Data Processing

Contents

Overview

In this script the ESC data is processed and cleaned for the data modeling stage. This will incorporate the following: 1. Redefine variables as factor or numeric 2. Dividing the variables into the three predefined groups; i. Performance ii. External iii. Competition 3. Normalize the Numeric Data to have mean 0 and standard deviation 1 4. Dummy Encoding all Categorical Factor levels. 5. Data Reduction i. Redundant Variables ii. Variables of Linear Combinations iii. Categorical Variables via Chi-Squared Tests of Association

It is also possible to: 1. Derive Interaction Terms 2. Derive Polynomial Terms However, it may be easier and more efficient to define these individually during the model building stage.

```r
# load required libraries
library(rmarkdown)
library(knitr)
library(ggplot2)
library(dplyr)
# load in custom utility functions
source("./scripts/utilities/column_to_factor.R")
source("./scripts/utilities/TC_FC_missing_observations.R")
source("./scripts/utilities/plot_missings_effect.R")
source("./scripts/utilities/extract_numeric_data.R")
source("./scripts/utilities/extract_factor_data.R")
source("./scripts/utilities/categorical_dummy_encoding.R")
source("./scripts/utilities/range_standardise_data.R")
source("./scripts/utilities/normalise_data.R")
source("./scripts/utilities/range_normalisation.R")
source("./scripts/utilities/data_normalisation.R")
source("./scripts/utilities/data_reduction_chisq.R")
source("./scripts/utilities/data_reduction_corr.R")
```

```
# load in the raw ESC 2016 data for the analysis
ESCdata <- read.csv(file = "./data/ESC_2016_voting_data.csv", header = T)
# order the data by To_country, From_country, Round and OOA
ESCdata <- ESCdata %>% arrange(To_country, From_country, Round, OOA)
# drop the id variable
ESCdata <- ESCdata %>% select(-c(id))
```

Redefine Factor Variables

Some of the numeric music features need to be redefined as nominal variables, the variables are key, mode and time signature.

```
# define the columns to be converted to factor variables
to_factor_cols <- c('key', 'mode', 'time_signature', 'VBlocs1_FC', 'VBlocs2_FC', 'VBlocs1_TC', 'VBlocs2
# call the column to factor function
ESCdata <- column_to_factor(dataset = ESCdata, col_names = to_factor_cols)
```

Remove Missing Observations

There is a substantial amount of data being lost in this section. This is by far the biggest limitation in the research, and definitely a possible area of improvement for future research. Could use an alternative source such as the world bank, however these do not store the data based on country of birth / citizenship.

http://www.worldbank.org/en/topic/migrationremittancesdiasporaissues/brief/migration-remittances-data

```
# NA values per column
na_count_per_column <- sapply(ESCdata, function(y) sum(length(which(is.na(y)))) / nrow(ESCdata))
# write to a csv file
write.csv(na_count_per_column, './report/stats/NA_prop_per_columns.csv')
# there are 1022 rows with missing data values
nrow(ESCdata) - nrow(ESCdata %>% na.omit)
```

```
## [1] 1022
```

```
# This accounts for just over 60% of the data
(nrow(ESCdata) - nrow(ESCdata %>% na.omit)) / nrow(ESCdata) * 100
```

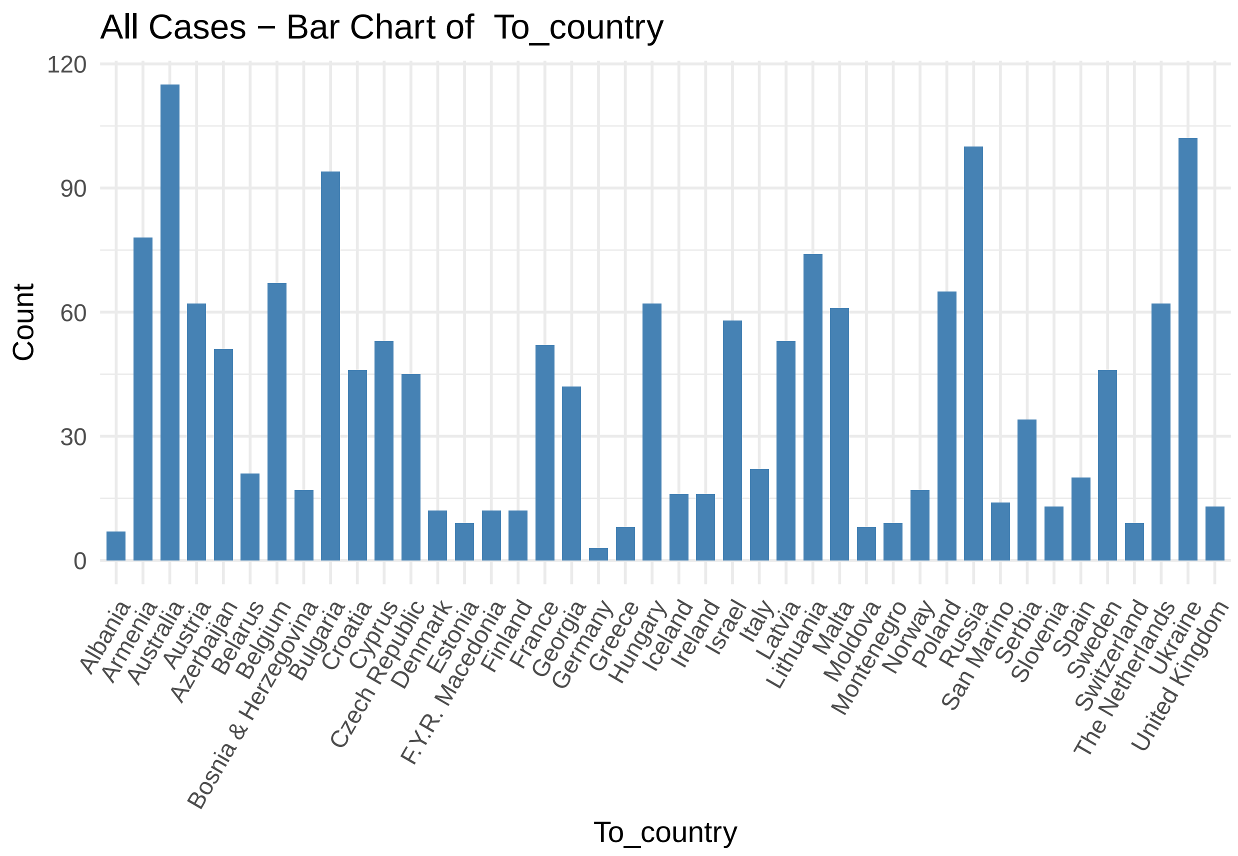
```
## [1] 60.83333
```

```
# filter our rows with missing data values
completedata <- ESCdata %>% na.omit
# have 658 complete observations
nrow(completedata)
```

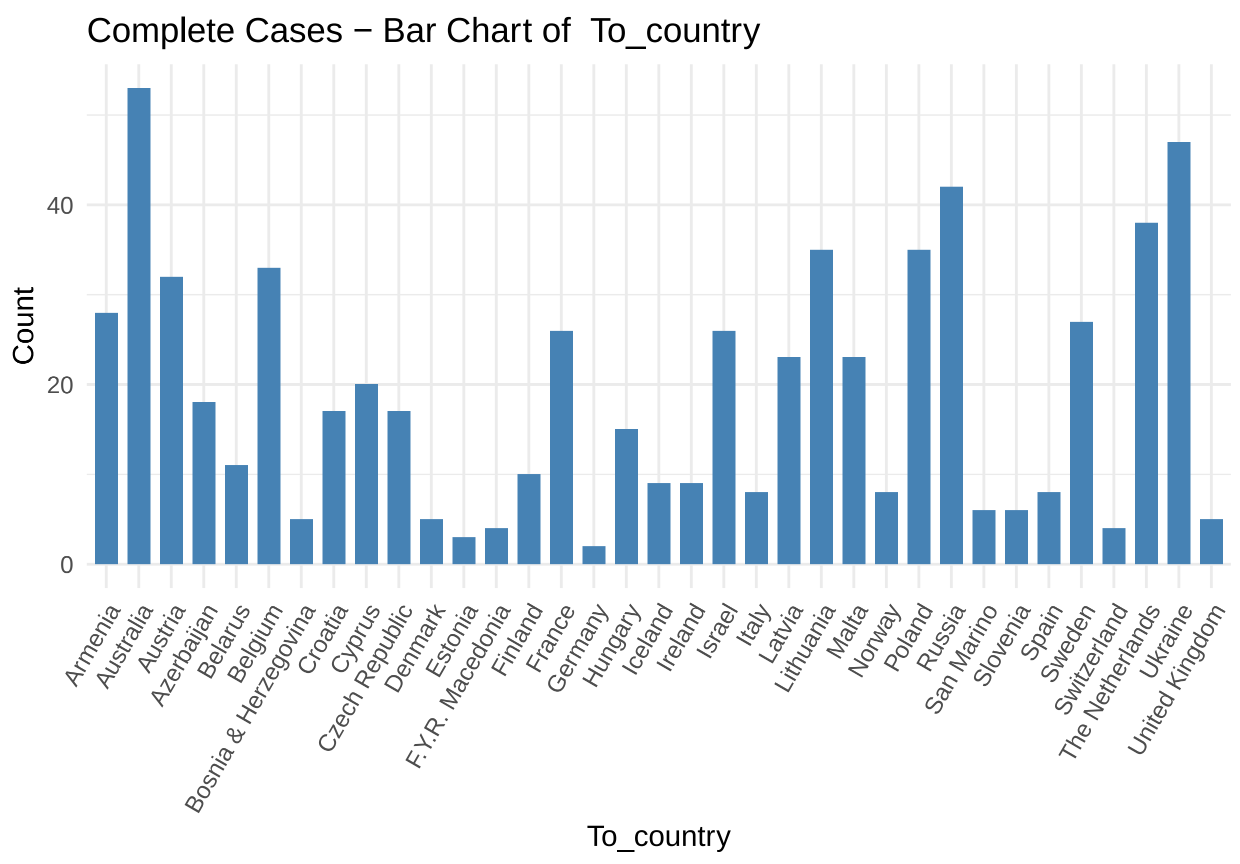
```
## [1] 658
```

```
# extract out unique from countries in order
from_countries <- unique(ESCdata$From_countr) %>% sort()
# generate count stats for the from countries before and after missings are removed
TC_FC_missing_obseration_df <- TC_FC_missing_observations(from_countries = from_countries, orig_data = E
# write the TC_FC_missing_observations_df to a csv file
write.csv(x = TC_FC_missing_obseration_df, file = "./report/stats/TC_FC_missing_obseration_df.csv")

# The Effect on To_country, From_country, Points
voting_factors <- c('To_country', 'From_country', 'Points')
# call the plot missing function
plot_missings_effect(col_name = voting_factors, orig_data = ESCdata, comp_data = completedata, output_d
```

```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.

## Saving 6.5 x 4.5 in image

## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.

## Saving 6.5 x 4.5 in image
```

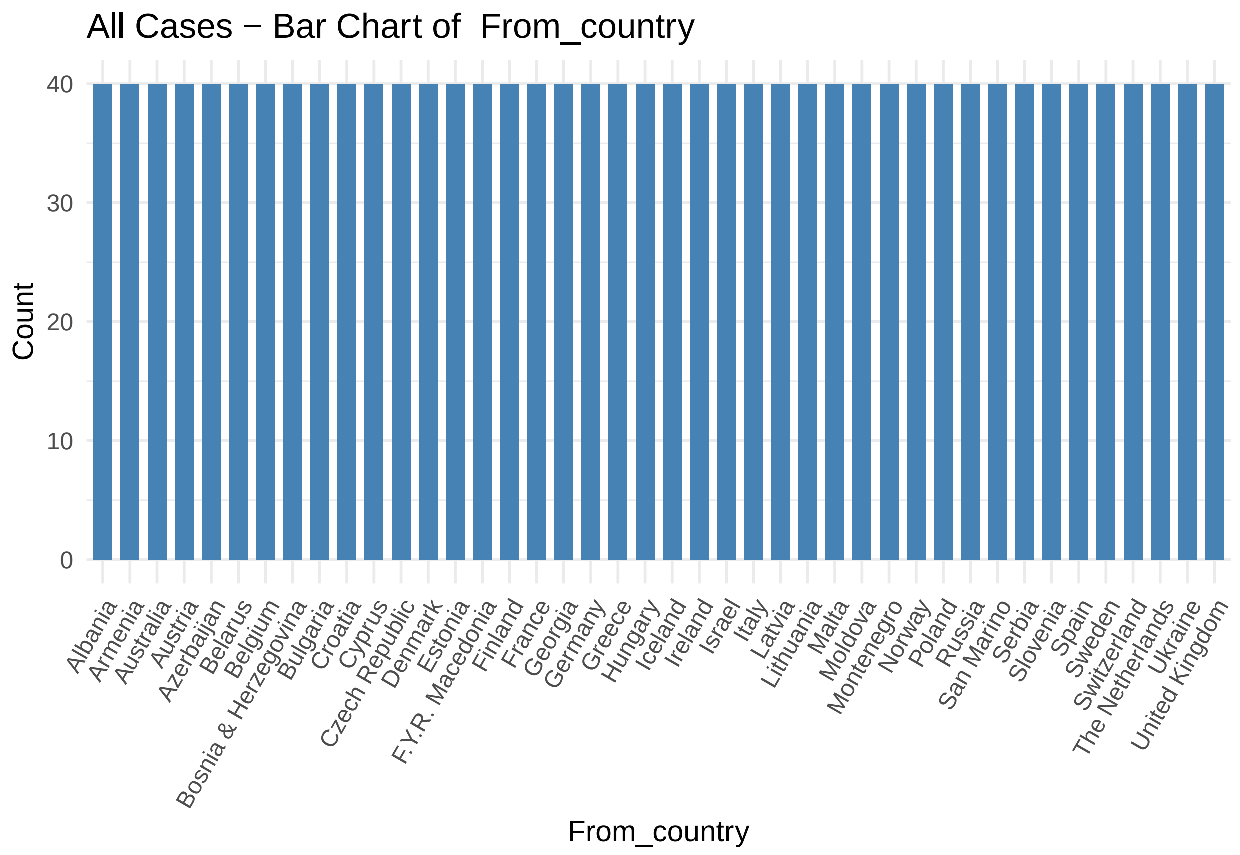


```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.

## Saving 6.5 x 4.5 in image
```



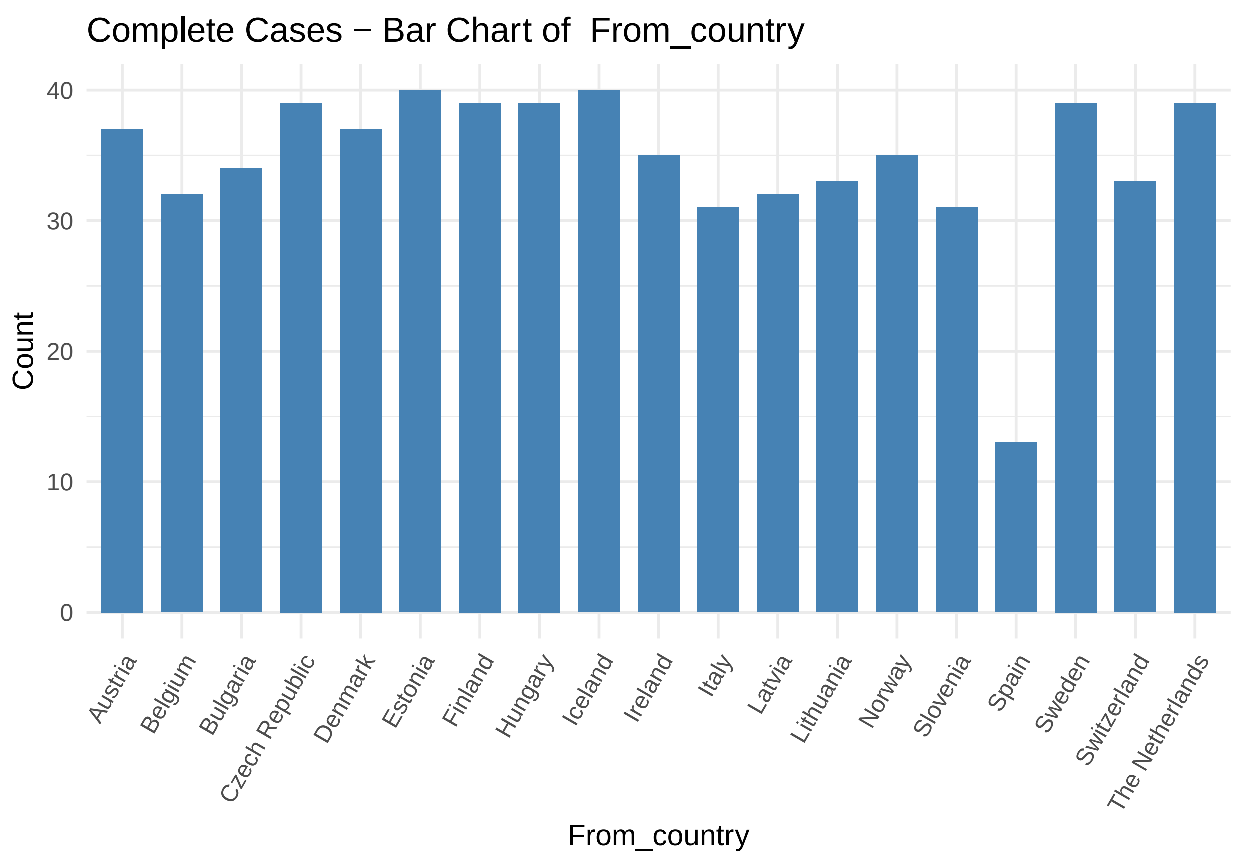
```
## Warning: 'guides(<scale> = FALSE)' is deprecated. Please use 'guides(<scale> =
## "none")' instead.
```

```
## Saving 6.5 x 4.5 in image
```



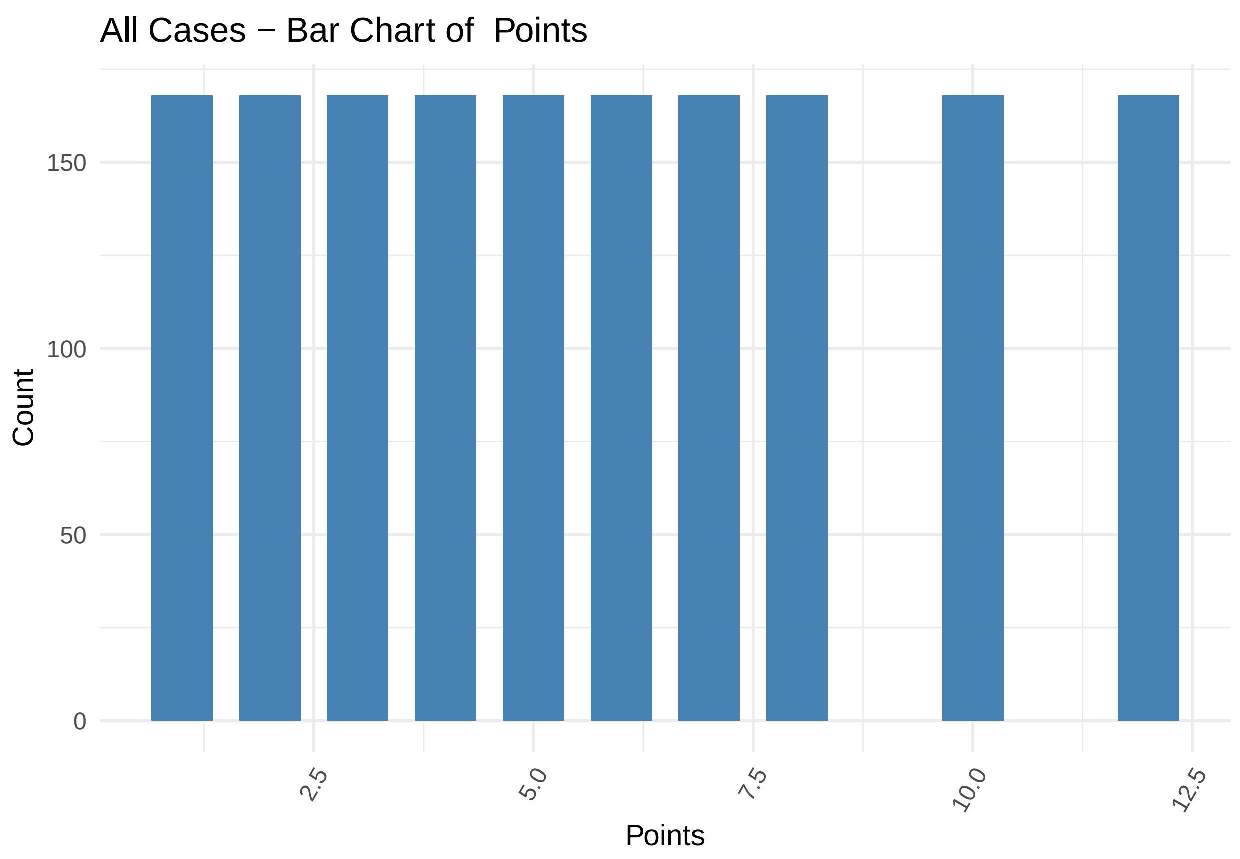
```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.
```

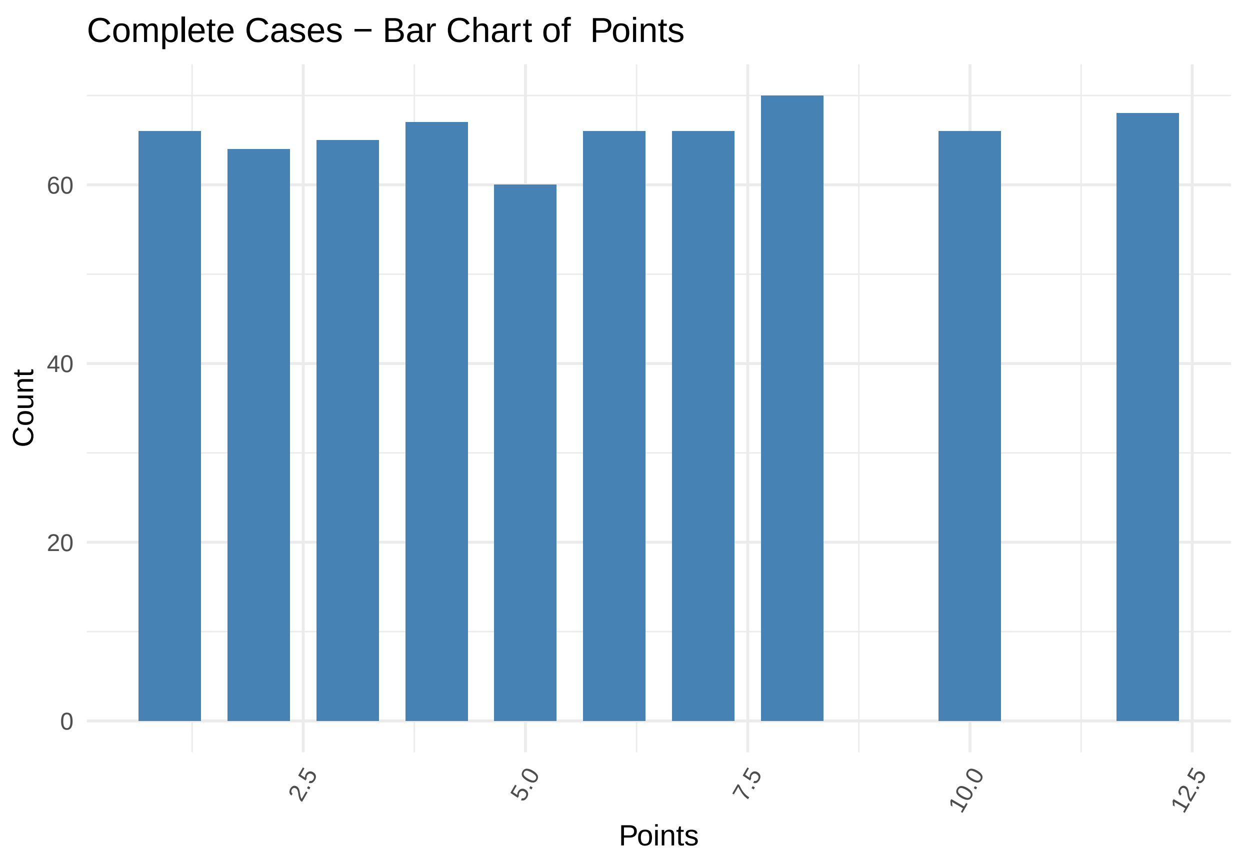
```
## Saving 6.5 x 4.5 in image
```



```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.
```

```
## Saving 6.5 x 4.5 in image
```


All Cases − Bar Chart of Points
Count
150
100
50
0
2.5
5.0
7.5
10.0
12.5
Points



```
## [1] 0
```

Generate Factor Blocs

```
# extract all the voting factors
voting_factors <- completedata %>% subset(select = voting_factors)
# There 3 variables in the voting factors bloc
ncol(voting_factors)
```

```
## [1] 3
```

```
# define the competition factor column names
comp_factors_names = c('Average_Points', 'Round', 'Voting_Method', 'Host_Nation', 'OOA')
# extract the competition factor column names
competition_factors <- completedata %>% subset(select = comp_factors_names)
# There are 5 variables in the competition factors bloc
ncol(competition_factors)
```

```
## [1] 5
```

```
# define the external factor column names
ext_factors_names = c('VBlocs1_FC', 'VBlocs2_FC', 'VBlocs1_TC', 'VBlocs2_TC', 'ComVBlocs1', 'ComVBlocs2
                      'FC_LANGFAM', 'TC_LANGFAM', 'ComLANGFAM', 'Neighbours', 'TC_NumNeigh', 'FC_NonCOB
                      'FC_NonCitzens', 'FC_COB', 'FC_Citizens', 'FC_Population', 'METRIC_COB', 'METRIC_
                      'METRIC_COBCit', 'FC_GDP_mil', 'TC_GDP_mil', 'GDP_PROP', 'FC_CAP_LAT', 'FC_CAP_LO
                      'TC_CAP_LAT', 'TC_CAP_LON', 'CAP_DIST_km')
# extract the external factor column names
external_factors <- completedata %>% subset(select = ext_factors_names)
# There are 27 variables in the external factors bloc
ncol(external_factors)
```

```
## [1] 27
```

```
# define the performance factor column names
perf_factors_names = c('TC_PerfType', 'TC_SingerGender', 'FC_SONGLANG', 'TC_SONGLANG', 'ComSONGLAN',
                       'danceability', 'energy', 'key', 'loudiness', 'mode', 'speechiness', 'acousticne
                       'instrumentalness', 'liveness', 'valence', 'tempo', 'duration_ms', 'time_signatu
# extract the performance factor column names
performance_factors <- completedata %>% subset(select = perf_factors_names)
# There are 18 variables in the performance factors bloc
ncol(performance_factors)
```

```
## [1] 18
```

Extract Numeric & Categorical Features

Objective is to extract the numeric & categorical features for each variable bloc. Two functions are defined to extract the numeric variables and the categorical variables separately

```
# (i) Numeric Variables
competition_factors_numeric <- extract_numeric_data(dataset = competition_factors)
# (ii) Categorical competition Variables
competition_factors_categorical <- extract_factor_data(dataset = competition_factors)
```

```
# (i) Numeric Variables
external_factors_numeric <- extract_numeric_data(dataset = external_factors)
# (ii) Categorical Variables
external_factors_categorical <- extract_factor_data(dataset = external_factors)
```

```
# (i) Numeric Variables
performance_factors_numeric <- extract_numeric_data(dataset = performance_factors)
# (ii) Categorical Variables
performance_factors_categorical <- extract_factor_data(dataset = performance_factors)
```

```
# (i) Numeric Variables
voting_factors_numeric <- extract_numeric_data(dataset = voting_factors)
#  (ii) Categorical Variables
voting_factors_categorical <- extract_factor_data(dataset = voting_factors)
```

Dummy Encoding Categorical Variables

This section dummy encodes the categorical variables for each variable bloc. A dummy encoding function is defined and applied it to each categorical bloc. It is not necessary to dummy encode the voting factors To_country and From_country as we will not be incorporated as model predictor variables

```
# Dummy encode competition factors
competition_factors_categorical <- categorical_dummy_encoding(dataset = competition_factors_categorical
# Dummy encode external factors
external_factors_categorical <- categorical_dummy_encoding(dataset = external_factors_categorical)
# Dummy encode performance factors
performance_factors_categorical <- categorical_dummy_encoding(dataset = performance_factors_categorical
```

Standardise Numeric Data

This section normalizes or range standardizes the numeric variables in each block. There is also a special case for OOA in the competition Bloc. OOA will need to be standardized in relation to each round. This shall be done after the numeric variables from each bloc have been standardized

```
# (1) Competition Factors
competition_factors_numeric <- normalise_data(dataset = competition_factors_numeric)
# Check that each column has mean 0
round(apply(X = competition_factors_numeric, MARGIN = 2, FUN = mean), digits = 6)
```

```
## Average_Points            OOA
##              0              0
```

```
# Check that each column has standard deviation 1
round(apply(X = competition_factors_numeric, MARGIN = 2, FUN = sd), digits = 6)
```

```
## Average_Points            OOA
##              1              1
```

```
# (2) External Factors
external_factors_numeric <- normalise_data(dataset = external_factors_numeric)
# Check that each column has mean 0
round(apply(X = external_factors_numeric, MARGIN = 2, FUN = mean), digits = 6)
```

```
##     TC_NumNeigh       FC_NonCOB   FC_NonCitzens          FC_COB     FC_Citizens
##               0               0               0               0               0
##   FC_Population      METRIC_COB METRIC_Citizens   METRIC_COBCit      FC_GDP_mil
##               0               0               0               0               0
##      TC_GDP_mil        GDP_PROP      FC_CAP_LAT      FC_CAP_LON      TC_CAP_LAT
##               0               0               0               0               0
##      TC_CAP_LON     CAP_DIST_km
##               0               0
```

```
# Check that each column has standard deviation 1
round(apply(X = external_factors_numeric, MARGIN = 2, FUN = sd), digits = 6)
```

```
##      TC_NumNeigh        FC_NonCOB    FC_NonCitzens           FC_COB      FC_Citizens
##                1                1                1                1                1
##    FC_Population       METRIC_COB METRIC_Citizens    METRIC_COBCit       FC_GDP_mil
##                1                1                1                1                1
##       TC_GDP_mil         GDP_PROP       FC_CAP_LAT       FC_CAP_LON       TC_CAP_LAT
##                1                1                1                1                1
##       TC_CAP_LON      CAP_DIST_km
##                1                1
```

```
# (3) Performance Factors
performance_factors_numeric <- normalise_data(dataset = performance_factors_numeric)
# Check that each column has mean 0
round(apply(X = performance_factors_numeric, MARGIN = 2, FUN = mean), digits = 6)
```

```
##        ComSONGLAN     danceability           energy         loudiness
##                 0                0                0                 0
##       speechiness     acousticness instrumentalness          liveness
##                 0                0                0                 0
##           valence            tempo      duration_ms
##                 0                0                0
```

```
# Check that each column has standard deviation 1
round(apply(X = performance_factors_numeric, MARGIN = 2, FUN = sd), digits = 6)
```

```
##        ComSONGLAN     danceability           energy         loudiness
##                 1                1                1                 1
##       speechiness     acousticness instrumentalness          liveness
##                 1                1                1                 1
##           valence            tempo      duration_ms
##                 1                1                1
```

```
#-- OOA --#

# Extract the relevant features
OOA_df <- completedata %>% subset(select = c('From_country', 'To_country', 'Round', 'OOA'))
# Order the Features
OOA_df <- OOA_df %>% arrange(Round, OOA, To_country, From_country)
# check head of results
head(OOA_df)
```

From_country	To_country	Round	OOA
Austria	Belgium	f	1
Bulgaria	Belgium	f	1
Czech Republic	Belgium	f	1
Denmark	Belgium	f	1
Denmark	Belgium	f	1
Iceland	Belgium	f	1

```
# Apply the range standardization
OOA_df[OOA_df$Round == "f", 4] <- range_normalisation(dataset = OOA_df[OOA_df$Round == "f", 4], lb = 0,
OOA_df[OOA_df$Round == "sf1", 4] <- range_normalisation(dataset = OOA_df[OOA_df$Round == "sf1", 4], lb =
OOA_df[OOA_df$Round == "sf2", 4] <- range_normalisation(dataset = OOA_df[OOA_df$Round == "sf2", 4], lb =
head(OOA_df)
```

From_country	To_country	Round	OOA
Austria	Belgium	f	0
Bulgaria	Belgium	f	0
Czech Republic	Belgium	f	0
Denmark	Belgium	f	0
Denmark	Belgium	f	0
Iceland	Belgium	f	0

```
# Re-Order the data frame
OOA_df <- OOA_df %>% arrange(To_country, From_country, Round, OOA)
# check the results
head(OOA_df)
```

From_country	To_country	Round	OOA
Austria	Armenia	f	1.0000000
Austria	Armenia	sf1	0.3529412
Austria	Armenia	sf1	0.3529412
Belgium	Armenia	f	1.0000000
Bulgaria	Armenia	f	1.0000000
Bulgaria	Armenia	f	1.0000000

```
# compare with original
head(completedata %>% subset(select = c('From_country', 'To_country', 'Round', 'OOA')))
```

	From_country	To_country	Round	OOA
9	Austria	Armenia	f	26
10	Austria	Armenia	sf1	7
11	Austria	Armenia	sf1	7
14	Belgium	Armenia	f	26
18	Bulgaria	Armenia	f	26
19	Bulgaria	Armenia	f	26

```
# check target to overwrite
head(competition_factors_numeric)
```

	Average_Points	OOA
9	1.339881	2.2137693
10	1.339881	-0.6879442
11	1.339881	-0.6879442

	Average_Points	OOA
14	1.470934	2.2137693
18	1.034092	2.2137693
19	1.034092	2.2137693

```r
# Overwrite the OOA in competition_factors_numeric with the new standardized OOA
competition_factors_numeric[,"OOA"] <- OOA_df[,"OOA"]
```

Data Reduction

This section performs data reduction whereby Categorical / Numeric variables are removed if: 1. They have only a single type of observation, 0 or 1 2. They form part of a linear combination with other variables 3. They are strongly associated / correlated with other variables

```r
# There are 13 categorical variables that have only 0 observations
zero_ext_fact_cats <- apply(X = external_factors_categorical, MARGIN = 2, FUN = sum) == 0
# extract out the columns with all zero values
zero_ext_fact_cats_cols <- names(which(zero_ext_fact_cats))
# find the columns which do not have all zero values
non_zero_ext_fact_cols <- colnames(external_factors_categorical)[!(colnames(external_factors_categorica
# determine how many columns have zero values
length(zero_ext_fact_cats_cols)
```

```
## [1] 13
```

```r
# extract the external categorical features
external_factors_categorical <- external_factors_categorical %>% subset(select = non_zero_ext_fact_cols
```

```r
# There are no competition categorical variables with only 0 observations
sum(apply(X = competition_factors_categorical, MARGIN = 2, FUN = sum) == 0)
```

```
## [1] 0
```

```r
# There are 0 performance categorical variables with only 0 observations
sum(apply(X = performance_factors_categorical,  MARGIN = 2, FUN = sum) == 0)
```

```
## [1] 0
```

If two variables are feature a lot of 0s or 1s then there will be a strong association between the two variables as they share a lot of common observations. In such cases, one of the variables can be removed as they both measure the same entity. This lowers the chance of collinearity and reduces the number of dimensions

```r
# extract out the relevant voting blocks for each county
FC_VBlocs1_facts <- external_factors_categorical %>% select(starts_with('VBlocs1_FC')) %>% colnames()
FC_VBlocs2_facts <- external_factors_categorical %>% select(starts_with('VBlocs2_FC')) %>% colnames()
TC_VBlocs1_facts <- external_factors_categorical %>% select(starts_with('VBlocs1_TC')) %>% colnames()
TC_VBlocs2_facts <- external_factors_categorical %>% select(starts_with('VBlocs2_TC')) %>% colnames()
FC_LANGFAM_facts <- external_factors_categorical %>% select(starts_with('FC_LANGFAM')) %>% colnames()
TC_LANGFAM_facts <- external_factors_categorical %>% select(starts_with('TC_LANGFAM')) %>% colnames()
```

```
# extract out the column names for significant columns
chisq_tests_FC_VBlocs1_cols <- data_reduction_chisq(dataset = external_factors_categorical, col_names =
chisq_tests_FC_VBlocs2_cols <- data_reduction_chisq(dataset = external_factors_categorical, col_names =
chisq_tests_TC_VBlocs1_cols <- data_reduction_chisq(dataset = external_factors_categorical, col_names =
chisq_tests_TC_VBlocs2_cols <- data_reduction_chisq(dataset = external_factors_categorical, col_names =
chisq_tests_FC_LANGFAM_cols <- data_reduction_chisq(dataset = external_factors_categorical, col_names =
chisq_tests_TC_LANGFAM_cols <- data_reduction_chisq(dataset = external_factors_categorical, col_names =
# concatenate all columns to be removed
remove_vblocs_cols <- c(chisq_tests_FC_VBlocs1_cols, chisq_tests_FC_VBlocs2_cols, chisq_tests_TC_VBlocs
remove_langfam_cols <- c(chisq_tests_FC_LANGFAM_cols, chisq_tests_TC_LANGFAM_cols)
remove_ext_cols <- c(remove_vblocs_cols, remove_langfam_cols)
# extract all the external factors categorical variables
external_factors_categorical_cols <- colnames(external_factors_categorical)
# determine the columns to keep
keep_ext_cols <- external_factors_categorical_cols[!(external_factors_categorical_cols %in% remove_ext_
# Remove Unnecessary Categorical Variables
external_factors_categorical <- external_factors_categorical %>% subset(select = keep_ext_cols)
```

```
# extract out the relevant from country and to country song language
FC_SONGLANG_facts <- performance_factors_categorical %>% select(starts_with('FC_SONGLANG')) %>% colname
TC_SONGLANG_facts <- performance_factors_categorical %>% select(starts_with('TC_SONGLANG')) %>% colname
keys_fact <- performance_factors_categorical %>% select(starts_with('key_')) %>% colnames()
# extract out the column names for significant columns
chisq_tests_FC_SONGLANG_cols <- data_reduction_chisq(dataset = performance_factors_categorical, col_nam
chisq_tests_TC_SONGLANG_cols <- data_reduction_chisq(dataset = performance_factors_categorical, col_nam
chisq_tests_keys_cols <- data_reduction_chisq(dataset = performance_factors_categorical, col_names = ke
# concatenate all columns to be removed
remove_songlang_cols <- c(chisq_tests_FC_SONGLANG_cols, chisq_tests_TC_SONGLANG_cols)
remove_perf_cols <- c(remove_songlang_cols, chisq_tests_keys_cols)
# extract all the external factors categorical variables
performance_factors_categorical_cols <- colnames(performance_factors_categorical)
# determine the columns to keep
keep_perf_cols <- performance_factors_categorical_cols[!(performance_factors_categorical_cols %in% remo
# Remove Unnecessary Categorical Variables
performance_factors_categorical <- performance_factors_categorical %>% subset(select = keep_perf_cols)
```

If two numeric variables are very highly correlated and represent the same entity then it is unnecessary to include them in the data modeling stage. This is particular the case for the migration data. The correlation test function implement is the same as the one that was used during the exploratory analysis section.

```
# Perform the correlation tests
mig_nums = c('FC_NonCOB', 'FC_NonCitzens', 'FC_COB', 'FC_Citizens', 'FC_Population', 'METRIC_COB','METR
# extract out the column names for significant columns
corr_tests_mig_cols <- data_reduction_corr(dataset = external_factors_numeric, col_names = mig_nums)
# extract all the external factors categorical variables
external_factors_numeric_cols <- colnames(external_factors_numeric)
# determine the columns to keep
keep_ext_num_cols <- external_factors_numeric_cols[!(external_factors_numeric_cols %in% corr_tests_mig_
# Remove Unnecessary Categorical Variables
external_factors_numeric <- external_factors_numeric %>% subset(select = keep_ext_num_cols)
```

Data Output

```
# combine the various factor blocks together
processed_voting_factors <- cbind(voting_factors_categorical, voting_factors_numeric)
processed_competition_factors <- cbind(competition_factors_categorical, competition_factors_numeric)
processed_external_factors <- cbind(external_factors_categorical, external_factors_numeric)
processed_performance_factors <- cbind(performance_factors_categorical, performance_factors_numeric)
# combine and create the final processed data set
processed_data <- cbind(processed_voting_factors, processed_competition_factors, processed_external_fact
# write the processed data to a csv file
write.csv(processed_data, './data/processed_data.csv', row.names = F)
```