Apprentissage Statistique

New York city Airbnb

Duong Nguyen & Julien Le Mauff

28 avril, 2020

Import Packages

Import Raw data

Data manipulations / Cleaning the data

Dealing with missing values

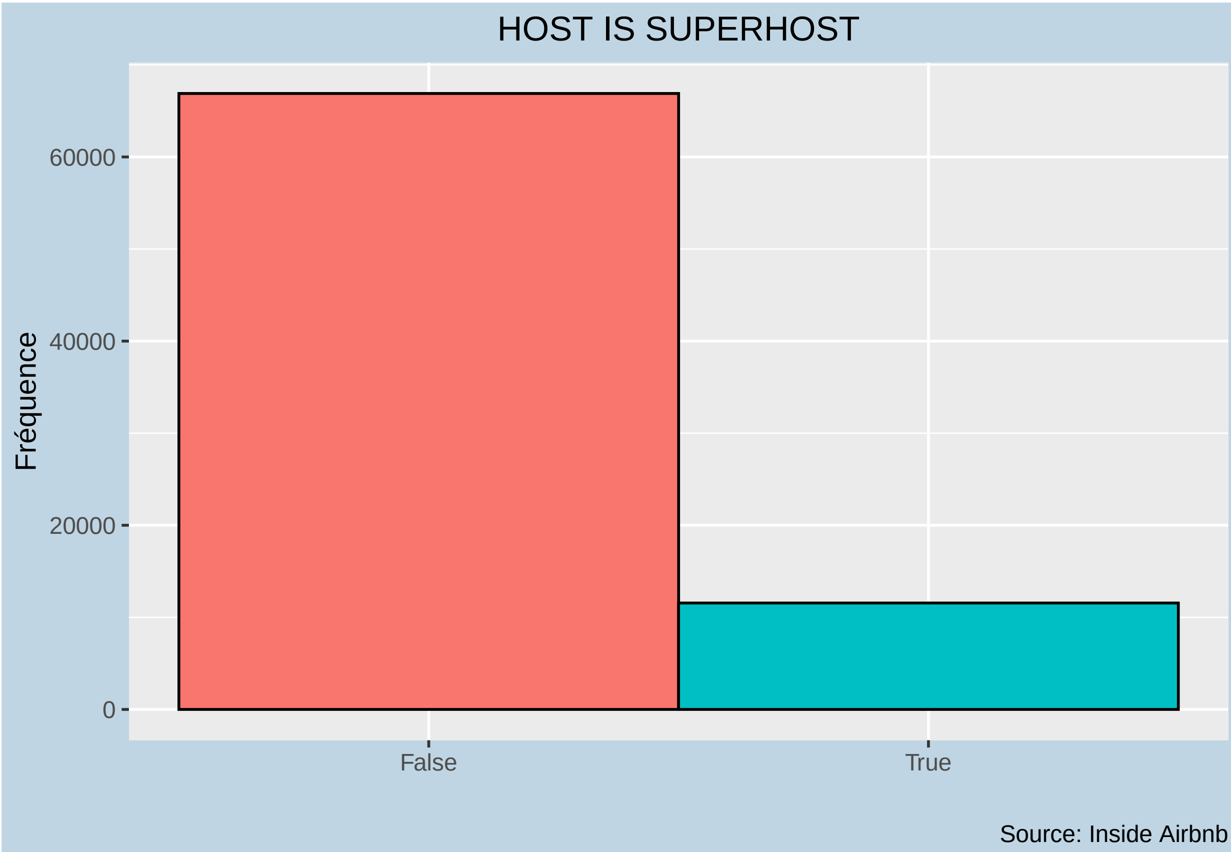
Dealing with categorical variables

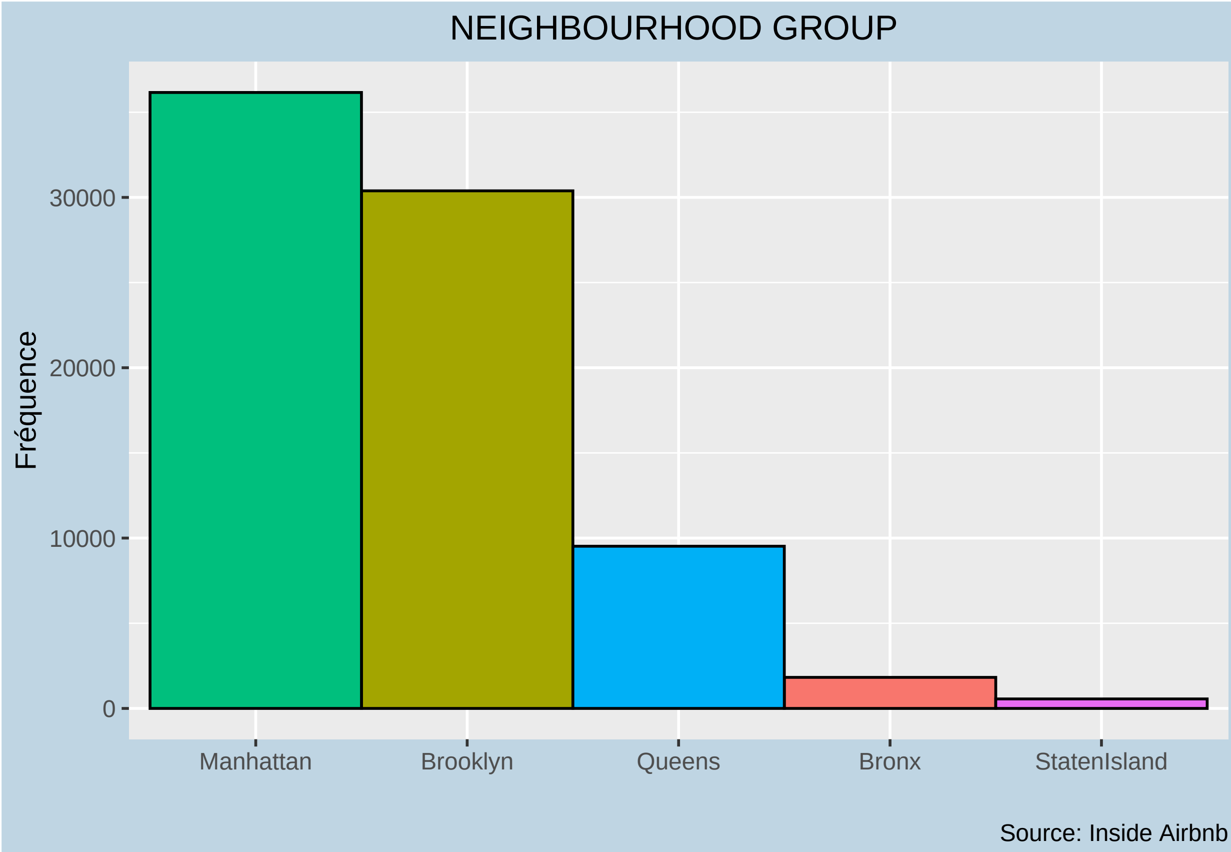
Data analysis / Data visualisation

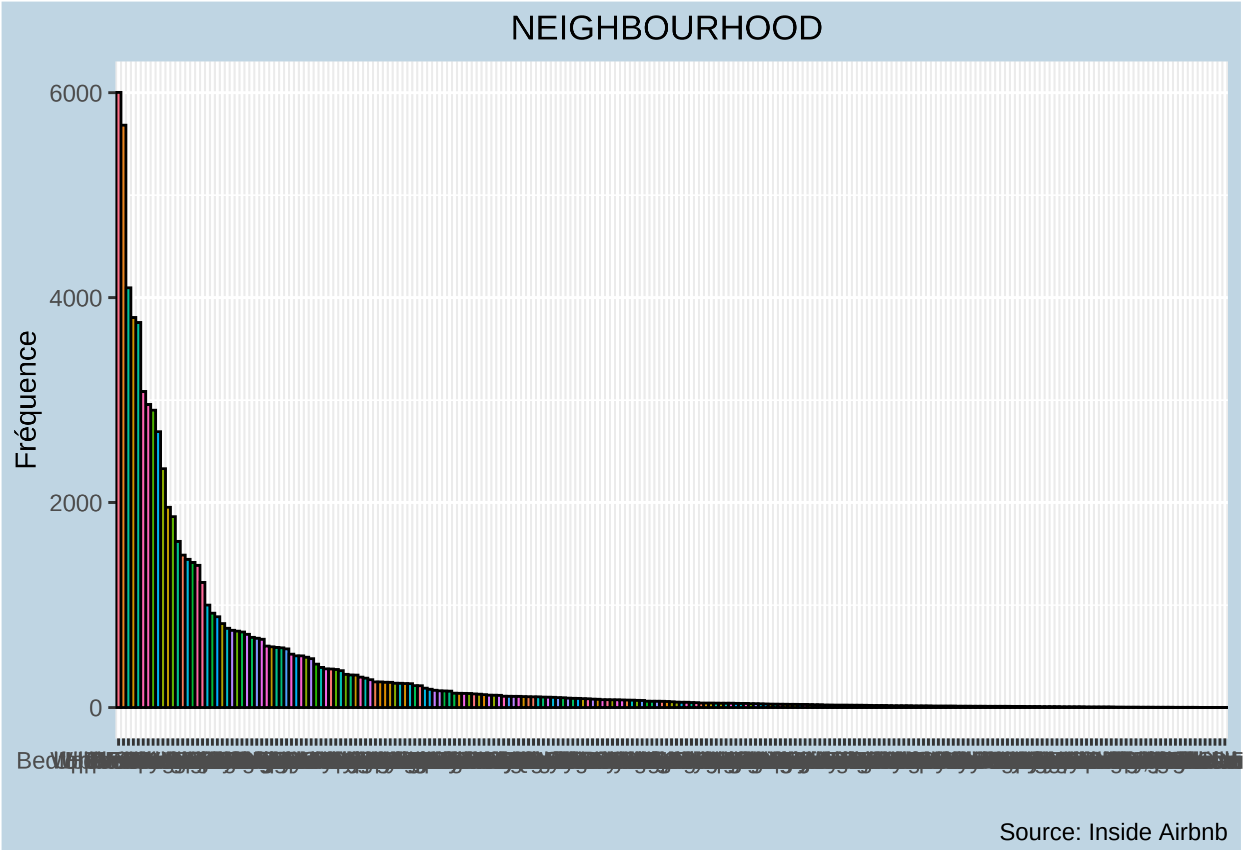
Data visualisation

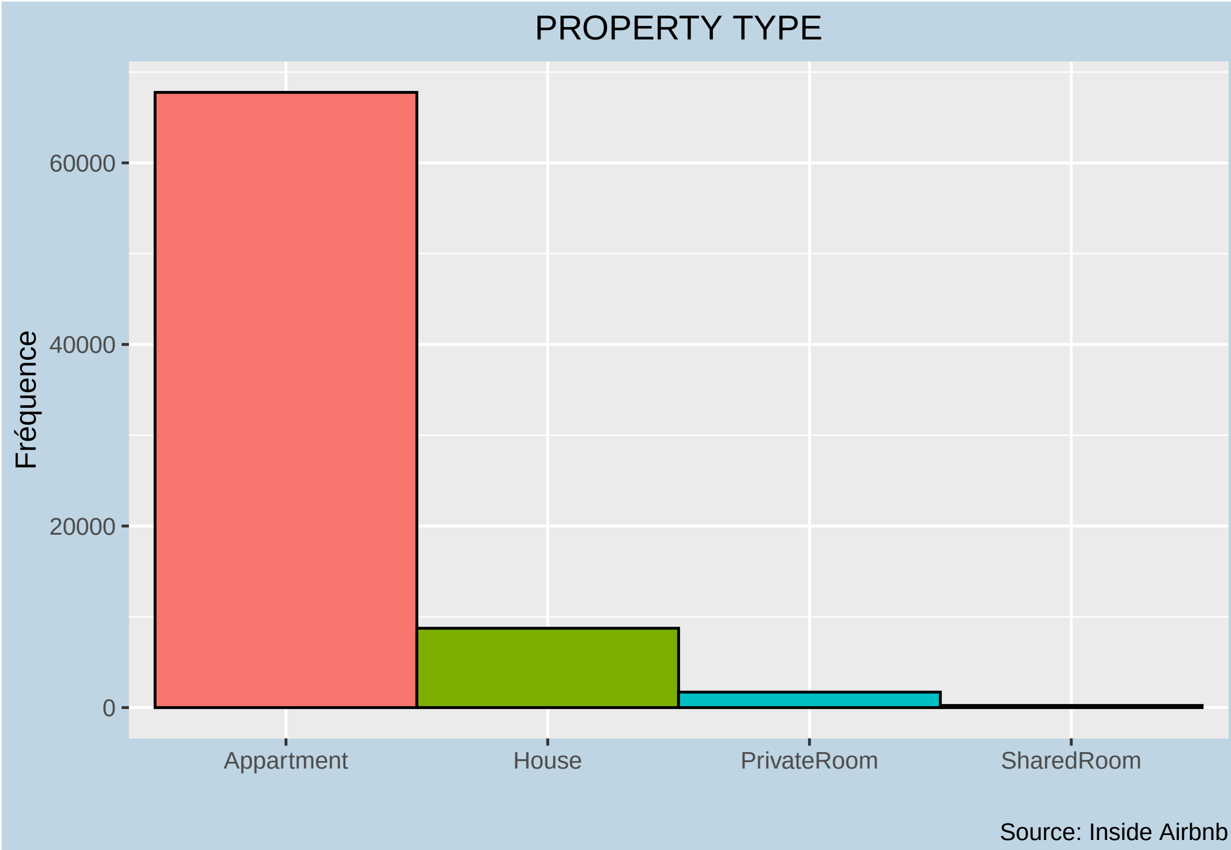
Frequence of discrete variables

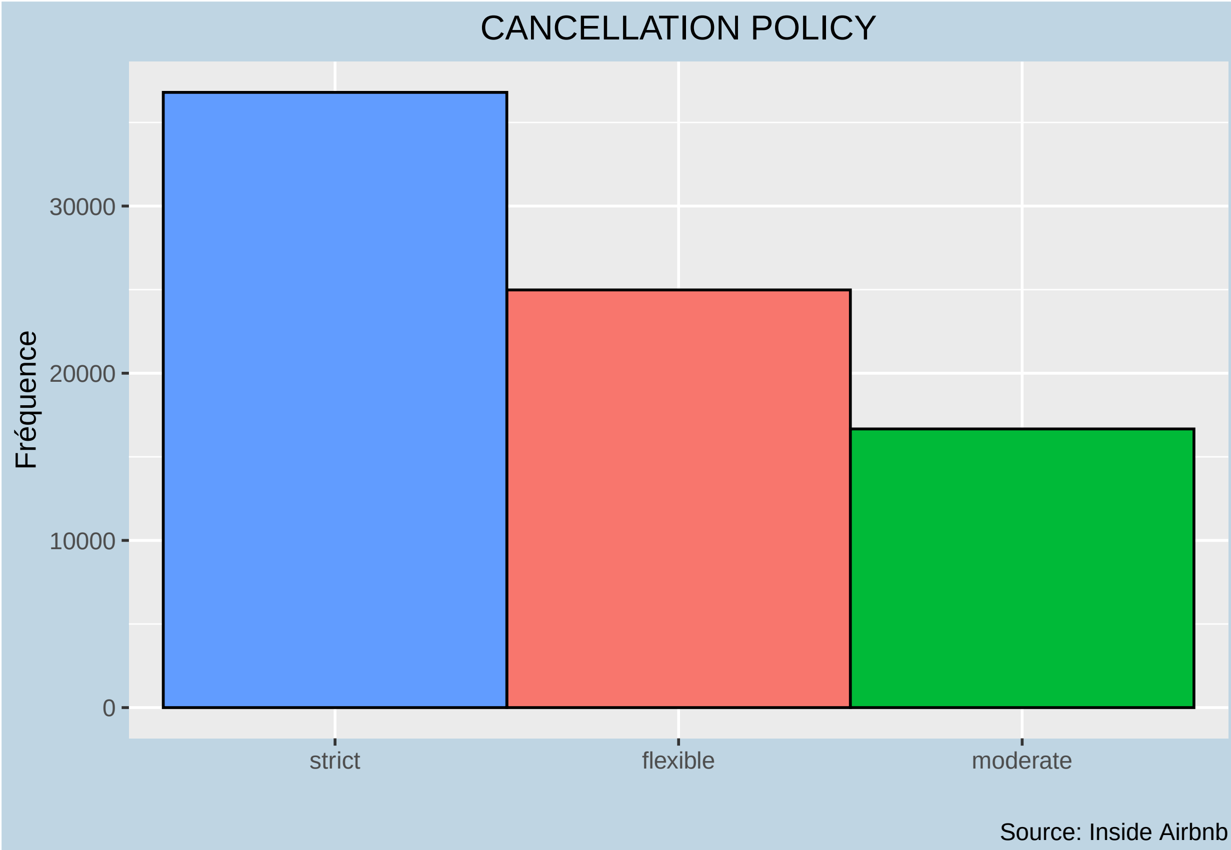
```
features_discrete <- names(select_if(data[,-c(1)],is.factor))
# Without id

for (i in features_discrete) {
  plot <-ggplot(mapping = aes_string(x = fct_infreq(data[,i]),
                                     fill = data[,i])) +
    geom_bar(width = 1, colour = "black", show.legend = F) +
    theme + labs(x = "", y = "Fréquence", caption = tag_source) +
    ggtitle(str_replace_all(toupper(i), c("_" = " ", "CLEANSED" = "")))
  print(plot)
}
```


HOST IS SUPERHOST
Fréquence
60000
40000
20000
0
False
True
Source: Inside Airbnb


NEIGHBOURHOOD GROUP
Fréquence
30000
20000
10000
0
Manhattan
Brooklyn
Queens
Bronx
StatenIsland
Source: Inside Airbnb


NEIGHBOURHOOD
Fréquence
6000
4000
2000
0
Source: Inside Airbnb

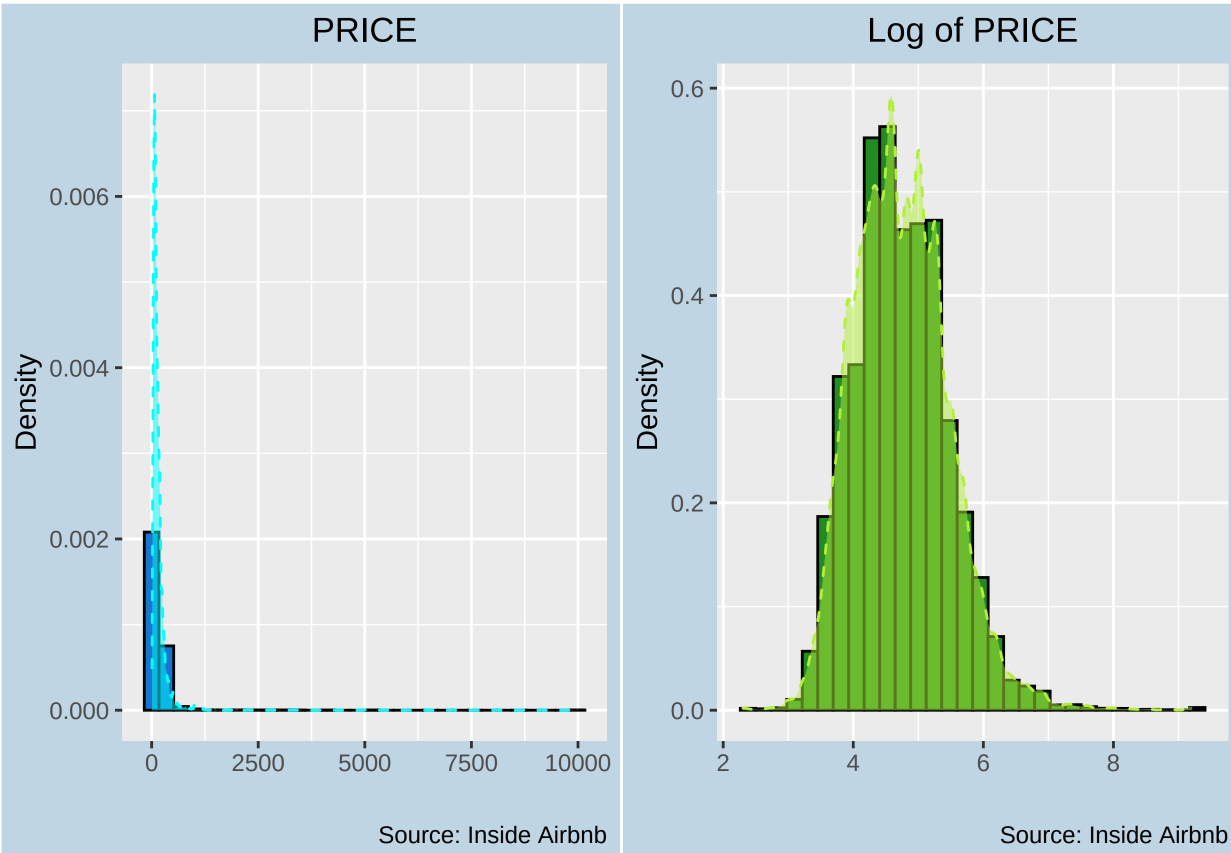

PROPERTY TYPE
Fréquence
60000
40000
20000
0
Appartment
House
PrivateRoom
SharedRoom
Source: Inside Airbnb

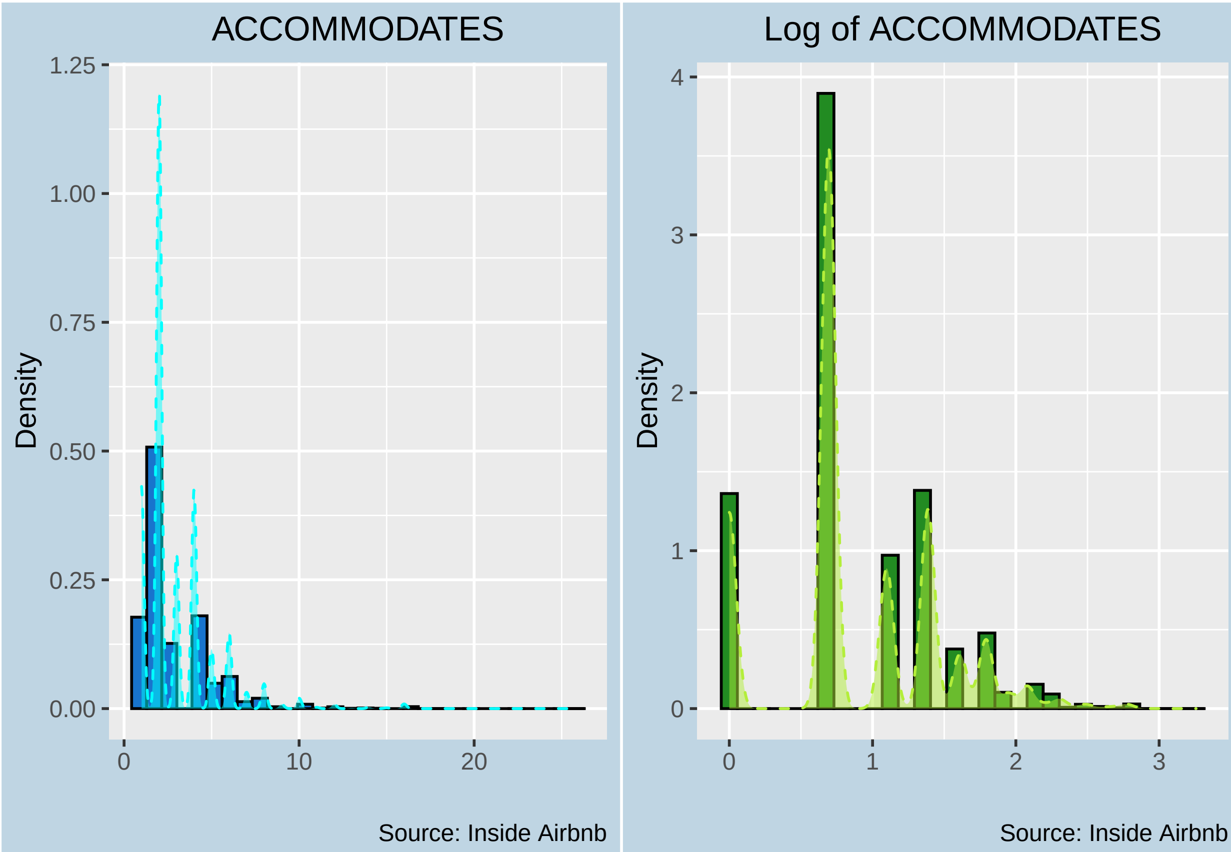


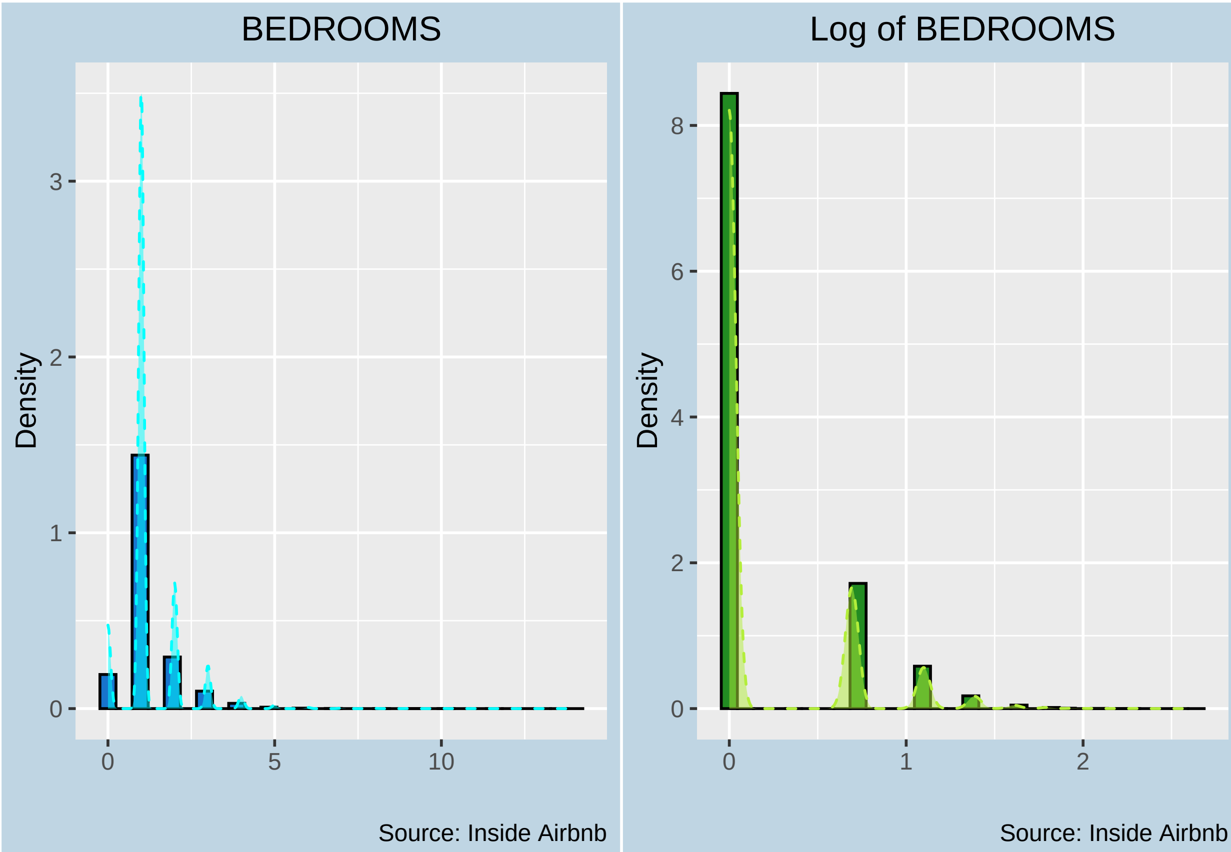
Density and log of continuous variables

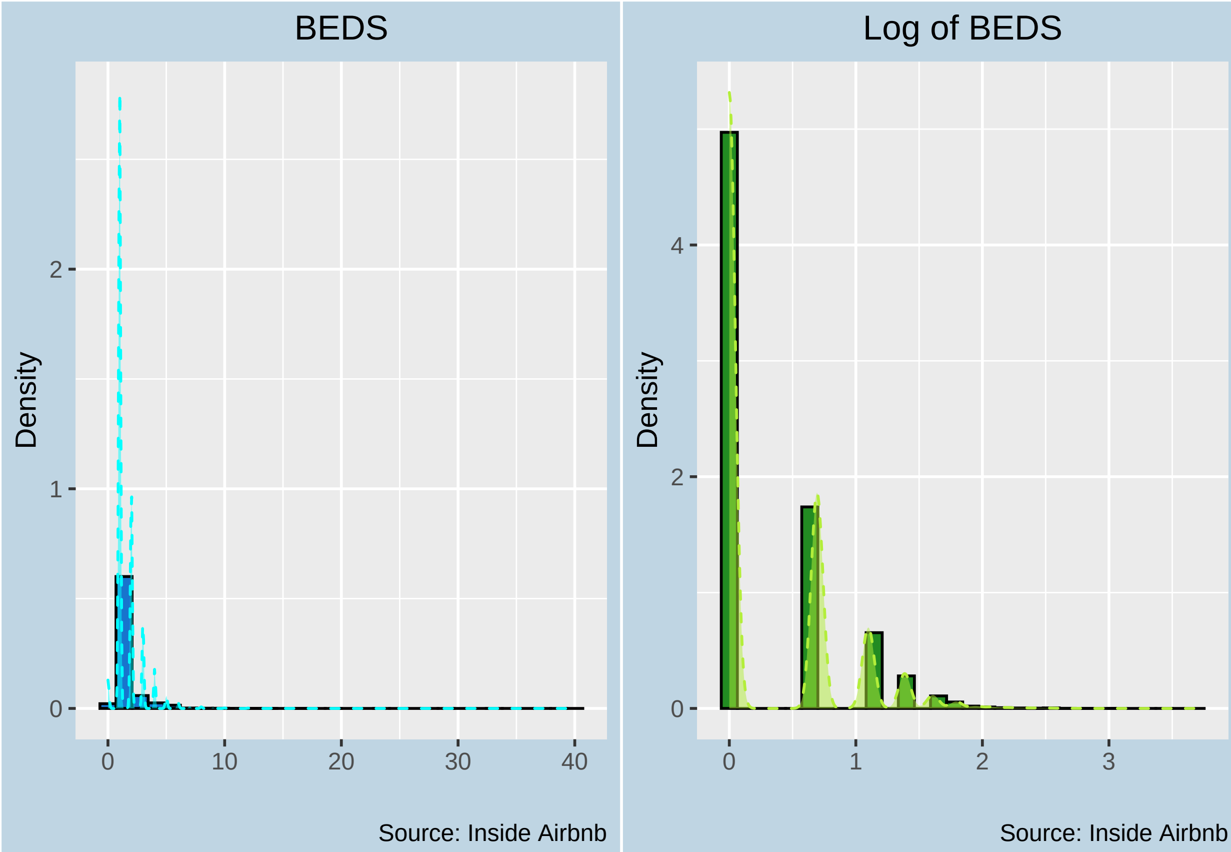
```
features_numeric <- names(select_if(data[,-c(1,6:7)],is.numeric))
# Without id, latitude and longitude

for (i in features_numeric){

  plot <- ggplot(mapping = aes(x = data[,i])) +
    geom_histogram(colour="black", fill="dodgerblue3",
                   aes(y = ..density..)) + theme +
    ggtitle(str_replace_all(toupper(i), c("_" = " "))) +
    labs(x = "", y = "Density", caption = tag_source) +
    geom_density(fill = "cyan", colour = "cyan",
                 alpha = 0.5, lwd=0.5, linetype = "dashed")

  log_plot <- ggplot(mapping = aes(x = log(data[,i]))) +
    geom_histogram(colour="black", fill="forestgreen",
                   aes(y= ..density..)) + theme +
    labs(x = "", y = "Density", caption = tag_source) +
    ggtitle(str_replace_all(paste("Log of",toupper(i)),
                            c("_" = " "))) +
    geom_density(fill = "olivedrab2", colour = "olivedrab2",
                 alpha = 0.5, lwd=0.5, linetype = "dashed")
```

```
  grid.arrange(plot, log_plot, ncol=2)
}
```

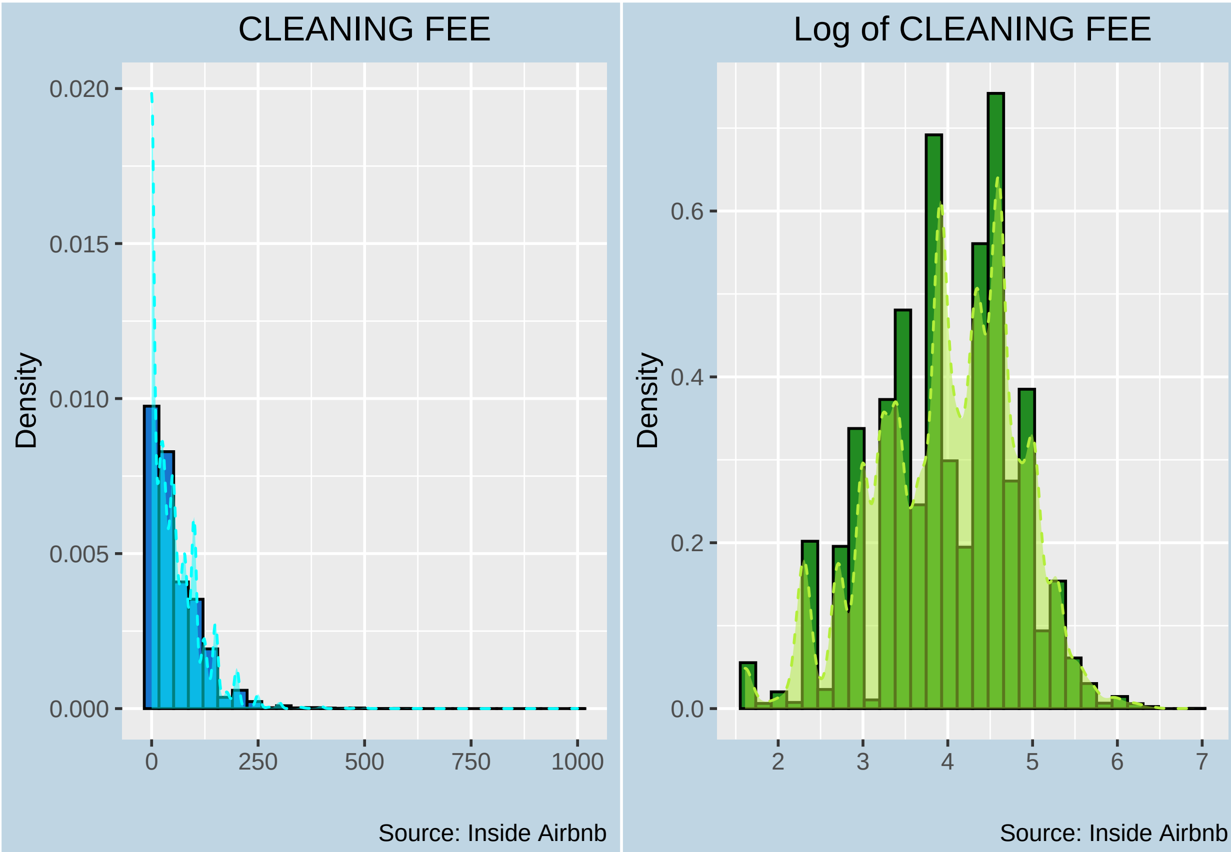


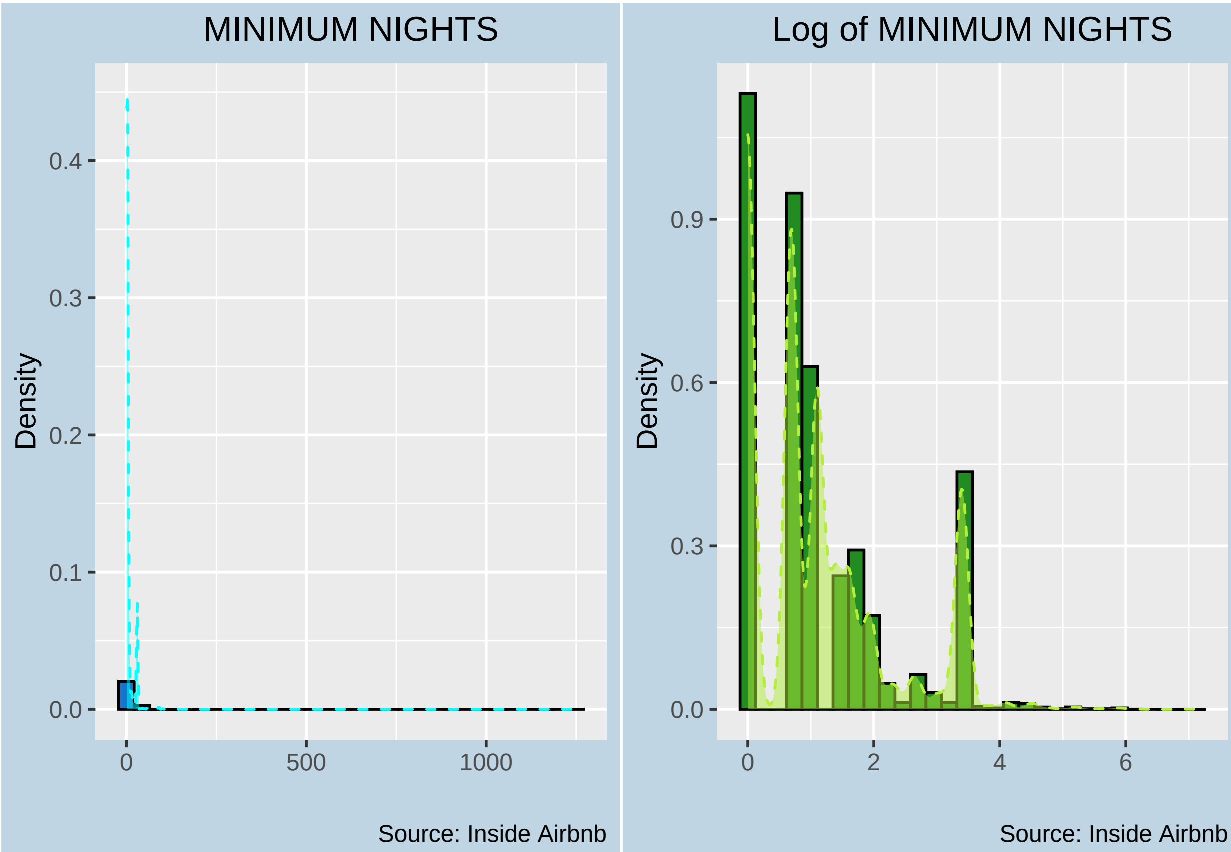

ACCOMMODATES
Density
1.25
1.00
0.75
0.50
0.25
0.00
0
10
20
Source: Inside Airbnb
Log of ACCOMMODATES
Density
4
3
2
1
0
0
1
2
3
Source: Inside Airbnb

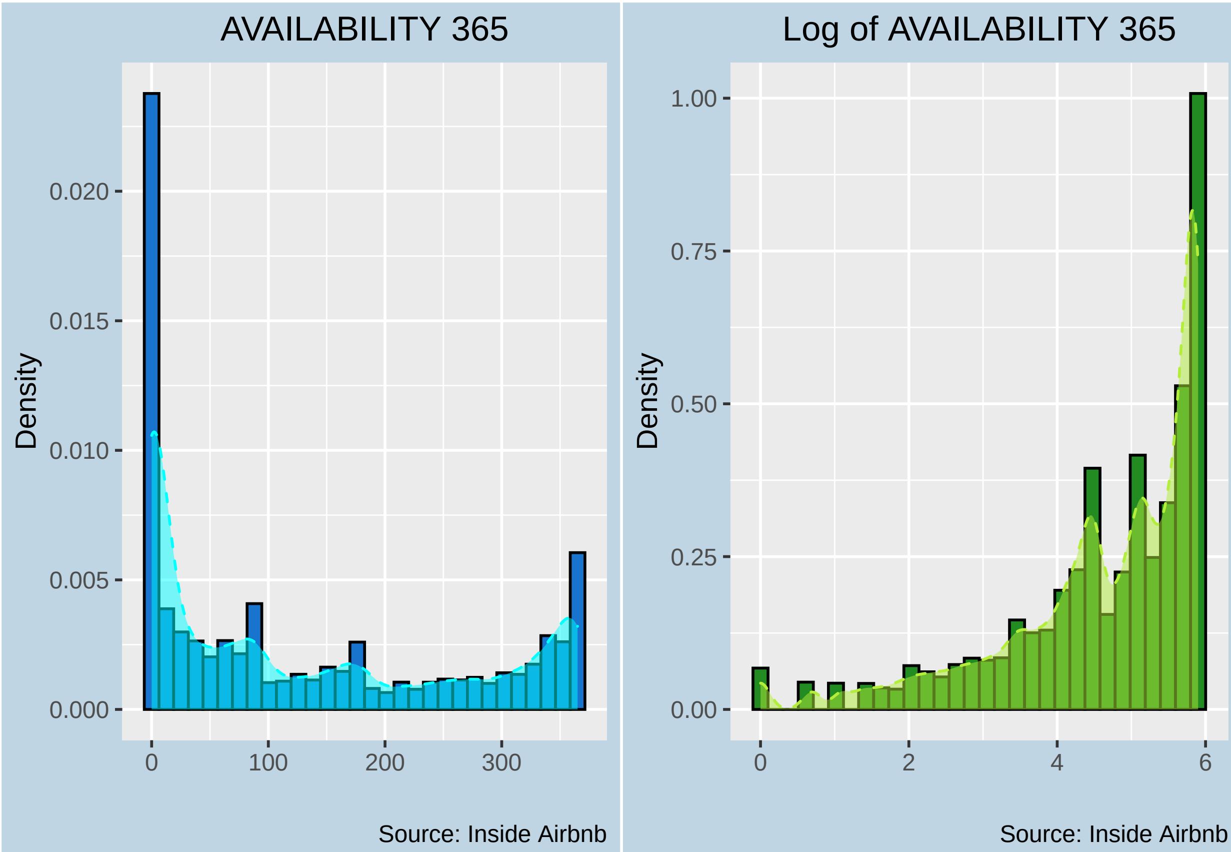

BEDROOMS
Density
0
1
2
3
0
5
10
Source: Inside Airbnb
Log of BEDROOMS
Density
0
2
4
6
8
0
1
2
Source: Inside Airbnb

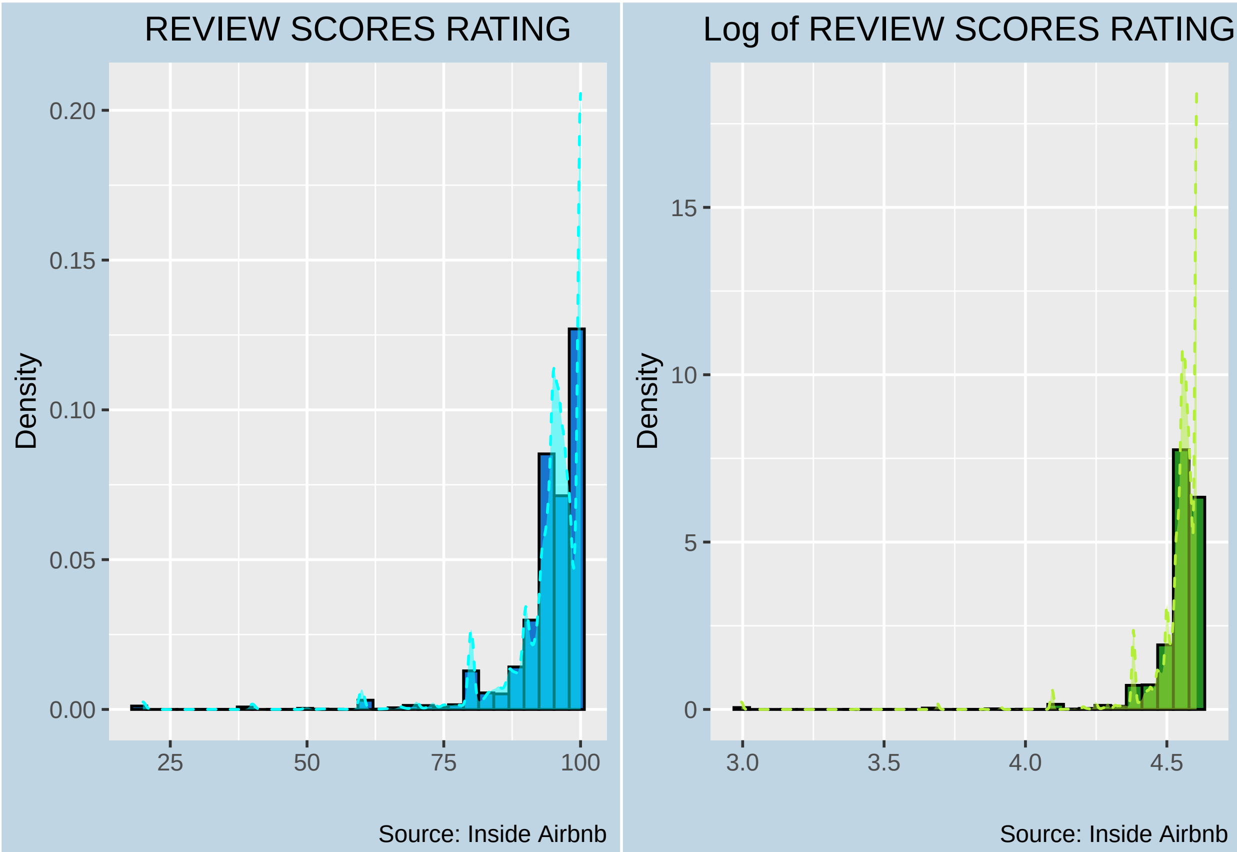


BEDS
Density
0
1
2
0
10
20
30
40
Source: Inside Airbnb
Log of BEDS
Density
0
2
4
0
1
2
3
Source: Inside Airbnb


CLEANING FEE
Density
0.020
0.015
0.010
0.005
0.000
0
250
500
750
1000
Source: Inside Airbnb
Log of CLEANING FEE
Density
0.6
0.4
0.2
0.0
2
3
4
5
6
7
Source: Inside Airbnb

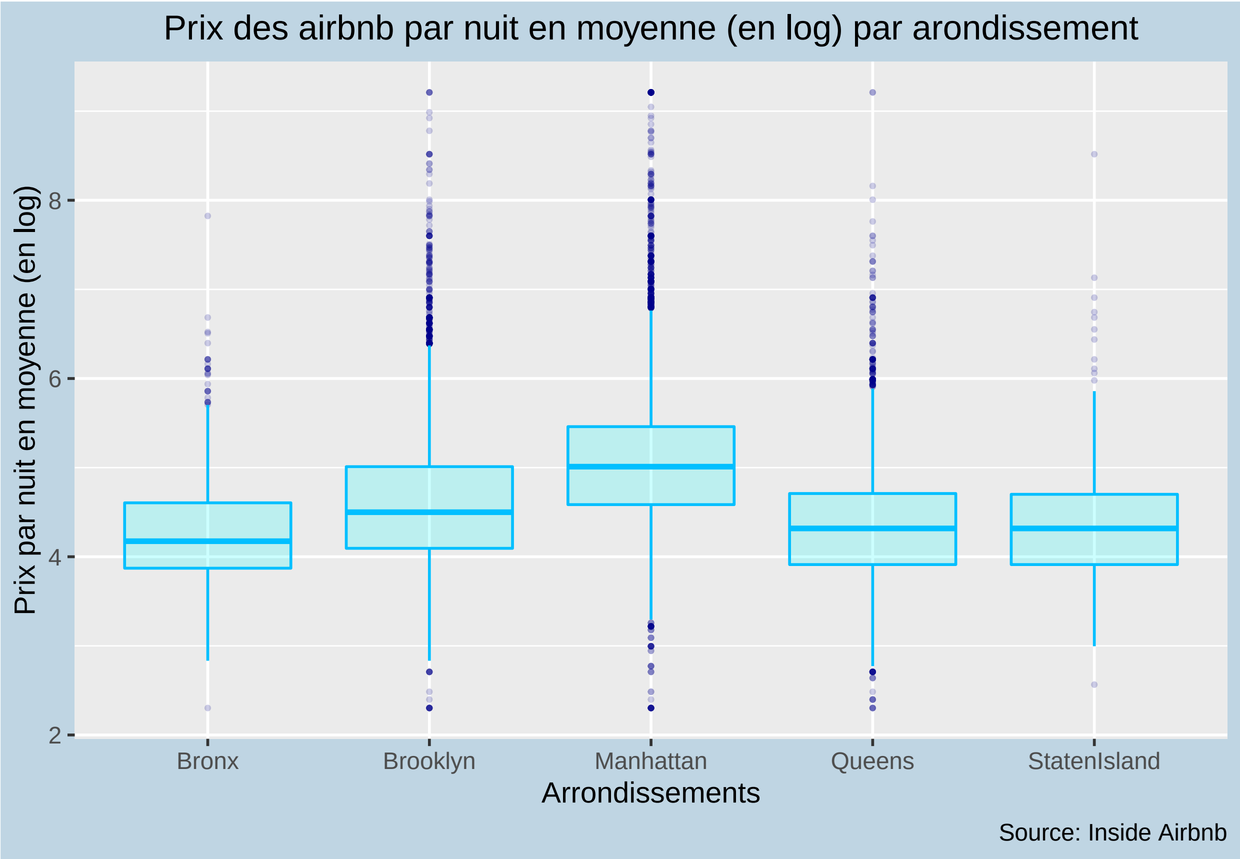

MINIMUM NIGHTS
Density
0.4
0.3
0.2
0.1
0.0
0
500
1000
Source: Inside Airbnb
Log of MINIMUM NIGHTS
Density
0.9
0.6
0.3
0.0
0
2
4
6
Source: Inside Airbnb


AVAILABILITY 365
Density
0.020
0.015
0.010
0.005
0.000
0
100
200
300
Source: Inside Airbnb
Log of AVAILABILITY 365
Density
1.00
0.75
0.50
0.25
0.00
0
2
4
6
Source: Inside Airbnb

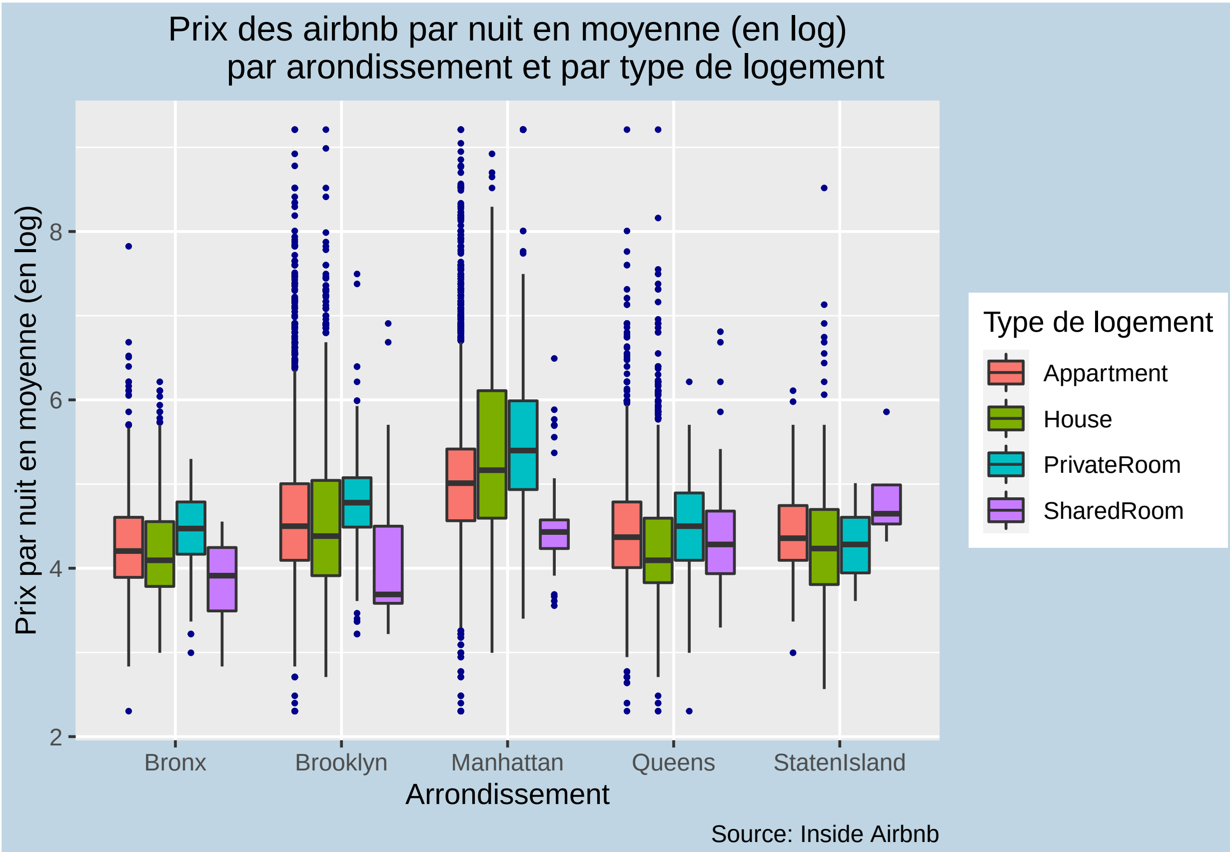


Boxplot

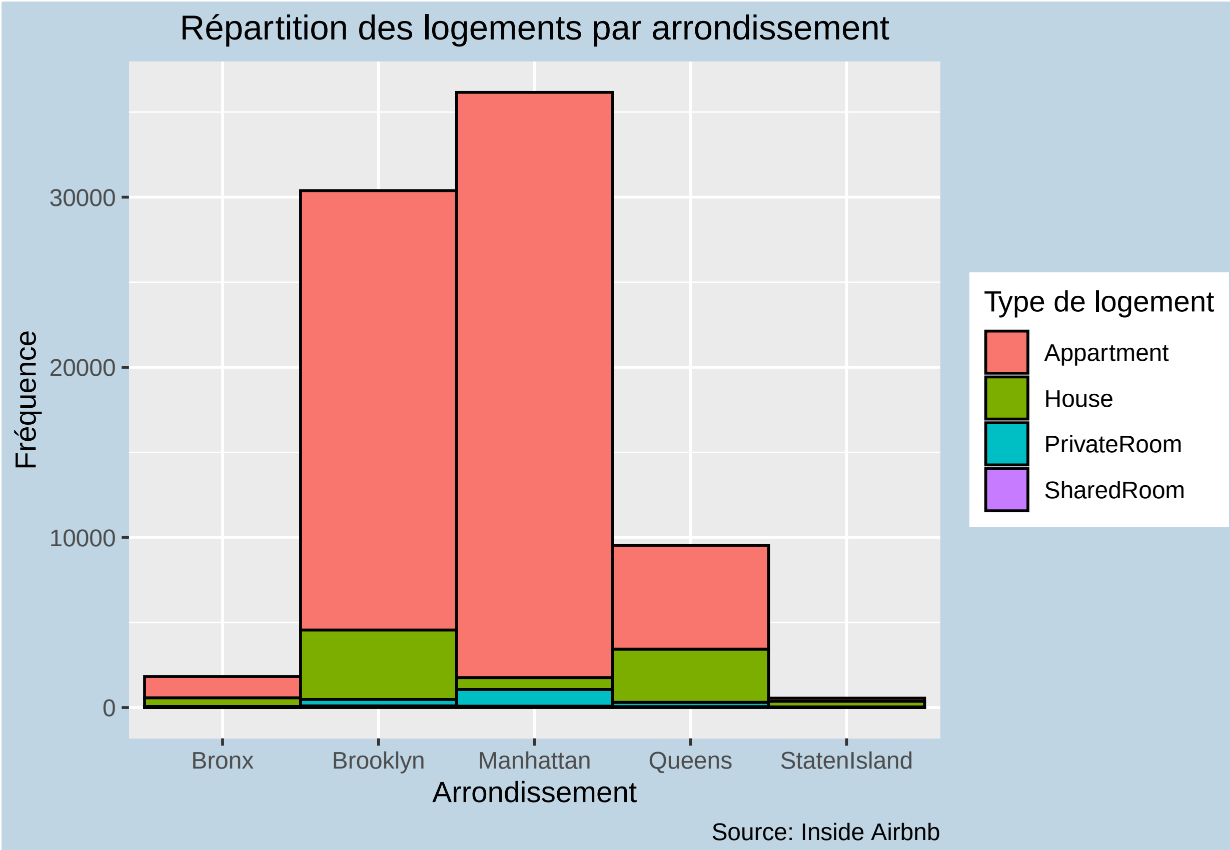
```
# # ===== ===== Prix par arrondissement ===== =====
ggplot(data, aes(x = neighbourhood_group_cleansed, y = log(price))) +
  geom_boxplot(outlier.colour = "darkblue", outlier.size = 0.5,
               color="deepskyblue", fill="cyan", alpha=0.2) +
  ggtitle("Prix des airbnb par nuit en moyenne (en log) par arondissement") +
  labs(x = "Arrondissements", y = "Prix par nuit en moyenne (en log)",
       caption = tag_source) + theme
```




```r
# # ===== ===== Prix par arrondissement et type de logement ===== =====

ggplot(data, aes(x = neighbourhood_group_cleansed, y = log(price),
                 fill = property_type)) +
  geom_boxplot(outlier.colour = "darkblue", outlier.size = 0.5) +
  ggtitle("Prix des airbnb par nuit en moyenne (en log)
        par arondissement et par type de logement") +
  scale_fill_discrete(name = "Type de logement") + theme +
  labs(x = "Arrondissement", y = "Prix par nuit en moyenne (en log)",
      caption = tag_source)
```

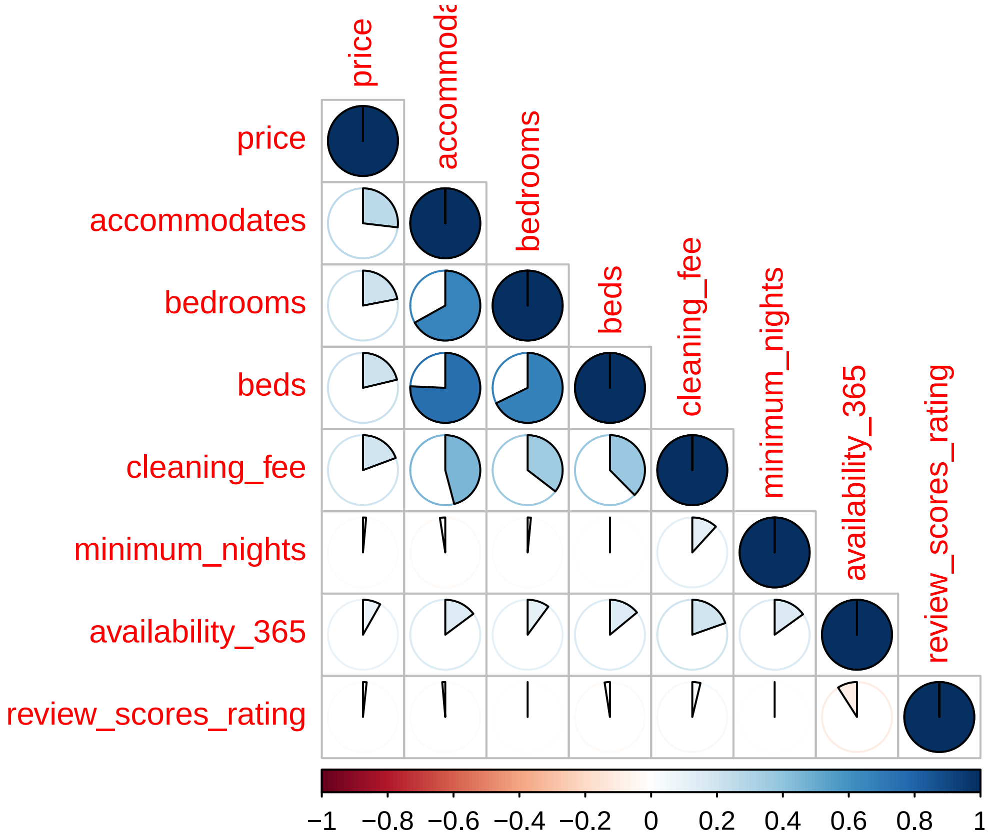



```r
# # ===== ===== Répartition des logements par arrondissement ===== =====

ggplot(data, aes(x = neighbourhood_group_cleansed, fill = property_type)) +
  geom_bar(width = 1, colour = "black", show.legend = T) +
  ggtitle("Répartition des logements par arrondissement") +
  scale_fill_discrete(name = "Type de logement") + theme +
  labs(x = "Arrondissement", y = "Fréquence", caption = tag_source)
```



Correlation matrix

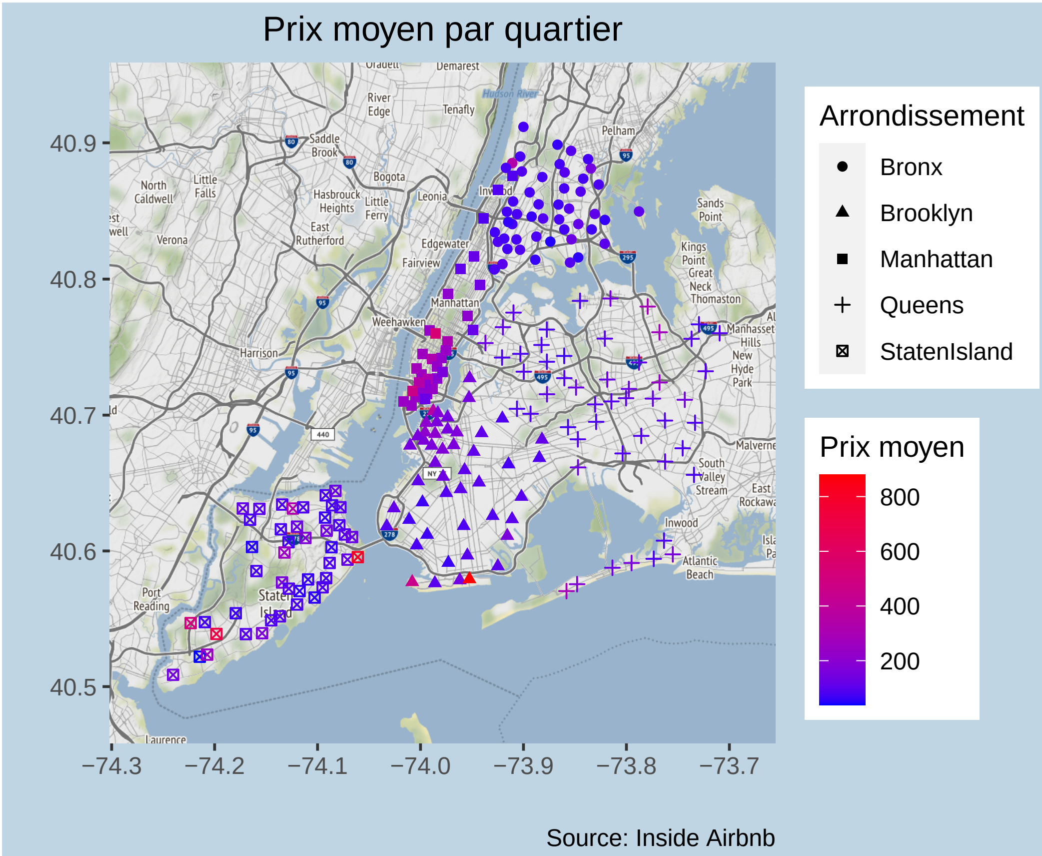
```
corrplot(cor(data[,features_numeric]), method = "pie", type = "lower")
```



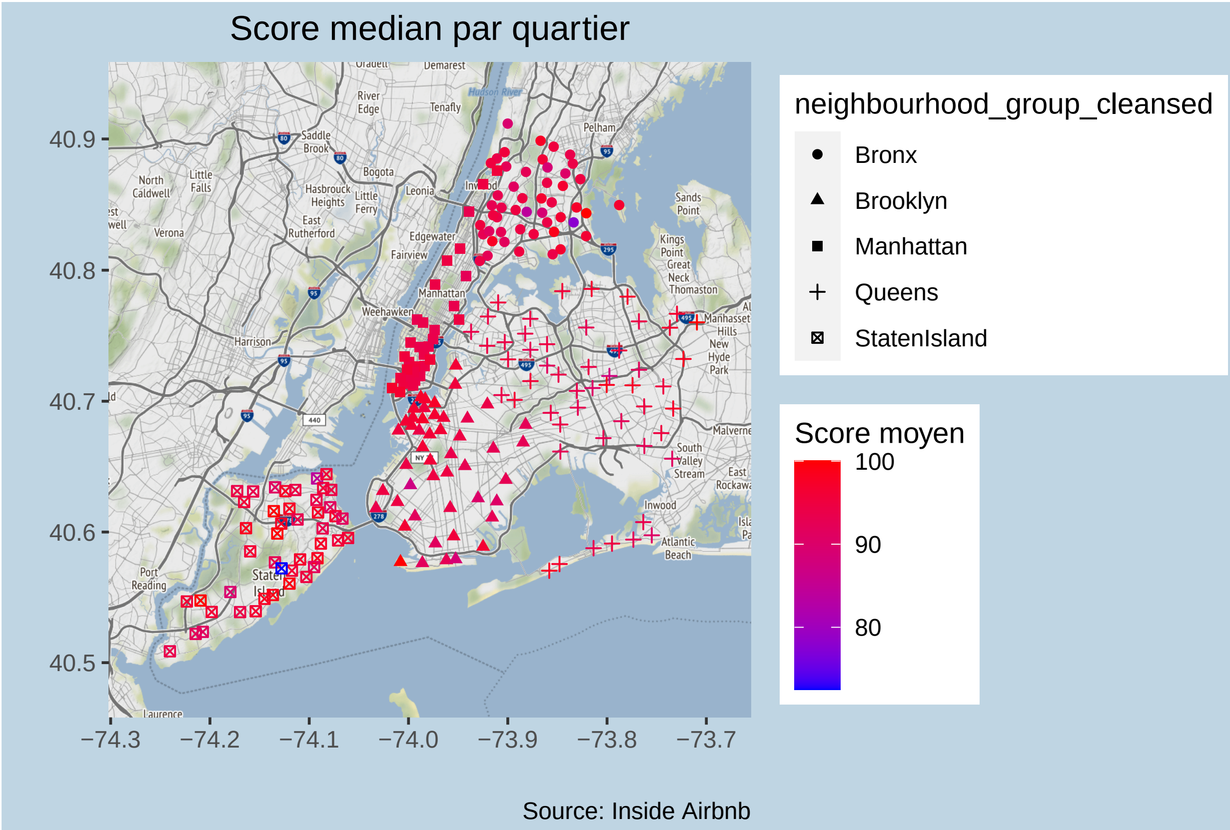
Spatial Heatmap

Evolution des prix par quartier

```
# # ===== ===== Prix moyen par quartiers ===== =====
ggmap(map) +
  geom_point(by_neighbourhood,
             mapping = aes(x = longitude, y = latitude,
                           col = prix_moyen,
                           shape = neighbourhood_group_cleansed)) +
  scale_colour_gradient(low = "blue", high = "red") + theme +
  labs(x = "", y = "", caption = tag_source,
       col = "Prix moyen", shape = "Arrondissement") +
  ggtitle("Prix moyen par quartier")
```

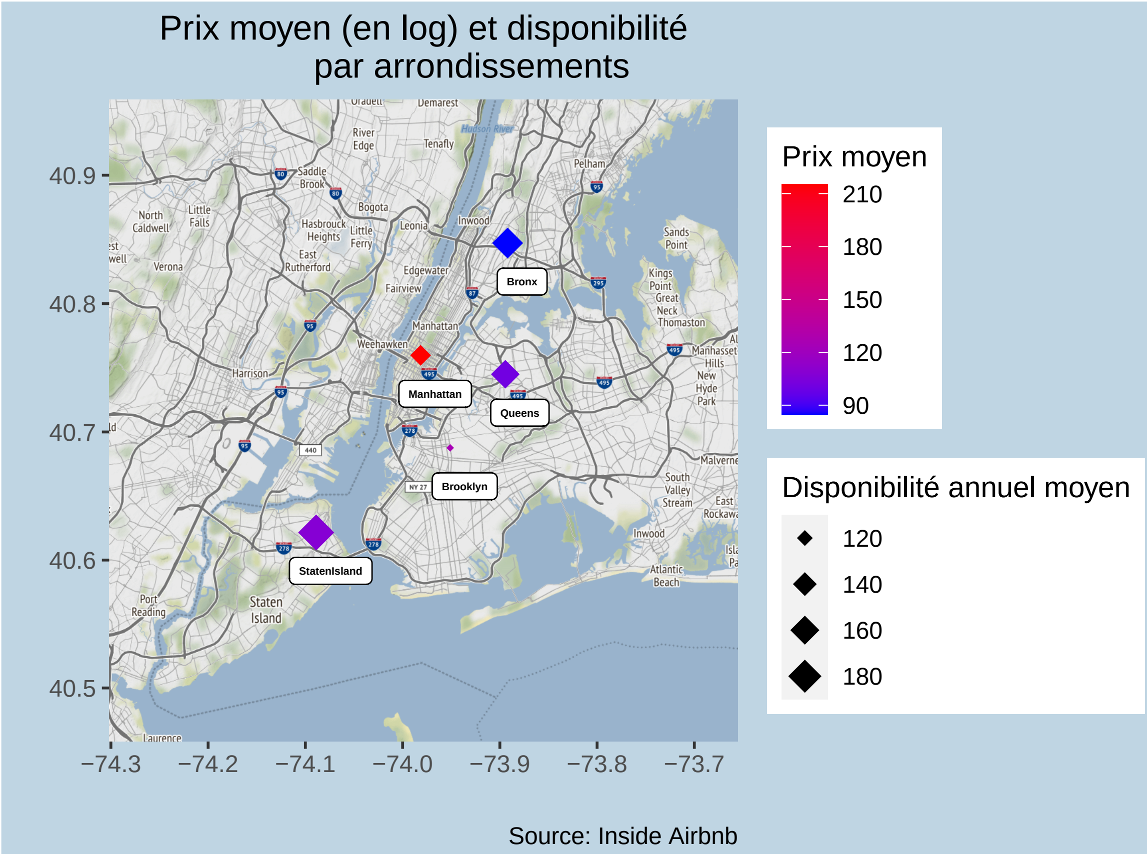



```r
# # ===== ===== Score median par quartiers ===== =====
ggmap(map) +
  geom_point(by_neighbourhood,
             mapping = aes(x = longitude, y = latitude,
                           col = scores_moyen,
                           shape = neighbourhood_group_cleansed)) +
  scale_colour_gradient(low = "blue", high = "red") + theme +
  labs(x = "", y = "", caption = tag_source, col = "Score moyen") +
  ggtitle("Score median par quartier")
```



Evolution des prix par Arrondissement

```
# # ===== ===== Prix moyen par arrondissements ===== =====
ggmap(map) +
  geom_point(by_borough, shape = 18,
             mapping = aes(x = longitude, y = latitude,
                           col = prix_moyen, size = disponibilité_moyen)) +
  scale_colour_gradient(low = "blue", high = "red") +
  theme + labs(x = "", y = "", caption = tag_source,
               size = "Disponibilité annuel moyen",
               col = "Prix moyen") +
  ggtitle("Prix moyen (en log) et disponibilité
        par arrondissements") +
  geom_label(by_borough,
             mapping = aes(longitude, latitude,
                           label = neighbourhood_group_cleansed),
             size = 1.5, fontface = "bold",
             nudge_x = 0.015, nudge_y = -0.03)
```



Models

Setting up the theme

Split the data and setting seeds

Allow for Parallel computing

Ordinary Least Square model

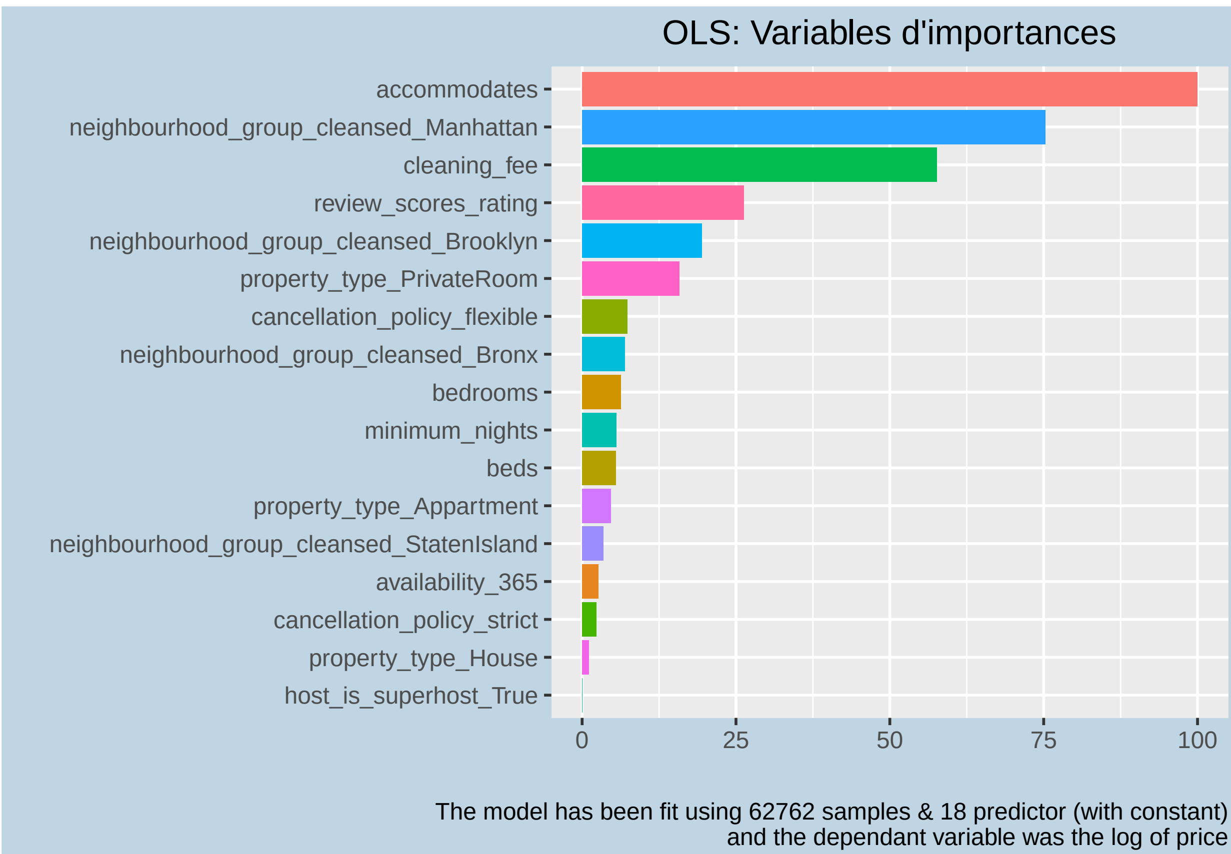
```
set.seed(777)

ols_fit <- train(log_price ~ . -
                   host_is_superhost_False -
                   property_type_SharedRoom -
                   neighbourhood_group_cleansed_Queens -
                   cancellation_policy_moderate,
                 data = data_train,
                 method = "lm" ,
                 trControl = K5_CV_seed)

summary(ols_fit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0010 -0.3237 -0.0203  0.2907  5.0917
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                2.951e+00  4.543e-02  64.954
## host_is_superhost_True                    -6.473e-03  6.052e-03  -1.069
## neighbourhood_group_cleansed_Bronx        -1.170e-01  1.492e-02  -7.843
## neighbourhood_group_cleansed_Brooklyn      1.412e-01  7.072e-03  19.963
## neighbourhood_group_cleansed_Manhattan     5.374e-01  7.250e-03  74.123
## neighbourhood_group_cleansed_StatenIsland -1.120e-01  2.515e-02  -4.454
## property_type_Appartment                   2.154e-01  3.819e-02   5.639
## property_type_House                        8.116e-02  3.861e-02   2.102
## property_type_PrivateRoom                  6.662e-01  4.057e-02  16.420
## accommodates                               1.708e-01  1.742e-03  98.065
## bedrooms                                   2.694e-02  3.760e-03   7.165
## beds                                      -1.875e-02  2.940e-03  -6.376
## cleaning_fee                               2.397e-03  4.205e-05  57.002
## minimum_nights                            -7.098e-04  1.095e-04  -6.482
## availability_365                           6.086e-05  1.665e-05   3.655
## review_scores_rating                       6.824e-03  2.566e-04  26.590
## cancellation_policy_flexible               4.845e-02  5.909e-03   8.199
## cancellation_policy_strict                -1.829e-02  5.578e-03  -3.279
##                                           Pr(>|t|)
## (Intercept)                                < 2e-16 ***
## host_is_superhost_True                    0.284868
## neighbourhood_group_cleansed_Bronx        4.46e-15 ***
## neighbourhood_group_cleansed_Brooklyn      < 2e-16 ***
## neighbourhood_group_cleansed_Manhattan     < 2e-16 ***
## neighbourhood_group_cleansed_StatenIsland 8.45e-06 ***
## property_type_Appartment                  1.72e-08 ***
## property_type_House                       0.035568 *
## property_type_PrivateRoom                  < 2e-16 ***
## accommodates                               < 2e-16 ***
## bedrooms                                  7.84e-13 ***
## beds                                      1.83e-10 ***
## cleaning_fee                               < 2e-16 ***
## minimum_nights                            9.12e-11 ***
## availability_365                          0.000258 ***
## review_scores_rating                       < 2e-16 ***
## cancellation_policy_flexible              2.48e-16 ***
## cancellation_policy_strict                0.001042 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5221 on 62744 degrees of freedom
## Multiple R-squared:  0.4917, Adjusted R-squared:  0.4915
## F-statistic:  3570 on 17 and 62744 DF,  p-value: < 2.2e-16
```

```r
ols_pred <- predict(ols_fit, data_test)

postResample(pred = ols_pred, obs = data_test$log_price)
```

```
##      RMSE  Rsquared       MAE
## 0.5149610 0.4929084 0.3863929
```

```r
ols_varImp <- data.frame(variables = row.names(varImp(ols_fit)$importance),
                         varImp(ols_fit)$importance)

ggplot(data = ols_varImp, mapping = aes(x=reorder(variables, Overall),
                                        y=Overall,
                                        fill=variables)) +
  coord_flip() + geom_bar(stat = "identity", position = "dodge") +
  theme_models + labs(x = "", y = "", caption = tag_source_models) +
  ggtitle("OLS: Variables d'importances")
```



Ridge regression

```r
set.seed(777)

ridge_fit <- train(log_price ~ . -
                     host_is_superhost_False -
                     property_type_SharedRoom -
                     neighbourhood_group_cleansed_Queens -
```

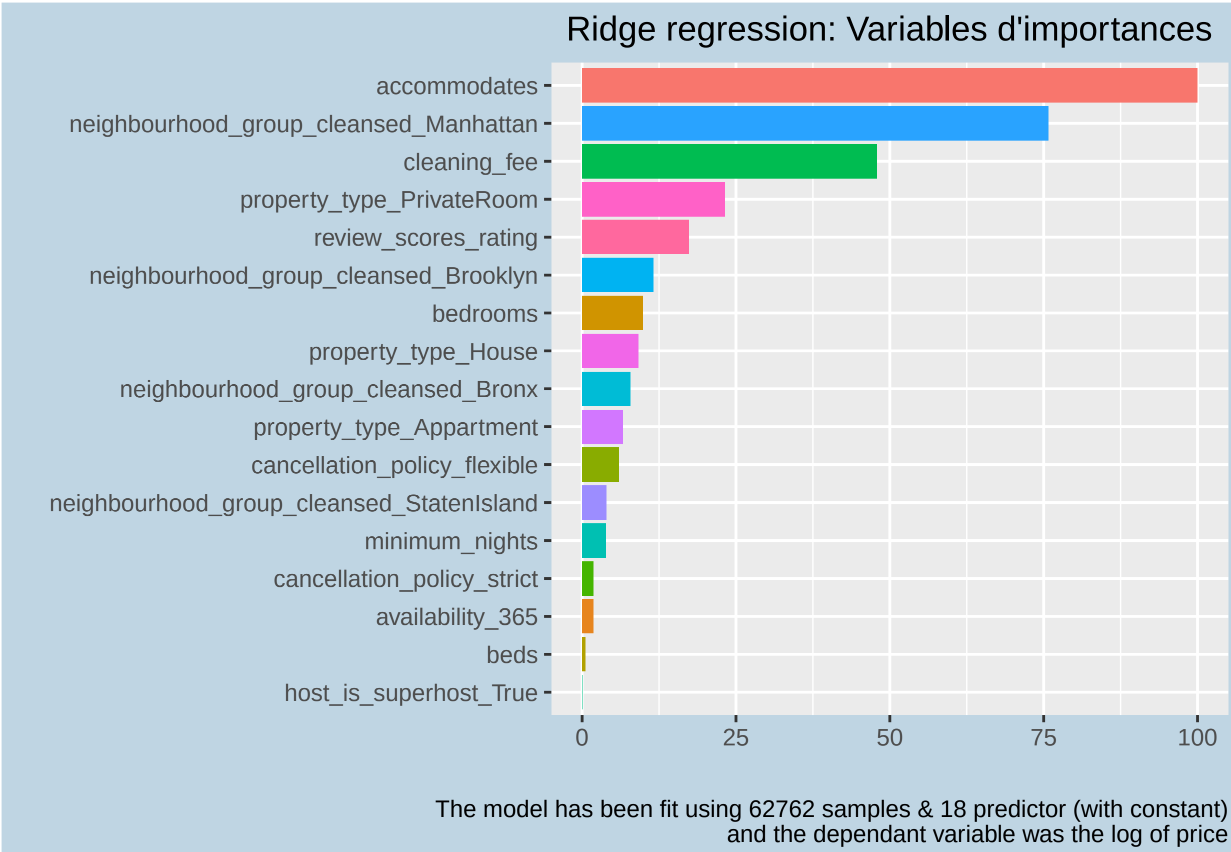
```r
                   cancellation_policy_moderate,
                   data = data_train,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 0,
                                          lambda = seq(0, 1, 0.01)),
                   preProcess = c("scale","center"),
                   trControl = K5_CV_seed)

ridge_fit$bestTune
```

```
##   alpha lambda
## 5     0   0.04
```

```r
ridge_pred <- predict(ridge_fit, data_test)

postResample(pred = ridge_pred, obs = data_test$log_price)
```

```
##      RMSE  Rsquared       MAE
## 0.5157762 0.4914722 0.3880052
```

```r
ridge_varImp <- data.frame(variables = row.names(varImp(ridge_fit)$importance),
                           varImp(ridge_fit)$importance)

ggplot(data = ridge_varImp, mapping = aes(x=reorder(variables, Overall),
                                        y=Overall,
                                        fill=variables)) +
  coord_flip() + geom_bar(stat = "identity", position = "dodge") +
  theme_models + labs(x = "", y = "", caption = tag_source_models) +
  ggtitle("Ridge regression: Variables d'importances")
```



Lasso regression

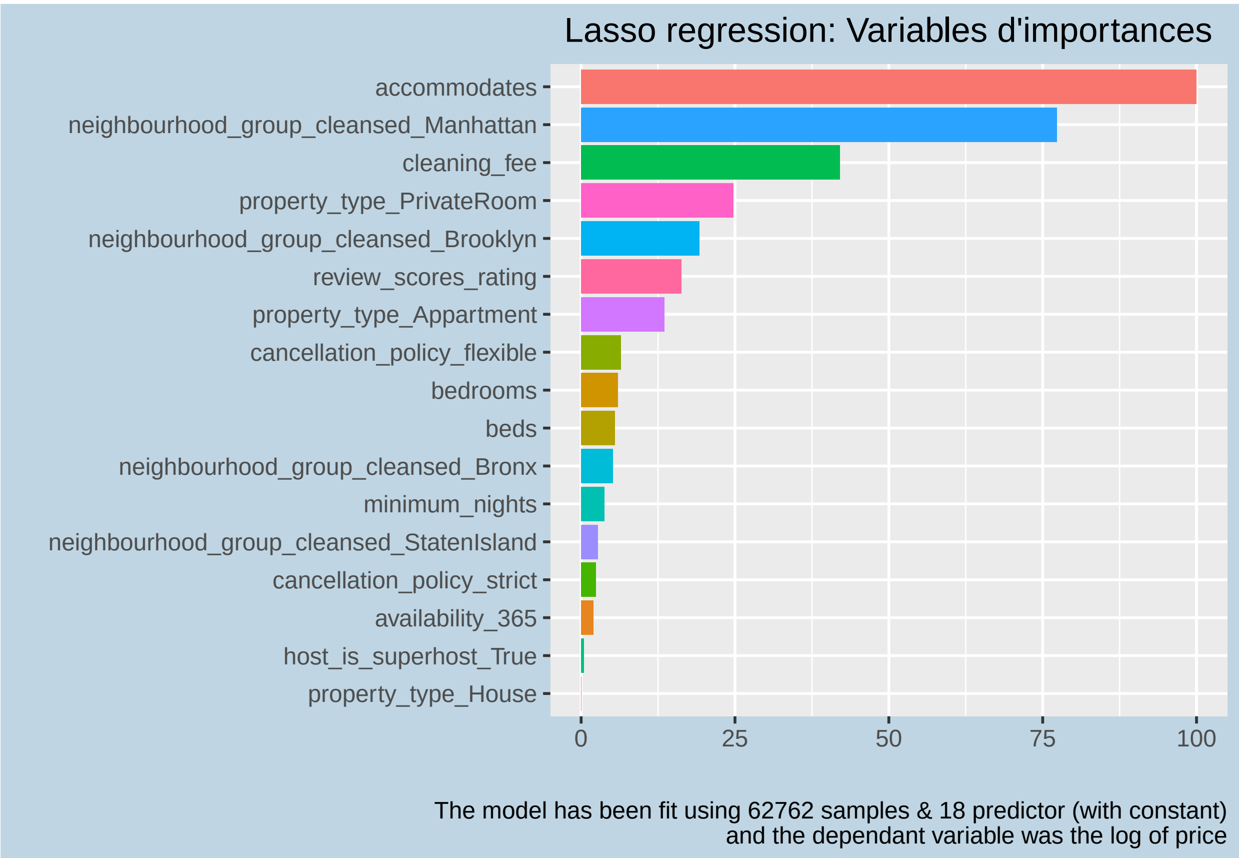
```
set.seed(777)

lasso_fit <- train(log_price ~ . -
                     host_is_superhost_False -
                     property_type_SharedRoom -
                     neighbourhood_group_cleansed_Queens -
                     cancellation_policy_moderate,
                   data = data_train,
                   method = "glmnet",
                   tuneGrid = expand.grid(alpha = 1,
                                          lambda = seq(0, 1, 0.01)),
                   preProcess = c("scale","center"),
                   trControl = K5_CV_seed)

lasso_fit$bestTune
```

```
##   alpha lambda
## 1     1      0
```

```
lasso_pred <- predict(lasso_fit, data_test)

postResample(pred = lasso_pred, obs = data_test$log_price)
```

```
##      RMSE  Rsquared       MAE
## 0.5149834 0.4928203 0.3864483
```

```r
lasso_varImp <- data.frame(variables = row.names(varImp(lasso_fit)$importance),
                           varImp(lasso_fit)$importance)

ggplot(data = lasso_varImp, mapping = aes(x=reorder(variables, Overall),
                                          y=Overall,
                                          fill=variables)) +
  coord_flip() + geom_bar(stat = "identity", position = "dodge") +
  theme_models + labs(x = "", y = "", caption = tag_source_models) +
  ggtitle("Lasso regression: Variables d'importances")
```

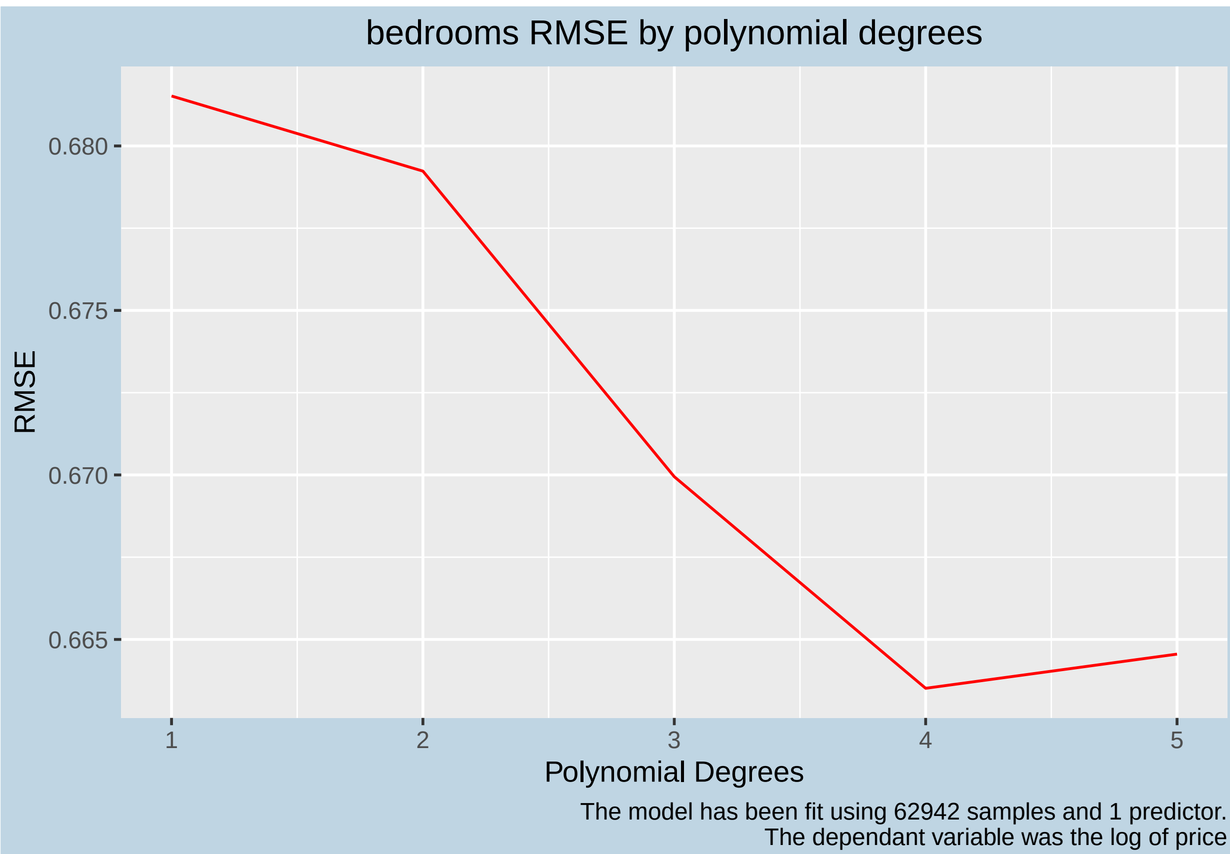


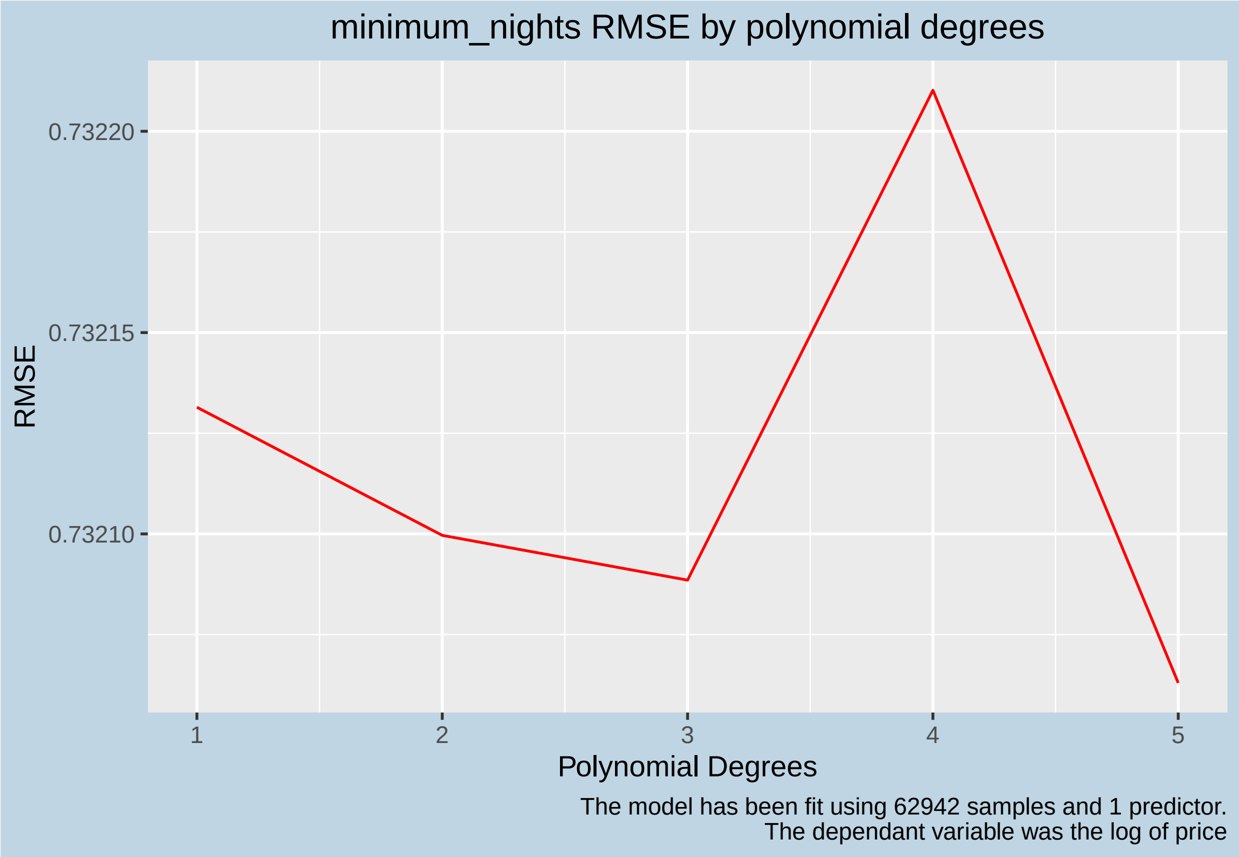
Polynomial Models

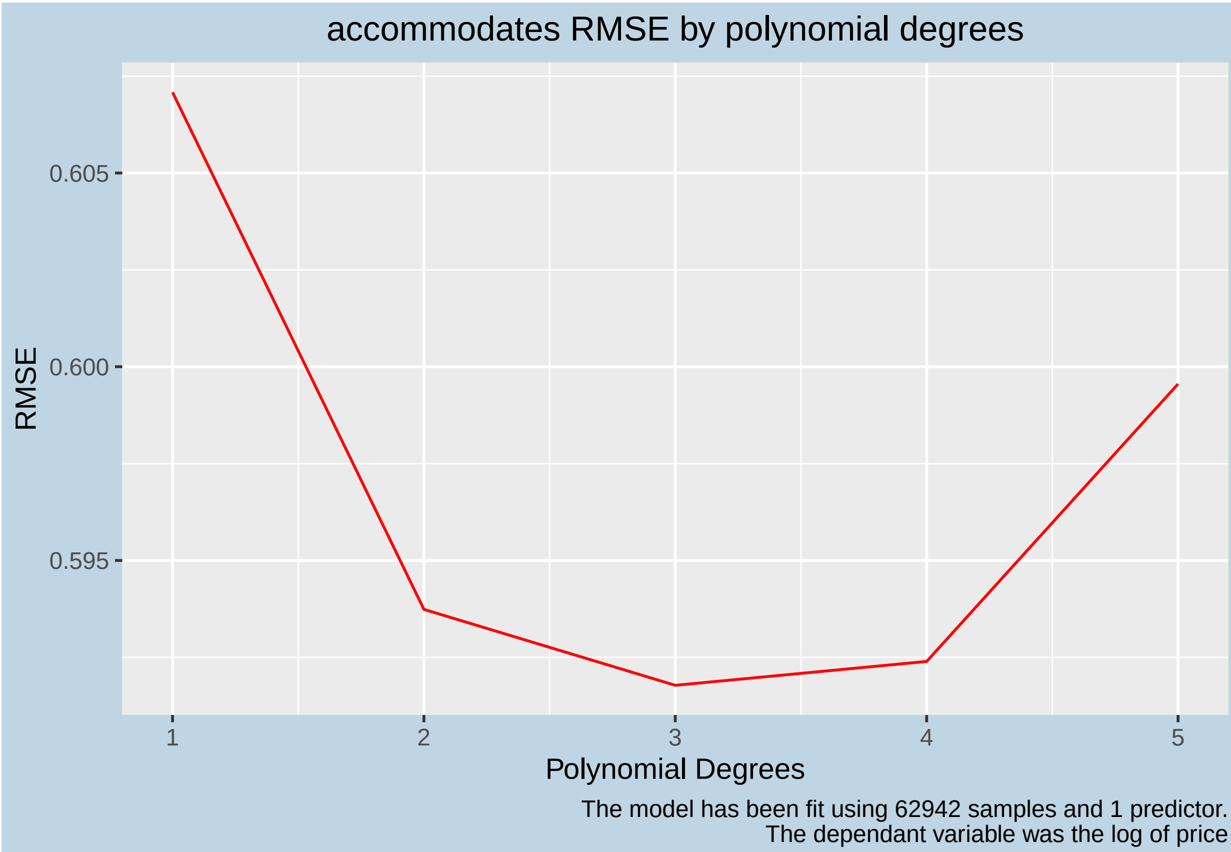
Polynomial plots

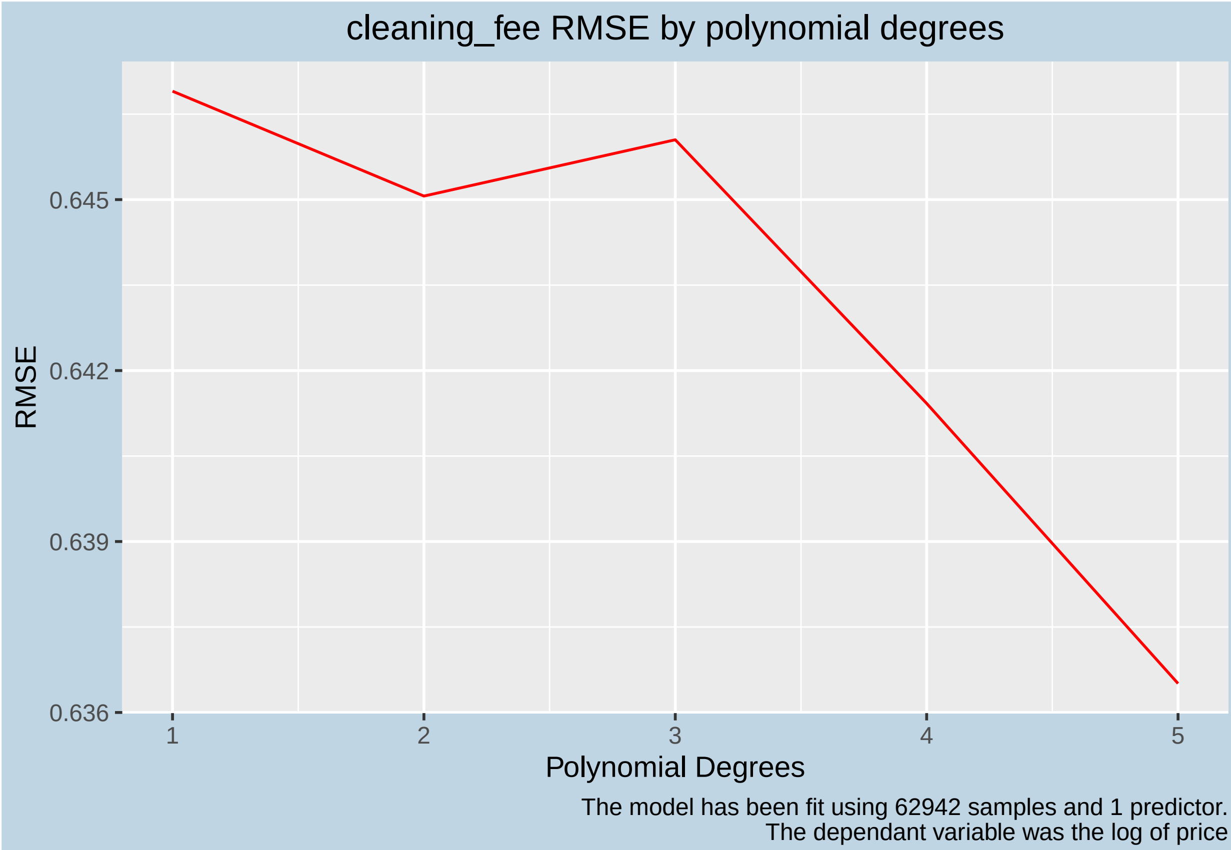
```r
set.seed(777)

# We choose the number of maximum polynomial degree
max_degree <- 5

list <- c("bedrooms","minimum_nights","accommodates",
          "cleaning_fee","availability_365","beds")
```

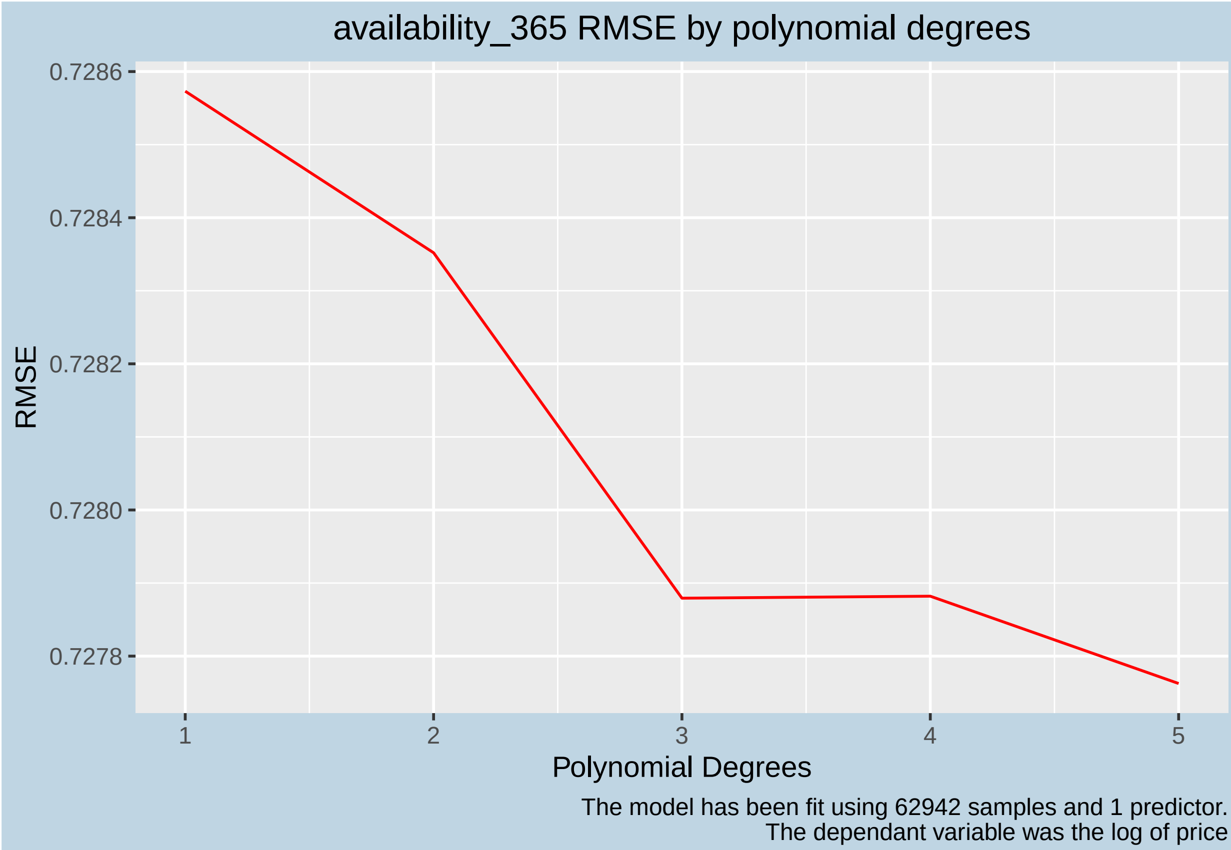
```r
for (l in list){
  RMSE <- rep(0,max_degree)
  for (d in 1:max_degree) {
    PolyRegressor <- train(as.formula(bquote(log_price ~ poly(.(as.name(l)), .(d)))),
                           data = data_train,
                           method = "lm",
                           trControl = K5_CV_seed)
    RMSE[d]<- PolyRegressor$results$RMSE
  }
  tem_data = data.frame(1:max_degree, RMSE)
  plot <- ggplot(aes(x = 1:max_degree,y = RMSE), data = tem_data) +
    geom_line(col="red") + theme_models +
    labs(x = "Polynomial Degrees", y = "RMSE",
         caption = "The model has been fit using 62942 samples and 1 predictor.
         The dependant variable was the log of price") +
    ggtitle(paste(l,"RMSE by polynomial degrees"))
  print(plot)
}
```

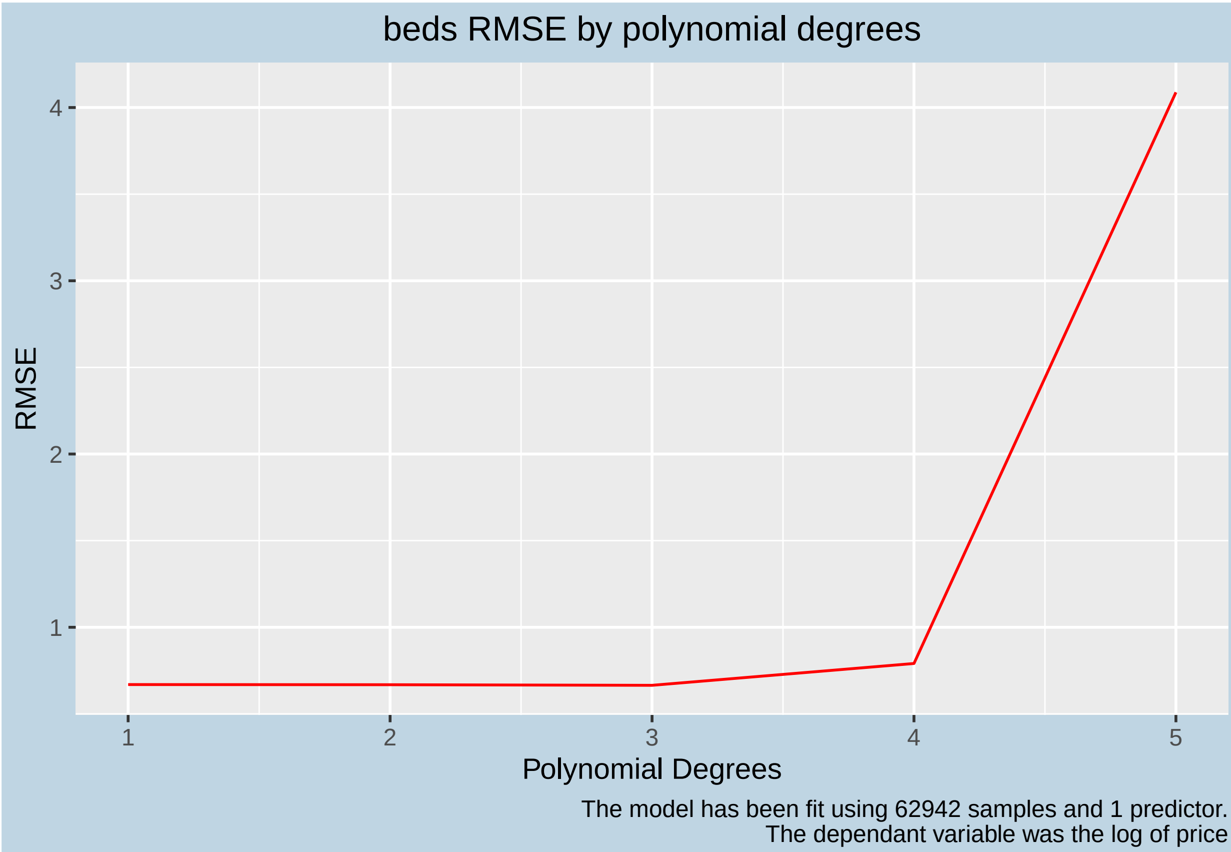



minimum_nights RMSE by polynomial degrees
RMSE
0.73220
0.73215
0.73210
1
2
3
4
5
Polynomial Degrees
The model has been fit using 62942 samples and 1 predictor.
The dependant variable was the log of price


accommodates RMSE by polynomial degrees
RMSE
0.605
0.600
0.595
1
2
3
4
5
Polynomial Degrees
The model has been fit using 62942 samples and 1 predictor.
The dependant variable was the log of price


cleaning_fee RMSE by polynomial degrees
RMSE
0.645
0.642
0.639
0.636
1
2
3
4
5
Polynomial Degrees
The model has been fit using 62942 samples and 1 predictor.
The dependant variable was the log of price


availability_365 RMSE by polynomial degrees
RMSE
0.7286
0.7284
0.7282
0.7280
0.7278
1
2
3
4
5
Polynomial Degrees
The model has been fit using 62942 samples and 1 predictor.
The dependant variable was the log of price



Polynomial Regression

```
set.seed(777)

poly_fit <- train(log_price ~ poly(accommodates,3) +
                    poly(availability_365,3) +
                    poly(bedrooms,4) +
                    poly(cleaning_fee,5) +. -
                    host_is_superhost_False -
                    property_type_SharedRoom -
                    neighbourhood_group_cleansed_Queens -
                    cancellation_policy_moderate -
                    accommodates -
                    availability_365 -
                    bedrooms -
                    cleaning_fee,
                  data = data_train,
                  method = "lm" ,
                  trControl = K5_CV_seed)

summary(poly_fit)
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3662 -0.3077 -0.0224  0.2698  5.2174
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                3.741e+00  4.386e-02  85.297
## `poly(accommodates, 3)1`                   7.230e+01  8.971e-01  80.596
## `poly(accommodates, 3)2`                  -2.752e+01  5.938e-01 -46.354
## `poly(accommodates, 3)3`                   1.585e+01  5.553e-01  28.553
## `poly(availability_365, 3)1`               3.066e+00  5.393e-01   5.685
## `poly(availability_365, 3)2`               4.758e+00  5.129e-01   9.276
## `poly(availability_365, 3)3`               4.287e+00  5.041e-01   8.504
## `poly(bedrooms, 4)1`                       2.066e+01  7.992e-01  25.846
## `poly(bedrooms, 4)2`                       1.170e+01  5.857e-01  19.984
## `poly(bedrooms, 4)3`                      -1.333e+01  5.429e-01 -24.558
## `poly(bedrooms, 4)4`                       1.065e+01  5.529e-01  19.257
## `poly(cleaning_fee, 5)1`                   3.396e+01  6.166e-01  55.072
## `poly(cleaning_fee, 5)2`                  -3.590e+00  5.255e-01  -6.832
## `poly(cleaning_fee, 5)3`                   2.008e+00  5.158e-01   3.893
## `poly(cleaning_fee, 5)4`                   7.498e+00  5.150e-01  14.558
## `poly(cleaning_fee, 5)5`                  -1.105e+01  5.083e-01 -21.737
## host_is_superhost_True                     5.616e-03  5.861e-03   0.958
## neighbourhood_group_cleansed_Bronx        -1.179e-01  1.433e-02  -8.229
## neighbourhood_group_cleansed_Brooklyn      1.279e-01  6.803e-03  18.798
## neighbourhood_group_cleansed_Manhattan     5.034e-01  6.999e-03  71.918
## neighbourhood_group_cleansed_StatenIsland -1.345e-01  2.415e-02  -5.567
## property_type_Appartment                   1.449e-01  3.670e-02   3.949
## property_type_House                        3.228e-02  3.709e-02   0.870
## property_type_PrivateRoom                  5.704e-01  3.900e-02  14.627
## beds                                      -2.525e-02  2.903e-03  -8.700
## minimum_nights                            -1.050e-03  1.057e-04  -9.936
## review_scores_rating                       6.716e-03  2.465e-04  27.244
## cancellation_policy_flexible               4.403e-02  5.778e-03   7.621
## cancellation_policy_strict                -2.508e-02  5.363e-03  -4.677
##                                           Pr(>|t|)
## (Intercept)                                < 2e-16 ***
## `poly(accommodates, 3)1`                   < 2e-16 ***
## `poly(accommodates, 3)2`                   < 2e-16 ***
## `poly(accommodates, 3)3`                   < 2e-16 ***
## `poly(availability_365, 3)1`              1.32e-08 ***
## `poly(availability_365, 3)2`               < 2e-16 ***
## `poly(availability_365, 3)3`               < 2e-16 ***
## `poly(bedrooms, 4)1`                       < 2e-16 ***
## `poly(bedrooms, 4)2`                       < 2e-16 ***
## `poly(bedrooms, 4)3`                       < 2e-16 ***
## `poly(bedrooms, 4)4`                       < 2e-16 ***
## `poly(cleaning_fee, 5)1`                   < 2e-16 ***
## `poly(cleaning_fee, 5)2`                  8.43e-12 ***
## `poly(cleaning_fee, 5)3`                  9.91e-05 ***
## `poly(cleaning_fee, 5)4`                   < 2e-16 ***
## `poly(cleaning_fee, 5)5`                   < 2e-16 ***
## host_is_superhost_True                       0.338
```

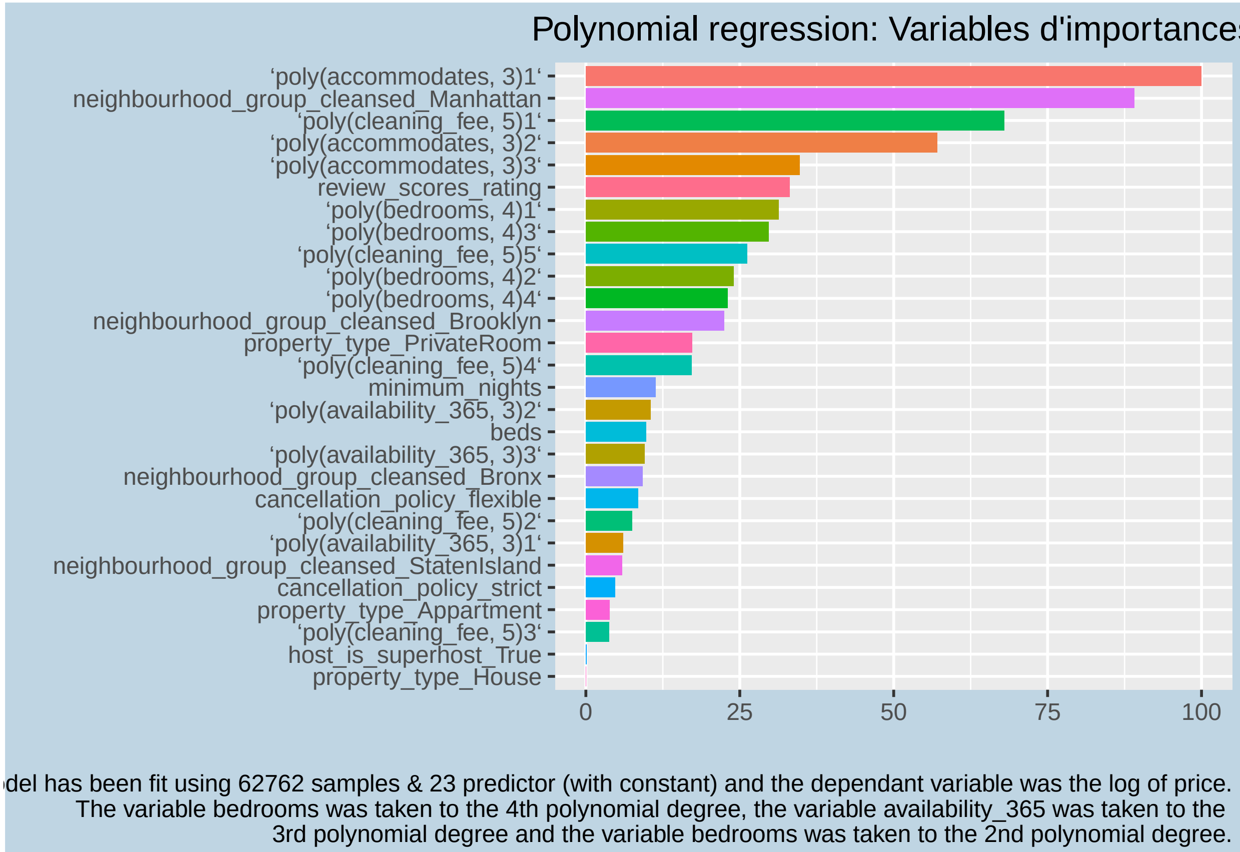
```
## neighbourhood_group_cleansed_Bronx         < 2e-16 ***
## neighbourhood_group_cleansed_Brooklyn      < 2e-16 ***
## neighbourhood_group_cleansed_Manhattan     < 2e-16 ***
## neighbourhood_group_cleansed_StatenIsland 2.61e-08 ***
## property_type_Appartment                  7.85e-05 ***
## property_type_House                          0.384
## property_type_PrivateRoom                  < 2e-16 ***
## beds                                       < 2e-16 ***
## minimum_nights                             < 2e-16 ***
## review_scores_rating                       < 2e-16 ***
## cancellation_policy_flexible              2.55e-14 ***
## cancellation_policy_strict                2.92e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5012 on 62733 degrees of freedom
## Multiple R-squared:  0.5317, Adjusted R-squared:  0.5315
## F-statistic:  2544 on 28 and 62733 DF,  p-value: < 2.2e-16
```

```r
poly_pred <- predict(poly_fit, data_test)

postResample(pred = poly_pred, obs = data_test$log_price)
```

```
##      RMSE  Rsquared       MAE
## 0.4950583 0.5313251 0.3693402
```

```r
# RMSE 0.5016245
```

```r
poly_varImp <- data.frame(variables = row.names(varImp(poly_fit)$importance),
                          varImp(poly_fit)$importance)

ggplot(data = poly_varImp, mapping = aes(x=reorder(variables, Overall),
                                         y=Overall,
                                         fill=variables)) +
  coord_flip() + geom_bar(stat = "identity", position = "dodge") +
  theme_models +
  labs(x = "", y = "",
       caption = "The model has been fit using 62762 samples & 23 predictor (with constant) and the depe
       The variable bedrooms was taken to the 4th polynomial degree, the variable availability_365 was t
       3rd polynomial degree and the variable bedrooms was taken to the 2nd polynomial degree.") +
  ggtitle("Polynomial regression: Variables d'importances")
```

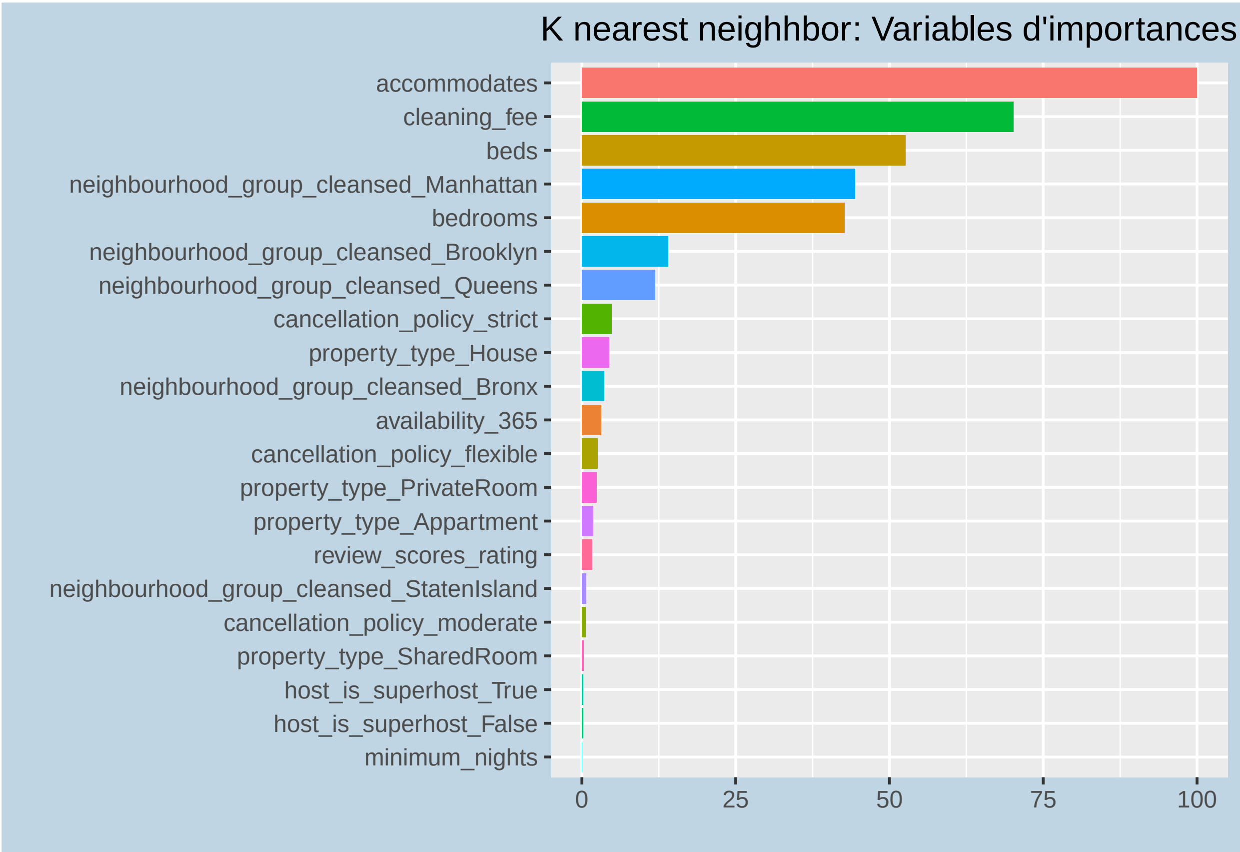


KNN

PreProcess / Standardize

KNN Regression

```
knn_varImp <- data.frame(variables = row.names(varImp(knn_fit)$importance),
                         varImp(knn_fit)$importance)

ggplot(data = knn_varImp, mapping = aes(x=reorder(variables, Overall),
                                        y=Overall,
                                        fill=variables)) +
  coord_flip() + geom_bar(stat = "identity", position = "dodge") +
  theme_models + labs(x = "", y = "") +
  ggtitle("K nearest neighhbor: Variables d'importances")
```



Boosting

XGboost

```
# Grid <- expand.grid(nrounds = 0:50,
#                         max_depth = 5,
#                         eta = 0.3,
#                         gamma = 0.01,
#                         colsample_bytree = 0.5,
#                         min_child_weight = 0,
#                         subsample = 0.5)

Grid <- expand.grid(nrounds = 47,
                    max_depth = 5,
                    eta = 0.3,
                    gamma = 0.01,
                    colsample_bytree = 0.5,
                    min_child_weight = 0,
                    subsample = 0.5)

set.seed(777)
xgboost_fit  <-  train(log_price~.,
                       data = data_train,
                       method = "xgbTree",
```

```r
                        bag.fraction = 0.5,
                        trControl = K5_CV_seed,
                        tuneGrid = Grid,
                        allowParallel = TRUE)

# plot(xgboost_fit)
# xgboost_fit$bestTune

xgboost_pred <- predict(xgboost_fit, data_test)

#RMSE for BoostingTrees
postResample(pred = xgboost_pred, obs = data_test$log_price)
```

```
##      RMSE  Rsquared       MAE
## 0.4524891 0.6083713 0.3340023
```

```r
#      RMSE  Rsquared       MAE
# 0.4524891 0.6083713 0.3340023
```

Random forest

```r
# Grid <- expand.grid(.mtry=c(1:5))
Grid <- expand.grid(.mtry=5)

set.seed(777)
rf_fit  <-  train(log_price ~ .,
                  data = data_train,
                  method = "rf",
                  trControl = K5_CV_seed,
                  tuneGrid = Grid,
                  allowParallel = TRUE)

# plot(rf_fit)
# rf_fit$bestTune

rf_pred <- predict(rf_fit, newdata = data_test)

#RMSE for Random Forest
postResample(pred = rf_pred, obs = data_test$log_price)
```

```
##      RMSE  Rsquared       MAE
## 0.4418751 0.6271526 0.3233835
```

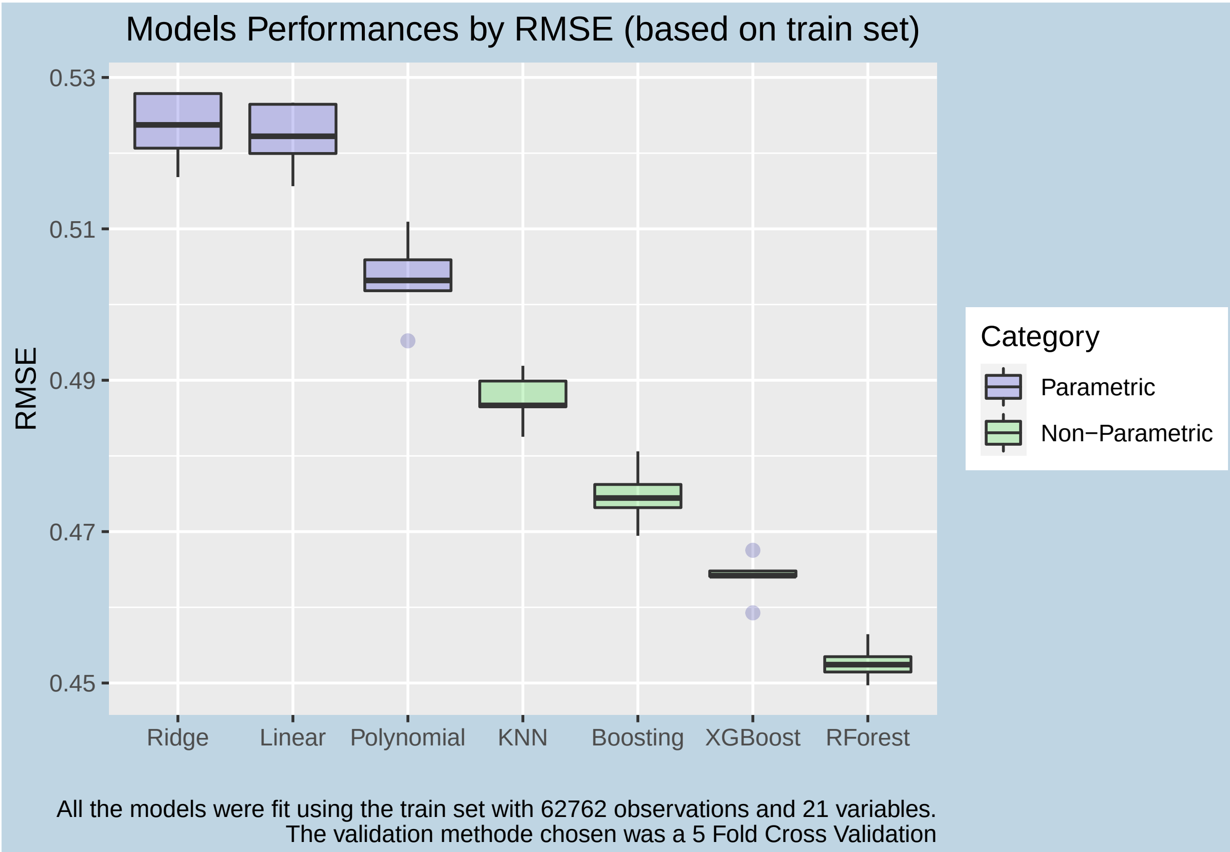
```r
#      RMSE  Rsquared       MAE
# 0.4418751 0.6271526 0.3233835
```

Comparaison of models (Train set)

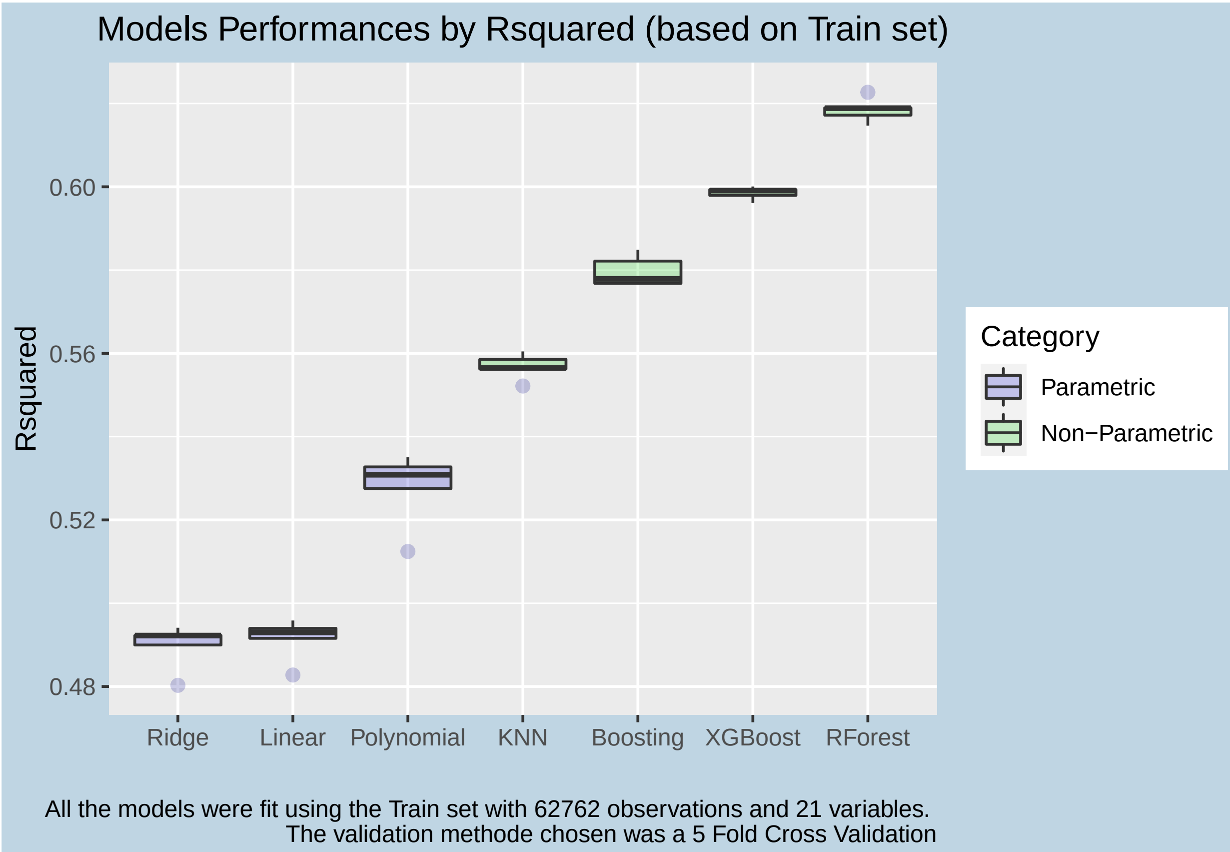
RMSE and R2 based on Train set

```r
models <- list(Linear = ols_fit,
               Ridge = ridge_fit,
```

```r
                Polynomial = poly_fit,
                KNN =  knn_fit,
                Boosting = gbm_fit,
                XGBoost = xgboost_fit,
                RForest = rf_fit)
res <- resamples(models)

models_list = c("Linear", "Ridge", "Polynomial", "KNN",
                "Boosting", "XGBoost", "RForest")

df = data.frame()
for (i in models_list){
  x = data.frame (model = i,
                  RMSE = res$values[paste(i,"RMSE", sep = "~")],
                  Rsquared = res$values[paste(i,"Rsquared", sep = "~")])
  x <- x %>% rename(RMSE = paste(i,".RMSE", sep = ""))
  x <- x %>% rename(Rsquared = paste(i,".Rsquared", sep = ""))
  df <- rbind(df,x)
}

df$Category <- factor(ifelse(df$model %in% c("KNN", "Boosting", "XGBoost",
                                             "RForest"), 1, 0),
                      labels = c("Parametric", "Non-Parametric"))

theme_models2 <- theme(plot.title = element_text(hjust = 0.5),
               plot.background = element_rect(fill = "#BFD5E3"))

# RMSE
ggplot(data = df, aes(x = fct_reorder(model, RMSE, .desc = T),
                      y = RMSE, fill = Category)) +
  geom_boxplot(outlier.colour = "darkblue", outlier.size = 2, alpha=0.2) +
  theme_models2 +
  labs(x = "", y = "RMSE",
       caption = "All the models were fit using the train set with 62762 observations and 21 variables.
       The validation methode chosen was a 5 Fold Cross Validation") +
  ggtitle("Models Performances by RMSE (based on train set)") +
  scale_fill_manual(values=c("blue3", "green3"))
```



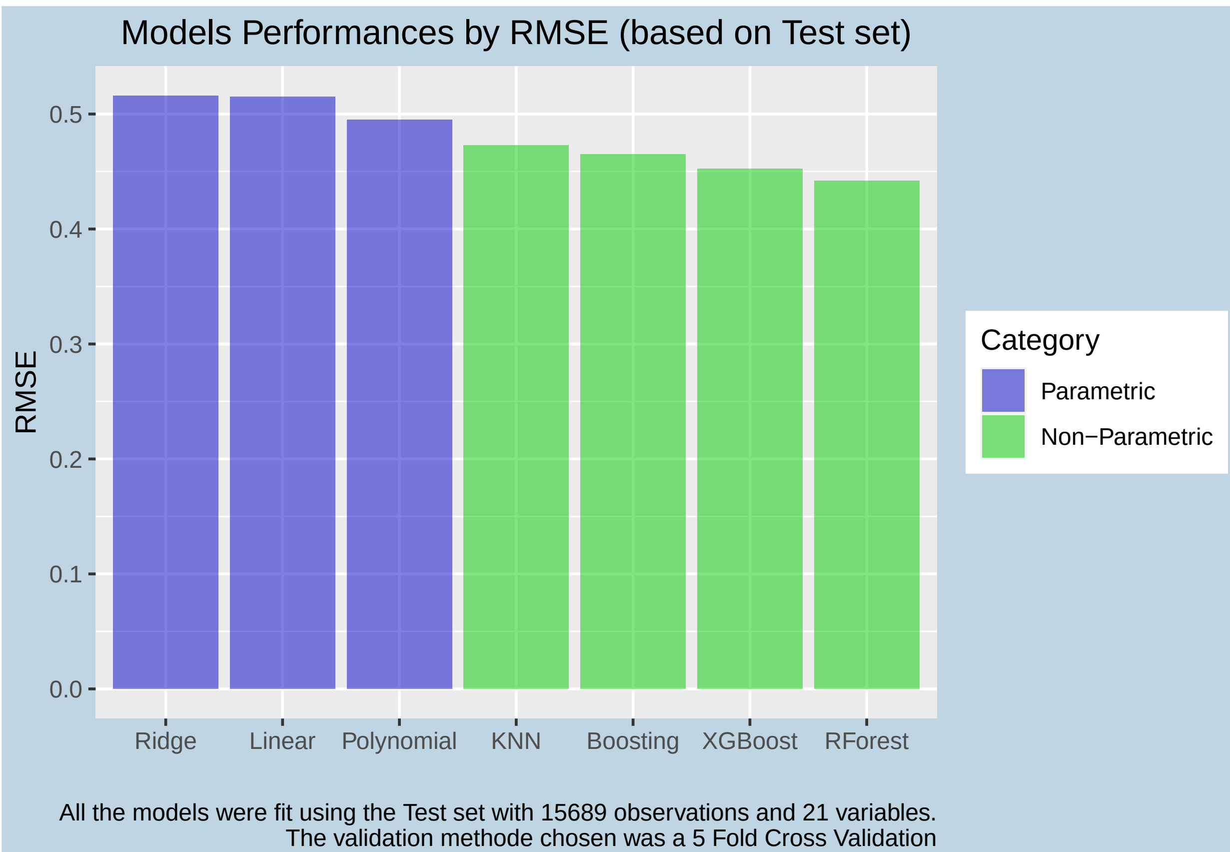

```
# Rsquared
ggplot(data = df, aes(x = fct_reorder(model, Rsquared, .desc = F),
                      y = Rsquared, fill = Category)) +
  geom_boxplot(outlier.colour = "darkblue", outlier.size = 2, alpha=0.2) +
  theme_models2 +
  labs(x = "", y = "Rsquared",
       caption = "All the models were fit using the Train set with 62762 observations and 21 variables.
       The validation methode chosen was a 5 Fold Cross Validation") +
  ggtitle("Models Performances by Rsquared (based on Train set)") +
  scale_fill_manual(values=c("blue3", "green3"))
```



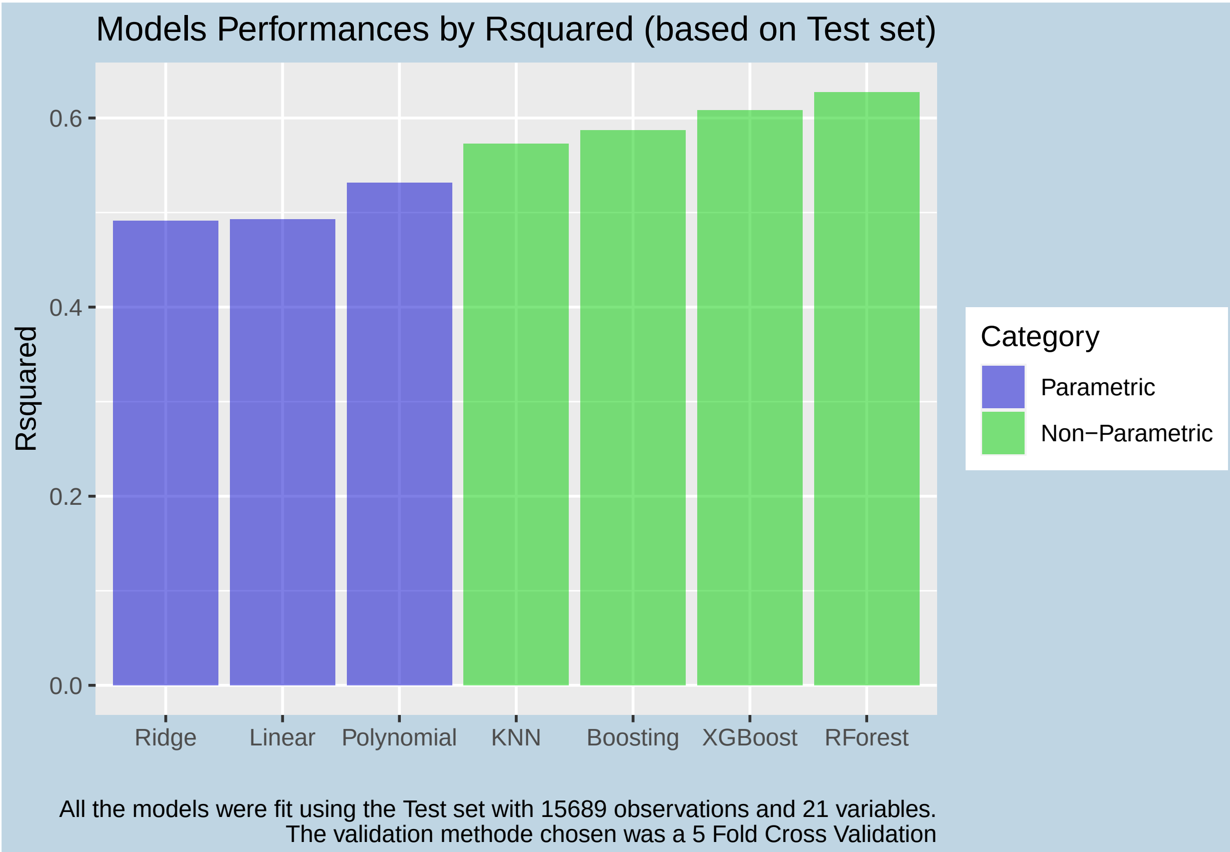
Comparaison of models (Test set)

```r
pred_list = c("ols_pred", "ridge_pred", "poly_pred", "knn_pred",
                "gbm_pred", "xgboost_pred", "rf_pred")

models_list = c("Linear", "Ridge", "Polynomial", "KNN",
                "Boosting", "XGBoost", "RForest")

df = data.frame()
for (i in pred_list){
  if (i == "knn_pred"){
    x = as.data.frame(postResample(pred = get(i),
                                   obs = data_test_standardized$log_price))
    names(x)[1] <- i
    x <- t(x)
  }
  else {
    x = as.data.frame(postResample(pred = get(i), obs = data_test$log_price))
    names(x)[1] <- i
    x <- t(x)
  }
  df <- rbind(df,x)
}
```

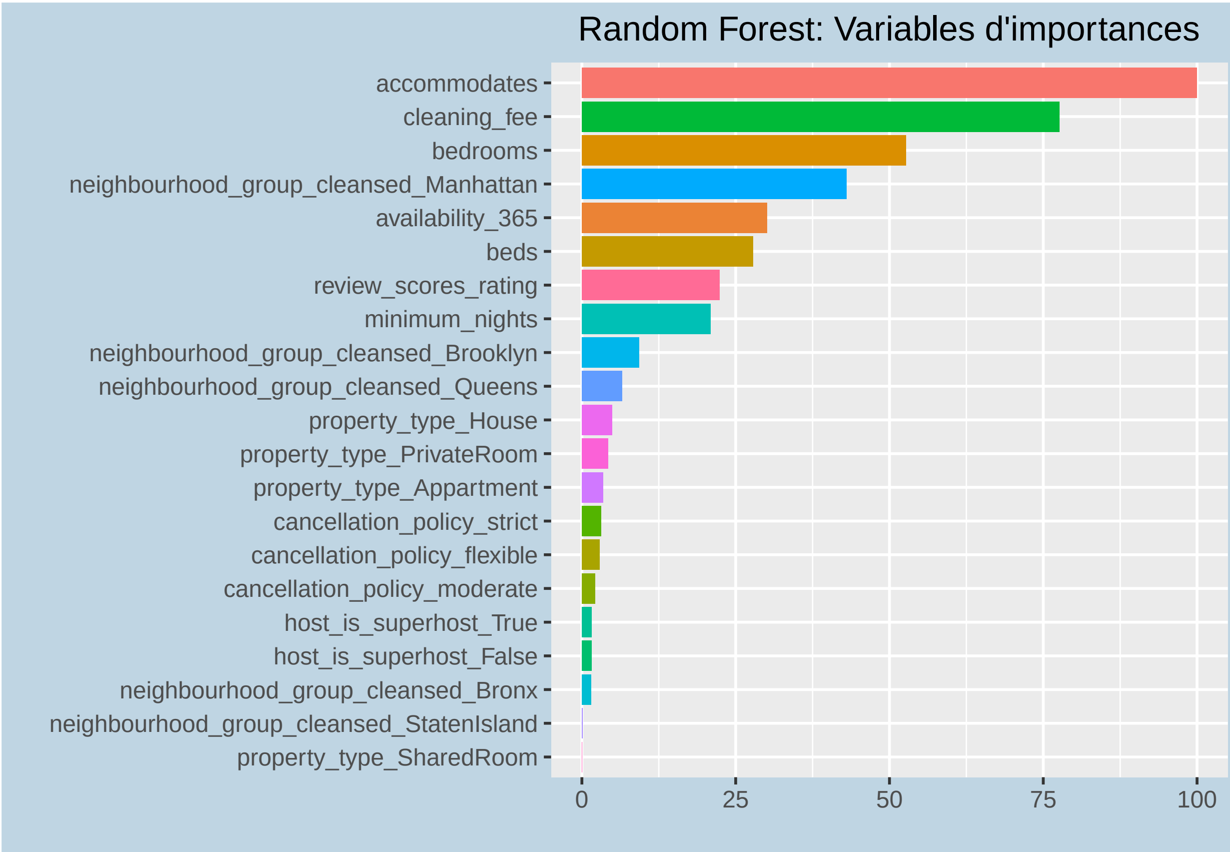
```r
df$model <- models_list

df$Category <- factor(ifelse(df$model %in% c("KNN", "Boosting", "XGBoost",
                                             "RForest"), 1, 0),
                      labels = c("Parametric", "Non-Parametric"))

# RMSE
ggplot(data = df, aes(x = fct_reorder(model, RMSE, .desc = T),
                      y = RMSE, fill = Category)) +
  geom_bar(stat = "identity", alpha=0.5) + theme_models2 +
  labs(x = "", y = "RMSE",
       caption = "All the models were fit using the Test set with 15689 observations and 21 variables.
       The validation methode chosen was a 5 Fold Cross Validation") +
  ggtitle("Models Performances by RMSE (based on Test set)") +
  scale_fill_manual(values=c("blue3", "green3"))
```




```r
# Rsquared
ggplot(data = df, aes(x = fct_reorder(model, Rsquared, .desc = F),
                      y = Rsquared, fill = Category)) +
  geom_bar(stat = "identity", alpha=0.5) + theme_models2 +
  labs(x = "", y = "Rsquared",
       caption = "All the models were fit using the Test set with 15689 observations and 21 variables.
       The validation methode chosen was a 5 Fold Cross Validation") +
  ggtitle("Models Performances by Rsquared (based on Test set)") +
  scale_fill_manual(values=c("blue3", "green3"))
```




```
rf_varImp <- data.frame(variables = row.names(varImp(rf_fit)$importance),
                        varImp(rf_fit)$importance)

ggplot(data = rf_varImp, mapping = aes(x=reorder(variables, Overall),
                                       y=Overall,
                                       fill=variables)) +
  coord_flip() + geom_bar(stat = "identity", position = "dodge") +
  theme(plot.title = element_text(hjust = 0.5),
             plot.background = element_rect(fill = "#BFD5E3"),
             legend.position = "none") +
  labs(x = "", y = "") +
  ggtitle("Random Forest: Variables d'importances")
```



Random Forest: Variables d'importances
accommodates
cleaning_fee
bedrooms
neighbourhood_group_cleansed_Manhattan
availability_365
beds
review_scores_rating
minimum_nights
neighbourhood_group_cleansed_Brooklyn
neighbourhood_group_cleansed_Queens
property_type_House
property_type_PrivateRoom
property_type_Appartment
cancellation_policy_strict
cancellation_policy_flexible
cancellation_policy_moderate
host_is_superhost_True
host_is_superhost_False
neighbourhood_group_cleansed_Bronx
neighbourhood_group_cleansed_StatenIsland
property_type_SharedRoom
0
25
50
75
100