Maye2018: Hidden Secrets of the NEIPA (TQ-55-4-1218-01.pdf)

import, tidy, scale, normalize

```
## c("Beer", "Humulinones", "Iso-a-acid", "a-Acids", "Myrcene",
## "Xanthohumol", "ß-Acids", "Turbidity (NTU)")
```

```
## Using beer as id variables
## Using beer as id variables
## Using beer as id variables
```

```
##    beer ox.alpha iso.alpha alpha myrcene xantho beta  NTU
## 1     A     34.6      18.2  31.8     1.2    3.5  9.1 1774
## 2     B     37.9      26.7  72.1     2.5    3.0  8.3 1328
## 3     C     38.4      11.4  48.0     2.4    3.1 14.0 1071
## 4     D     23.5      21.3  31.8     2.3    2.1  5.6  654
## 5     E     12.0      20.0  32.2     1.7    2.0  5.4  410
## 6     F     34.5      31.7  34.4     1.7    1.5  4.3  299
## 7     G     16.2      22.8  17.2     0.6    1.7  1.3  226
## 8     H     19.6      21.8  27.7     1.3    1.3  3.6  224
## 9     I     25.4      16.9  20.7     0.5    1.8  2.3  173
## 10    J     25.5       5.5  23.1     0.7    1.0  1.9  147
## 11    K     16.0      16.5  16.9     0.6    2.0  1.3  137
## 12    L     28.4      29.5  17.6     0.6    1.1  1.3  119
```

These NTU values seem low compared to the FTU numbers we get from Optek DT9011. But the instrument and sample prep were different. Not sure how to compare. We could purchase formazin standard if we want to get to the bottom of it.

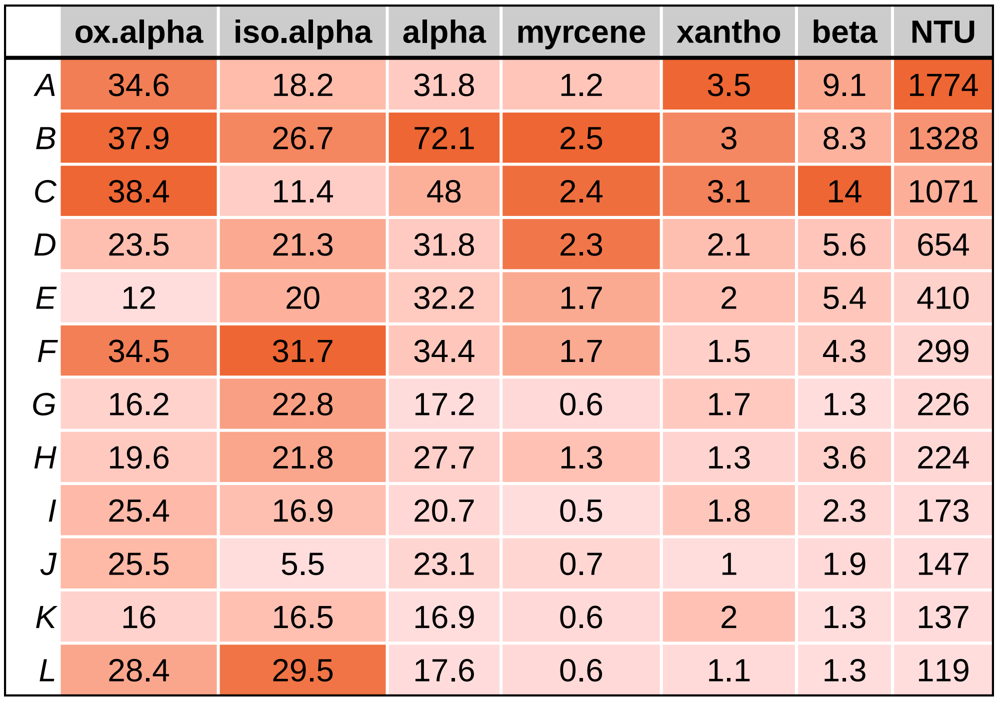
"Turbidity measurements of the NEIPAs (brought to room temperature and degassed via bath sonication) were made using a VWR Scientific model 34100-787 turbidity meter. For beer samples with turbidity >200 NTU, samples were diluted with reverse osmosis (RO) water, and the turbidity measurement was multiplied by the dilution factor. A 1,000 NTU turbidity standard (formazin standard from Aldrich Chemical Co.) was diluted with RO water to calibrate the turbidity meter; the calibration curve required a second-order polynomial fit."

```
summary(neipas)  ## of Table 2. Detailed HPLC analyses of hop compounds (mg/L) of all 12 New England IPA
```

```
##     ox.alpha       iso.alpha         alpha          myrcene     
##  Min.   :12.00   Min.   : 5.50   Min.   :16.90   Min.   :0.500  
##  1st Qu.:18.75   1st Qu.:16.80   1st Qu.:19.93   1st Qu.:0.600  
##  Median :25.45   Median :20.65   Median :29.75   Median :1.250  
##  Mean   :26.00   Mean   :20.19   Mean   :31.12   Mean   :1.342  
##  3rd Qu.:34.52   3rd Qu.:23.77   3rd Qu.:32.75   3rd Qu.:1.850  
##  Max.   :38.40   Max.   :31.70   Max.   :72.10   Max.   :2.500  
##      xantho           beta             NTU        
##  Min.   :1.000   Min.   : 1.300   Min.   : 119.0  
##  1st Qu.:1.450   1st Qu.: 1.750   1st Qu.: 166.5  
##  Median :1.900   Median : 3.950   Median : 262.5  
##  Mean   :2.008   Mean   : 4.867   Mean   : 546.8  
##  3rd Qu.:2.325   3rd Qu.: 6.275   3rd Qu.: 758.2  
##  Max.   :3.500   Max.   :14.000   Max.   :1774.0  
```

columnwise heatmap

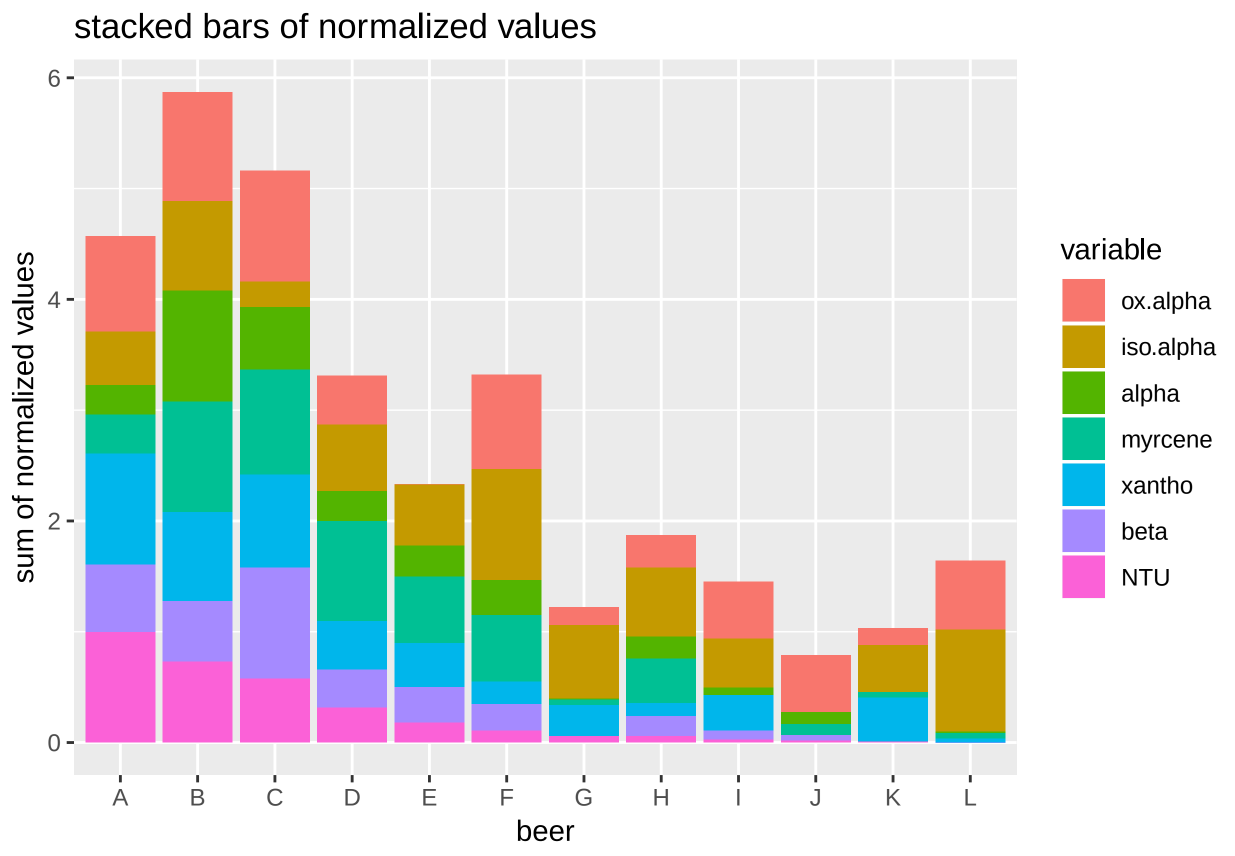
```r
# based on https://stackoverflow.com/questions/44141060/how-to-formatting-numbers-by-column-in-a-table-

mydata <- neipas
# a simple function to scale each column to the range [0, 1]
norm <- function(x) {
    apply(x, 2, function(y){(y-min(y))/(max(y)-min(y))})
}
bluecol <- colorRamp(c("#DDDDFF", "#AABBFF", "#3366EE"))(norm(mydata))
bluecol <- rgb(bluecol[, 3], bluecol[, 2], bluecol[, 1], max=255)
tt <- ttheme_default(core=list(bg_params=list(fill=bluecol)))
g <- tableGrob(mydata, theme=tt)
g <- gtable_add_grob(g,
    grobs = rectGrob(gp = gpar(fill = NA, lwd = 2)),
    t = 2, b = nrow(g), l = 1, r = ncol(g))
g <- gtable_add_grob(g,
    grobs = rectGrob(gp = gpar(fill = NA, lwd = 2)),
    t = 1, l = 1, r = ncol(g))
grid.newpage()  ## newpage must be called for draw to appear in R Notebooks
grid.draw(g)
```



	ox.alpha	iso.alpha	alpha	myrcene	xantho	beta	NTU
A	34.6	18.2	31.8	1.2	3.5	9.1	1774
B	37.9	26.7	72.1	2.5	3	8.3	1328
C	38.4	11.4	48	2.4	3.1	14	1071
D	23.5	21.3	31.8	2.3	2.1	5.6	654
E	12	20	32.2	1.7	2	5.4	410
F	34.5	31.7	34.4	1.7	1.5	4.3	299
G	16.2	22.8	17.2	0.6	1.7	1.3	226
H	19.6	21.8	27.7	1.3	1.3	3.6	224
I	25.4	16.9	20.7	0.5	1.8	2.3	173
J	25.5	5.5	23.1	0.7	1	1.9	147
K	16	16.5	16.9	0.6	2	1.3	137
L	28.4	29.5	17.6	0.6	1.1	1.3	119

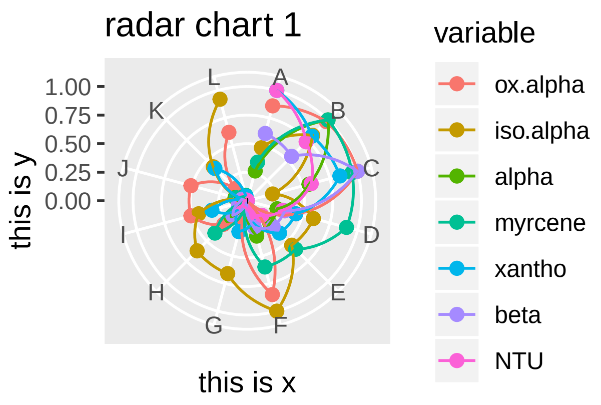
stacked bar plot

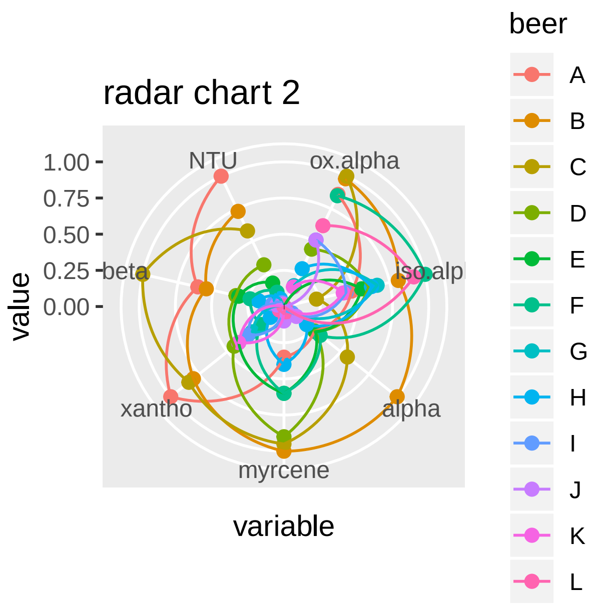
```
mydata <- melted.norm.t2_NEIPAs
ggplot() + geom_bar(aes(y=value,
                        x=beer,
                        fill=variable),
                    data=mydata,
                    stat="identity") + ylab("sum of normalized values") + ggtitle("stacked bars of norma
```



radar charts

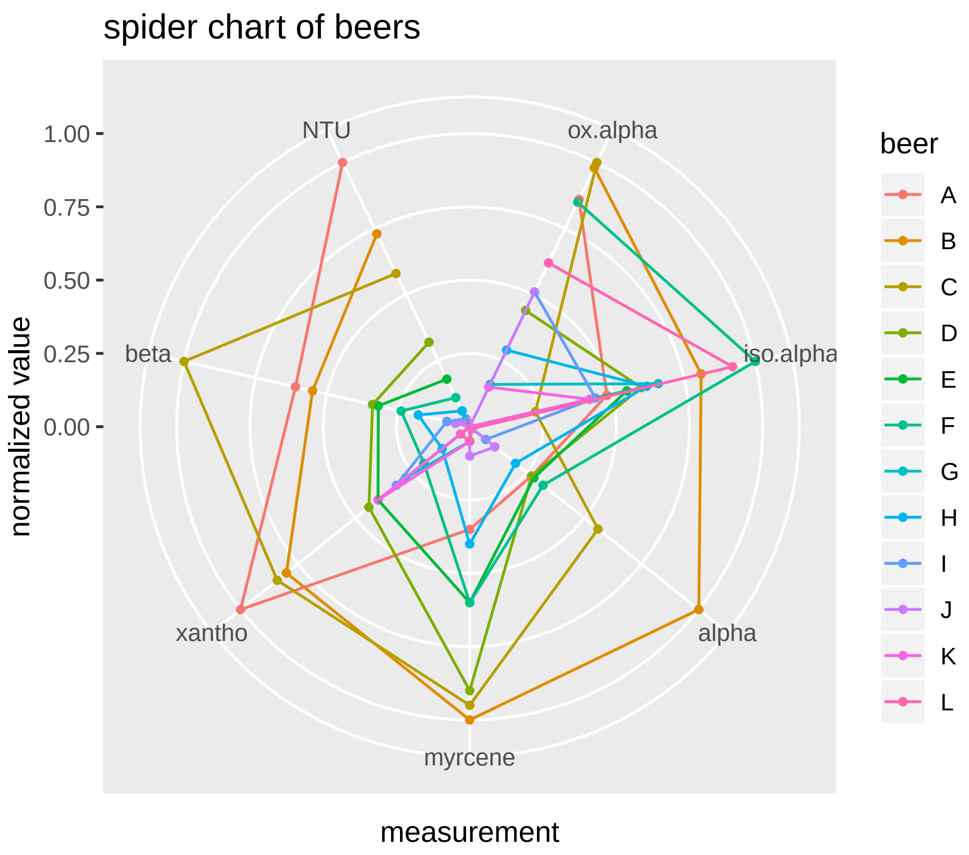
```
mydata<- melted.norm.t2_NEIPAs

p1<- ggplot(data=mydata,  aes(x=beer, y=value, group=variable, colour=variable)) +
  geom_point(size=2) + geom_line() +
  xlab("this is x") +  ylab("this is y") +
  ylim(0,1) + ggtitle("radar chart 1")  +
  geom_hline(aes(yintercept=0), lwd=1, lty=2) + coord_polar()

p2<- ggplot(data=mydata,  aes(x=variable, y=value, group=beer, colour=beer)) +
  geom_point(size=2) + geom_line() +
  ylim(0,1) + ggtitle("radar chart 2")  +
  geom_hline(aes(yintercept=0), lwd=1, lty=2) + coord_polar()

grid.arrange(p1, p2, ncol = 2)
```





coord_radar function for spider charts (straight lines connecting dots)

```
mydata<- melted.norm.t2_NEIPAs

# function from Erwan Le Pennec: From Parallel Plot to Radar Plot as cited at https://stackoverflow.com
coord_radar <- function (theta = "x", start = 0, direction = 1) {
  theta <- match.arg(theta, c("x", "y"))
  r <- if (theta == "x") "y" else "x"
  ggproto("CordRadar", CoordPolar, theta = theta, r = r, start = start,
          direction = sign(direction),
          is_linear = function(coord) TRUE)
}

ggplot(data=mydata,  aes(x=variable, y=value, group=beer, colour=beer)) + geom_point(size=1) + geom_line
  xlab("measurement") +  ylab("normalized value") +
  ylim(0,1) + ggtitle("spider chart of beers")  + coord_radar()
```



spider chart of subsets (beers with high and low Z score for NTUs)

```
## create separate dataframes for most and least hazy
df<- norm.t2_NEIPAs %>% arrange(desc(NTU))
```

```
## Warning: `as_dictionary()` is soft-deprecated as of rlang 0.3.0.
## Please use `as_data_pronoun()` instead
## This warning is displayed once per session.

## Warning: `new_overscope()` is soft-deprecated as of rlang 0.2.0.
## Please use `new_data_mask()` instead
## This warning is displayed once per session.

## Warning: The `parent` argument of `new_data_mask()` is deprecated.
## The parent of the data mask is determined from either:
##
##   * The `env` argument of `eval_tidy()`
##   * Quosure environments when applicable
## This warning is displayed once per session.

## Warning: `overscope_clean()` is soft-deprecated as of rlang 0.2.0.
## This warning is displayed once per session.
```

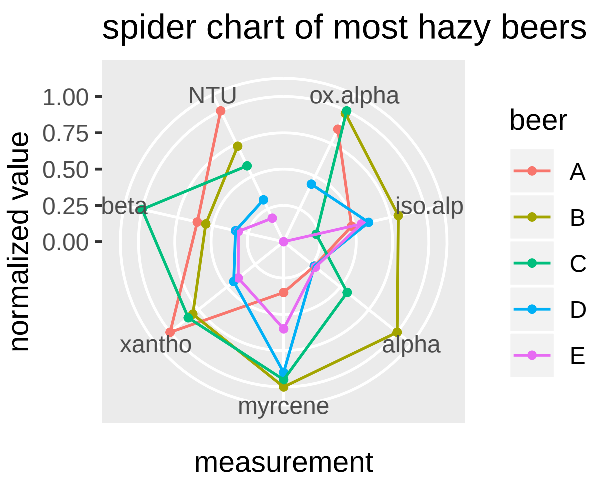
```
mosthazy <-melt(df[1:5,])
```

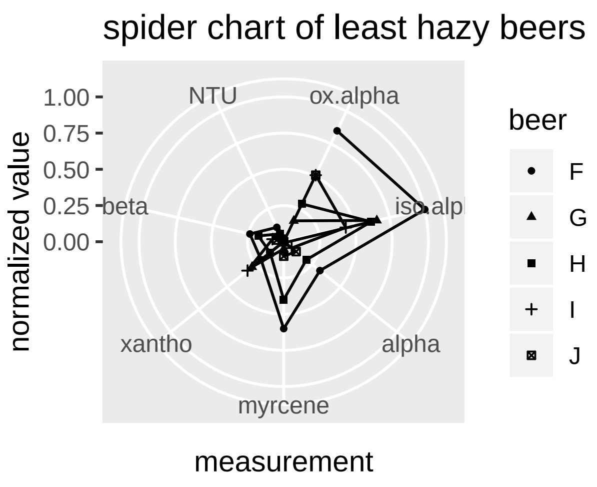
```
## Using beer as id variables
```

```
leasthazy <-melt(df[6:10,])
```

```
## Using beer as id variables
```

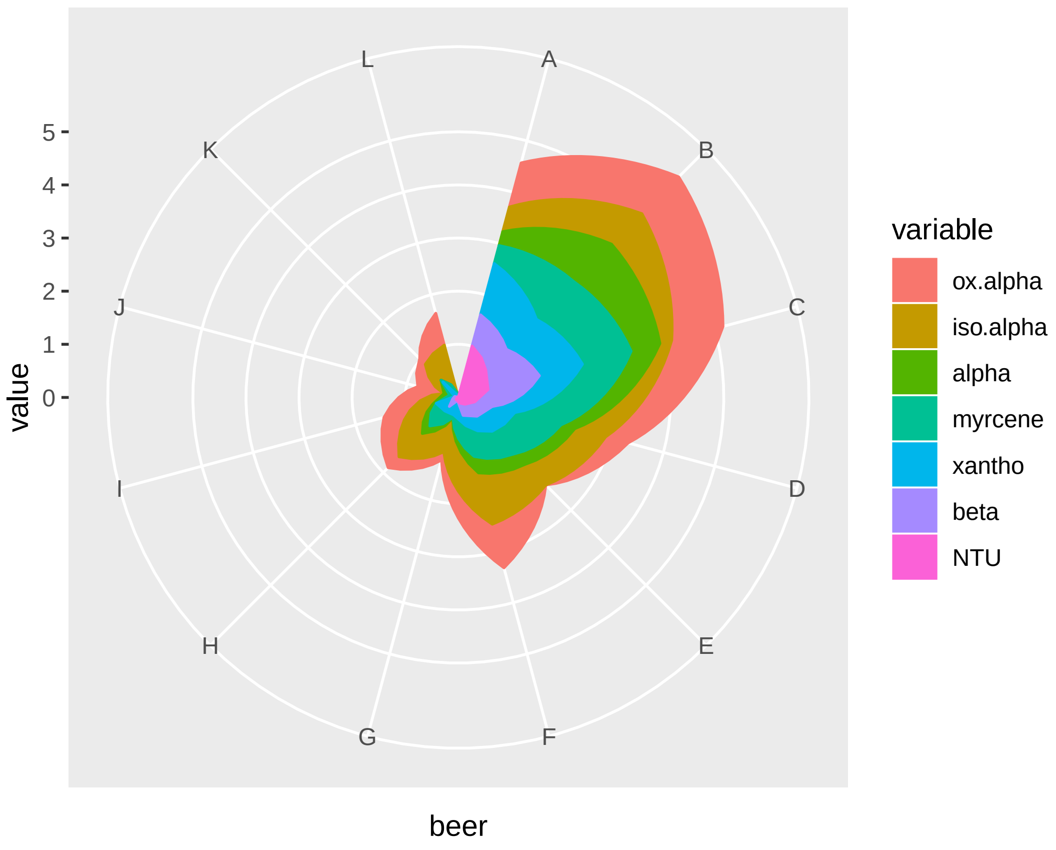
```
# function from Erwan Le Pennec: From Parallel Plot to Radar Plot as cited at https://stackoverflow.com
coord_radar <- function (theta = "x", start = 0, direction = 1) {
  theta <- match.arg(theta, c("x", "y"))
  r <- if (theta == "x") "y" else "x"
  ggproto("CordRadar", CoordPolar, theta = theta, r = r, start = start,
          direction = sign(direction),
          is_linear = function(coord) TRUE)
}
p1 <- ggplot(data=mosthazy,  aes(x=variable, y=value, group=beer, colour=beer)) + geom_point(size=1) +
  xlab("measurement") +  ylab("normalized value") +
  ylim(0,1) + ggtitle("spider chart of most hazy beers")  + coord_radar()
p2 <- ggplot(data=leasthazy,  aes(x=variable, y=value, group=beer, shape=beer)) + geom_point(size=1) +
  xlab("measurement") +  ylab("normalized value") +
  ylim(0,1) + ggtitle("spider chart of least hazy beers")  + coord_radar()
grid.arrange(p1, p2, ncol = 2)
```





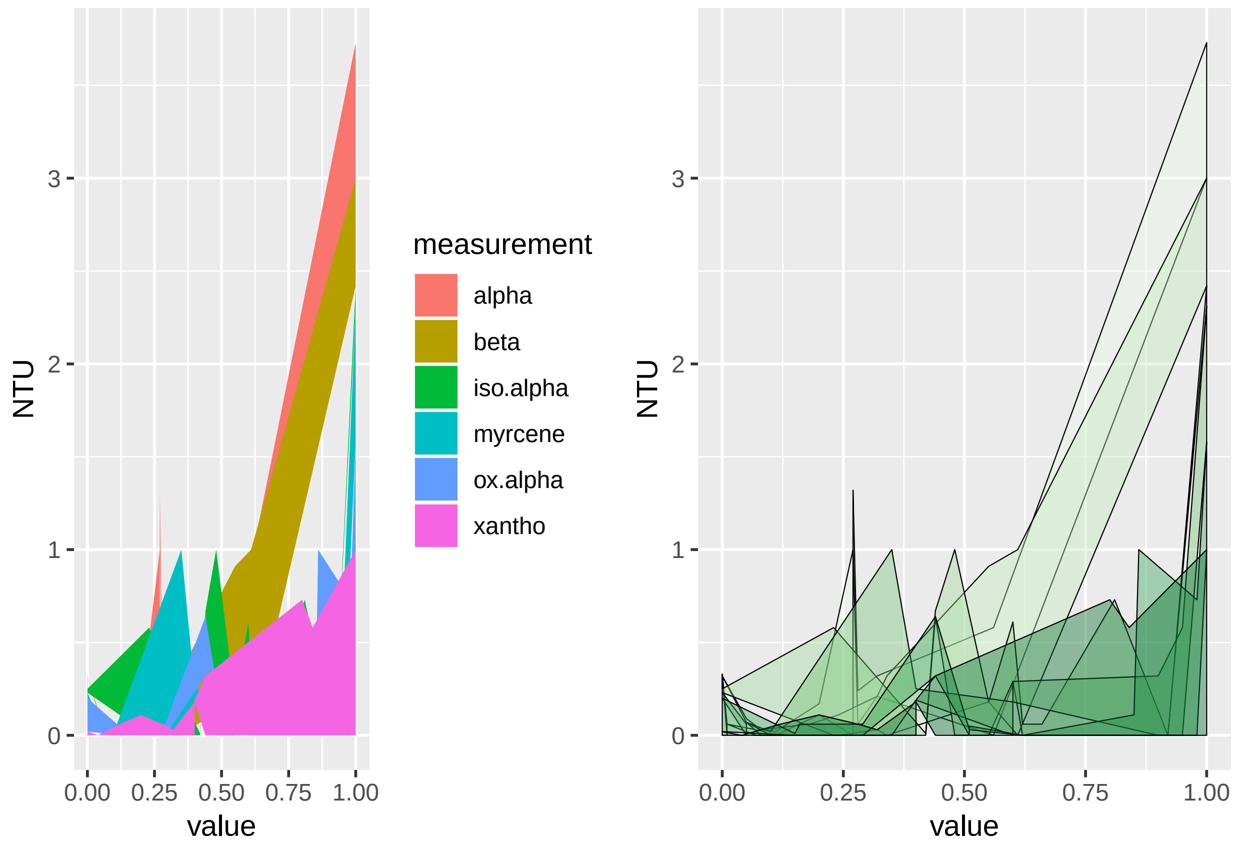
“stacked spider chart”

```
mydata <- melted.norm.t2_NEIPAs

p = ggplot(data=mydata, aes(x=beer, y=value, group=variable))
p + geom_area(aes(color=variable, fill=variable)) + coord_polar()
```


value
5
4
3
2
1
0
L
A
B
C
D
E
F
G
H
I
J
K
beer
variable
ox.alpha
iso.alpha
alpha
myrcene
xantho
beta
NTU

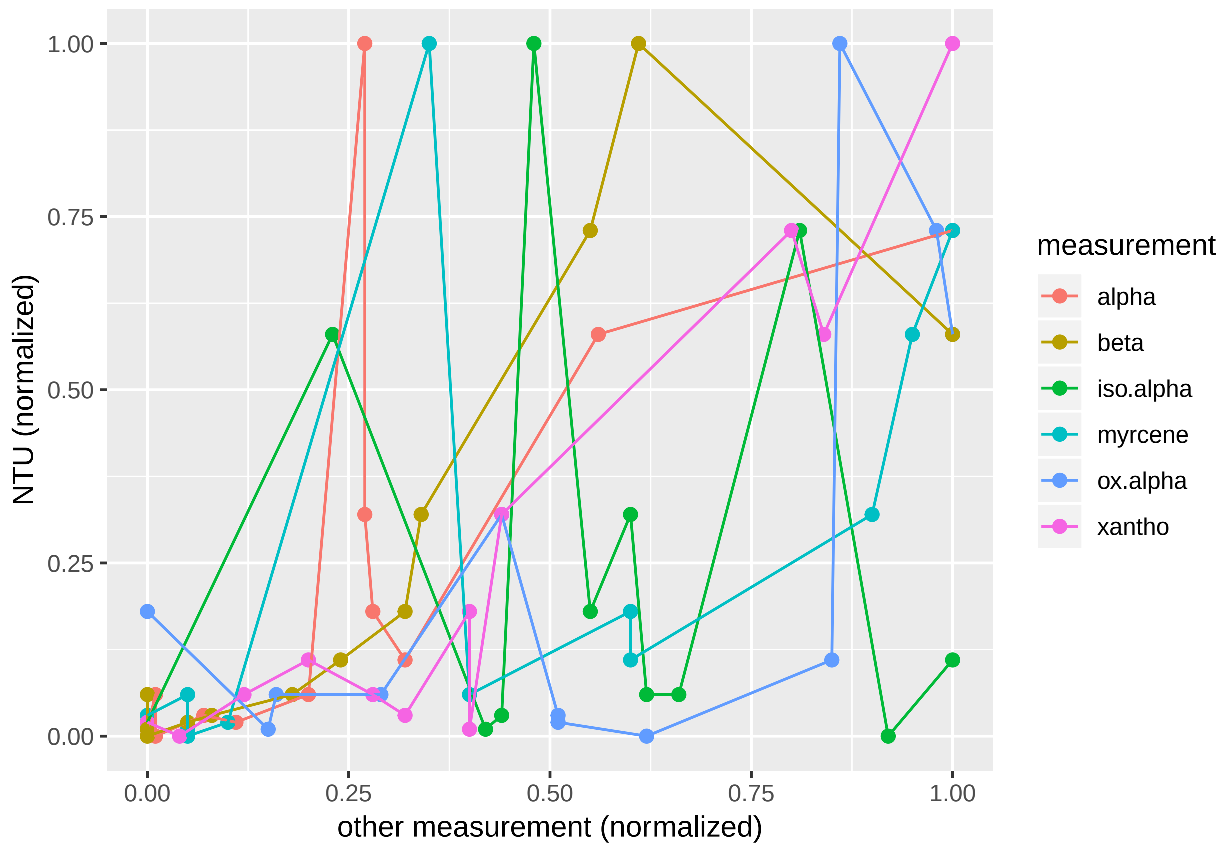
stacked area chart

```
mydata<- norm.gatherNTU
p1<-ggplot(mydata, aes(x=value, y=NTU, fill=measurement)) + geom_area()
p2<- ggplot(mydata, aes(x=value, y=NTU, fill=measurement)) +
    geom_area(colour="black", size=.2, alpha=.4) +
    scale_fill_brewer(palette="Greens", breaks=rev(levels(norm.gatherNTU$measurement)))

grid.arrange(p1, p2, ncol = 2)
```



scatter plot of normalized values (NTU vs other measurements)

```
mydata<- norm.gatherNTU

ggplot(mydata, aes(y=NTU,
                   x=value,
                   color=measurement)) +
#                  , shape=beer)) +
geom_point(size=2) +
geom_line() +
ylab("NTU (normalized)") +
xlab("other measurement (normalized)")
```



note: in the case above, each beer is a horizontal row of dots

correlation matrix

```r
mydata<- neipas
## following  Manuel Amunategui  https://www.youtube.com/watch?v=igPQ-pI8Bjo
## Using Correlations To Understand Your Data: Machine Learning With R
##functions for flattenSquareMatrix
cor.prob <- function (X, dfr=nrow(X) -2) {
  R<- cor(X, use="pairwise.complete.obs")
  above<- row(R) < col(R)
  r2 <- R[above]^2
  Fstat<- r2 * dfr/(1-r2)
  R[above] <- 1- pf(Fstat, 1, dfr)
  R[row(R) == col(R)] <- NA
  R
}
flattenSquareMatrix <- function(m) {
  if( (class(m) != "matrix") | (nrow(m)!=ncol(m))) stop("Must be a square matrix.")
  if(!identical(rownames(m), colnames(m))) stop("Row and column names must be equal.")
  ut <- upper.tri(m)
  data.frame(i = rownames(m)[row(m)[ut]],
             j = rownames(m)[col(m)[ut]],
             cor=t(m)[ut],
             p=m[ut])
}
corMasterList<- flattenSquareMatrix(cor.prob(mydata))    ## list of all correlations
corlist<- corMasterList[order(-abs(corMasterList$cor)),]  ## order by strength of correlation
corlist[corlist$j=="NTU",]
```

```
##             i   j         cor            p
## 20    xantho NTU  0.92392722 1.764096e-05
## 21      beta NTU  0.81956568 1.102987e-03
## 18     alpha NTU  0.68031852 1.490403e-02
## 16  ox.alpha NTU  0.63551566 2.635946e-02
## 19   myrcene NTU  0.58143915 4.737256e-02
## 17 iso.alpha NTU -0.03135355 9.229419e-01
```

Strongest correlations with NTU in descending order are xanthohumol (R= 0.9239272) and lupulones (R= 0.8195657). Fairly strong correlations with everything except isohumulones (R= -0.0313535).

linear model (NTU as a function of normalized measurements)

```
mymodel <- lm(NTU~alpha*beta*xantho, data=norm.neipas)
summary(mymodel)
```

```
##
## Call:
## lm(formula = NTU ~ alpha * beta * xantho, data = norm.neipas)
##
## Residuals:
##          A          B          C          D          E          F
##  0.0019339  0.0012525 -0.0020268  0.0542210 -0.0572521 -0.0064642
##          G          H          I          J          K          L
##  0.0401071  0.0284773 -0.0436069 -0.0008448 -0.0026616 -0.0131354
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)        0.007253   0.057793   0.125    0.906
## alpha              0.475647   0.672917   0.707    0.519
## beta              -0.725841   0.931950  -0.779    0.480
## xantho             0.013522   0.192157   0.070    0.947
## alpha:beta        -0.443097   3.247966  -0.136    0.898
## alpha:xantho       1.463377   1.020564   1.434    0.225
## beta:xantho        2.423264   1.144702   2.117    0.102
## alpha:beta:xantho -3.088618   3.987586  -0.775    0.482
##
## Residual standard error: 0.05189 on 4 degrees of freedom
## Multiple R-squared:  0.9912, Adjusted R-squared:  0.9758
## F-statistic: 64.37 on 7 and 4 DF,  p-value: 0.0006009
```

According to this model of these data, NTUs can be predicted almost entirely from alpha,beta, and xathohumol values. Interactions with xanthohumol are the most impactful components.

compare multiple models with mtable function in package memisc

```
lm1<-lm(NTU~xantho, data= norm.t2_NEIPAs)
lm2<-lm(NTU~beta*xantho, data= norm.t2_NEIPAs)
lm3<-lm(NTU~myrcene*beta*xantho, data= norm.t2_NEIPAs)
lm4<-lm(NTU~alpha*beta*xantho, data=norm.neipas)

mtable1234 <- mtable("Model 1"=lm1,"Model 2"=lm2,"Model 3"=lm3, "Model 4"=lm4,
                    summary.stats=c("sigma","R-squared","F","p","N"),show.eqnames=T)
mtable1234b <- relabel(mtable1234,
                      "(Intercept)" = "Constant",
                      SG = "Specific Gravity",
                      ABW = "ABW = Ethanol (w/w)",
                      Er = "Er = Residual Extract (g/100mL)"
                      )
mtable1234
```

```
##
## Calls:
## Model 1: lm(formula = NTU ~ xantho, data = norm.t2_NEIPAs)
## Model 2: lm(formula = NTU ~ beta * xantho, data = norm.t2_NEIPAs)
## Model 3: lm(formula = NTU ~ myrcene * beta * xantho, data = norm.t2_NEIPAs)
## Model 4: lm(formula = NTU ~ alpha * beta * xantho, data = norm.neipas)
##
## ==================================================================
##                              Model 1   Model 2  Model 3   Model 4
##                             ---------- -------- --------- --------
##                                NTU       NTU      NTU       NTU
## ------------------------------------------------------------------
##   (Intercept)                -0.127    -0.080   -0.015    0.007
##                              (0.064)   (0.098)  (0.035)  (0.058)
##   xantho                      0.956***  0.705*   0.027    0.014
##                              (0.125)   (0.293)  (0.114)  (0.192)
##   beta                                 -0.070   -0.525   -0.726
##                                        (0.533)  (0.434)  (0.932)
##   beta x xantho                         0.393    2.011*   2.423
##                                        (0.703)  (0.475)  (1.145)
##   myrcene                                        0.663
##                                                 (0.302)
##   myrcene x beta                                -2.416
##                                                 (0.941)
##   myrcene x xantho                               0.555
##                                                 (0.384)
##   myrcene x beta x xantho                        0.791
##                                                 (1.170)
##   alpha                                                   0.476
##                                                          (0.673)
##   alpha x beta                                           -0.443
##                                                          (3.248)
##   alpha x xantho                                          1.463
##                                                          (1.021)
##   alpha x beta x xantho                                  -3.089
##                                                          (3.988)
## ------------------------------------------------------------------
##   sigma                     0.134      0.142     0.033    0.052
##   R-squared                 0.854      0.869     0.997    0.991
##   F                        58.354     17.693   162.744   64.367
##   p                         0.000      0.001     0.000    0.001
##   N                        12         12        12       12
## ==================================================================
```

```
#show_html(mtable1234b)
```

According to this model of these data, NTUs can be predicted almost entirely from myrcene,beta, and xanthohumol values. The most impactful components of this model (in terms of positive contribution to haze) are interactions between lupulones and xanthohumol (R =

r model.frame(lm4)[4, 1])

Table2: beer humulinones isoalpha alpha myrcene xanthohumol beta NTU A 34.6 18.2 31.8 1.2 3.5 9.1 1774 B 37.9 26.7 72.1 2.5 3.0 8.3 1328 C 38.4 11.4 48.0 2.4 3.1 14.0 1071 D 23.5 21.3 31.8 2.3 2.1 5.6 654 E 12.0 20.0 32.2 1.7 2.0 5.4 410 F 34.5 31.7 34.4 1.7 1.5 4.3 299 G 16.2 22.8 17.2 0.6 1.7 1.3 226 H 19.6 21.8 27.7 1.3 1.3 3.6 224 I 25.4 16.9 20.7 0.5 1.8 2.3 173 J 25.5 5.5 23.1 0.7 1.0 1.9 147

```
sessionInfo()
```

```
## R version 3.5.2 (2018-12-20)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## 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] grid      stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] bindrcpp_0.2.2    memisc_0.99.14.12 MASS_7.3-51.1
##  [4] lattice_0.20-38   data.table_1.12.0 gridExtra_2.3
##  [7] gtable_0.2.0      forcats_0.3.0     stringr_1.3.1
## [10] purrr_0.2.4       readr_1.1.1       tibble_1.4.2
## [13] tidyverse_1.2.1   ggplot2_3.0.0     tidyr_0.8.0
## [16] dplyr_0.7.4       readxl_1.1.0      pdftools_2.1
##
## loaded via a namespace (and not attached):
##  [1] tidyselect_0.2.4   xfun_0.4           repr_0.19.1
##  [4] reshape2_1.4.3     haven_1.1.1        colorspace_1.3-2
##  [7] htmltools_0.3.6    base64enc_0.1-3    yaml_2.1.19
## [10] rlang_0.3.1        pillar_1.2.2       foreign_0.8-71
## [13] glue_1.2.0         withr_2.1.2        RColorBrewer_1.1-2
## [16] modelr_0.1.2       bindr_0.1.1        plyr_1.8.4
## [19] munsell_0.4.3      cellranger_1.1.0   rvest_0.3.2
## [22] psych_1.8.4        evaluate_0.10.1    labeling_0.3
## [25] knitr_1.21         parallel_3.5.2     broom_0.4.4
## [28] Rcpp_0.12.16       scales_0.5.0       backports_1.1.2
## [31] jsonlite_1.5       mnormt_1.5-5       hms_0.4.2
## [34] digest_0.6.15      stringi_1.1.7      rprojroot_1.3-2
## [37] cli_1.0.0          tools_3.5.2        magrittr_1.5
## [40] lazyeval_0.2.1     crayon_1.3.4       pkgconfig_2.0.1
## [43] xml2_1.2.0         lubridate_1.7.4    assertthat_0.2.0
## [46] rmarkdown_1.9      httr_1.3.1         rstudioapi_0.7
## [49] R6_2.2.2           nlme_3.1-137       compiler_3.5.2
```