stats

```
require(dplyr)
```

```
## Loading required package: dplyr
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
require(tidyr)
```

```
## Loading required package: tidyr
```

```
require(knitr)
```

```
## Loading required package: knitr
```

```
require(ggplot2)
```

```
## Loading required package: ggplot2
```

```
require(maps)
```

```
## Loading required package: maps
```

```
require(RColorBrewer)
```

```
## Loading required package: RColorBrewer
```

```
require(summarytools)
```

```
## Loading required package: summarytools
```

```
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 16299)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
```

```
## other attached packages:
##  [1] summarytools_0.8.0   RColorBrewer_1.1-2   maps_3.2.0
##  [4] ggplot2_2.2.1        knitr_1.18           tidyr_0.7.2
##  [7] dplyr_0.7.4          RevoUtilsMath_10.0.1 RevoUtils_10.0.7
## [10] RevoMods_11.0.0      MicrosoftML_9.3.0    mrsdeploy_1.1.3
## [13] RevoScaleR_9.3.0     lattice_0.20-35      rpart_4.1-11
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.14           pryr_0.1.3             plyr_1.8.4
##  [4] compiler_3.4.3         pillar_1.0.1           bindr_0.1
##  [7] bitops_1.0-6           iterators_1.0.9        tools_3.4.3
## [10] digest_0.6.13          jsonlite_1.5           evaluate_0.10.1
## [13] tibble_1.4.1           gtable_0.2.0           pkgconfig_2.0.1
## [16] rlang_0.1.6            foreach_1.4.5          CompatibilityAPI_1.1.0
## [19] curl_3.1               yaml_2.1.16            bindrcpp_0.2
## [22] stringr_1.2.0          rprojroot_1.3-1        grid_3.4.3
## [25] glue_1.2.0             R6_2.2.2               rmarkdown_1.8
## [28] pander_0.6.1           purrr_0.2.4            magrittr_1.5
## [31] rapportools_1.0        matrixStats_0.52.2     backports_1.1.2
## [34] scales_0.5.0           codetools_0.2-15       htmltools_0.3.6
## [37] assertthat_0.2.0       colorspace_1.3-2       stringi_1.1.6
## [40] RCurl_1.95-4.9         lazyeval_0.2.1         munsell_0.4.3
```

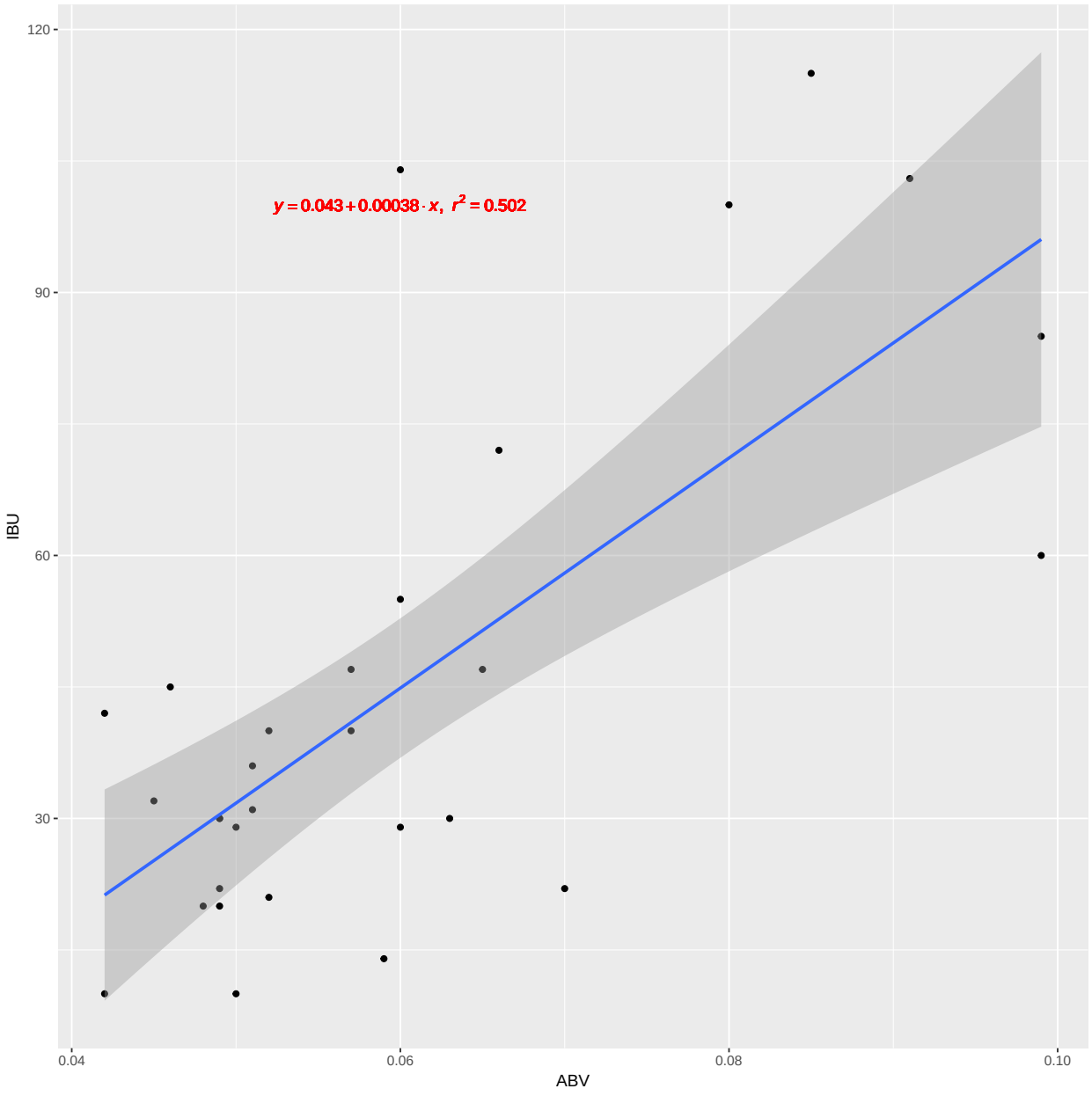
```
beer_data <- read.csv("../data/Beers.csv", header=TRUE)
head(beer_data)
```

```
##                  Name Beer_ID   ABV IBU Brewery_id
## 1            Pub Beer    1436 0.050  NA        409
## 2         Devil's Cup    2265 0.066  NA        178
## 3 Rise of the Phoenix    2264 0.071  NA        178
## 4            Sinister    2263 0.090  NA        178
## 5       Sex and Candy    2262 0.075  NA        178
## 6        Black Exodus    2261 0.077  NA        178
##                            Style Ounces
## 1            American Pale Lager     12
## 2        American Pale Ale (APA)     12
## 3                  American IPA     12
## 4 American Double / Imperial IPA     12
## 5                  American IPA     12
## 6                 Oatmeal Stout     12
```

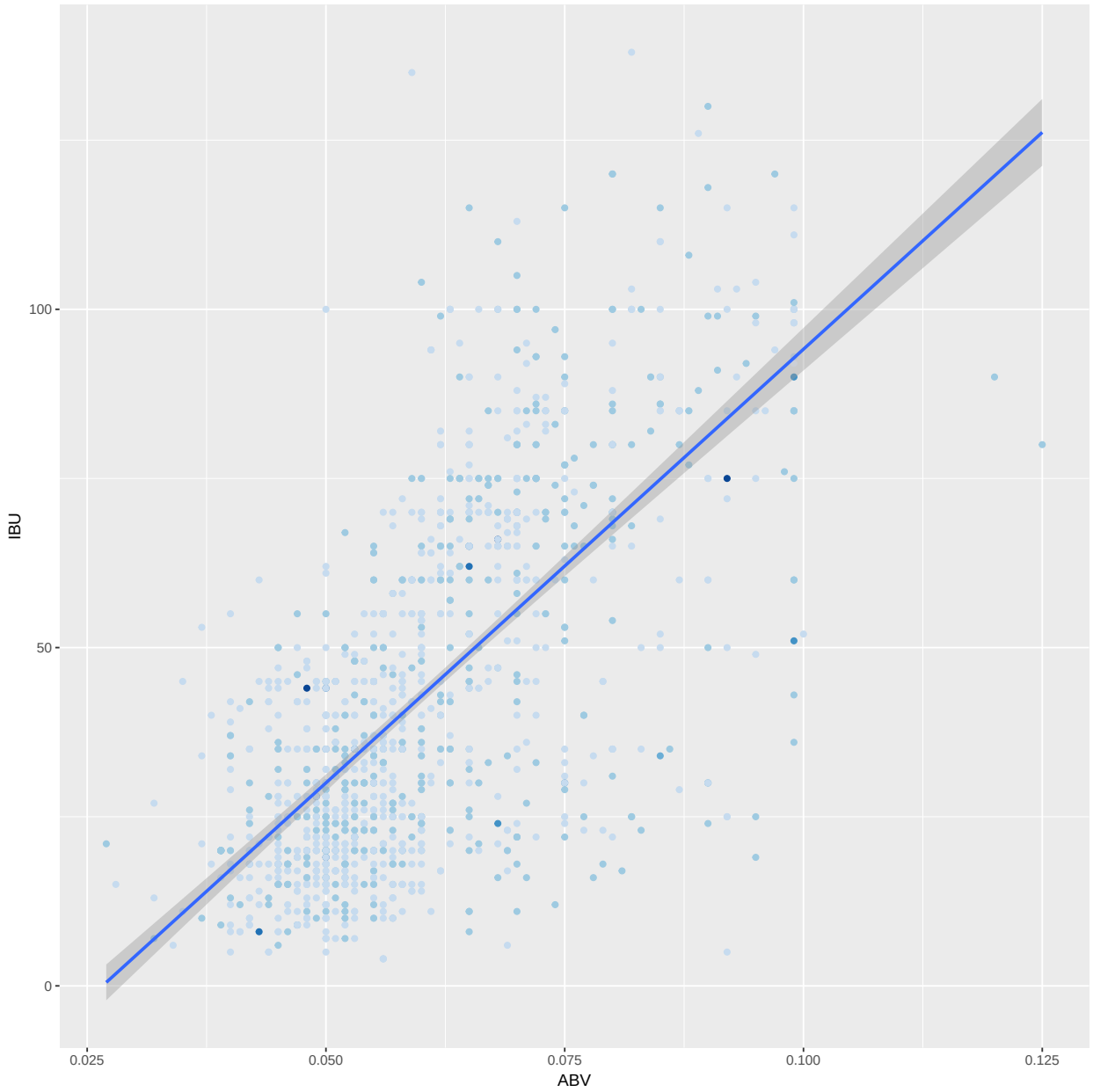
```
#beer_data$Brewery_id[(beer_data$Brewery_id %in% breweries_sk$Brew_ID)]  <- breweries_sk$Brew_SK # upda


beer_clean <- distinct(beer_data) %>% rename(Brew_ID = Brewery_id, Beer_Name = Name)

notnull <- beer_data %>% filter(is.na(IBU)) %>% distinct(Style)


isnull <- beer_data %>% filter(!is.na(IBU)) %>% distinct(Beer_ID, Style, IBU, ABV, Ounces) %>% arrange(S


#A distinct list of beer styles. A distinct beer style is noted as having a unique style name, IBU, ABV
styles <- notnull %>%
          full_join(isnull, by = "Style") %>% arrange(Style) #%>%
```

```
          #mutate_each(funs(as.character), Style)

styles <- styles %>% na.omit(IBU, ABV) #omit any record with missing ABV or IBU

ss <- sample_n(styles, size = 30) #sample data frame rows so IBU and ABV remain paired
```

```
lm_eqn = function(m) {

  l <- list(a = format(coef(m)[1], digits = 2),
      b = format(abs(coef(m)[2]), digits = 2),
      r2 = format(summary(m)$r.squared, digits = 3));

  if (coef(m)[2] >= 0)  {
    eq <- substitute(italic(y) == a + b %.% italic(x)*","~~italic(r)^2~"="~r2,l)
  } else {
    eq <- substitute(italic(y) == a - b %.% italic(x)*","~~italic(r)^2~"="~r2,l)
  }

  as.character(as.expression(eq));
}

ggplot(ss, aes(x=ABV, y=IBU)) +
  geom_point() +
  geom_smooth(method = "lm") +
  geom_text(aes(x = .06, y = 100, label = lm_eqn(lm(ABV ~ IBU ,ss))), parse = TRUE, color = "red")
```



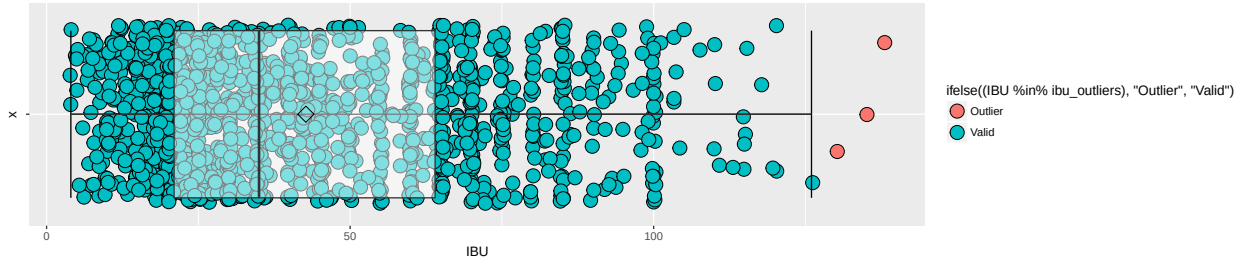

```
ggplot(styles, aes(x=ABV, y=IBU)) +
  geom_point(aes(colour=as.factor(Ounces))) +
  scale_colour_brewer() +
  geom_smooth(method = "lm") +
  theme(legend.position="none")
```




```
  #geom_text(aes(x = .06, y = 100, label = lm_eqn(lm(ABV ~ IBU ,styles))), parse = TRUE, color = "red")
```

Check for Outliers in IBU

```
ibu_outliers <- boxplot(styles$IBU, plot = FALSE)[["out"]]

ggplot((styles %>% drop_na(IBU)), aes(x="", y=IBU)) +
      geom_point(aes(fill = ifelse((IBU %in% ibu_outliers),"Outlier","Valid")), size = 5, shape = 21, p
      stat_boxplot(geom ='errorbar') +
      geom_boxplot(alpha=.5, outlier.shape = NA) +#, outlier.colour = "red") +
      stat_summary(fun.y=mean, geom="point", shape=5, size=4) +
      coord_flip()
```




```
      #geom_text(aes(label=ifelse((x>4*IQR(x)|y>4*IQR(y)),label,"")), hjust=1.1)
```

```
ibu_outliers
```

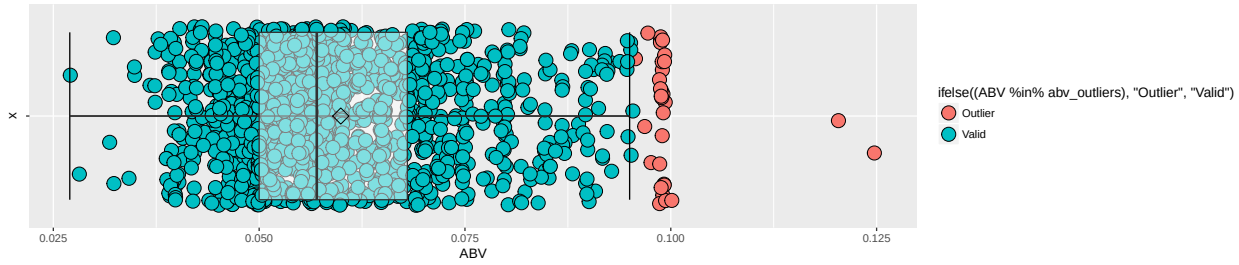
```
## [1] 138 130 135
```

Check for Outliers in ABV

```
abv_outliers <- boxplot(styles$ABV, plot = FALSE)[["out"]]

abv_outliers
```

```
##  [1] 0.099 0.099 0.099 0.099 0.099 0.097 0.098 0.099 0.096 0.099 0.099
## [12] 0.099 0.099 0.099 0.097 0.099 0.099 0.099 0.099 0.100 0.099 0.099
## [23] 0.125 0.099 0.099 0.099 0.099 0.099 0.099 0.099 0.120 0.099
```

```
ggplot((styles %>% drop_na(ABV)), aes(x="", y=ABV)) +
      #geom_point(aes(fill = ifelse((ABV>3.29*IQR(ABV)),"Outlier","Valid")), size = 5, shape = 21, posi
      geom_point(aes(fill = ifelse((ABV %in% abv_outliers),"Outlier","Valid")), size = 5, shape = 21, po
      stat_boxplot(geom ='errorbar') +
      geom_boxplot(alpha=.5, outlier.shape = NA) +#, outlier.colour = "red") +
      stat_summary(fun.y=mean, geom="point", shape=5, size=4) +
      coord_flip()
```

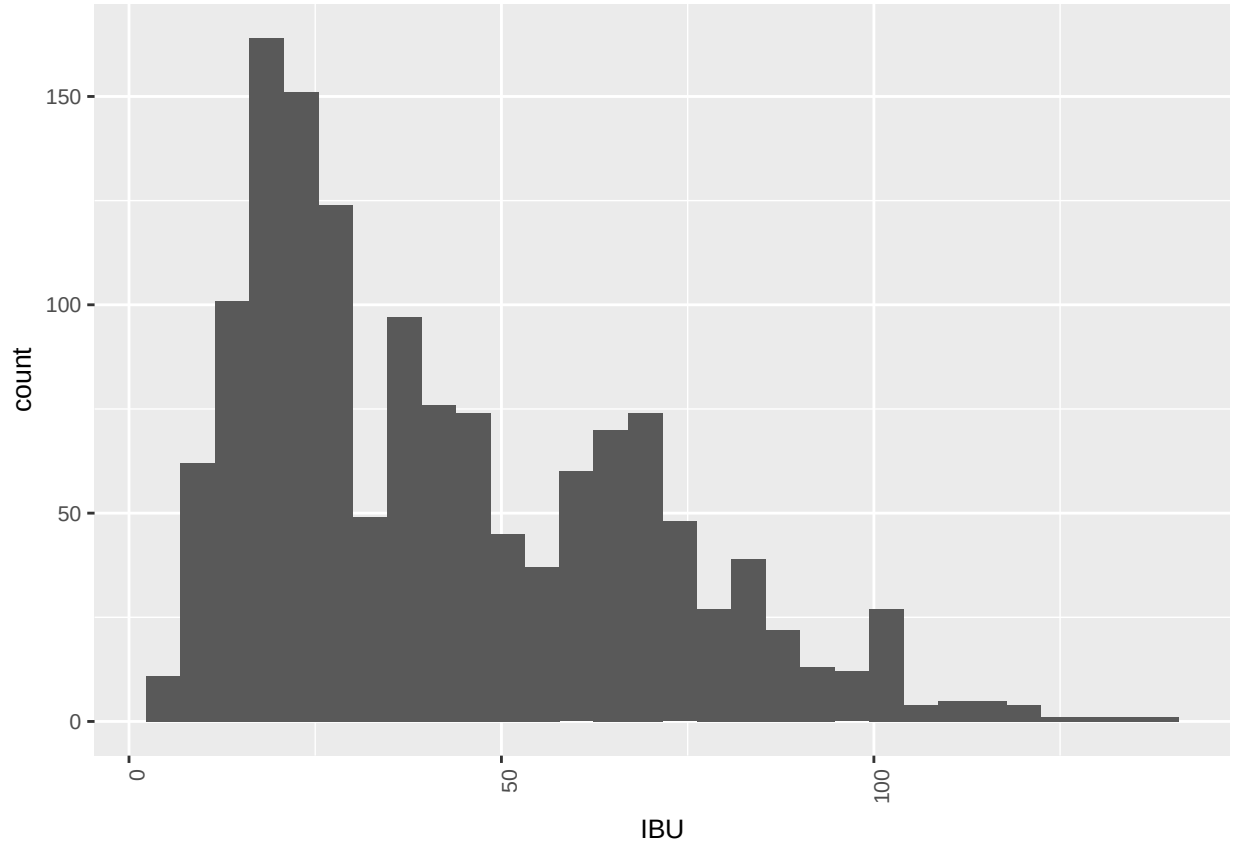



```
      #geom_text(aes(label=ifelse((x>4*IQR(x)|y>4*IQR(y)),label,"")), hjust=1.1)

#boxplot(styles$ABV, plot = TRUE)
```

check normality of full dataset and sample

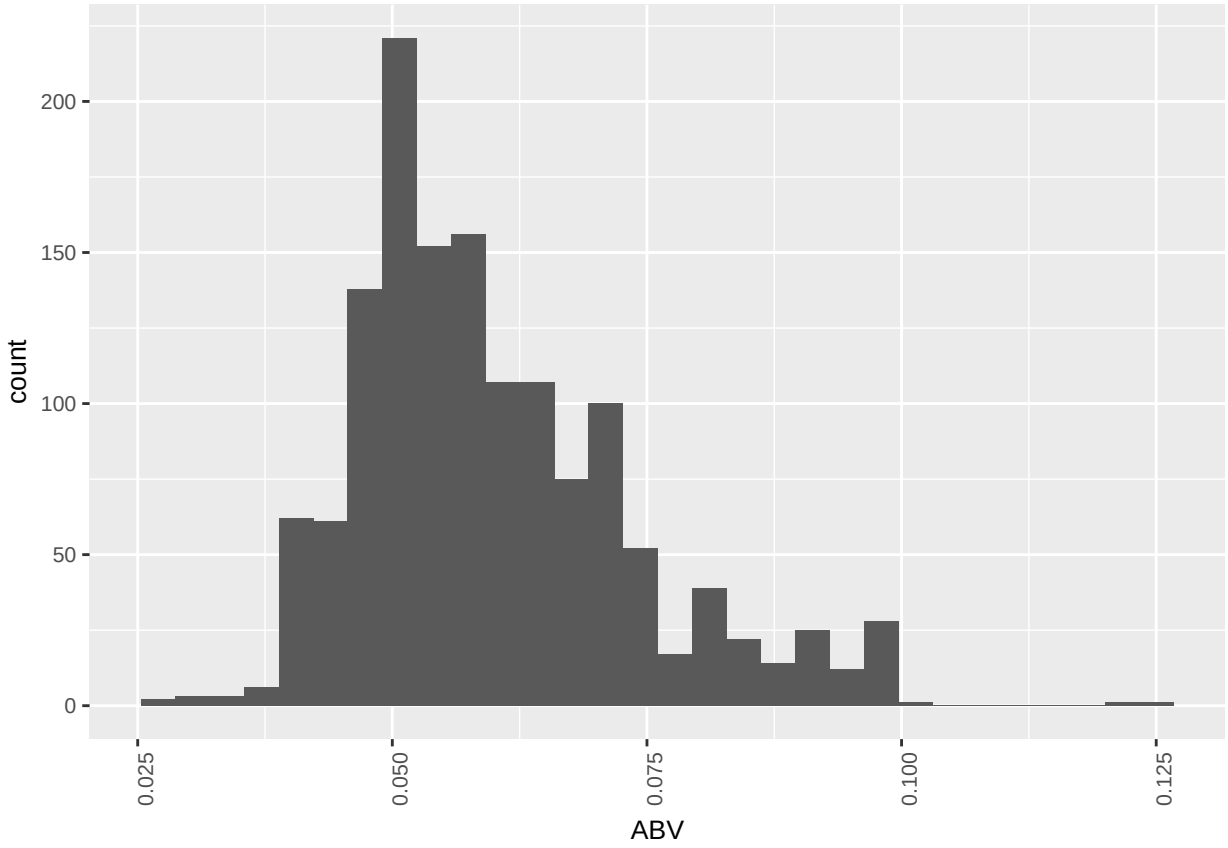
```
#full dataaset
ggplot(styles) +
    geom_histogram(aes(x=IBU)) +
    theme(text = element_text(size=10),
        axis.text.x = element_text(angle=90, hjust=1))
```

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




```
ggplot(styles) +
    geom_histogram(aes(x=ABV)) +
    theme(text = element_text(size=10),
        axis.text.x = element_text(angle=90, hjust=1))
```

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

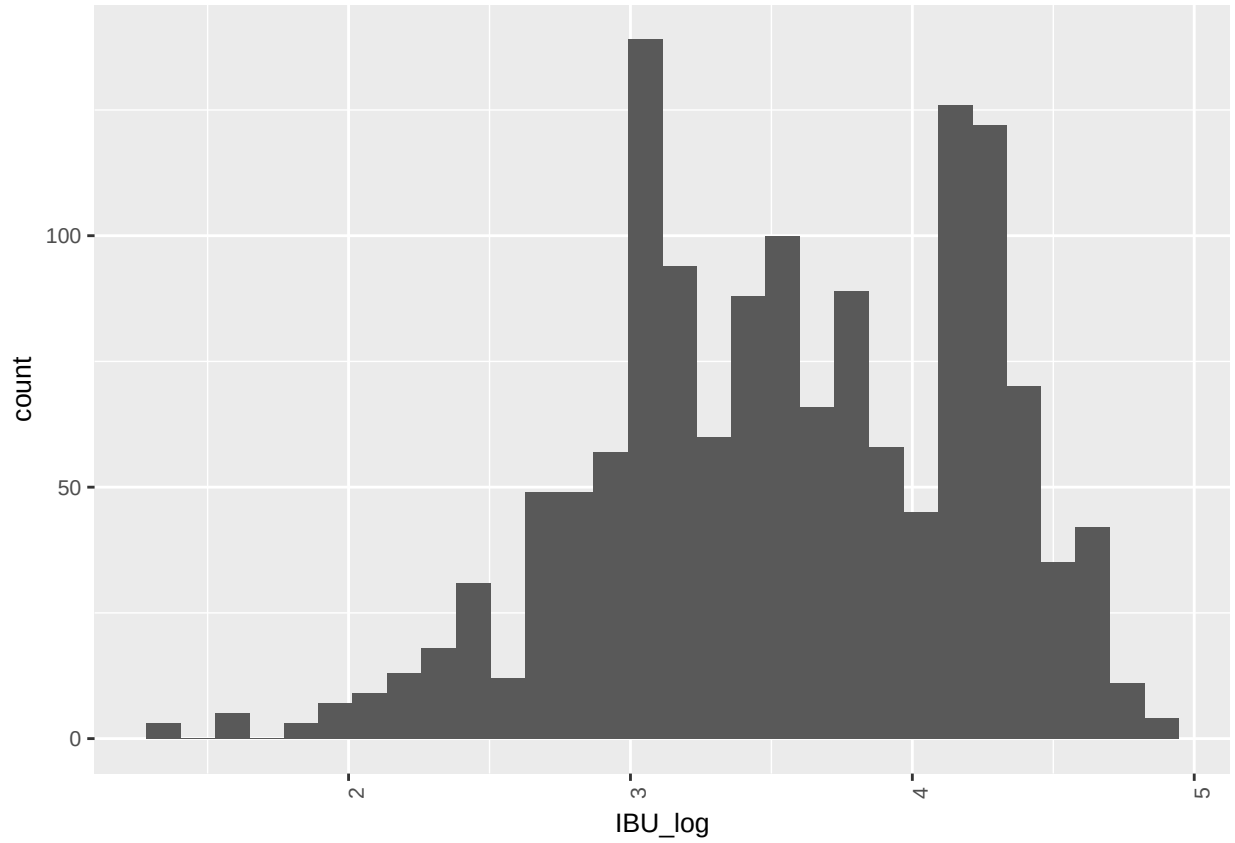



```
# #sample
# ggplot(ss) +
#     geom_histogram(aes(x=IBU)) +
#     theme(text = element_text(size=10),
#         axis.text.x = element_text(angle=90, hjust=1))
#
# ggplot(ss) +
#     geom_histogram(aes(x=ABV)) +
#     theme(text = element_text(size=10),
#         axis.text.x = element_text(angle=90, hjust=1))
#
#
```

Check normality of log sample

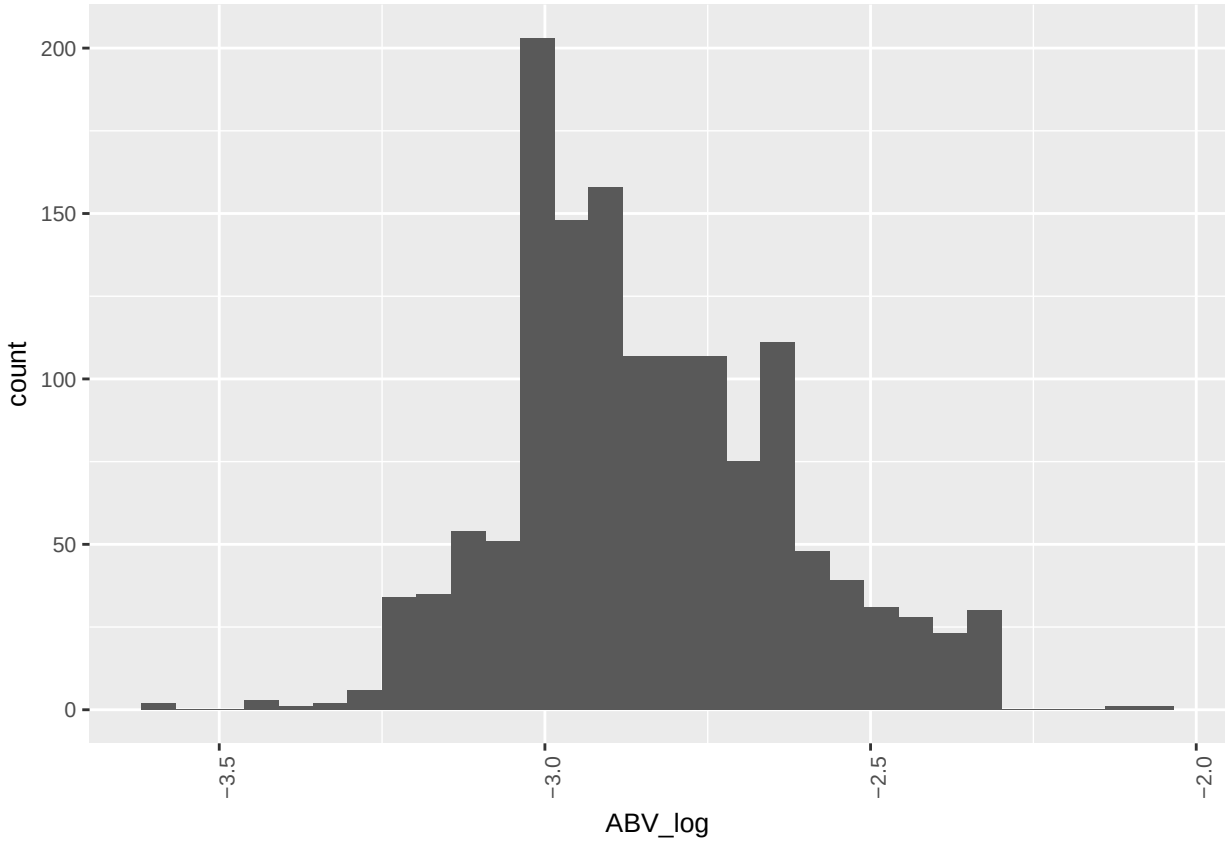
```
styles$IBU_log <- log(styles$IBU)
styles$ABV_log <- log(styles$ABV)

ggplot(styles) +
    geom_histogram(aes(x=IBU_log)) +
    theme(text = element_text(size=10),
        axis.text.x = element_text(angle=90, hjust=1))
```

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




```
ggplot(styles) +
    geom_histogram(aes(x=ABV_log)) +
    #geom_density() +
    theme(text = element_text(size=10),
        axis.text.x = element_text(angle=90, hjust=1))
```

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

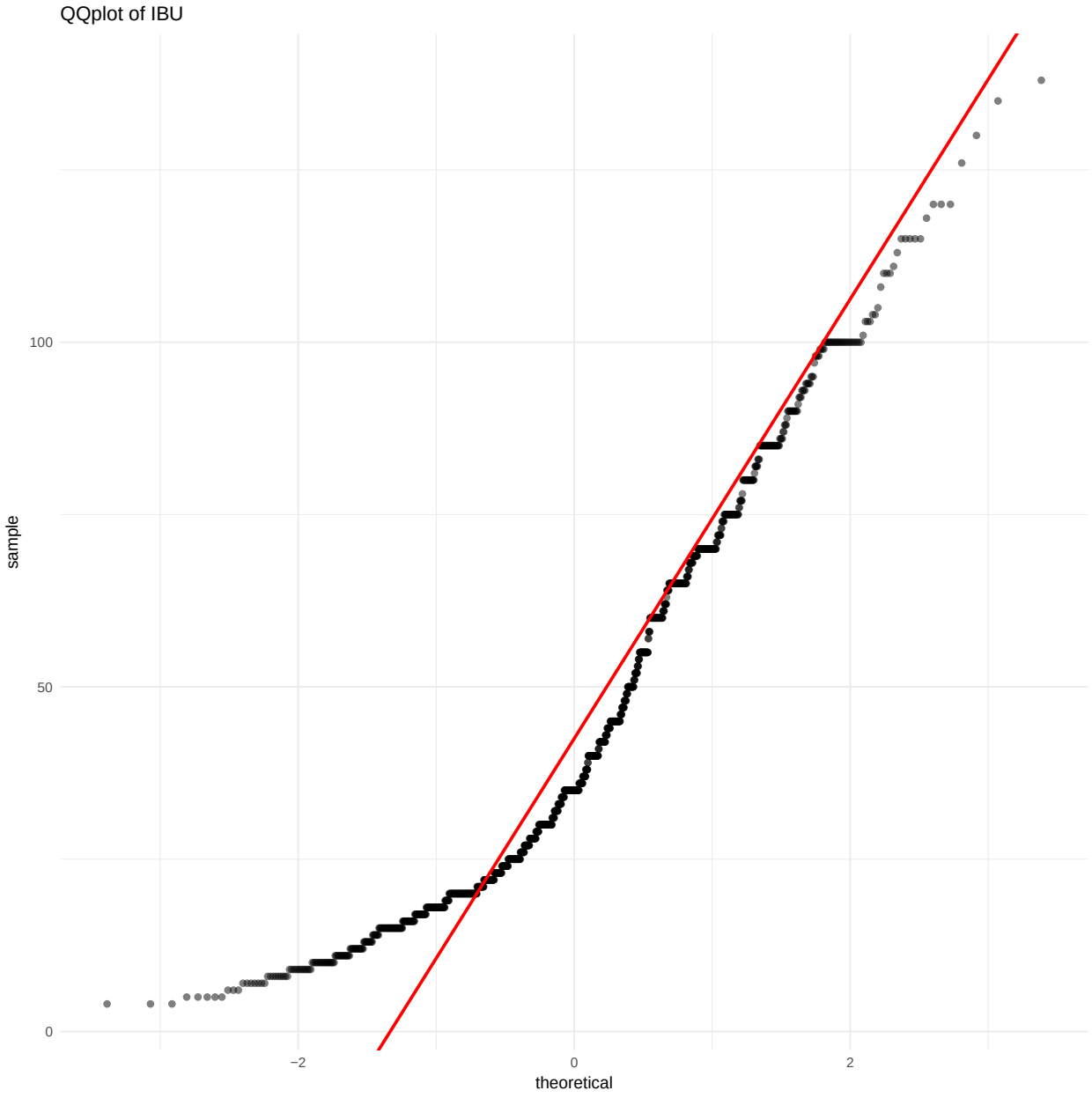



```
# ss$IBU_log <- log(ss$IBU)
# ss$ABV_log <- log(ss$ABV)
#
# ggplot(ss) +
#     geom_histogram(aes(x=IBU_log)) +
#     theme(text = element_text(size=10),
#         axis.text.x = element_text(angle=90, hjust=1))
#
# ggplot(ss) +
#     geom_histogram(aes(x=ABV_log)) +
#     theme(text = element_text(size=10),
#         axis.text.x = element_text(angle=90, hjust=1))


# log plots seem to also suggest spearman over pearson
```

QQ Plots of full and sample datasets

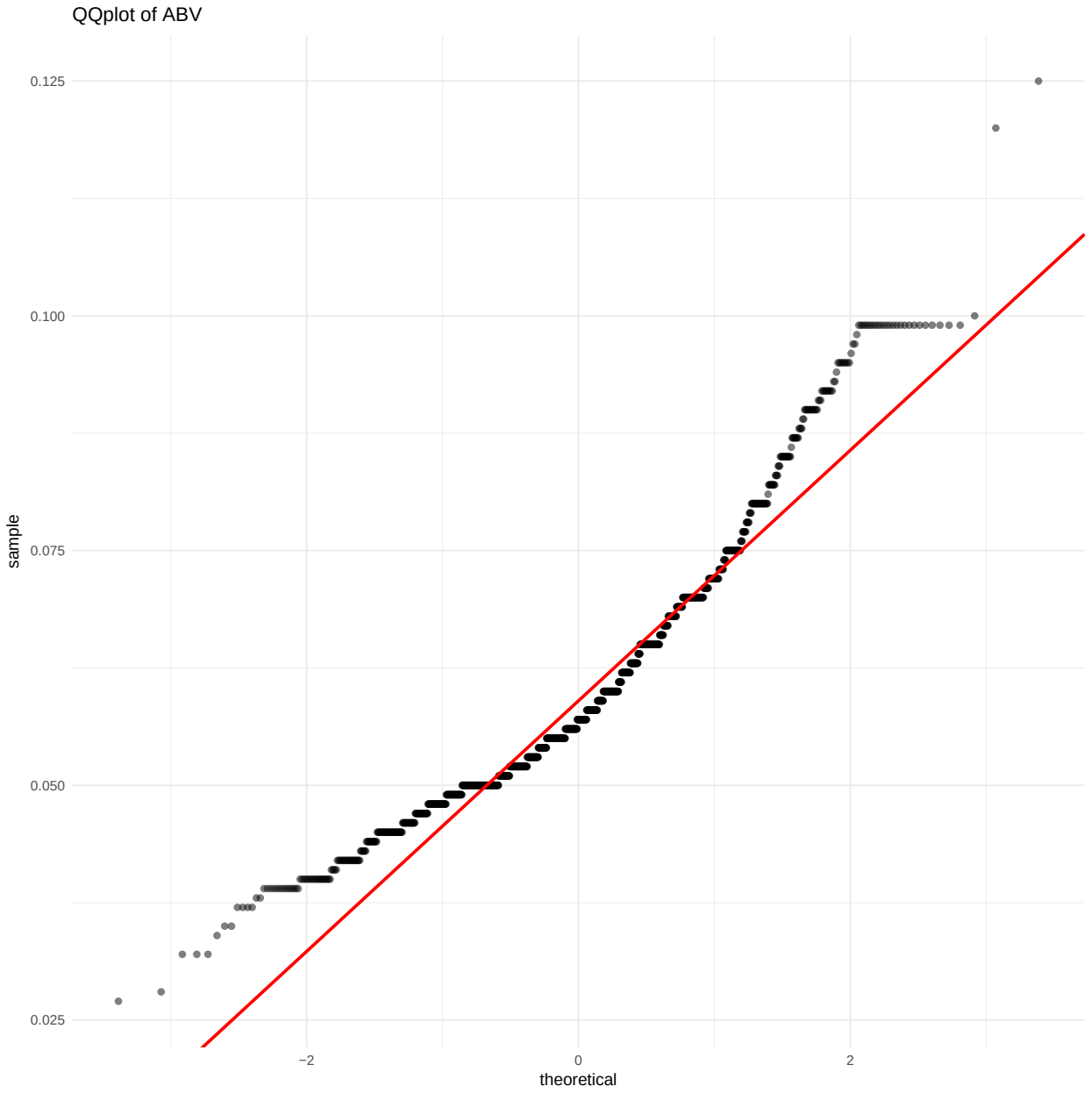
```
  #calulate line fit
  y <- quantile((styles$IBU %>% na.omit()), c(0.25, 0.75))
  x <- qnorm(c(0.25, 0.75))
  slope <- diff(y)/diff(x)
  y_int <- y[1] - slope * x[1]
```

```r
ggplot(styles, aes(sample = styles$IBU)) +
  geom_qq(shape = 16, size = 2, alpha = 0.5) +
  geom_abline(slope = slope, intercept = y_int, colour ='red', size = 1) +
  ggtitle("QQplot of IBU") +
  theme_minimal()
```




```r
#calulate line fit
y <- quantile((styles$ABV %>% na.omit()), c(0.25, 0.75))
x <- qnorm(c(0.25, 0.75))
slope <- diff(y)/diff(x)
y_int <- y[1] - slope * x[1]

ggplot(styles, aes(sample = styles$ABV)) +
  geom_qq(shape = 16, size = 2, alpha = 0.5) +
```

```
    geom_abline(slope = slope, intercept = y_int, colour ='red', size = 1) +
    ggtitle("QQplot of ABV") +
    theme_minimal()
```



Analysis using Spearman Rank-Order Correlation

`+ More info on Spearman test: https://statistics.laerd.com/statistical-guides/spearmans-rank-order-corre`

- H_o: $\rho = 0$
- H_A: $\rho \neq 0$

```
a=1-.05/2
```

```
ss
```

```
##                                Style Beer_ID   ABV IBU Ounces
## 294  American Double / Imperial IPA     628 0.091 103     12
## 842         American Pale Wheat Ale    2360 0.059  14     12
## 272  American Double / Imperial IPA    2668 0.080 100     16
## 166              American Blonde Ale    2195 0.049  20     12
## 1138                            Gose    2506 0.042  10     12
## 948                    Baltic Porter    1020 0.099  85     12
## 988                        Cream Ale    1351 0.063  30     16
## 1123                German Pilsener    2583 0.048  20     16
## 1166                     Hefeweizen    1837 0.049  30     12
## 992                        Cream Ale    2413 0.052  21     12
## 630          American Pale Ale (APA)    2683 0.042  42     16
## 790              American Pale Lager    1987 0.050  29     12
## 420                     American IPA    1610 0.060 104     16
## 920                   American Stout    2268 0.065  47     16
## 368                     American IPA     633 0.066  72     16
## 1029              English Barleywine     394 0.099  60     16
## 1286                     Pumpkin Ale    2310 0.050  10     12
## 881                 American Porter    1873 0.057  40     12
## 583                     American IPA    1749 0.046  45     12
## 608                     American IPA    1845 0.060  55     12
## 1291                         Radler    1838 0.049  30     12
## 214               American Brown Ale     655 0.070  22     16
## 1324         Saison / Farmhouse Ale     603 0.060  29     16
## 957                Belgian Pale Ale    2032 0.051  31     12
## 1335                    Schwarzbier      26 0.052  40     16
## 688          American Pale Ale (APA)    1615 0.045  32     16
## 3                   Abbey Single Ale    2505 0.049  22     12
## 308  American Double / Imperial IPA    1449 0.085 115     16
## 661          American Pale Ale (APA)    1182 0.051  36     16
## 699          American Pale Ale (APA)    2329 0.057  47     12
```

```
m = "spearman" # ABV and IBU are both ordinal

#plot(styles$IBU, styles$ABV)
results <- cor.test(styles$IBU, styles$ABV, method = m, conf.level = a)
```

```
## Warning in cor.test.default(styles$IBU, styles$ABV, method = m, conf.level
## = a): Cannot compute exact p-value with ties
```

```
results
```

```
##
##  Spearman's rank correlation rho
##
## data:  styles$IBU and styles$ABV
## S = 153570000, p-value < 2.2e-16
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.6677798
```

```
r_sq <- results[["estimate"]][["rho"]]^2
r_sq
```

```
## [1] 0.4459299
```

```
#qt(a, results[["parameter"]][["df"]])



results <- cor.test(ss$IBU, ss$ABV, method = m, conf.level = a)
```

```
## Warning in cor.test.default(ss$IBU, ss$ABV, method = m, conf.level = a):
## Cannot compute exact p-value with ties
```

```
results
```

```
##
##  Spearman's rank correlation rho
##
## data:  ss$IBU and ss$ABV
## S = 1853.2, p-value = 0.0006376
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.5877236
```

```
r_sq <- results[["estimate"]][["rho"]]^2
r_sq
```

```
## [1] 0.345419
```

```
#qt(a, results[["parameter"]][["df"]])

#plot(select(ss, IBU, ABV))

#print("There is strong evidence that the ABV and IBU are correlated (p-value < 0.001).  The IBU rating

#TODO: check number of ties
```

There is strong evidence that the ABV and IBU are correlated (p-value < 0.001). At a 95% confidence level, the IBU rating accounts for 31.5% of the variation in the ABV. While IBU and ABV certainly have a correlation, that correlation is not very strong. We can conclude that IBU rating and ABV are associated, but only that there is an association. No causality or extrapolation can be applied to these conclusions."

Alternative questions to ask: + Does the mean/median ABV of a brewery