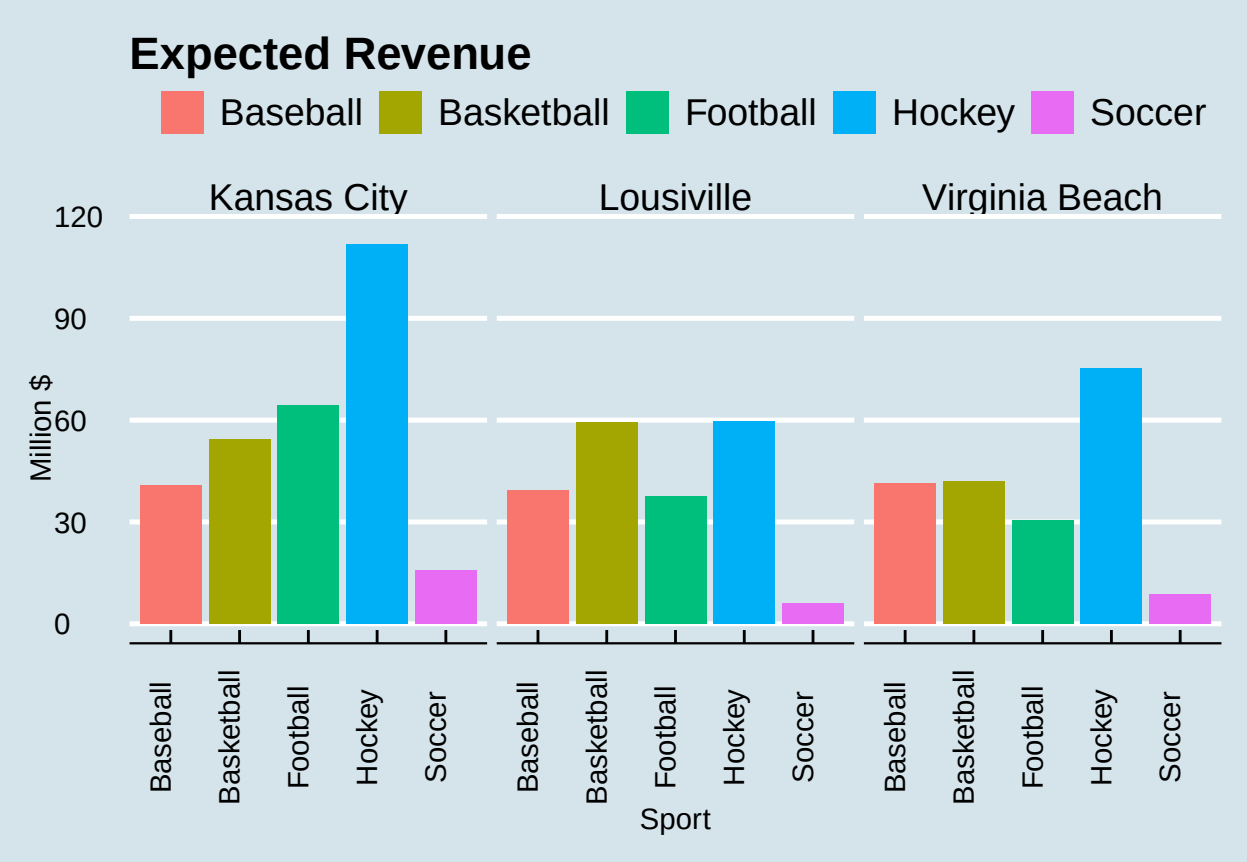

```
ggplot(
  fan_types_by_city, aes(x= Sport, y = expected_regular_fans/1000000, fill = Sport)) +
  geom_col() +
  facet_wrap(~CityName)+
  labs(x = "Sport", y = "Millions of Regular Fans", title = "Expected Regular Fans")+
  ggthemes::theme_economist()+
  theme(axis.text.x = element_text(angle = 90))+ylim(0,1.7)
```




```
ggplot(fan_types_by_city, aes(x= Sport, y = expected_super_fans/1000000, fill = Sport)) +
  geom_col() +
  facet_wrap(~CityName)+
  labs(x = "Sport", y = "Millions of Super Fans", title = "Expected Super Fans")+
  ggthemes::theme_economist()+
  theme(axis.text.x = element_text(angle = 90)) +ylim(0,1.7)
```




```
ggplot(fan_types_by_city, aes(x= Sport, y = FINAL_SPEND/1000000, fill = Sport)) +
  geom_col() +
  facet_wrap(~CityName) +
  labs(x = "Sport", y = "Million $", title = "Expected Revenue", fill = "") +
  ggthemes::theme_economist()+
  theme(axis.text.x = element_text(angle = 90)) + ylim(0,115)
```



$E[\text{Event\$}] \sim \text{FinancialStatus} + \text{LifeStage} + \text{PrevTixHistory})$

Concert more

```
fan_types_concerts <-
  continuous %>%
  select(contains("propn_score_minor_1"), CityName) %>%
  pivot_longer(values_to = "score", cols = c(contains("propn")), names_to = "Prop") %>%
  left_join(population, by = c("CityName" = "city"))%>%
  left_join(read_csv("prop_join.csv")) %>%
  rename(Genre = Sport) %>%
  mutate(prop_number = as.numeric(substring(Prop,nchar(Prop)-3+1), nchar(Prop))) %>%
  filter(between(prop_number, 100, 200)) %>%
  mutate(super_fan = ifelse(score >= 650, 1, 0),
         regular_fan= ifelse(score >= mean(score, na.rm=T), 1, 0),
         Genre = sub(".*:", "", Genre))
```

```
## Rows: 29 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (2): Prop, Sport
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## Joining, by = "Prop"
```

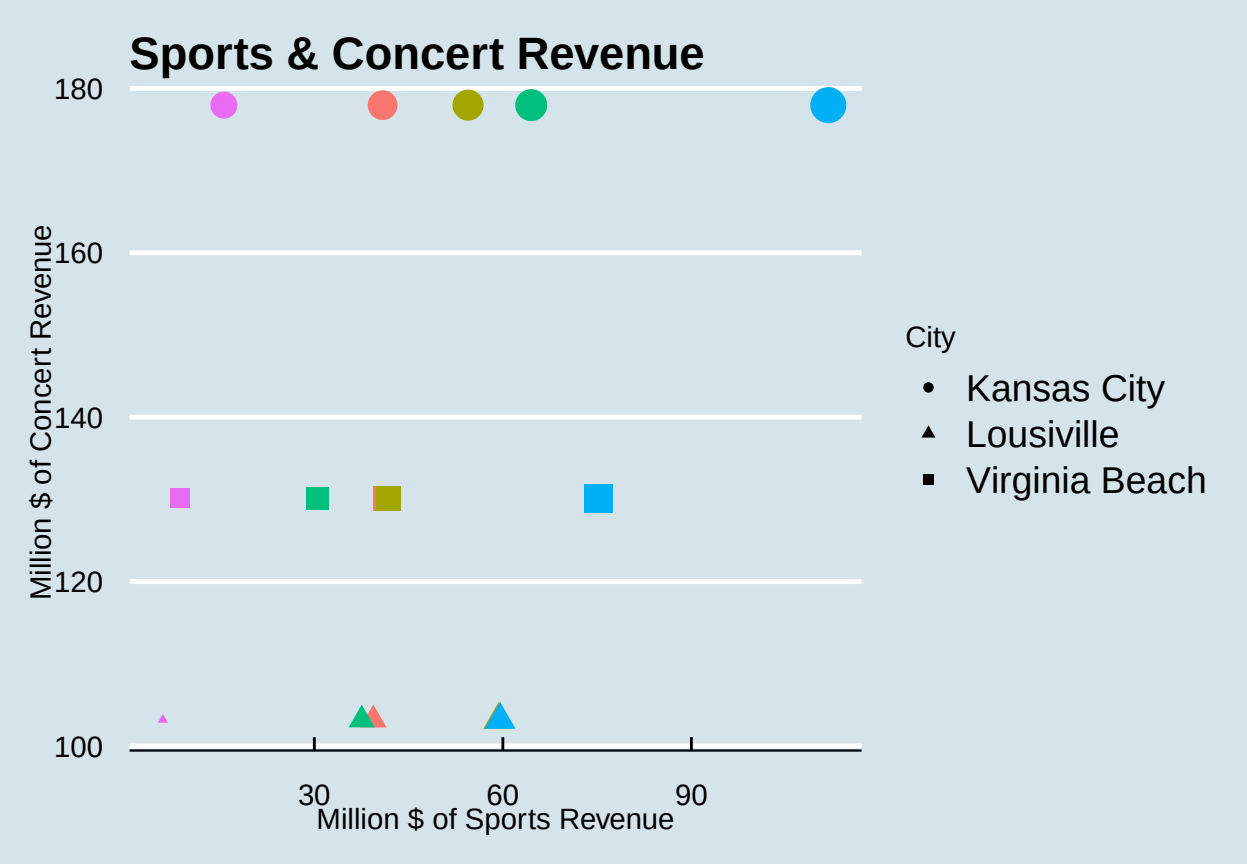
```
fan_totals_concerts <-
  fan_types_concerts %>%
  group_by(CityName) %>%
  summarise(city_total_regular_fans = sum(regular_fan, na.rm= T))

fan_types_by_city_concert <-
  fan_types_concerts %>%
  group_by(CityName) %>%
  summarise(population = mean(population),
            avg_score = mean(score, na.rm=T),
            regular_fans =  sum(regular_fan, na.rm=T),
            super_fans = sum(super_fan, na.rm=T),
            ct = n(),
            expected_super_fans = super_fans*population/ct,
            expected_regular_fans = regular_fans*population/ct,
            ) %>%
  left_join(fan_totals_concerts) %>%
  rowwise() %>%
  mutate(perc_by_sport = regular_fans/city_total_regular_fans) %>%
  left_join(spend_by_city_concert) %>%
  mutate(FINAL_SPEND_concert = total_spend_concert * expected_regular_fans/sample_size_concert)
```

```
## Joining, by = "CityName"
## Joining, by = "CityName"
```

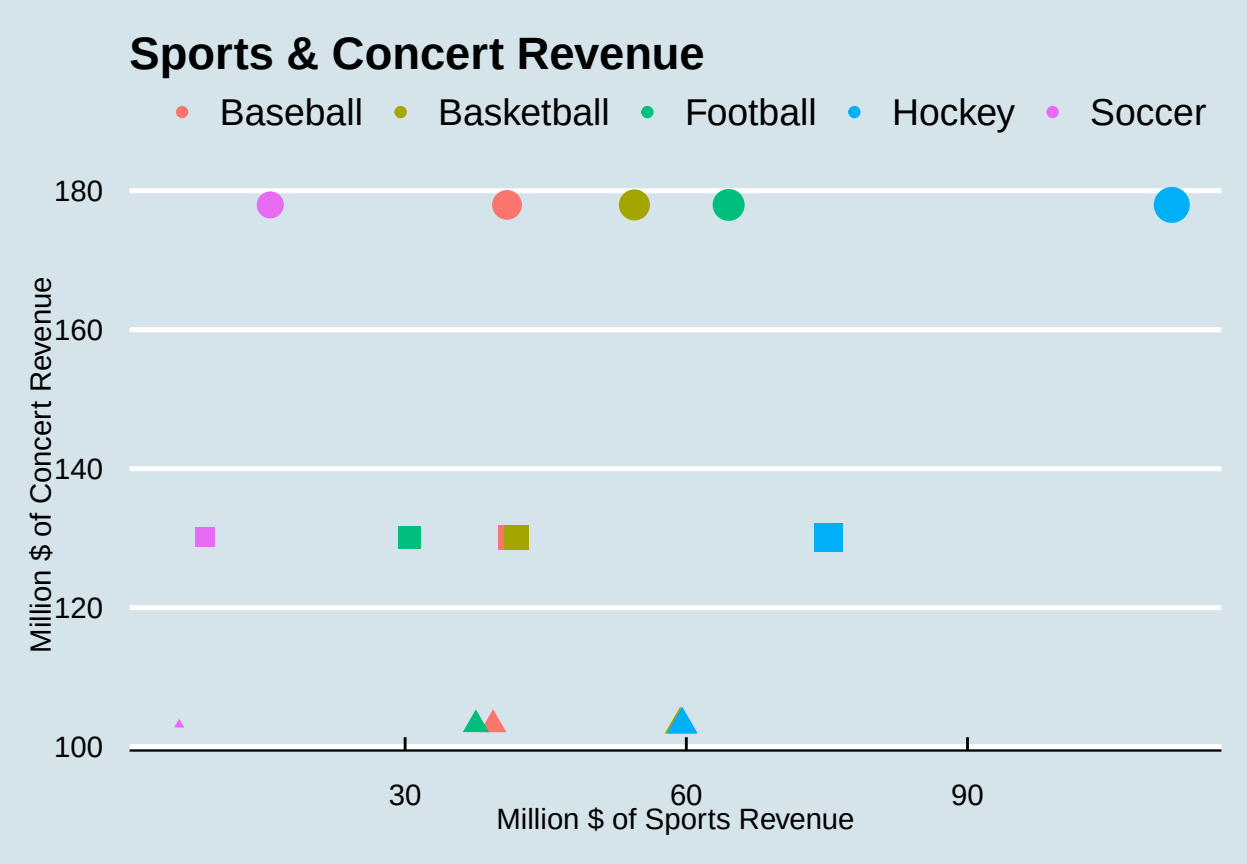
```
concert_matches <- left_join(fan_types_by_city %>% select(CityName, Sport, FINAL_SPEND),fan_types_by_ci
```

```
## Joining, by = "CityName"
```

```
ggplot(concert_matches, aes(y = FINAL_SPEND_concert/1000000, x= FINAL_SPEND/1000000, shape = CityName,
```




```
ggplot(concert_matches, aes(y = FINAL_SPEND_concert/1000000, x= FINAL_SPEND/1000000, shape = CityName,
```


Sports & Concert Revenue
Baseball
Basketball
Football
Hockey
Soccer
180
160
140
120
100
Million $ of Concert Revenue
30
60
90
Million $ of Sports Revenue