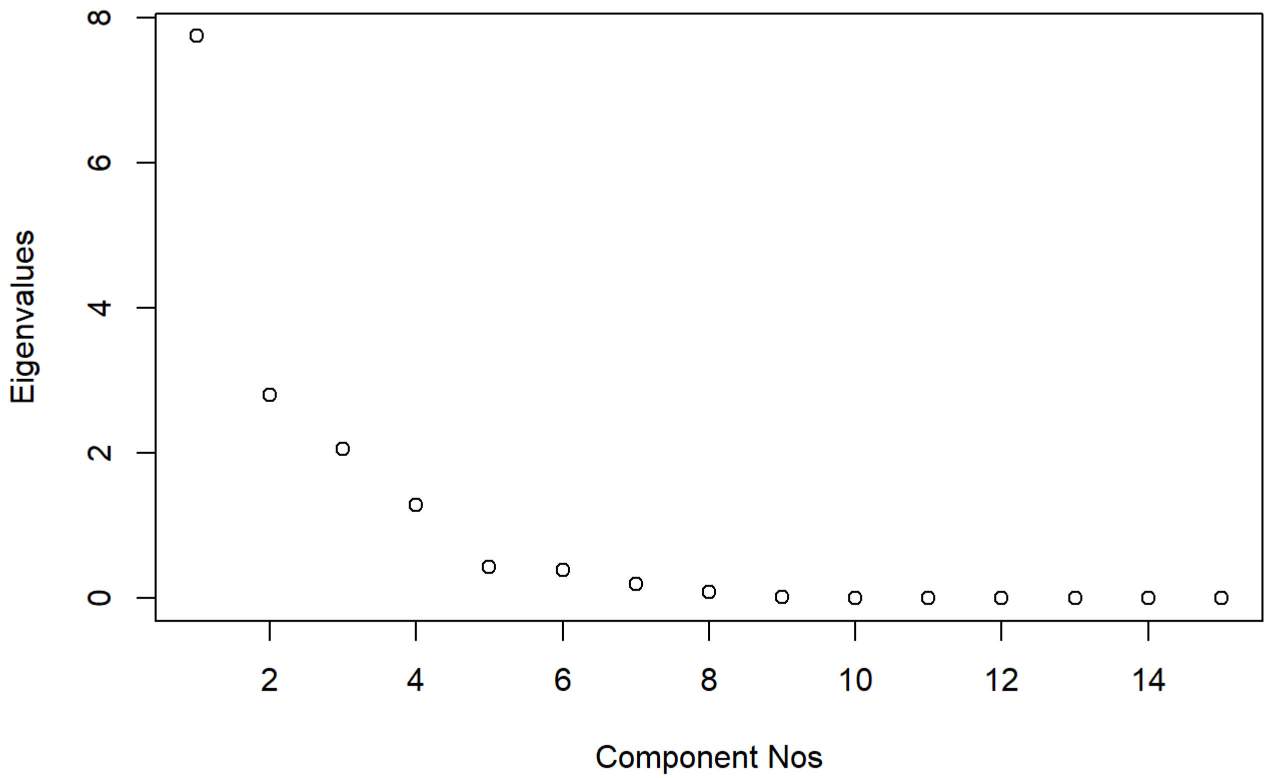
Brand Maps & PCA

1. Build brand maps for car brands for the client’s brand - Infinity.

```
rm(list=ls(all=TRUE))

cars <- read.csv("Cars_Data.csv", header=T)

y <-  cars[,17]
x <-  as.matrix(cars[,2:16])
data <- cbind(y,x)
cor_mat = cor(x)

out1 <-  eigen(cor_mat) # eigen decomposition of correlation matrix
va <-  out1$values          # eigenvalues
ve <-  out1$vectors     # eigenvector
```

2. Determine how many factors to retain?

```
# scree plot
plot(va, ylab = "Eigenvalues", xlab = "Component Nos")
```




```
ego <- va[va > 1]                          # eigenvalues > 1
nn <- nrow(as.matrix(ego))                 # number of factors to retain

print(paste("Factors to retain : ", nn))
```

```
## [1] "Factors to retain :  4"
```

3. Assign names to the retained factors

```r
out2 <- ve[,1:nn]                           # eigenvectors associated with the retained factors
out3 <- ifelse(abs(out2) < 0.3, 0, out2)        # ignore small values < 0.3

rownames(out3) <- c("Attractive", "Quiet", "Unreliable", "Poorly.Built", "Interesting", "Sporty", "Uncomfortable", "Roomy",
                    "Easy.Service", "Prestige", "Common", "Economical", "Successful", "AvantGarde", "Poor.Value")

z = x %*% out3          # Component Scores; coordinates of the brands in the map

pref_reg = lm(y ~ z)        # Preference Regression to estimate how benefits drive overall preferences = f(benefits)

summary(pref_reg)
```

```
##
## Call:
## lm(formula = y ~ z)
##
## Residuals:
##        1        2        3        4        5        6        7        8
## -0.41399  0.20488  0.21196  0.17883 -0.17872  0.29476  0.07001 -0.13008
##        9       10
## -0.11977 -0.11788
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.18458    0.43161   5.061 0.003895 **
## z1          -0.68944    0.07258  -9.499 0.000219 ***
## z2          -0.28516    0.11920  -2.392 0.062209 .
## z3           0.46138    0.11900   3.877 0.011675 *
## z4           0.21219    0.21267   0.998 0.364205
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.303 on 5 degrees of freedom
## Multiple R-squared:  0.9742, Adjusted R-squared:  0.9536
## F-statistic: 47.21 on 4 and 5 DF,  p-value: 0.0003671
```

```r
# Since estimates should be positive, flipping the eigenvectors for Z1 and Z2

out4  <- out3

out4[,1] <- (-1)*out4[,1]
out4[,2] <- (-1)*out4[,2]

z = x %*% out4

pref_reg = lm(y ~ z)

summary(pref_reg)
```

```
##
## Call:
## lm(formula = y ~ z)
##
## Residuals:
##        1        2        3        4        5        6        7        8
## -0.41399  0.20488  0.21196  0.17883 -0.17872  0.29476  0.07001 -0.13008
##        9       10
## -0.11977 -0.11788
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.18458    0.43161   5.061 0.003895 **
## z1           0.68944    0.07258   9.499 0.000219 ***
## z2           0.28516    0.11920   2.392 0.062209 .
## z3           0.46138    0.11900   3.877 0.011675 *
## z4           0.21219    0.21267   0.998 0.364205
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.303 on 5 degrees of freedom
## Multiple R-squared:  0.9742, Adjusted R-squared:  0.9536
## F-statistic: 47.21 on 4 and 5 DF,  p-value: 0.0003671
```

```
# eliminating Z4 as is it not significant

out4 <- out4[,1:3]

out4
```

```
##                    [,1]       [,2]       [,3]
## Attractive     0.3262814  0.0000000  0.0000000
## Quiet          0.3173818  0.0000000  0.0000000
## Unreliable     0.0000000  0.4011014  0.0000000
## Poorly.Built  -0.3319244  0.0000000  0.0000000
## Interesting    0.0000000  0.0000000 -0.4290683
## Sporty         0.0000000 -0.4288285  0.0000000
## Uncomfortable  0.0000000  0.0000000  0.0000000
## Roomy          0.0000000  0.3789204  0.0000000
## Easy.Service   0.0000000 -0.4279673  0.0000000
## Prestige       0.3322855  0.0000000  0.0000000
## Common         0.0000000  0.0000000  0.0000000
## Economical     0.0000000  0.0000000  0.6642165
## Successful     0.3155394  0.0000000  0.0000000
## AvantGarde     0.0000000  0.0000000  0.0000000
## Poor.Value     0.0000000  0.0000000 -0.4710589
```

Names

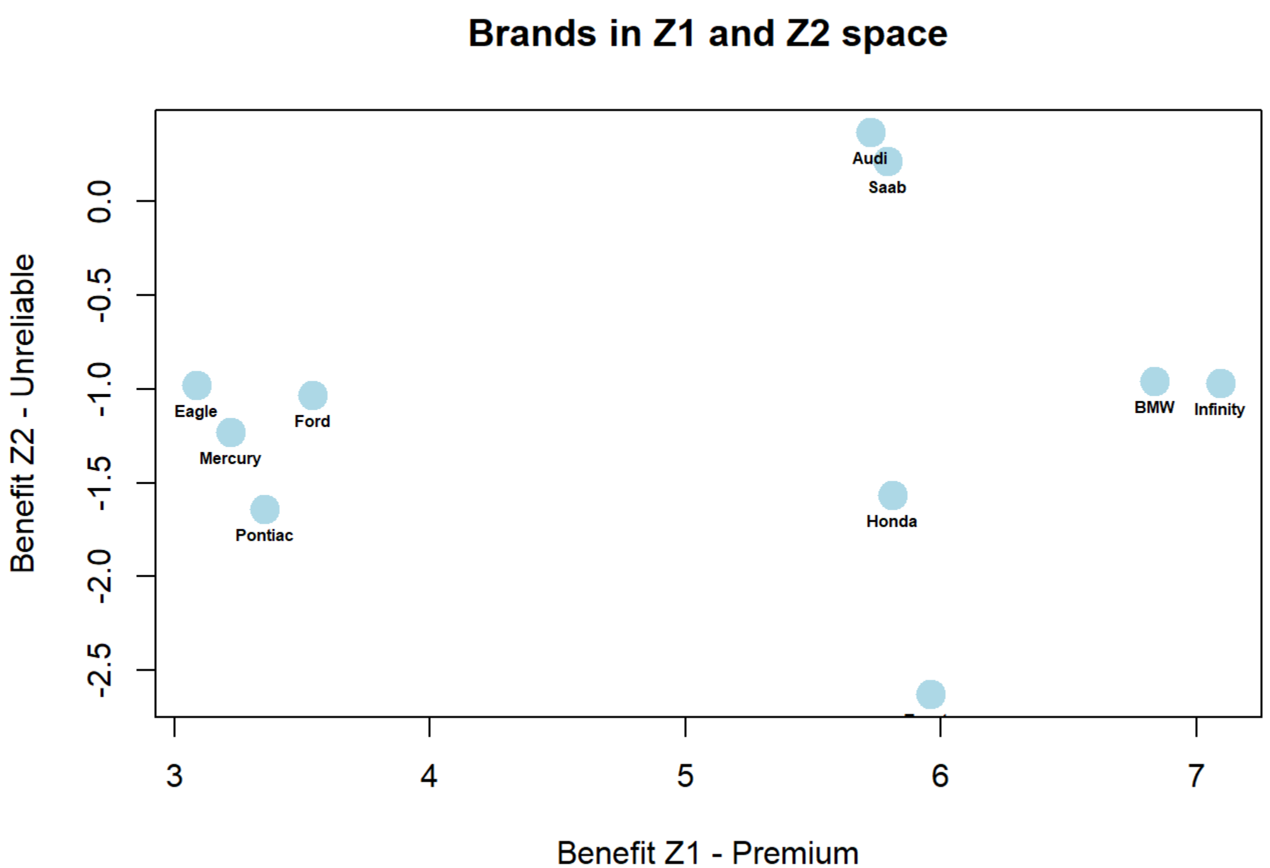
Z1 - Premium
Z2 - Unreliable
Z3 - Value for Money

4. Compute the angles of iso-preference line and ideal vector arrow

Brand Maps

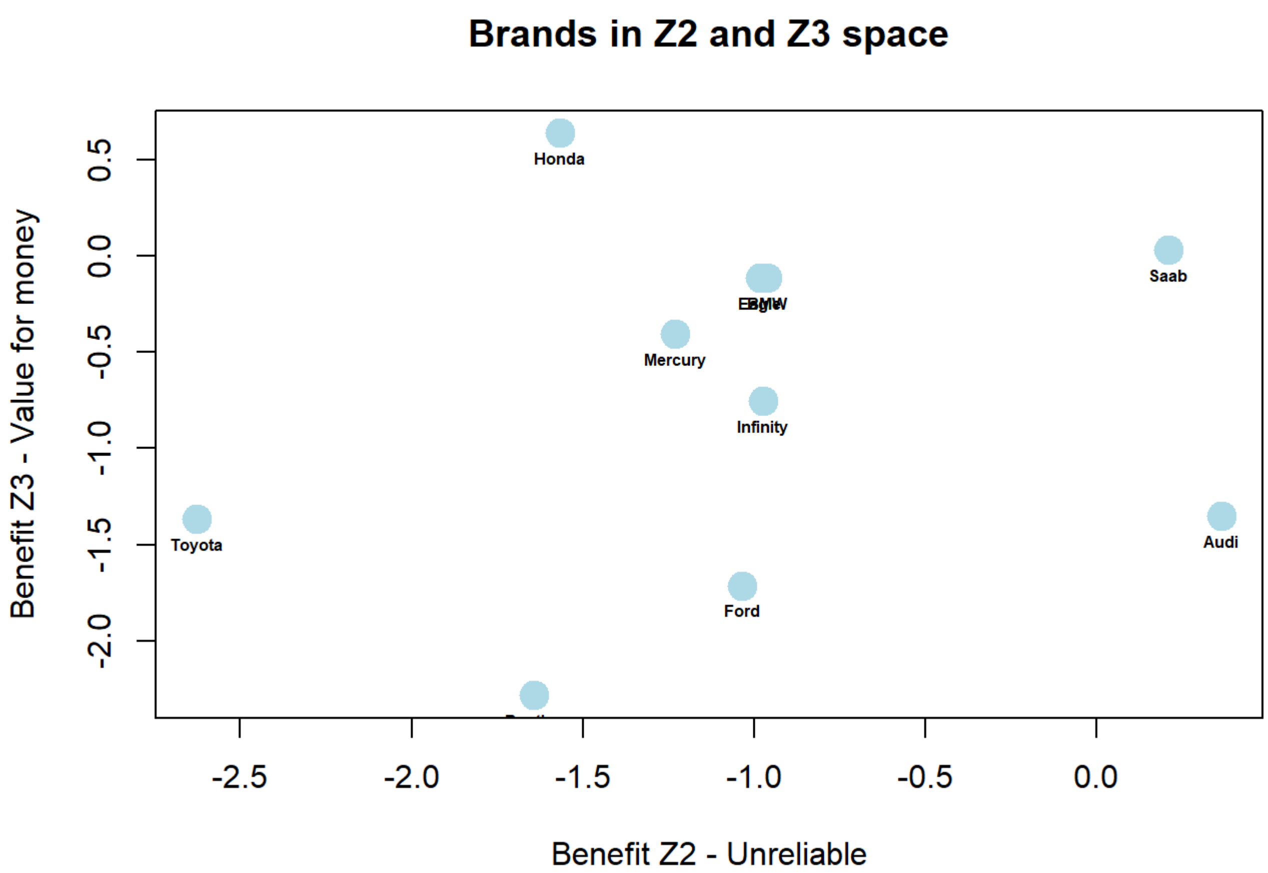
Z1, Z2

```
Z1 = z[,1]
Z2 = z[,2]
z.out = cbind(Z1, Z2)
rownames(z.out) = cars[,1]

plot(Z1, Z2, main = "Brands in Z1 and Z2 space", xlab = "Benefit Z1 - Premium", ylab = "Benefit Z2 - Unreliable", col = "lig
htblue", pch = 19, cex = 2) # Brand Map in Z1-Z2 space
text(z.out, labels = row.names(z.out), font = 2, cex = 0.5, pos = 1)                    # labeling brands
```



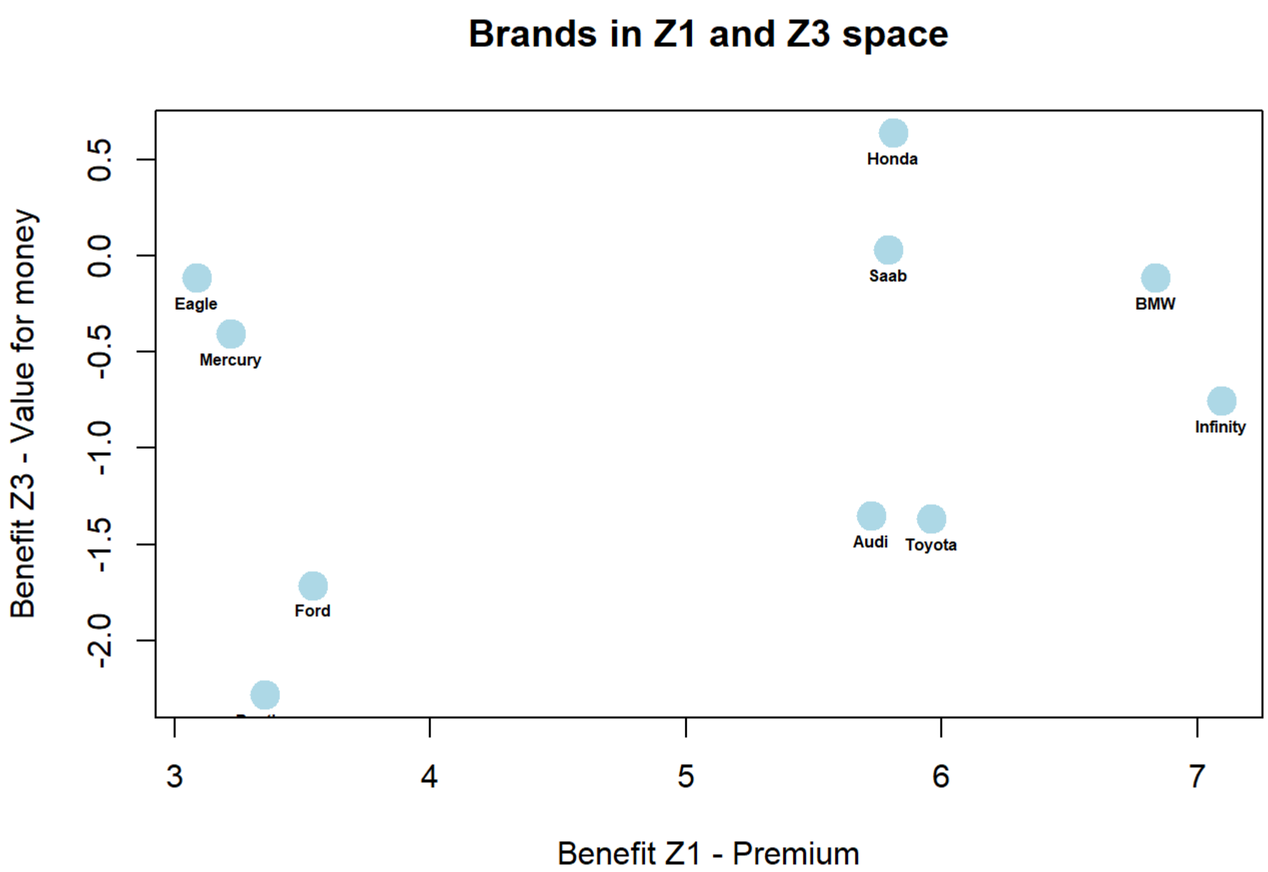
Z2, Z3

```
Z3 = z[,3]
z1.out = cbind(Z2, Z3)
rownames(z1.out) = cars[,1]

plot(Z2, Z3, main = "Brands in Z2 and Z3 space", xlab = "Benefit Z2 - Unreliable", ylab = "Benefit Z3 - Value for money", co
l = "lightblue", pch = 19, cex = 2) # Brand Map in Z2-Z3 space
text(z1.out, labels = row.names(z1.out), font = 2, cex = 0.5, pos = 1)                    # labeling brands
```



Z1, Z3

```
z2.out = cbind(Z1, Z3)
rownames(z2.out) = cars[,1]

plot(Z1, Z3, main = "Brands in Z1 and Z3 space", xlab = "Benefit Z1 - Premium", ylab = "Benefit Z3 - Value for money", col =
"lightblue", pch = 19, cex = 2) # Brand Map in Z1-Z2 space
text(z2.out, labels = row.names(z2.out), font = 2, cex = 0.5, pos = 1)                    # labeling brands
```




```
# Slopes of iso-preference and ideal vector
```

For Z1 and Z2

```
b1 = as.vector(coef(pref_reg)[2])
b2 = as.vector(coef(pref_reg)[3])
slope.iso.preference = - b1/b2
slope.ideal.vector = b2/b1

# Angles of iso-preference and ideal vector

angle.iso.preference = atan(slope.iso.preference)*180/pi
angle.ideal.vector = atan(slope.ideal.vector)*180/pi

print(paste("angle.iso.preference : ", angle.iso.preference))
```

```
## [1] "angle.iso.preference :  -67.5297112315307"
```

```
print(paste("angle.ideal.vector : ", angle.ideal.vector))
```

```
## [1] "angle.ideal.vector :  22.4702887684693"
```

For Z2 and Z3

```
b3 = as.vector(coef(pref_reg)[4])
slope.iso.preference = - b2/b3
slope.ideal.vector = b3/b2

# Angles of iso-preference and ideal vector

angle.iso.preference = atan(slope.iso.preference)*180/pi
angle.ideal.vector = atan(slope.ideal.vector)*180/pi

print(paste("angle.iso.preference : ", angle.iso.preference))
```

```
## [1] "angle.iso.preference :  -31.7182997006912"
```

```
print(paste("angle.ideal.vector : ", angle.ideal.vector))
```

```
## [1] "angle.ideal.vector :  58.2817002993088"
```

For Z1 and Z3

```
slope.iso.preference = - b1/b3
slope.ideal.vector = b3/b1

# Angles of iso-preference and ideal vector

angle.iso.preference = atan(slope.iso.preference)*180/pi
angle.ideal.vector = atan(slope.ideal.vector)*180/pi

print(paste("angle.iso.preference : ", angle.iso.preference))
```

```
## [1] "angle.iso.preference :  -56.2092740146157"
```

```
print(paste("angle.ideal.vector : ", angle.ideal.vector))
```

```
## [1] "angle.ideal.vector :  33.7907259853843"
```

5. Find 95% confidence interval for the angle of the ideal vector using data bootstrap method.

```
angle <- list()
r <- nrow(data)

set.seed(876)

# Do Data Bootstrap 1000 times to get 95% CI for R^2
for(i in 1:1000) {
    tryCatch({

              data.star <- data[sample(r, r, replace = T),]     # create (y*, x*) by resampling rows in original data matrix
            ystar <- data.star[,1]
            xstar <- data.star[,2:16]
            cor_mat = cor(xstar)
            o1 <-  eigen(cor_mat)   # eigen decomposition of correlation matrix
            va <-  o1$values            # eigenvalues
            ve <-  o1$vectors
            ego <- va[va > 1]                          # eigenvalues > 1
            n <- nrow(as.matrix(ego))
            o2 <- ve[,1:n]                         # eigenvectors associated with the retained factors
            o3 <- ifelse(abs(o2) < 0.3, 0, o2)      # ignore small values < 0.3
            rownames(o3) <- c("Attractive", "Quiet", "Unreliable", "Poorly.Built", "Interesting", "Sporty", "Uncomfortable",
"Roomy",
                                "Easy.Service", "Prestige", "Common", "Economical", "Successful", "AvantGarde", "Poor.Value"
)
            zstar = xstar %*% o3            # Component Scores; coordinates of the brands in the map
            pref_reg_star = lm(ystar ~ zstar)       # Preference Regression to estimate how benefits drive overall preferenc
es = f(benefits)

            # flipping o3 if coefficients are negative
            for (j in 2:length(coef(pref_reg_star))){
              if(coef(pref_reg_star)[j] < 0){
                o3[,j-1] = (-1)*o3[,j-1]
              }
            }
            # new model
            zstar = xstar %*% o3
            pref_reg_star = lm(ystar ~ zstar)

            # finding significant coefficients
```

```
      cf <- data.frame(summary(pref_reg_star)$coef) # collecting coefficients
      cf <- cf[-1,] # removing intercept
      a <- cf[cf[,4] <= .05, 1] # taking significant coefficients only

      # making sure that there are more than 1 significant variable
      if(length(a) > 1){
        b <- t(combn(a, 2)) # creating pairs
          c <- c()

          # calculating angle
          for (k in 1:nrow(b)) {
             c <- c(c, atan(b[k,2]/b[k,1])*180/pi)
          }

          # saving the output
          angle[[i]] <- c
      }
    }, error=function(e){})
}
```

```
## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable
```

```
## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable
```

```
## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable

## Warning in summary.lm(pref_reg_star): essentially perfect fit: summary may be
## unreliable
```

```
# removing nulls and converting it to data frame

df <- plyr::ldply(angle, rbind)

df <- df[,colSums(is.na(df))<nrow(df)]
```

printing the confidence intervals for each pair of signficant betas.

```
for (i in 1:ncol(df)){
  print(paste("For pair : ", i))
  print(quantile(df[,i], probs = c(0.025, 0.975), na.rm = TRUE))
  print(paste("mean : ", mean(df[,i], na.rm = TRUE)))
}
```

```
## [1] "For pair :  1"
##     2.5%    97.5%
##  4.45337 50.68285
## [1] "mean :  25.5694442549705"
## [1] "For pair :  2"
##      2.5%     97.5%
##  3.921029 54.509593
## [1] "mean :  27.981349621021"
## [1] "For pair :  3"
##      2.5%     97.5%
##  7.146867 75.734755
## [1] "mean :  46.0685047089444"
## [1] "For pair :  4"
##      2.5%     97.5%
##  5.413237 83.393219
## [1] "mean :  45.7872411258584"
## [1] "For pair :  5"
##      2.5%     97.5%
##  4.482864 83.213689
## [1] "mean :  44.1959489869967"
## [1] "For pair :  6"
##      2.5%     97.5%
##  4.095385 81.598240
## [1] "mean :  43.723438541603"
```

6. Recommend to Infinity's managers what they should do to improve their product design

We got three brand maps -

1. Premium vs unreliable - In this brand map, BMW is infinity's close competitor - with ideal vector at 22 degrees which leans towards premium feature.

Thus, infinity should work on premium cars which are attractive and well built.

2. unreliable vs value for money - In this brand map, Mercury, BMW, Eagle are infinity's close competitor - with ideal vector at 58 degrees which leans towards value for money feature.

Thus infinity should focus on economical and good value cars.

3. Premium vs value for money - In this brand map, BMW is infinity's close competitor - with ideal vector at 33 degrees which leans towards premium feature.

Thus, infinity should work on premium cars which are attractive and well built.