Regression-Based Artificial Neural Networks to Predict Airbnb Prices in Berlin and Listing Segmentation Through PCA Dimension Reduction and K-Means Clustering

Julie Bazalewski

04/25/2021

```
require(ggplot2)
require(gridExtra)
require(ggthemes)
library(nnet)
library(neuralnet)
library(NeuralNetTools)
library(dplyr)
library(HH)
library(factoextra)
library(ggfortify)

#data load
Berlin <- read.csv("AirbnbBerlin.csv")

#data preparation
Berlin <- subset(Berlin,select=-c(Review.ID,review_date,Reviewer.ID,Reviewer.
Name,Comments,Listing.URL,Listing.Name,Host.ID,Host.URL,Host.Name,Neighborhoo
d.Group,Is.Exact.Location,City,Country.Code,Country,Square.Feet,Postal.Code,n
eighbourhood,Host.Response.Rate,Property.Type))

#group by listing ID. The only thing that changes between rows of the same li
sting is the reviewer details and comments which have already been removed. S
imply group by listing ID and take all of the other details from the first ro
w of each listing.
Berlin <- Berlin %>% group_by(Listing.ID) %>%
  summarise(Host.Since = first(Host.Since),Host.Response.Time=first(Host.Resp
onse.Time),Is.Superhost=first(Is.Superhost),Latitude=first(Latitude),Longitud
e=first(Longitude),Room.Type=first(Room.Type),Accomodates=first(Accomodates),
Bathrooms=first(Bathrooms),Bedrooms=first(Bedrooms),Beds=first(Beds),Price=fi
rst(Price),Guests.Included=first(Guests.Included),Min.Nights=first(Min.Nights
),Reviews=first(Reviews),First.Review=first(First.Review),Last.Review=first(L
ast.Review),Overall.Rating=first(Overall.Rating),Accuracy.Rating=first(Accura
cy.Rating),Cleanliness.Rating=first(Cleanliness.Rating),Checkin.Rating=first(
Checkin.Rating),Communication.Rating=first(Communication.Rating),Location.Rat
ing=first(Location.Rating),Value.Rating=first(Value.Rating),Instant.Bookable=
first(Instant.Bookable)) %>%arrange(desc(Listing.ID))

## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
#convert factors to numerical
Berlin <- Berlin %>% mutate(Host.Response.Time = case_when(
  (Host.Response.Time == "within an hour") ~ 3,
  (Host.Response.Time == "within a few hours") ~ 2,
  (Host.Response.Time == "within a day") ~ 1,
  (Host.Response.Time == "within a few days") ~ 0)
  )

Berlin <- Berlin %>% mutate(Room.Type = case_when(
  (Room.Type == "Entire home/apt") ~ 2,
  (Room.Type == "Private room") ~ 1,
  (Room.Type == "Shared room") ~ 0)
  )

Berlin <- Berlin %>% mutate(Instant.Bookable = case_when(
  (Instant.Bookable == "t") ~ 1,
  (Instant.Bookable == "f") ~ 0))

Berlin <- Berlin %>% mutate(Is.Superhost = case_when(
  (Is.Superhost == "t") ~ 1,
  (Is.Superhost == "f") ~ 0))

#convert date variables to epoch time
Berlin$Host.Since = as.numeric(as.POSIXct(Berlin$Host.Since, format="%m/%d/%Y
"))
Berlin$First.Review = as.numeric(as.POSIXct(Berlin$First.Review, format="%m/%
d/%Y"))
Berlin$Last.Review = as.numeric(as.POSIXct(Berlin$Last.Review, format="%m/%d/
%Y"))

#convert numeric variables to numeric
Berlin$Price <- as.numeric(Berlin$Price)

## Warning: NAs introduced by coercion

#after removing entries without a rating, the remaining NA's are a very small
percent of the data, so just remove
Berlin <- na.omit(Berlin)

Berlin <- Berlin[!Berlin$Min.Nights>365, ] #filter entries with min nights gr
eater than 365 as unrealistic
Berlin <- Berlin[!Berlin$Price<=0, ] #filter entries with price of zero as in
correct
Berlin <- Berlin[!Berlin$Beds<=0,] #filter entries with zero beds as incorrec
t
Berlin <- Berlin[which(Berlin$Bathrooms <= Berlin$Accomodates),] #several ent
ries have unrealistic bathroom counts
Berlin <- Berlin[!Berlin$Price > 2500,] #several entries have unrealistic pri
ces
```

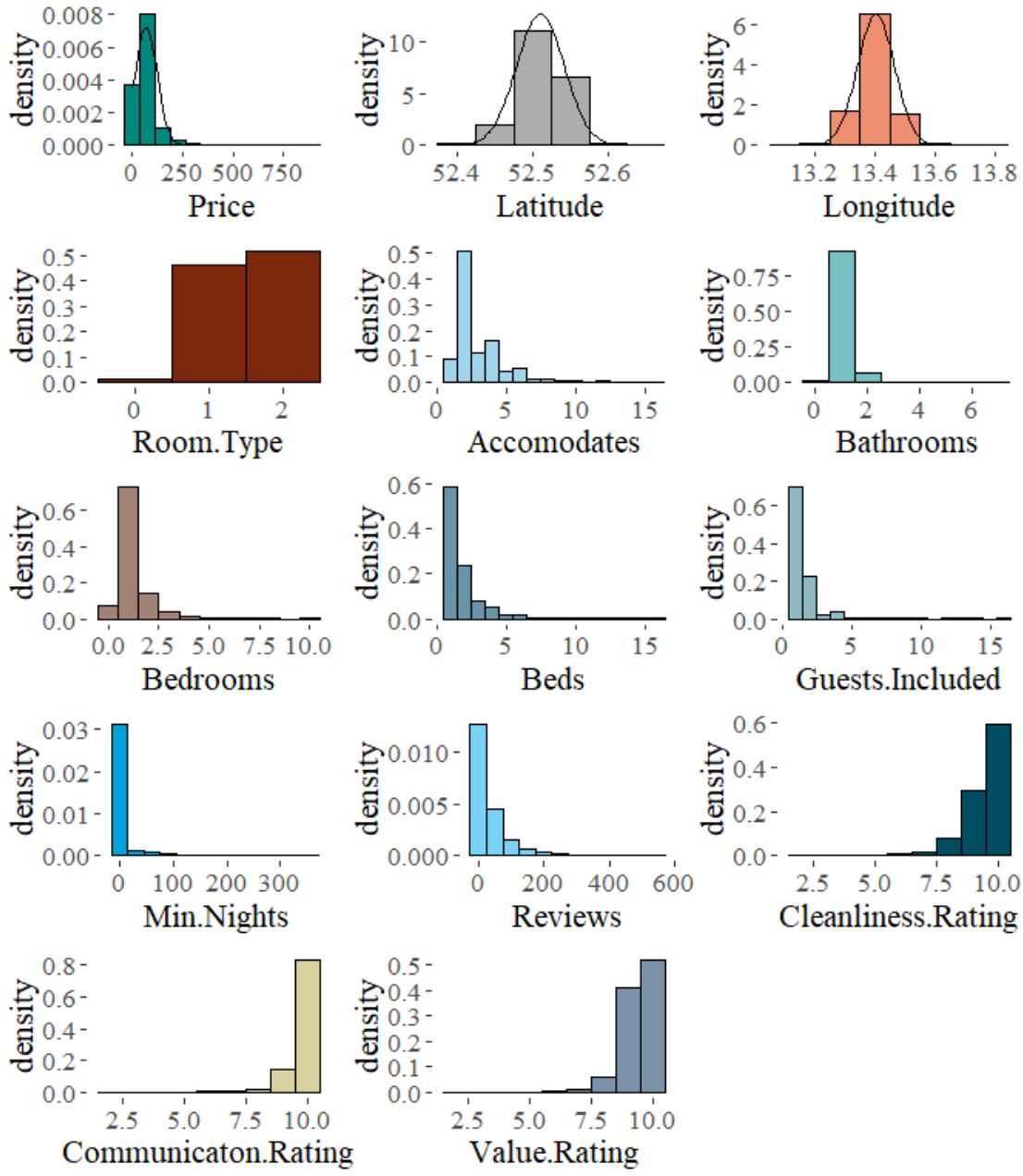
```r
summary(Berlin)
```

```
##    Listing.ID         Host.Since        Host.Response.Time  Is.Superhost
##  Min.   :    2695   Min.   :1.219e+09   Min.   :1.000      Min.   :0.0000
##  1st Qu.: 9841399   1st Qu.:1.376e+09   1st Qu.:2.000      1st Qu.:0.0000
##  Median :20030808   Median :1.431e+09   Median :3.000      Median :0.0000
##  Mean   :18569292   Mean   :1.429e+09   Mean   :2.349      Mean   :0.2854
##  3rd Qu.:27613200   3rd Qu.:1.481e+09   3rd Qu.:3.000      3rd Qu.:1.0000
##  Max.   :34465414   Max.   :1.557e+09   Max.   :3.000      Max.   :1.0000
##     Latitude       Longitude       Room.Type      Accomodates
##  Min.   :52.38   Min.   :13.12   Min.   :0.000   Min.   : 1.00
##  1st Qu.:52.49   1st Qu.:13.37   1st Qu.:1.000   1st Qu.: 2.00
##  Median :52.51   Median :13.41   Median :2.000   Median : 2.00
##  Mean   :52.51   Mean   :13.40   Mean   :1.507   Mean   : 2.94
##  3rd Qu.:52.53   3rd Qu.:13.44   3rd Qu.:2.000   3rd Qu.: 4.00
##  Max.   :52.64   Max.   :13.76   Max.   :2.000   Max.   :16.00
##    Bathrooms        Bedrooms         Beds            Price
##  Min.   :0.000   Min.   : 0.0   Min.   : 1.000   Min.   :  1.00
##  1st Qu.:1.000   1st Qu.: 1.0   1st Qu.: 1.000   1st Qu.: 35.00
##  Median :1.000   Median : 1.0   Median : 1.000   Median : 50.00
##  Mean   :1.094   Mean   : 1.2   Mean   : 1.828   Mean   : 67.84
##  3rd Qu.:1.000   3rd Qu.: 1.0   3rd Qu.: 2.000   3rd Qu.: 80.00
##  Max.   :7.000   Max.   :10.0   Max.   :16.000   Max.   :888.00
##  Guests.Included    Min.Nights         Reviews        First.Review
##  Min.   : 1.000   Min.   :  1.000   Min.   :  1.00   Min.   :1.245e+09
##  1st Qu.: 1.000   1st Qu.:  2.000   1st Qu.:  5.00   1st Qu.:1.463e+09
##  Median : 1.000   Median :  2.000   Median : 15.00   Median :1.508e+09
##  Mean   : 1.471   Mean   :  6.045   Mean   : 35.42   Mean   :1.494e+09
##  3rd Qu.: 2.000   3rd Qu.:  3.000   3rd Qu.: 42.00   3rd Qu.:1.538e+09
##  Max.   :16.000   Max.   :365.000   Max.   :545.00   Max.   :1.558e+09
##   Last.Review        Overall.Rating   Accuracy.Rating  Cleanliness.Rating
##  Min.   :1.342e+09   Min.   : 20.00   Min.   : 2.000   Min.   : 2.000
##  1st Qu.:1.551e+09   1st Qu.: 93.00   1st Qu.:10.000   1st Qu.: 9.000
##  Median :1.556e+09   Median : 96.00   Median :10.000   Median :10.000
##  Mean   :1.550e+09   Mean   : 94.87   Mean   : 9.718   Mean   : 9.442
##  3rd Qu.:1.557e+09   3rd Qu.:100.00   3rd Qu.:10.000   3rd Qu.:10.000
##  Max.   :1.558e+09   Max.   :100.00   Max.   :10.000   Max.   :10.000
##  Checkin.Rating   Communication.Rating Location.Rating   Value.Rating
##  Min.   : 2.000   Min.   : 2.000       Min.   : 2.000   Min.   : 2.000
##  1st Qu.:10.000   1st Qu.:10.000       1st Qu.: 9.000   1st Qu.: 9.000
##  Median :10.000   Median :10.000       Median :10.000   Median :10.000
##  Mean   : 9.768   Mean   : 9.782       Mean   : 9.609   Mean   : 9.417
##  3rd Qu.:10.000   3rd Qu.:10.000       3rd Qu.:10.000   3rd Qu.:10.000
##  Max.   :10.000   Max.   :10.000       Max.   :10.000   Max.   :10.000
##  Instant.Bookable
##  Min.   :0.0000
##  1st Qu.:0.0000
##  Median :0.0000
##  Mean   :0.3903
```

```
##  3rd Qu.:1.0000
##  Max.   :1.0000

#Examine linear fit to assist with variable selection and check multicollinea
rity
fit=lm(Price~Latitude+Longitude+Room.Type+Accomodates+Bathrooms+Bedrooms+Beds
+Guests.Included+Min.Nights+Reviews+Cleanliness.Rating+Communication.Rating+L
ocation.Rating+Value.Rating,data=Berlin)
summary(fit)

##
## Call:
## lm(formula = Price ~ Latitude + Longitude + Room.Type + Accomodates +
##     Bathrooms + Bedrooms + Beds + Guests.Included + Min.Nights +
##     Reviews + Cleanliness.Rating + Communication.Rating + Location.Rating
+
##     Value.Rating, data = Berlin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -292.69  -17.11   -3.58   11.33  816.87
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -1.928e+03  6.335e+02  -3.044 0.002342 **
## Latitude              4.081e+01  1.182e+01   3.452 0.000558 ***
## Longitude            -2.372e+01  6.079e+00  -3.902 9.60e-05 ***
## Room.Type             2.663e+01  7.891e-01  33.754  < 2e-16 ***
## Accomodates           9.353e+00  4.314e-01  21.683  < 2e-16 ***
## Bathrooms             3.585e+01  1.335e+00  26.856  < 2e-16 ***
## Bedrooms              1.255e+01  7.334e-01  17.117  < 2e-16 ***
## Beds                 -2.839e+00  4.822e-01  -5.887 4.06e-09 ***
## Guests.Included       6.242e+00  4.735e-01  13.182  < 2e-16 ***
## Min.Nights           -2.119e-01  2.317e-02  -9.145  < 2e-16 ***
## Reviews              -4.429e-02  7.079e-03  -6.257 4.07e-10 ***
## Cleanliness.Rating    7.715e+00  5.154e-01  14.969  < 2e-16 ***
## Communication.Rating -2.759e+00  7.404e-01  -3.726 0.000195 ***
## Location.Rating       7.997e+00  6.371e-01  12.553  < 2e-16 ***
## Value.Rating         -7.932e+00  6.478e-01 -12.244  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38.63 on 10883 degrees of freedom
## Multiple R-squared:  0.5141, Adjusted R-squared:  0.5135
## F-statistic: 822.6 on 14 and 10883 DF,  p-value: < 2.2e-16

vif(fit)

##             Latitude            Longitude            Room.Type
##             1.013553             1.016042             1.261822
##          Accomodates            Bathrooms             Bedrooms
```

```
##             4.429118             1.317356             2.008821
##                 Beds      Guests.Included          Min.Nights
##             3.599158             1.619143             1.055410
##              Reviews   Cleanliness.Rating Communication.Rating
##             1.032547             1.396054             1.355913
##      Location.Rating         Value.Rating
##             1.212625             1.663005

#hist.plot
#function for plotting histograms of predictors,
#takes arguements:
#data- dataframe containing the predictor)
#predictor- a column of the dataframe data (the variable of interest), call w
ith $ notation
#plot.color- string for bin fill color
#width- integer for bin width
#label- string for x plot label
#returns plot item

hist.plot <- function(data,predictor,plot.color="white",width="1",label="pred
ictor",cont=FALSE)
{
  if(cont==TRUE){
  p.item <- ggplot( data,  aes( x = predictor)) +
  geom_histogram( aes( y = ..density..),
                  binwidth = width,
                  colour = "black",
                  fill = plot.color) +
                  labs(x=label) +
                  stat_function( fun = dnorm,
                      args = list( mean = mean(predictor),
                                   sd = sd(predictor))) +
                  theme_tufte() + theme(text = element_text(size = 16))
  }
  else{p.item <- ggplot( data,  aes( x = predictor)) +
  geom_histogram( aes( y = ..density..),
                  binwidth = width,
                  colour = "black",
                  fill = plot.color) +
    labs(x=label) + theme_tufte() + theme(text = element_text(size = 16))
  }
  return(p.item)
}

#plot histograms of variables
p1 <- hist.plot(Berlin,Berlin$Price,"#00887d",75,"Price",cont=TRUE)
p2 <- hist.plot(Berlin,Berlin$Latitude,"#adadad",.05,"Latitude",cont=TRUE)
p3 <- hist.plot(Berlin,Berlin$Longitude,"#ee8f71",.1,"Longitude",cont=TRUE)
p4 <- hist.plot(Berlin,Berlin$Room.Type,"#7c260b",1,"Room.Type")
p5 <- hist.plot(Berlin,Berlin$Accomodates,"#a0d6ec",1,"Accomodates")
```

```r
p6 <- hist.plot(Berlin,Berlin$Bathrooms,"#76c0c1",1,"Bathrooms")
p7 <- hist.plot(Berlin,Berlin$Bedrooms,"#a18376",1,"Bedrooms")
p8 <- hist.plot(Berlin,Berlin$Beds,"#6794a7",1,"Beds")
p9 <- hist.plot(Berlin,Berlin$Guests.Included,"#8fb9c3",1,"Guests.Included")
p10 <- hist.plot(Berlin,Berlin$Min.Nights,"#01a2d9",30,"Min.Nights")
p11 <- hist.plot(Berlin,Berlin$Reviews,"#7ad2f6",50,"Reviews")
p12 <- hist.plot(Berlin,Berlin$Cleanliness.Rating,"#014d64",1,"Cleanliness.Ra
ting")
p13 <- hist.plot(Berlin,Berlin$Communication.Rating,"#D7D29E",1,"Communicaton
.Rating")
p14 <- hist.plot(Berlin,Berlin$Value.Rating,"#7B92A8",1,"Value.Rating")
grid.arrange( p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13, p14, nc
ol = 3)
```




```
fulldata.in = Berlin
suppressWarnings(set.seed(2, sample.kind = "Rounding"))
#split into 3 sets, 80% train, 10% validation, 10% test
#first do 80/20 split
train = sample(nrow(Berlin)*.8)
Berlin_train = Berlin[train,]
Berlin_rest = Berlin[-train,]
#then split 20% into validiation and test sets
valid = sample(nrow(Berlin_rest)*.5)
Berlin_valid = Berlin_rest[valid,]
Berlin_test = Berlin_rest[-valid,]
```

```r
#create another set including training and validiation sets
Berlin_trainvalid = rbind(Berlin_train,Berlin_valid)

#independently standardize numeric columns for each set
Berlin_train= Berlin_train %>%
    mutate_if(is.numeric, scale)
Berlin_valid = Berlin_valid %>%
    mutate_if(is.numeric, scale)
Berlin_test = Berlin_test %>%
    mutate_if(is.numeric, scale)
Berlin_trainvalid = Berlin_trainvalid %>%
    mutate_if(is.numeric, scale)

#total model count
decayRate <- seq(.01, .024, by = .001)
sizes = 5:10
nmodels = length(decayRate) * length(sizes)
formula = Price~Latitude+Longitude+Room.Type+Accomodates+Bathrooms+Bedrooms+B
eds+Guests.Included+Min.Nights+Reviews+Cleanliness.Rating+Communication.Ratin
g+Location.Rating+Value.Rating

suppressWarnings(set.seed(2, sample.kind = "Rounding"))

allmodelCV.in = rep(NA,nmodels) #place-holder for results
r2.in = rep(NA,nmodels) #place-holder for results
alllmfitCV.in = NA

# set up storage for predicted values across models
n.in = nrow(Berlin_valid)
allpredictedCV.in = matrix(rep(NA,n.in*nmodels),ncol=nmodels)

#choose decay and hidden layer nodes
m = 1
for(j in 1:length(decayRate)){
  for(s in 1:length(sizes)){
    lmfitCV.in = nnet(formula, data=Berlin_train, size = sizes[s], trace = F,
linout = T, maxit = 1000, MaxNWts=5000, decay = decayRate[j])
    allpredictedCV.in[,m] = predict(lmfitCV.in,newdata=Berlin_valid)
    allmodelCV.in[m] = mean((allpredictedCV.in[,m]-Berlin_valid$Price)^2)
    r2.in[m] = 1-sum((allpredictedCV.in[,m]-Berlin_valid$Price)^2)/sum((Berli
n_valid$Price-mean(Berlin_valid$Price))^2)
    print(m)
    if(lmfitCV.in$convergence == 1){
      print("model not converged") #check for convergence
    }
    #print(allmodelCV.in[m])
    #print(r2.in[m])
    m = m+1  #increment for next model
  }#end iteration over s
} # end iteration over j
```

```
## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5
## [1] 6
## [1] 7
## [1] 8
## [1] 9
## [1] 10
## [1] 11
## [1] 12
## [1] 13
## [1] 14
## [1] 15
## [1] 16
## [1] 17
## [1] 18
## [1] "model not converged"
## [1] 19
## [1] 20
## [1] 21
## [1] 22
## [1] 23
## [1] 24
## [1] 25
## [1] 26
## [1] 27
## [1] 28
## [1] 29
## [1] 30
## [1] "model not converged"
## [1] 31
## [1] 32
## [1] 33
## [1] 34
## [1] 35
## [1] 36
## [1] 37
## [1] 38
## [1] 39
## [1] 40
## [1] 41
## [1] 42
## [1] 43
## [1] 44
## [1] 45
## [1] 46
## [1] 47
## [1] 48
```

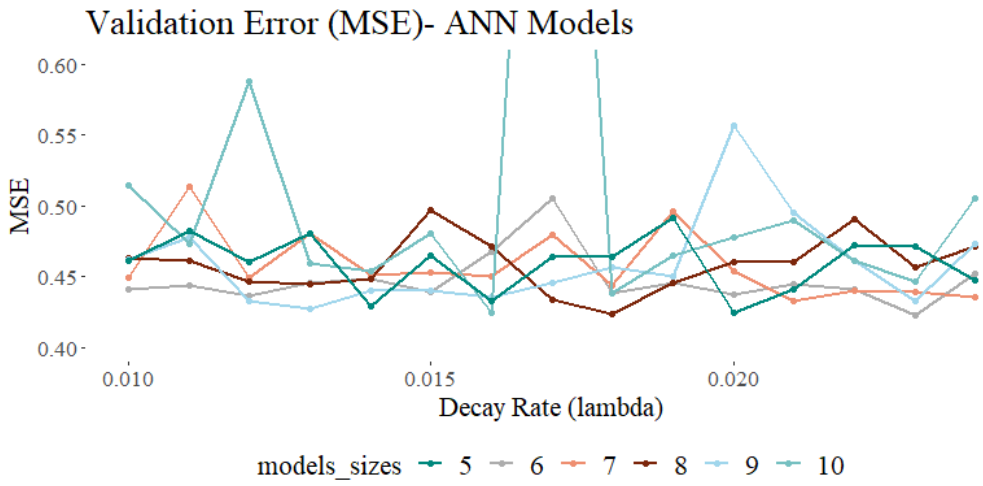
```
## [1] 49
## [1] 50
## [1] 51
## [1] 52
## [1] 53
## [1] 54
## [1] "model not converged"
## [1] 55
## [1] 56
## [1] 57
## [1] 58
## [1] 59
## [1] 60
## [1] 61
## [1] 62
## [1] 63
## [1] 64
## [1] 65
## [1] 66
## [1] 67
## [1] 68
## [1] 69
## [1] 70
## [1] 71
## [1] 72
## [1] 73
## [1] 74
## [1] 75
## [1] 76
## [1] 77
## [1] 78
## [1] 79
## [1] 80
## [1] 81
## [1] 82
## [1] 83
## [1] 84
## [1] 85
## [1] 86
## [1] 87
## [1] 88
## [1] 89
## [1] 90
```

```
bestmodel.in = (1:nmodels)[order(allmodelCV.in)[1]]  # actual selection
# state which is best model and minimum MSE value
bestmodel.in; min(allmodelCV.in, na.rm = TRUE)
```

```
## [1] 79
```

```
## [1] 0.4230812

#determine the decay rate and hidden nodes based on best model
models_decayRate = rep(NA,nmodels)
models_sizes = rep(NA,nmodels)
k=1
for (i in 1:length(decayRate))  {
  for (j in 1:length(sizes)){
    models_decayRate[k] = decayRate[i]; models_sizes[k] = sizes[j]
    k=k+1
    }
  }
best_dR = models_decayRate[bestmodel.in]; best_size = models_sizes[bestmodel.
in]
print(best_dR);print(best_size)

## [1] 0.023

## [1] 5

print(r2.in[bestmodel.in])

## [1] 0.5765303

# visualize error values across all models
model_index <- c(1:nmodels)
mse_df <- data.frame(allmodelCV.in,models_decayRate,models_sizes,model_index)
mse_df$models_sizes <- as.factor(mse_df$models_sizes)

myPalette <- c("#00887d","#adadad","#ee8f71","#7c260b","#a0d6ec","#76c0c1","#
a18376","#6794a7","#8fb9c3","#01a2d9","#7ad2f6","#014d64","#3E647D","#7B92A8"
,"#82C0E9","#2D6D66","#BFA19C","#008BBC","#97B6B0","#D7D29E")

#plot MSE vs decay rate by hidden layer nodes
mse_plot <- ggplot(mse_df[models_sizes==5,], aes(models_decayRate, allmodelCV
.in)) +
  geom_point(aes(color = models_sizes)) + geom_line(aes(color = models_sizes)
,size=1) +
  geom_point(data=mse_df[models_sizes==6,],aes(color = models_sizes))+ geom_l
ine(data=mse_df[models_sizes==6,],aes(color = models_sizes),size=1) +
  geom_point(data=mse_df[models_sizes==7,],aes(color = models_sizes))+ geom_l
ine(data=mse_df[models_sizes==7,],aes(color = models_sizes),size=1) +
  geom_point(data=mse_df[models_sizes==8,],aes(color = models_sizes))+ geom_l
ine(data=mse_df[models_sizes==8,],aes(color = models_sizes),size=1) +
  geom_point(data=mse_df[models_sizes==9,],aes(color = models_sizes))+ geom_l
ine(data=mse_df[models_sizes==9,],aes(color = models_sizes),size=1) +
  geom_point(data=mse_df[models_sizes==10,],aes(color = models_sizes))+
  geom_line(data=mse_df[models_sizes==10,],aes(color = models_sizes),size=1)
+
  theme_tufte() + theme(aspect.ratio=1/3, legend.position = "bottom",
        axis.text=element_text(size=13),axis.title=element_text(size=16),plot
```

```
.title=element_text(size=20),
        legend.text=element_text(size=16),legend.title=element_text(size=16))
+ coord_cartesian(ylim=c(0.4, .6)) +
  scale_color_manual(values=myPalette, labels=c(5,6,7,8,9,10)) + guides(colou
r = guide_legend(nrow = 1)) +
  ggtitle("Validation Error (MSE)- ANN Models") +ylab("MSE") + xlab("Decay Ra
te (lambda)")
mse_plot
```

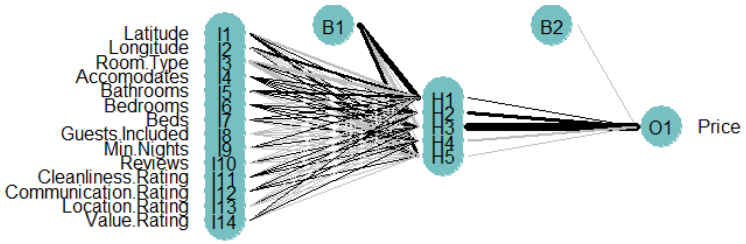



```
suppressWarnings(set.seed(2, sample.kind = "Rounding"))

#fit best model on training and validation data
bestfit <- nnet(Price~Latitude+Longitude+Room.Type+Accomodates+Bathrooms+Bedr
ooms+Beds+Guests.Included+Min.Nights+Reviews+Cleanliness.Rating+Communication
.Rating+Location.Rating+Value.Rating, data=Berlin_trainvalid, size = best_siz
e, trace = F, linout = T, maxit = 1000, MaxNWts=5000, decay = best_dR)

#predict new values with test data to evaluate model
allpredictedCV=predict(bestfit,newdata=Berlin_test)
allmodelCV = mean((allpredictedCV-Berlin_test$Price)^2)
bestfit$convergence

## [1] 0

print(allmodelCV)

## [1] 0.5208898

R2 = 1-sum((allpredictedCV-Berlin_test$Price)^2)/sum((Berlin_test$Price-mean(
Berlin_test$Price))^2); print(R2)

## [1] 0.4786319

##############################
## End of modeling process ##
##############################
```

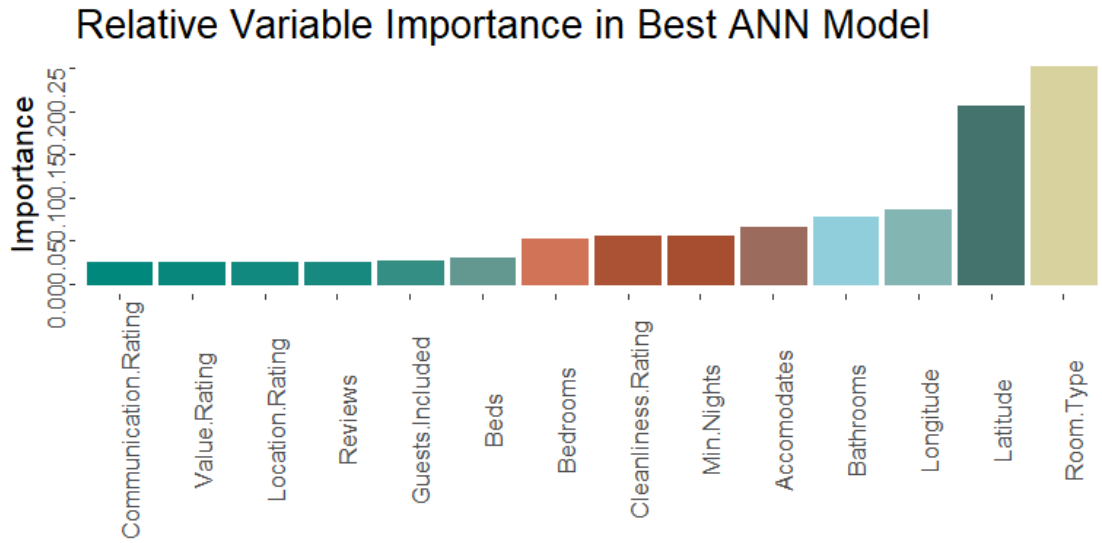
```r
summary(bestfit)
```

```
## a 14-5-1 network with 81 weights
## options were - linear output units  decay=0.023
##   b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i
9->h1
##   15.03   -3.43    3.86   -8.31    0.18    6.74   -0.72    0.32   -0.23
0.30
## i10->h1 i11->h1 i12->h1 i13->h1 i14->h1
##   -0.20   -0.27   -0.04   -0.29    0.15
##   b->h2  i1->h2  i2->h2  i3->h2  i4->h2  i5->h2  i6->h2  i7->h2  i8->h2  i
9->h2
##   -1.78    0.75   -0.19    0.07    0.34    0.07    0.26    0.01    0.00
-0.31
## i10->h2 i11->h2 i12->h2 i13->h2 i14->h2
##   -0.11    0.12    0.09    0.00   -0.06
##   b->h3  i1->h3  i2->h3  i3->h3  i4->h3  i5->h3  i6->h3  i7->h3  i8->h3  i
9->h3
##   -9.61   -0.04   -0.05    3.19    0.47    0.56    0.16   -0.94   -0.42
0.06
## i10->h3 i11->h3 i12->h3 i13->h3 i14->h3
##    0.07    1.41   -0.16    0.14   -0.47
##   b->h4  i1->h4  i2->h4  i3->h4  i4->h4  i5->h4  i6->h4  i7->h4  i8->h4  i
9->h4
##   -2.37    1.06   -0.22   -0.13    0.28   -0.03    0.22    0.04   -0.12
-0.29
## i10->h4 i11->h4 i12->h4 i13->h4 i14->h4
##   -0.07    0.08    0.13   -0.04    0.00
##   b->h5  i1->h5  i2->h5  i3->h5  i4->h5  i5->h5  i6->h5  i7->h5  i8->h5  i
9->h5
##    8.31    4.54    2.72  -11.49    0.27   -0.10    0.39    0.00   -0.58
-0.50
## i10->h5 i11->h5 i12->h5 i13->h5 i14->h5
##   -0.86   -0.34    0.25   -1.83   -0.76
##   b->o  h1->o  h2->o  h3->o  h4->o  h5->o
##  -0.98   0.60   8.27  20.23  -6.69  -0.41
```

```r
plotnet(bestfit, cex_val=1, circle_col="#76c0c1", bord_col="#76c0c1",pad_x=.6
)
```

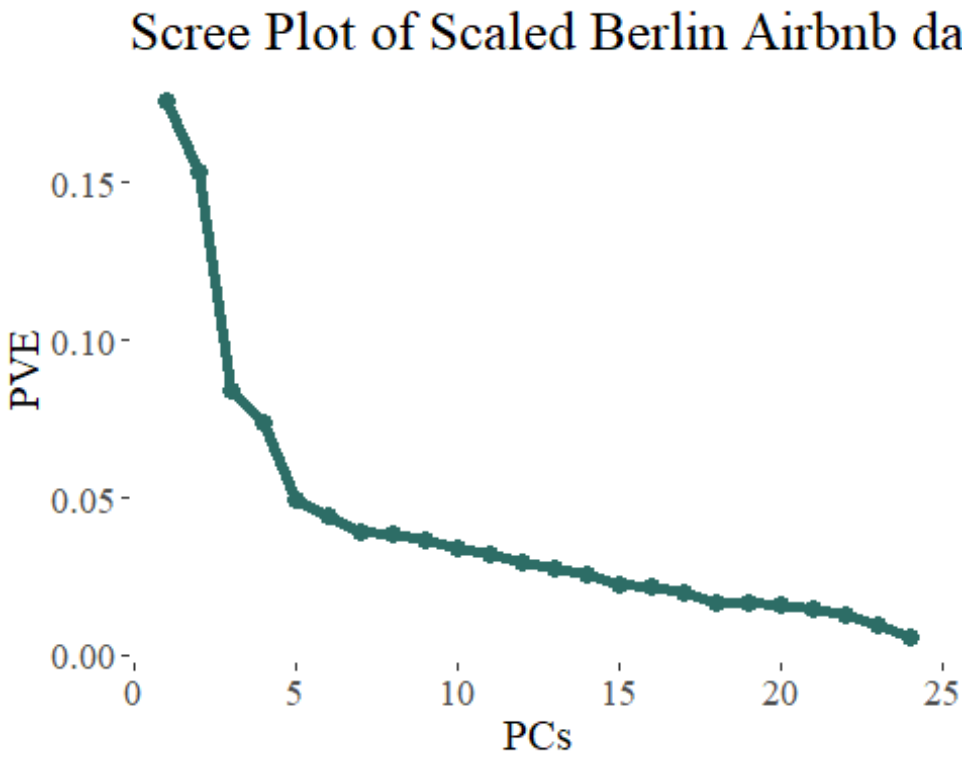



```
garson(bestfit) +
  scale_fill_gradientn(colours = myPalette) +
  scale_colour_gradientn(colours = myPalette) +
  theme(panel.border = element_blank(),panel.grid.major = element_blank(),pan
el.grid.minor = element_blank(),
        axis.text=element_text(angle =   90,size=12),axis.title=element_text(
size=16),plot.title=element_text(size=20)) +
  ggtitle("Relative Variable Importance in Best ANN Model")
```

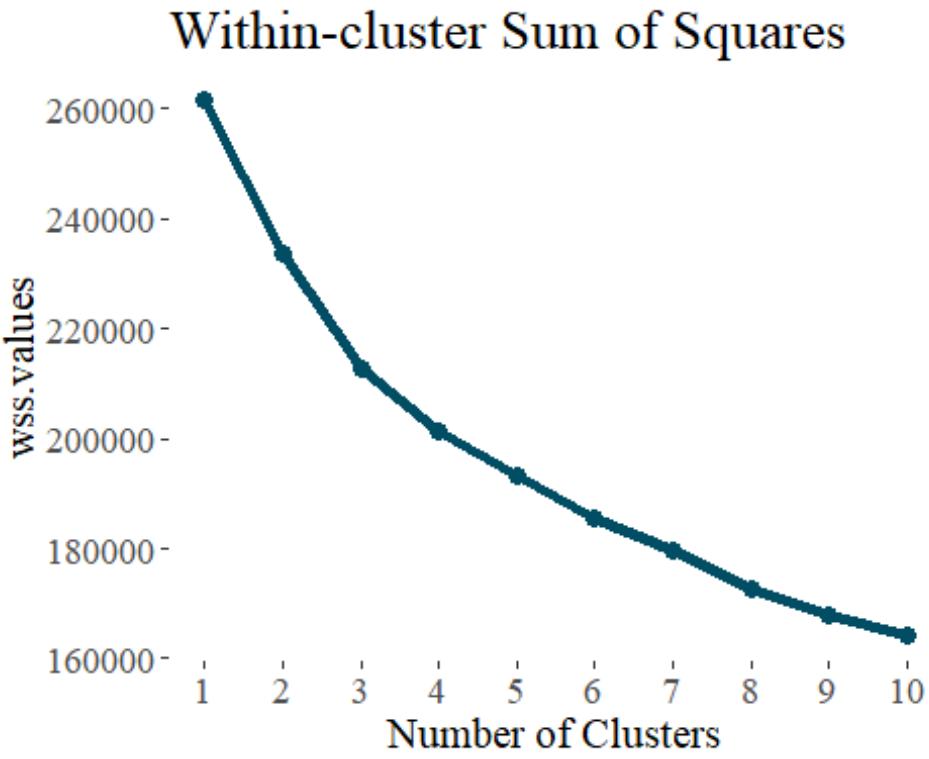



```
#```{r fig.width=20, fig.height=20}
n=10898
p=24
pc.info=prcomp(Berlin[-c(1)],center=T,scale=T)

#pc.info$rotation  #loadings
#pc.info$sdev

#calculate percentage of variation explained (PVE)
```

```
vjs = pc.info$sdev^2
PVE = vjs/sum(vjs)

PCs=1:24
var.df <- data.frame(PCs,PVE)
  ggplot(var.df, aes(x=PCs,y=PVE))+geom_point(size=3,color="#2d6d66")+geom_li
ne(size=2,color="#2d6d66")+
    theme_tufte() +
    theme(axis.text=element_text(size=14),axis.title=element_text(size=16),pl
ot.title=element_text(size=20))+
    ggtitle("Scree Plot of Scaled Berlin Airbnb data")
```

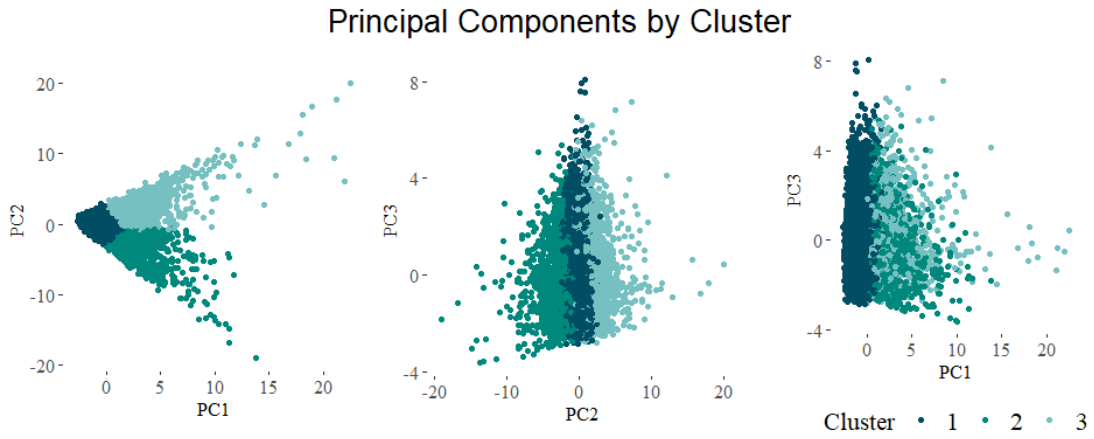



```
#determine number of clusters with elbow method

k.values <- 1:10
wss.values=rep(NA,length(k.values))
for(i in k.values){
  wss.values[i] = kmeans(pc.info$x, i)$tot.withinss
}

k.df <- data.frame(k.values, wss.values)
  ggplot(k.df, aes(x=k.values,y=wss.values))+geom_point(size=3,color="#014d64
")+geom_line(size=2,color="#014d64")+
    theme_tufte() + theme(axis.text=element_text(size=14),axis.title=element_
text(size=16),plot.title=element_text(size=20))+
    ggtitle("Within-cluster Sum of Squares") + scale_x_continuous(name="Numbe
r of Clusters", limits=c(1,10),breaks=1:10)
```

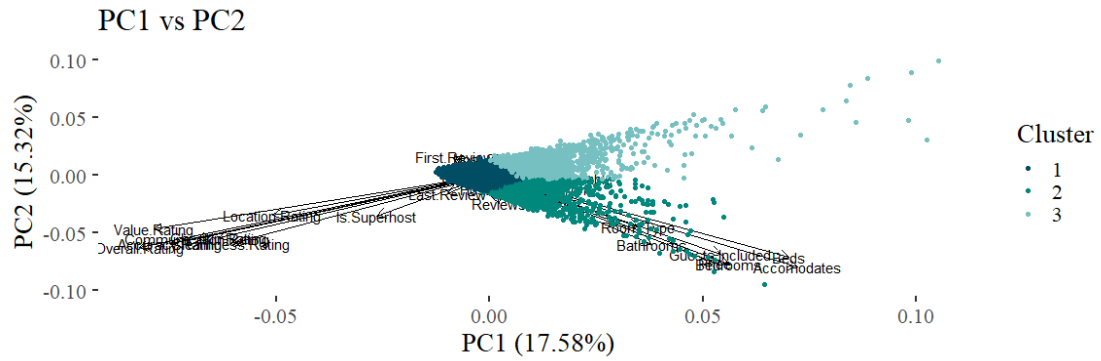



```
suppressWarnings(set.seed(4, sample.kind = "Rounding"))
myPaletteRand <- sample(myPalette)

#place first three PCs in data frame
Berlin.pcaScores <- data.frame(pc.info$x[,1:3])

#perform k-means with k=3 on reduced data
suppressWarnings(set.seed(2, sample.kind = "Rounding"))
k <- kmeans(pc.info$x,3)
Cluster <- as.factor(k$clust)

#plot first three PCs against each other
pca1 <- ggplot(Berlin.pcaScores, aes(y = PC2, x = PC1)) + geom_point(aes(col=
Cluster)) + scale_color_manual(values=myPaletteRand) +theme_tufte() + theme(a
spect.ratio=1, legend.position = "none", axis.text=element_text(size=12),axis
.title=element_text(size=12))
pca2 <- ggplot(Berlin.pcaScores, aes(y = PC3, x = PC2)) + geom_point(aes(col=
Cluster)) + scale_color_manual(values=myPaletteRand) +theme_tufte() + theme(a
spect.ratio=1, legend.position = "none", axis.text=element_text(size=12),axis
.title=element_text(size=12))
pca3 <- ggplot(Berlin.pcaScores, aes(y = PC3, x = PC1)) + geom_point(aes(col=
Cluster)) + scale_color_manual(values=myPaletteRand) +theme_tufte() + theme(a
spect.ratio=1.15, legend.position = "bottom", axis.text=element_text(size=12)
,axis.title=element_text(size=12),legend.text=element_text(size=16),legend.ti
tle=element_text(size=16))
```

```
grid.arrange(pca1,pca2,pca3, ncol=3, top = textGrob("Principal Components by
Cluster",gp=gpar(fontsize=20)))
```




```
#look at more detailed plot of just first two PCs
suppressWarnings(set.seed(4, sample.kind = "Rounding"))
myPaletteRand <- sample(myPalette)

autoplot(pc.info, data = Berlin[-c(1)], colour = as.factor(k$clust), loadings
.label.colour = 'black',
         loadings = TRUE, loadings.colour = 'black', loadings.label = TRUE, l
oadings.label.size = 4) +
  geom_point(aes(col=Cluster)) +scale_color_manual(values=myPaletteRand) + th
eme_tufte(base_size=20) + ggtitle("PC1 vs PC2")

## Warning: `select_()` was deprecated in dplyr 0.7.0.
## Please use `select()` instead.

## Warning in if (value %in% columns) {: the condition has length > 1 and onl
y the
## first element will be used
```




```
#create dataframe with column for cluster assignment
Berlin.clusters <- Berlin
Berlin.clusters<-data.frame(Berlin.clusters,k$cluster)
```

```r
#convert variables to factor
Berlin.clusters$Room.Type <- factor(Berlin.clusters$Room.Type, labels = c("Sh
ared Room", "Private Room", "Entire House"))
Berlin.clusters$Host.Response.Time <- as.factor(Berlin.clusters$Host.Response
.Time)
Berlin.clusters$Instant.Bookable <- as.factor(Berlin.clusters$Instant.Bookabl
e)
Berlin.clusters$Is.Superhost <- as.factor(Berlin.clusters$Is.Superhost)
Berlin.clusters$k.cluster <- as.factor(Berlin.clusters$k.cluster)

#examine each cluster
print("Cluster 1 Summary")
```

```
## [1] "Cluster 1 Summary"
```

```r
summary(Berlin.clusters[which(Berlin.clusters$k.cluster==1),])
```

```
##    Listing.ID         Host.Since        Host.Response.Time Is.Superhost
##  Min.   :    2695   Min.   :1.219e+09   1:1719             0:5229
##  1st Qu.:10737336   1st Qu.:1.377e+09   2:1825             1:2617
##  Median :20600017   Median :1.429e+09   3:4302
##  Mean   :18968201   Mean   :1.429e+09
##  3rd Qu.:27869319   3rd Qu.:1.479e+09
##  Max.   :34465414   Max.   :1.557e+09
##     Latitude       Longitude            Room.Type     Accomodates
##  Min.   :52.38   Min.   :13.12   Shared Room :  74   Min.   :1.000
##  1st Qu.:52.49   1st Qu.:13.37   Private Room:4273   1st Qu.:2.000
##  Median :52.51   Median :13.42   Entire House:3499   Median :2.000
##  Mean   :52.51   Mean   :13.40                       Mean   :2.382
##  3rd Qu.:52.53   3rd Qu.:13.44                       3rd Qu.:3.000
##  Max.   :52.64   Max.   :13.76                       Max.   :7.000
##    Bathrooms       Bedrooms          Beds           Price
##  Min.   :0.00   Min.   :0.000   Min.   :1.000   Min.   :  1.00
##  1st Qu.:1.00   1st Qu.:1.000   1st Qu.:1.000   1st Qu.: 35.00
##  Median :1.00   Median :1.000   Median :1.000   Median : 49.00
##  Mean   :1.04   Mean   :1.008   Mean   :1.389   Mean   : 55.17
##  3rd Qu.:1.00   3rd Qu.:1.000   3rd Qu.:2.000   3rd Qu.: 70.00
##  Max.   :2.50   Max.   :4.000   Max.   :6.000   Max.   :345.00
##  Guests.Included   Min.Nights         Reviews        First.Review
##  Min.   :1.000   Min.   :  1.000   Min.   :  1.00   Min.   :1.251e+09
##  1st Qu.:1.000   1st Qu.:  2.000   1st Qu.:  5.00   1st Qu.:1.467e+09
##  Median :1.000   Median :  2.000   Median : 14.50   Median :1.511e+09
##  Mean   :1.244   Mean   :  5.936   Mean   : 34.59   Mean   :1.497e+09
##  3rd Qu.:1.000   3rd Qu.:  3.000   3rd Qu.: 40.00   3rd Qu.:1.539e+09
##  Max.   :5.000   Max.   :365.000   Max.   :545.00   Max.   :1.558e+09
##   Last.Review        Overall.Rating   Accuracy.Rating  Cleanliness.Rating
##  Min.   :1.376e+09   Min.   : 80.00   Min.   : 6.000   Min.   : 6.00
##  1st Qu.:1.552e+09   1st Qu.: 95.00   1st Qu.:10.000   1st Qu.: 9.00
##  Median :1.556e+09   Median : 98.00   Median :10.000   Median :10.00
##  Mean   :1.551e+09   Mean   : 96.81   Mean   : 9.901   Mean   : 9.65
```

```
##  3rd Qu.:1.557e+09   3rd Qu.:100.00   3rd Qu.:10.000   3rd Qu.:10.00
##  Max.   :1.558e+09   Max.   :100.00   Max.   :10.000   Max.   :10.00
##  Checkin.Rating   Communication.Rating Location.Rating   Value.Rating
##  Min.   : 8.000   Min.   : 6.000       Min.   : 6.000   Min.   : 6.000
##  1st Qu.:10.000   1st Qu.:10.000       1st Qu.: 9.000   1st Qu.: 9.000
##  Median :10.000   Median :10.000       Median :10.000   Median :10.000
##  Mean   : 9.913   Mean   : 9.935       Mean   : 9.724   Mean   : 9.624
##  3rd Qu.:10.000   3rd Qu.:10.000       3rd Qu.:10.000   3rd Qu.:10.000
##  Max.   :10.000   Max.   :10.000       Max.   :10.000   Max.   :10.000
##  Instant.Bookable k.cluster
##  0:5003           1:7846
##  1:2843           2:   0
##                   3:   0
##
##
##
```

```
print("Cluster 2 Summary")
```

```
## [1] "Cluster 2 Summary"
```

```
summary(Berlin.clusters[which(Berlin.clusters$k.cluster==2),])
```

```
##    Listing.ID         Host.Since        Host.Response.Time Is.Superhost
##  Min.   :    9991   Min.   :1.219e+09   1:208              0:881
##  1st Qu.: 7775175   1st Qu.:1.358e+09   2:291              1:434
##  Median :17382908   Median :1.428e+09   3:816
##  Mean   :16951561   Mean   :1.423e+09
##  3rd Qu.:26035160   3rd Qu.:1.485e+09
##  Max.   :34246775   Max.   :1.556e+09
##     Latitude       Longitude            Room.Type     Accomodates
##  Min.   :52.38   Min.   :13.13   Shared Room :   6   Min.   : 1.00
##  1st Qu.:52.49   1st Qu.:13.37   Private Room:  56   1st Qu.: 5.00
##  Median :52.51   Median :13.41   Entire House:1253   Median : 6.00
##  Mean   :52.51   Mean   :13.40                       Mean   : 6.26
##  3rd Qu.:52.53   3rd Qu.:13.43                       3rd Qu.: 7.00
##  Max.   :52.64   Max.   :13.72                       Max.   :16.00
##    Bathrooms        Bedrooms           Beds            Price
##  Min.   :1.000   Min.   : 0.000   Min.   : 1.000   Min.   : 15.0
##  1st Qu.:1.000   1st Qu.: 2.000   1st Qu.: 3.000   1st Qu.: 98.5
##  Median :1.000   Median : 2.000   Median : 4.000   Median :135.0
##  Mean   :1.479   Mean   : 2.494   Mean   : 4.351   Mean   :157.2
##  3rd Qu.:2.000   3rd Qu.: 3.000   3rd Qu.: 5.000   3rd Qu.:197.5
##  Max.   :7.000   Max.   :10.000   Max.   :16.000   Max.   :888.0
##  Guests.Included    Min.Nights         Reviews        First.Review
##  Min.   : 1.000   Min.   :  1.000   Min.   :  1.00   Min.   :1.282e+09
##  1st Qu.: 1.000   1st Qu.:  2.000   1st Qu.:  7.00   1st Qu.:1.449e+09
##  Median : 2.000   Median :  3.000   Median : 23.00   Median :1.497e+09
##  Mean   : 2.975   Mean   :  4.972   Mean   : 47.88   Mean   :1.484e+09
##  3rd Qu.: 4.000   3rd Qu.:  3.000   3rd Qu.: 67.00   3rd Qu.:1.534e+09
##  Max.   :16.000   Max.   :180.000   Max.   :445.00   Max.   :1.558e+09
```

```
##   Last.Review        Overall.Rating   Accuracy.Rating Cleanliness.Rating
##  Min.   :1.418e+09   Min.   : 72.00   Min.   : 8.00   Min.   : 6.000
##  1st Qu.:1.554e+09   1st Qu.: 93.00   1st Qu.:10.00   1st Qu.: 9.000
##  Median :1.556e+09   Median : 96.00   Median :10.00   Median :10.000
##  Mean   :1.552e+09   Mean   : 95.28   Mean   : 9.76   Mean   : 9.581
##  3rd Qu.:1.557e+09   3rd Qu.: 99.00   3rd Qu.:10.00   3rd Qu.:10.000
##  Max.   :1.558e+09   Max.   :100.00   Max.   :10.00   Max.   :10.000
##  Checkin.Rating   Communication.Rating Location.Rating  Value.Rating
##  Min.   : 6.000   Min.   : 8.00        Min.   : 6.0    Min.   : 6.000
##  1st Qu.:10.000   1st Qu.:10.00        1st Qu.: 9.0    1st Qu.: 9.000
##  Median :10.000   Median :10.00        Median :10.0    Median : 9.000
##  Mean   : 9.841   Mean   : 9.84        Mean   : 9.6    Mean   : 9.321
##  3rd Qu.:10.000   3rd Qu.:10.00        3rd Qu.:10.0    3rd Qu.:10.000
##  Max.   :10.000   Max.   :10.00        Max.   :10.0    Max.   :10.000
##  Instant.Bookable k.cluster
##  0:720            1:   0
##  1:595            2:1315
##                   3:   0
##
##
##
```

```
print("Cluster 3 Summary")
```

```
## [1] "Cluster 3 Summary"
```

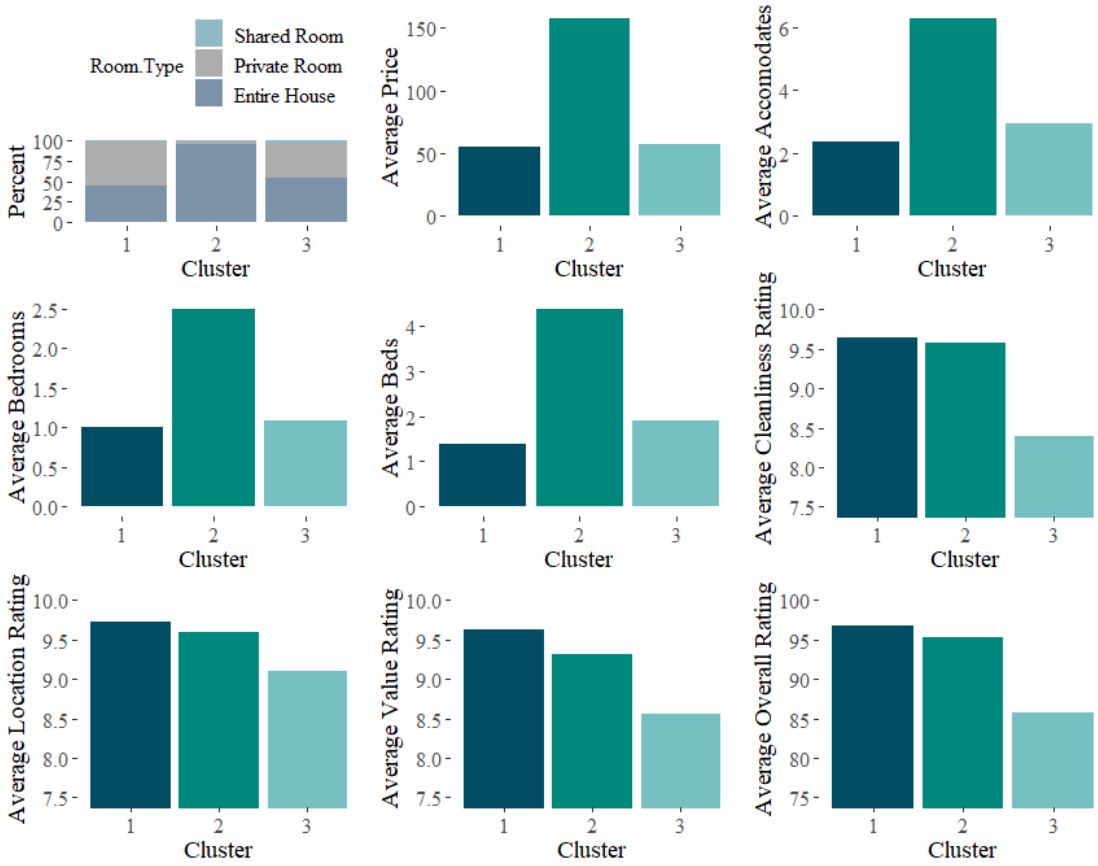
```
summary(Berlin.clusters[which(Berlin.clusters$k.cluster==3),])
```

```
##    Listing.ID         Host.Since        Host.Response.Time Is.Superhost
##  Min.   :    3176   Min.   :1.224e+09   1:361              0:1678
##  1st Qu.: 8266446   1st Qu.:1.377e+09   2:401              1:  59
##  Median :19197162   Median :1.439e+09   3:975
##  Mean   :17992135   Mean   :1.432e+09
##  3rd Qu.:27454977   3rd Qu.:1.485e+09
##  Max.   :34305674   Max.   :1.556e+09
##     Latitude       Longitude            Room.Type    Accomodates
##  Min.   :52.38   Min.   :13.12   Shared Room : 70   Min.   : 1.000
##  1st Qu.:52.49   1st Qu.:13.36   Private Room:741   1st Qu.: 2.000
##  Median :52.51   Median :13.41   Entire House:926   Median : 2.000
##  Mean   :52.51   Mean   :13.40                      Mean   : 2.944
##  3rd Qu.:52.53   3rd Qu.:13.44                      3rd Qu.: 4.000
##  Max.   :52.63   Max.   :13.71                      Max.   :11.000
##    Bathrooms        Bedrooms          Beds          Price
##  Min.   :0.000   Min.   :0.000   Min.   : 1.0   Min.   :  9.00
##  1st Qu.:1.000   1st Qu.:1.000   1st Qu.: 1.0   1st Qu.: 35.00
##  Median :1.000   Median :1.000   Median : 2.0   Median : 50.00
##  Mean   :1.049   Mean   :1.088   Mean   : 1.9   Mean   : 57.39
##  3rd Qu.:1.000   3rd Qu.:1.000   3rd Qu.: 2.0   3rd Qu.: 70.00
##  Max.   :4.000   Max.   :4.000   Max.   :11.0   Max.   :888.00
##  Guests.Included   Min.Nights         Reviews       First.Review
##  Min.   :1.000   Min.   :  1.000   Min.   :  1.0   Min.   :1.245e+09
```

```
##  1st Qu.:1.000   1st Qu.:  1.000   1st Qu.:  4.0   1st Qu.:1.456e+09
##  Median :1.000   Median :  2.000   Median : 12.0   Median :1.505e+09
##  Mean   :1.359   Mean   :  7.354   Mean   : 29.7   Mean   :1.491e+09
##  3rd Qu.:2.000   3rd Qu.:  4.000   3rd Qu.: 33.0   3rd Qu.:1.538e+09
##  Max.   :6.000   Max.   :365.000   Max.   :336.0   Max.   :1.558e+09
##   Last.Review        Overall.Rating   Accuracy.Rating  Cleanliness.Rating
##  Min.   :1.342e+09   Min.   : 20.00   Min.   : 2.000   Min.   : 2.000
##  1st Qu.:1.546e+09   1st Qu.: 83.00   1st Qu.: 9.000   1st Qu.: 8.000
##  Median :1.555e+09   Median : 88.00   Median : 9.000   Median : 9.000
##  Mean   :1.546e+09   Mean   : 85.79   Mean   : 8.858   Mean   : 8.398
##  3rd Qu.:1.556e+09   3rd Qu.: 91.00   3rd Qu.: 9.000   3rd Qu.: 9.000
##  Max.   :1.558e+09   Max.   :100.00   Max.   :10.000   Max.   :10.000
##  Checkin.Rating   Communication.Rating Location.Rating   Value.Rating
##  Min.   : 2.000   Min.   : 2.000       Min.   : 2.000   Min.   : 2.000
##  1st Qu.: 9.000   1st Qu.: 9.000       1st Qu.: 9.000   1st Qu.: 8.000
##  Median : 9.000   Median : 9.000       Median : 9.000   Median : 9.000
##  Mean   : 9.062   Mean   : 9.047       Mean   : 9.097   Mean   : 8.554
##  3rd Qu.:10.000   3rd Qu.:10.000       3rd Qu.:10.000   3rd Qu.: 9.000
##  Max.   :10.000   Max.   :10.000       Max.   :10.000   Max.   :10.000
##  Instant.Bookable k.cluster
##  0:922            1:   0
##  1:815            2:   0
##                   3:1737
##
##
##
```

```r
#make_bar_plot
#function for plotting bar plots by cluster
#takes arguements:
#var1, variable of interest
#bar_df- dataframe containing the data)
#y_lab- label for y-axis
#returns plot item
make_bar_plot <- function(var1, bar_df, y_lab){
  p.item<-ggplot(data=bar_df, aes(x=k.cluster, y=var1, fill=k.cluster)) +
  geom_bar(stat="identity")+theme_tufte()+ scale_fill_manual(values=myPalette
Rand)+ theme(plot.title=element_text(size=20),legend.title=element_text(size=
16),legend.position="none",legend.text=element_text(size=14),axis.text=elemen
t_text(size=12),axis.title=element_text(size=14))+labs(y=y_lab,x="Cluster",fi
ll="Cluster")
  return(p.item)
}

#calculate percentages for comparison in bar plots rather than count
bar_df <- Berlin.clusters %>%
  count(k.cluster,Room.Type) %>%
  group_by(k.cluster) %>%
  mutate(Percent = (n/sum(n))*100) %>%
  ungroup()
```

```r
suppressWarnings(set.seed(88, sample.kind = "Rounding"))
cbar <- ggplot( data = bar_df, aes( x = k.cluster, y=Percent, fill = Room.Type) ) +
      geom_bar(stat="identity") + scale_fill_manual(values=sample(myPalette), guide=guide_legend(nrow=3)) +
      xlab("Cluster") +theme_tufte()  + theme(plot.title=element_text(size=20),legend.title=element_text(size=12),legend.text=element_text(size=12),axis.text=element_text(size=12),
                                                                  axis.title=element_text(size=14), legend.position = "top")

#regular bar graphs
bar_df2 <- Berlin.clusters %>%
  group_by(k.cluster) %>%
  summarize(avgPrice = mean(Price),avgAccom = mean(Accomodates),avgBeds = mean(Beds),avgBedrooms=mean(Bedrooms),avgClean=mean(Cleanliness.Rating),avgLoc=mean(Location.Rating),avgVal=mean(Value.Rating),avgCom=mean(Communication.Rating),avgRating=mean(Overall.Rating)) %>%
  ungroup()

## `summarise()` ungrouping output (override with `.groups` argument)

attach(bar_df2)
bar1 <- make_bar_plot(avgPrice,bar_df2,"Average Price")
bar2 <- make_bar_plot(avgAccom,bar_df2,"Average Accomodates")
bar3 <- make_bar_plot(avgBedrooms,bar_df2,"Average Bedrooms")
bar4 <- make_bar_plot(avgBeds,bar_df2,"Average Beds")
bar5 <- make_bar_plot(avgClean,bar_df2,"Average Cleanliness Rating")+ coord_cartesian(ylim=c(7.5, 10))
bar6 <- make_bar_plot(avgLoc,bar_df2,"Average Location Rating")+ coord_cartesian(ylim=c(7.5, 10))
bar7 <- make_bar_plot(avgVal,bar_df2,"Average Value Rating")+ coord_cartesian(ylim=c(7.5, 10))
bar8 <- make_bar_plot(avgRating,bar_df2,"Average Overall Rating")+ coord_cartesian(ylim=c(75, 100))

grid.arrange(cbar,bar1,bar2,bar3,bar4,bar5,bar6,bar7,bar8, ncol=3)
```


Room.Type
Shared Room
Private Room
Entire House
Percent
Cluster
Average Price
Average Accomodates
Average Bedrooms
Average Beds
Average Cleanliness Rating
Average Location Rating
Average Value Rating
Average Overall Rating