Measuring Typological and Goegraphical Distances in R

Yingqi Jing

April 5, 2022

Contents

1 Alternative functions to ifelse & loop in a data.frame

1.1 Generalize `ifelse` to `case_when`

```r
lang_df = data.frame(id = 1:4,
                     lang = c("Estonian", "Finnish", "Hungarian", "North Saami"))
```

1.2 Generalize `for` loop to `rowwise` operation

2 Typological similarities between Uralic languages

2.1 Function

```r
typological_sim = function(data = ut_final, x, y){
  subdata = data %>%
    dplyr::select(-matches("subfamily|area", ignore.case = T)) %>%
    column_to_rownames(var = "Name")
  sim = sum(abs(as.vector(subdata[x, ]) == as.vector(subdata[y, ])))
  sim_p = sim/ncol(subdata)
  return(sim_p)
}
```

2.2 Data preprocessing

```r
uratyp_df = read.csv("../data/values.csv")
lang_df = read.csv("../data/languages.csv")
ut_data = uratyp_df %>%
  inner_join(., lang_df, by = c("Language_ID" = "ID")) %>%
  dplyr::select(Name, Parameter_ID, Value, Subfamily) %>%
  filter(grepl("UT", Parameter_ID))
```

(1) Convert all data into binary (0, 1)

```r
ut_wide = ut_data %>%
  mutate(Value = case_when(
    Value == "0" ~ 0L,
    Value == "1" ~ 1L,
    TRUE ~ NA_integer_ # convert all "?" into NA
  )) %>%
  pivot_wider(., names_from = Parameter_ID, values_from = Value)
```

(2) Remove all columns with missing values

```r
ut_wide = ut_wide %>%
  select_if(function(x) !any(is.na(x)))
# alternatively, select_if(~ !any(is.na(.x)))
# select_if(~ sum(is.na(.x)) == 0)
# select(where(~ sum(is.na(.x)) == 0))
```

(3) Remove all constant columns

```r
ut_final = ut_wide %>%
  remove_constant(.)
  # select_if(~ length(unique(.x)) > 1)
```

(4) Visualize data via heatmap

```
lang_sorted = ut_final %>%
  arrange(Subfamily) %>%
  pull(Name)
ut_final_long = ut_final %>%
  pivot_longer(., names_to = "feature", values_to = "value", -c("Name", "Subfamily")) %>%
  mutate(value = factor(value),
         Name = factor(Name, levels = lang_sorted))
ut_final_long %>%
  ggplot(., aes(feature, Name, fill = value)) +
  geom_tile() +
  scale_fill_manual(values = alpha(c("blue", "red"), 0.65)) +
  theme(axis.text.x = element_text(angle = 90, size = 6, hjust = 0),
        axis.ticks = element_blank()) +
  labs(x = NULL, y = NULL)
```
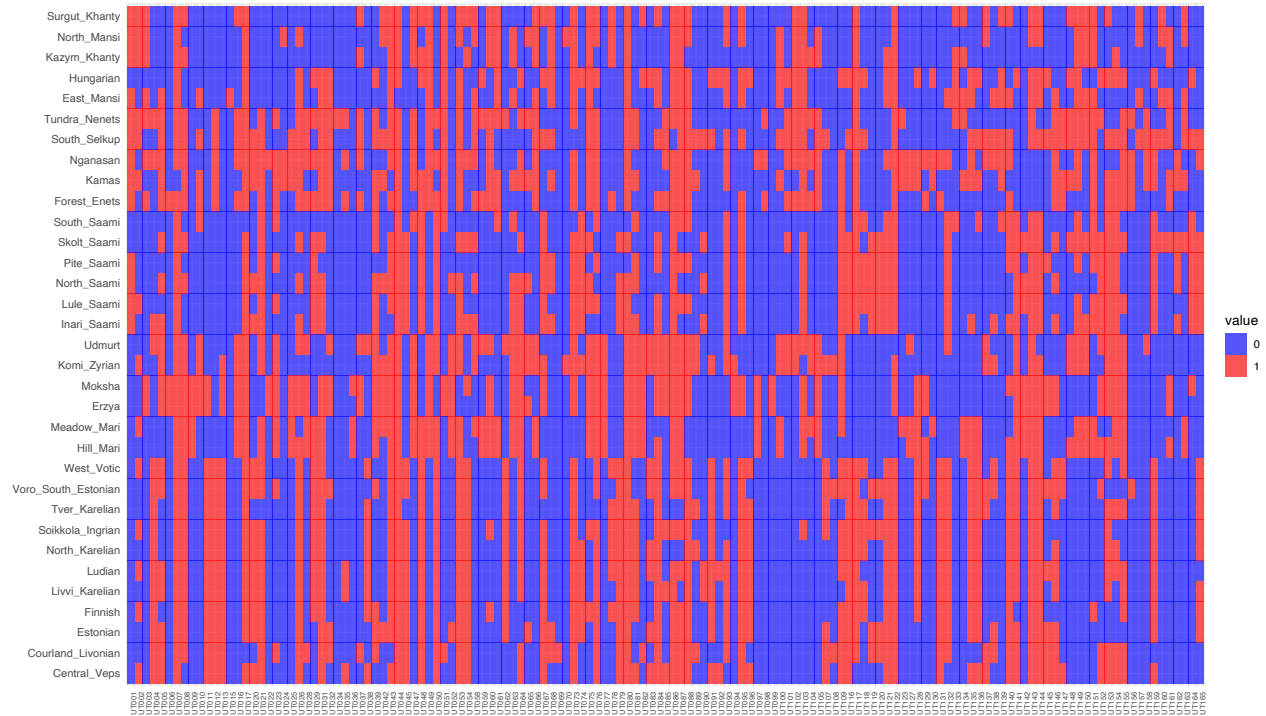

Figure 1: Overview of typological data in UT database

2.3 Calculating typological similarities

```r
lgs = ut_final$Name
lgs_sim = expand.grid(lgs, lgs, stringsAsFactors = F) %>%
  rowwise() %>%
  mutate(similarity = typological_sim(ut_final, Var1, Var2))
```

```r
lgs_sim_sorted = lgs_sim %>%
  mutate(Var1 = factor(Var1, levels = lang_sorted),
         Var2 = factor(Var2, levels = lang_sorted))
ggplot(lgs_sim_sorted, aes(Var1, Var2, fill = similarity)) +
  geom_tile() +
  geom_text(data = lgs_sim_sorted,
            mapping = aes(Var1, Var2,
                          label = round(similarity, digit = 1))) +
  scale_fill_continuous(type = "viridis") +
  labs(x = NULL, y = NULL) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        axis.text.y = element_text(size = 9),
        axis.text.x = element_text(angle = 90, size = 9, hjust = 1),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank()) +
  coord_fixed()
```
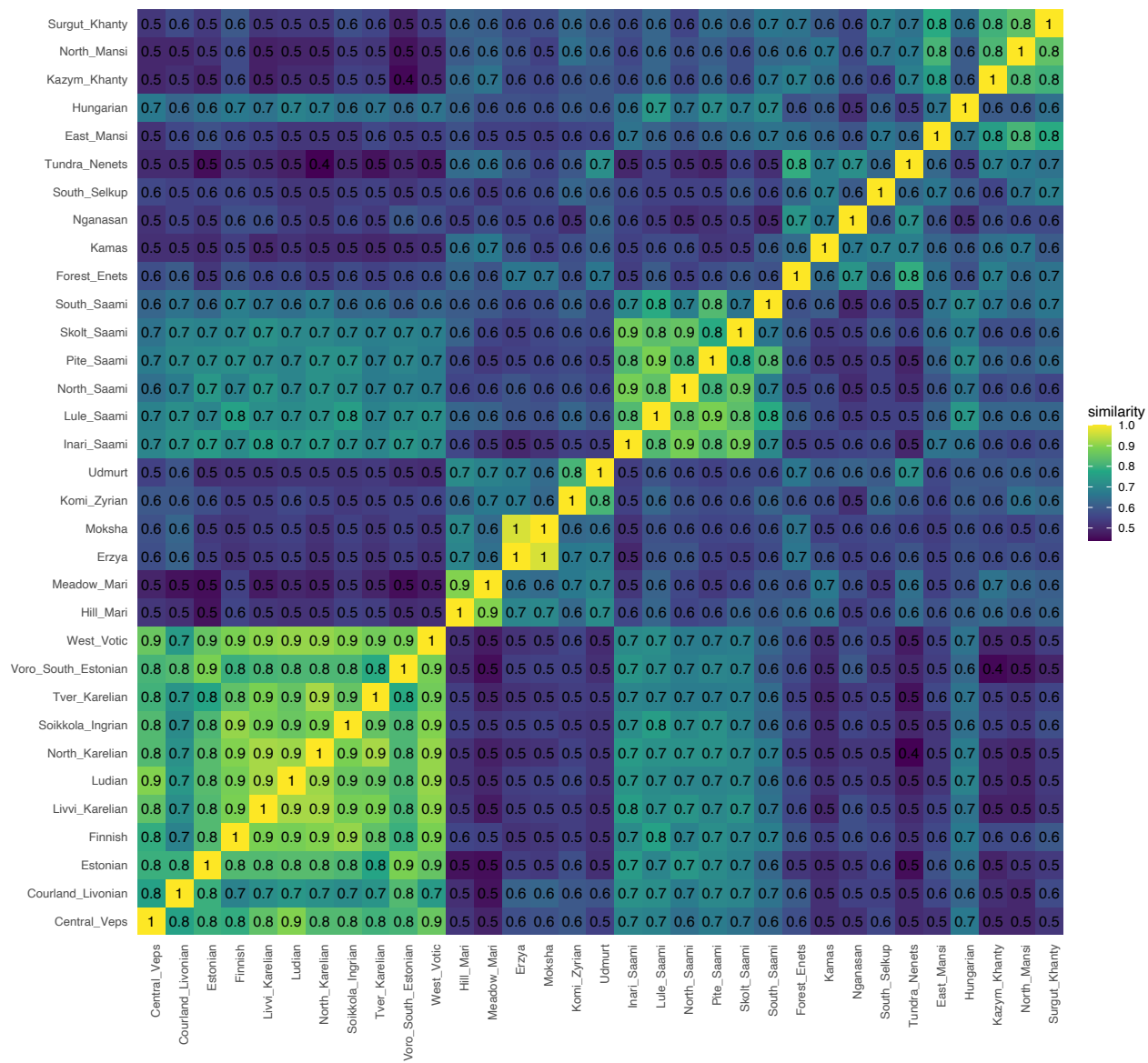
2.4   Exercises

(1) Pls add the information of typological areas (*parameters.csv*) to UT dataset. Note: you can use the preprocessed UT data **ut_final_long**.

(2) Pls calculate the typological similarities between languages across typological areas (phonology, morphology and syntax) in UT dataset, and plot them as heatmaps separately. Note: you can remove the lexicon features, and use `facet_wrap` function to create subpanels.

3 Geographical distances between languages

3.1 Function

```
geographical_dist = function(data = lang_geo, x = lang1, y = lang2){
  lang1_location = data[x, ]
  lang2_location = data[y, ]
  return(distHaversine(lang1_location, lang2_location)/1000)
}
```

3.2 Measuring geographical distances

```
lang_geo = lang_df %>%
  dplyr::select(Name, Longitude, Latitude) %>%
  column_to_rownames(var = "Name")
```

```
geo_dist = expand.grid(lgs, lgs, stringsAsFactors = F) %>%
  rowwise() %>%
  mutate(distance = geographical_dist(data = lang_geo,
                                      x = Var1,
                                      y = Var2)) %>%
  ungroup %>%
  mutate(dist_scaled = distance/max(distance))
geo_dist_sorted = geo_dist %>%
  mutate(Var1 = factor(Var1, levels = lang_sorted),
         Var2 = factor(Var2, levels = lang_sorted))
```

```
geo_dist_sorted %>%
  ggplot(., aes(Var1, Var2, fill = distance)) +
  geom_tile() +
  geom_text(data = geo_dist_sorted,
            mapping = aes(Var1, Var2,
                          label = round(dist_scaled, digit = 1))) +
  scale_fill_continuous(type = "viridis", direction = -1) +
  labs(x = NULL, y = NULL) +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"),
        axis.text.y = element_text(size = 9),
        axis.text.x = element_text(angle = 90, size = 9, hjust = 1),
        axis.ticks = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank()) +
  coord_fixed()
```
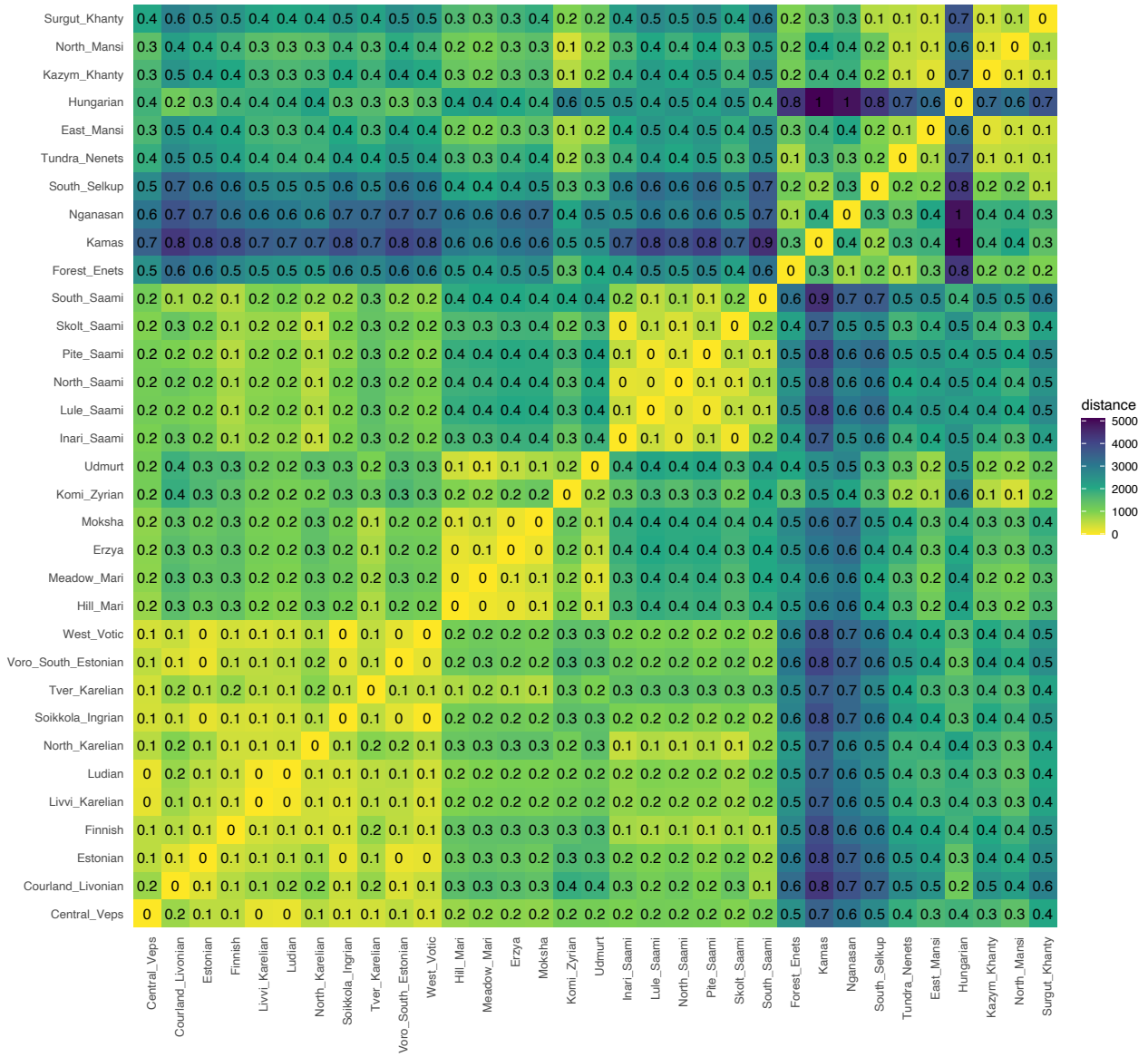
Figure 2: Geographical distances (scaled) between Uralic languages