Measuring Typological and Goegraphical Distances in R

Yingqi Jing

April 6, 2022

Contents

1 R programming language

1.1 Conditional statement (`ifelse` & `case_when`)

```
pi = 3.14
if(is.numeric(pi)){
  print("You have a numeric pi :)")
}else{
  print("You do not have a numeric pi :(")
}
```

```
## [1] "You have a numeric pi :)"
```

```
mylogic = TRUE
if(is.logical(mylogic)){
  print("You have a logical variable :)")
}else{
  print("You do not have a logical variable :(")
}
```

```
## [1] "You have a logical variable :)"
```

```
lang = "Estonian"
if(is.character(lang) & lang == "Estonian"){
  print("It is a character variable and it is Estonian!")
}else{
  print("It is not Estonian!")
}
```

```
## [1] "It is a character variable and it is Estonian!"
```

```
lang = "Estonian"
if(lang == "Finnish" | lang == "Estonian"){
  print("It is Finnish or it is Estonian!")
}else{
  print("It is neither Finnish or Estonian!")
}
```

```
## [1] "It is Finnish or it is Estonian!"
```

```
# alternative way
lang = "Estonian"
if(lang %in% c("Finnish", "Estonian")){
  print("It is Finnish or it is Estonian!")
}else{
  print("It is neither Finnish or Estonian!")
}
```

```
## [1] "It is Finnish or it is Estonian!"
```

```
lang_df = data.frame(id = 1:4,
          lang = c("Estonian", "English", "Chinese", "Esperanto"))
lang_df
```

```
##   id      lang
## 1  1  Estonian
## 2  2   English
## 3  3   Chinese
## 4  4 Esperanto
```

```
lang_df %>%
  mutate(family = ifelse(lang == "Estonian", "Uralic", "Unknown"))
```

```
##   id      lang  family
## 1  1  Estonian  Uralic
## 2  2   English Unknown
## 3  3   Chinese Unknown
## 4  4 Esperanto Unknown
```

We can generalize the `ifelse` into a case of multiple conditions, so that you have not only two options but multiple options.

```
lang_df %>%
  mutate(family = case_when(lang %in% c("Estonian") ~ "Uralic",
                            lang %in% c("English") ~ "IE",
                            lang %in% c("Chinese") ~ "Sino-Tibetan",
                            TRUE ~ "Unknown"))
```

```
##   id      lang       family
## 1  1  Estonian       Uralic
## 2  2   English           IE
## 3  3   Chinese Sino-Tibetan
## 4  4 Esperanto      Unknown
```

1.2 For loop (for and rowwise)

```r
names = c("Yingqi", "John", "Emily", "David")
for(name in names){
  print(nchar(name))
}
```

```
## [1] 6
## [1] 4
## [1] 5
## [1] 5
```

Dplyr provides a convenient `rowwise` operation to work on each row of a data.frame. This can be viewed as an alternative to `for` loop in a data.frame. Note: `rowwise` groups your data by row (class: rowwise_df), and it is best to `ungroup` immediately.[1]

To facilitate `rowwise` operation, you may need to use `c_across` to select specific columns. For example, if you want to calculate the number of characters of language names in a data.frame.

```r
mydf = data.frame(id = 1:5,
           names = c("Yingqi", "Michael", "John", "Emily", "David"))
name_length = mydf %>%
  rowwise() %>%
  mutate(length = paste(c_across(names), "'s name has:",
                        nchar(c_across(names)),
                        "charaters.")) %>%
  ungroup
name_length
```

```
## # A tibble: 5 x 3
##      id names   length
##   <int> <chr>   <chr>
## 1     1 Yingqi  Yingqi 's name has: 6 charaters.
## 2     2 Michael Michael 's name has: 7 charaters.
## 3     3 John    John 's name has: 4 charaters.
## 4     4 Emily   Emily 's name has: 5 charaters.
## 5     5 David   David 's name has: 5 charaters.
```

[1] https://medium.com/p/da3638b5f46c

1.3 Function

```
myaverage = function(myvec){
  if(is.numeric(myvec)){
    average = sum(myvec)/length(myvec)
  }else{
    average = sum(nchar(myvec))/length(myvec)
  }
  return(average)
}
```

```
myvec = c(2, 3)
myaverage(myvec)
```

```
## [1] 2.5
```

```
name_length %>%
  pull(names) %>%
  myaverage()
```

```
## [1] 5.4
```

2 Typological similarities between Uralic languages

2.1 Function

```
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

```
uratyp_df = read.csv("https://raw.githubusercontent.com/cldf-datasets/uratyp/v1.1/cldf/values.csv")
lang_df = read.csv("https://raw.githubusercontent.com/cldf-datasets/uratyp/v1.1/cldf/languages.csv")
# 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)

```
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

```
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

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

```
## # A tibble: 6 x 143
##   Name     Subfamily UT001 UT002 UT003 UT004 UT005 UT006 UT007 UT008 UT009 UT010
##   <chr>    <chr>     <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
## 1 Central~ Finnic        0     1     0     1     1     0     1     0     0     0
## 2 Courlan~ Finnic        0     0     0     1     1     0     1     1     0     0
## 3 East_Ma~ Ugric         1     0     1     0     1     0     1     0     0     1
## 4 Erzya    Mordvin       0     0     1     0     1     1     1     1     1     1
```

```
## 5 Estonian Finnic        0     0     0     1     1     0     1     1     0     0
## 6 Finnish  Finnic        0     1     0     1     0     0     1     1     0     0
## # ... with 131 more variables: UT011 <int>, UT012 <int>, UT013 <int>,
## #   UT015 <int>, UT016 <int>, UT017 <int>, UT020 <int>, UT021 <int>,
## #   UT022 <int>, UT023 <int>, UT024 <int>, UT025 <int>, UT026 <int>,
## #   UT028 <int>, UT029 <int>, UT031 <int>, UT032 <int>, UT034 <int>,
## #   UT035 <int>, UT036 <int>, UT037 <int>, UT038 <int>, UT039 <int>,
## #   UT042 <int>, UT043 <int>, UT044 <int>, UT045 <int>, UT047 <int>,
## #   UT048 <int>, UT049 <int>, UT050 <int>, UT051 <int>, UT052 <int>, ...
```

(4) reshape the data into long format

```
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))
head(ut_final_long)
```

```
## # A tibble: 6 x 4
##   Name         Subfamily feature value
##   <fct>        <chr>     <chr>   <fct>
## 1 Central_Veps Finnic    UT001   0
## 2 Central_Veps Finnic    UT002   1
## 3 Central_Veps Finnic    UT003   0
## 4 Central_Veps Finnic    UT004   1
## 5 Central_Veps Finnic    UT005   1
## 6 Central_Veps Finnic    UT006   0
```

(5) Visualize data via heatmap

```
ut_final_long %>%
  ggplot(., aes(x = feature, y = 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)
```

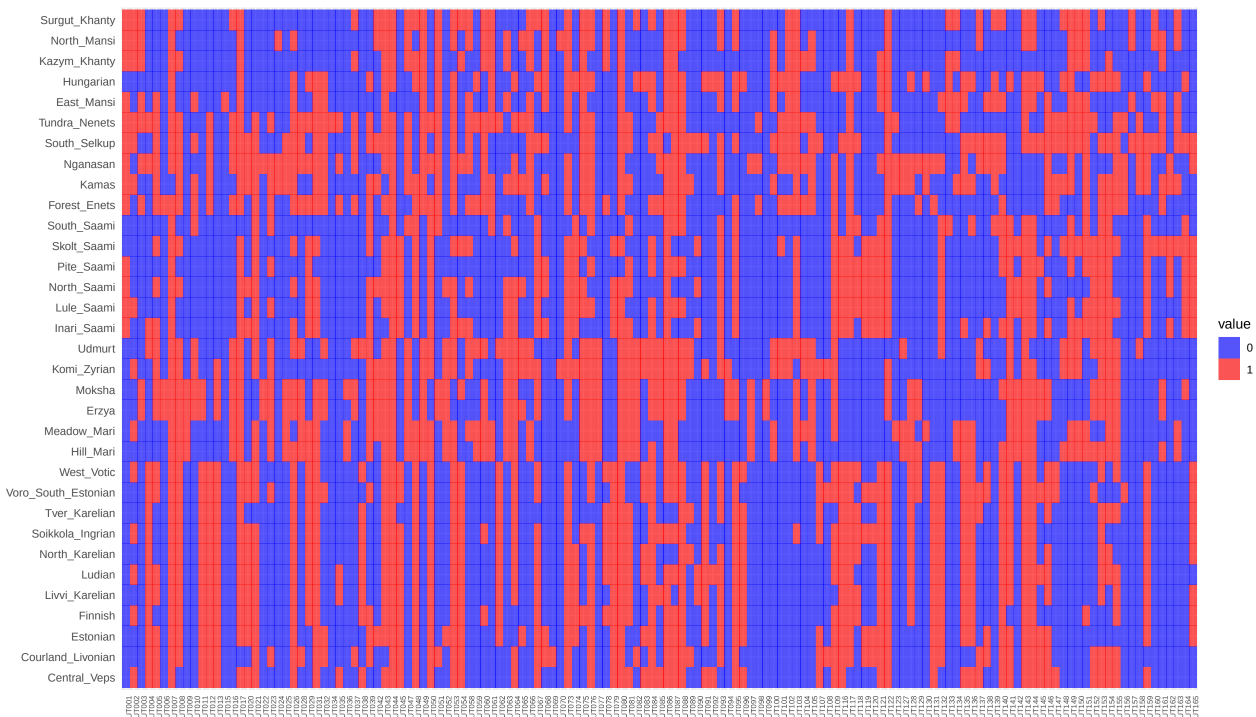
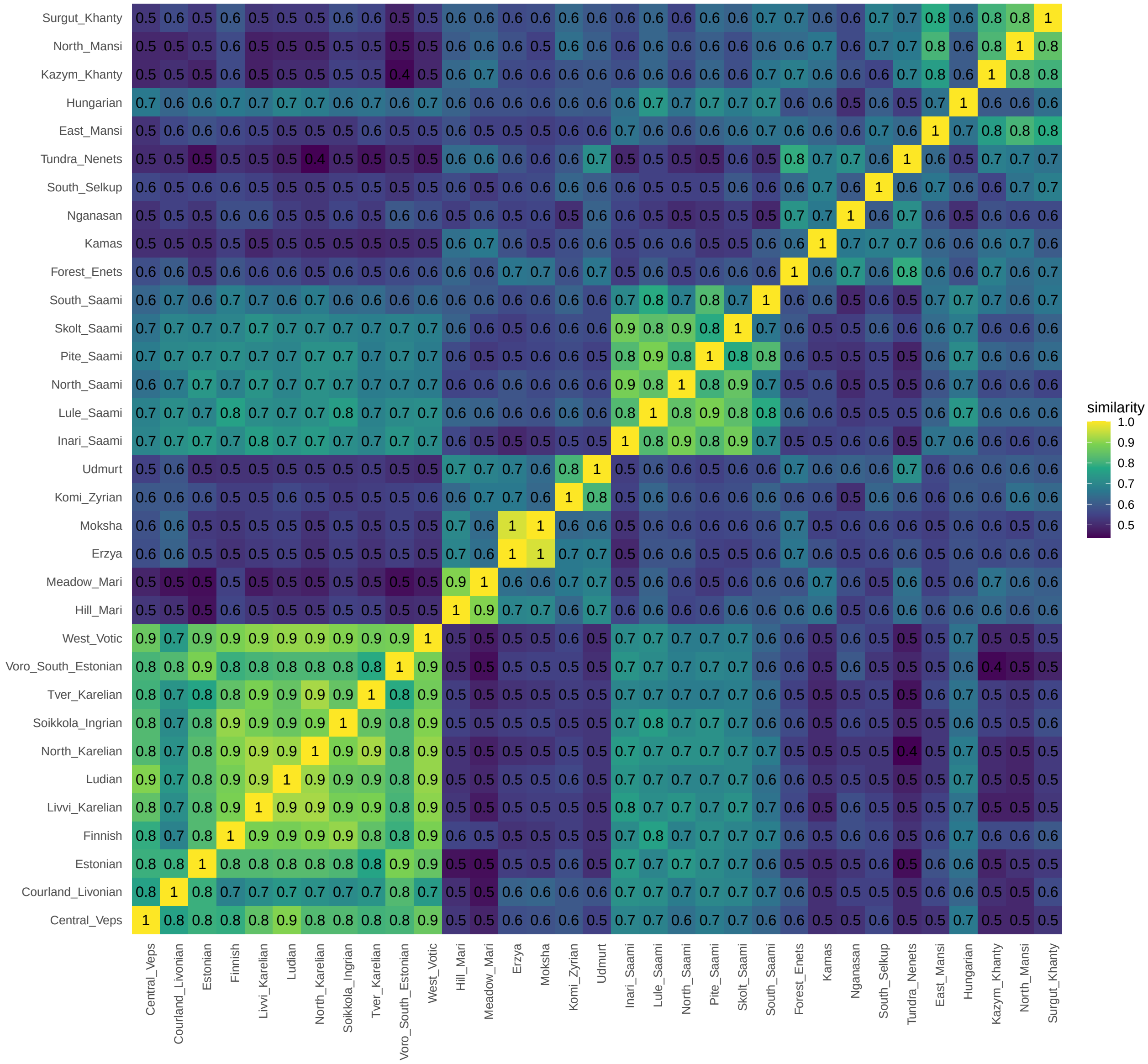


Figure 1: Overview of typological data in UT database

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()
```


Surgut_Khanty
North_Mansi
Kazym_Khanty
Hungarian
East_Mansi
Tundra_Nenets
South_Selkup
Nganasan
Kamas
Forest_Enets
South_Saami
Skolt_Saami
Pite_Saami
North_Saami
Lule_Saami
Inari_Saami
Udmurt
Komi_Zyrian
Moksha
Erzya
Meadow_Mari
Hill_Mari
West_Votic
Voro_South_Estonian
Tver_Karelian
Soikkola_Ingrian
North_Karelian
Ludian
Livvi_Karelian
Finnish
Estonian
Courland_Livonian
Central_Veps
similarity
1.0
0.9
0.8
0.7
0.6
0.5

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**.

```
feat_areas = read.csv("https://raw.githubusercontent.com/cldf-datasets/uratyp/v1.1/cldf/parameters.csv")
# feat_areas = read.csv("../Data/parameters.csv")
feat_areas = feat_areas %>%
  dplyr::select(ID, Area)
ut_areas = ut_final_long %>%
  inner_join(., feat_areas, by = c("feature" = "ID"))
```

(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.

```
ut_area_sim = ut_areas %>%
  filter(Area %in% c("Phonology", "Morphology", "Syntax")) %>%
  split(.$Area) %>%
  map_dfr(., ~{subdata = .x %>% spread(., key = feature, value = value)
              expand.grid(lgs, lgs, stringsAsFactors = F) %>%
                rowwise() %>%
                mutate(similarity = typological_sim(subdata, Var1, Var2)) %>%
                mutate(Area = subdata$Area[1])})
```

```
ut_area_sim_sorted = ut_area_sim %>%
  mutate(Var1 = factor(Var1, levels = lang_sorted),
         Var2 = factor(Var2, levels = lang_sorted),
         Area = factor(Area, levels = c("Phonology", "Morphology", "Syntax")))
ut_area_sim_sorted %>%
  ggplot(., 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") +
  facet_wrap(~Area) +
  # scale_x_discrete(position = "bottom") +
  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()
```

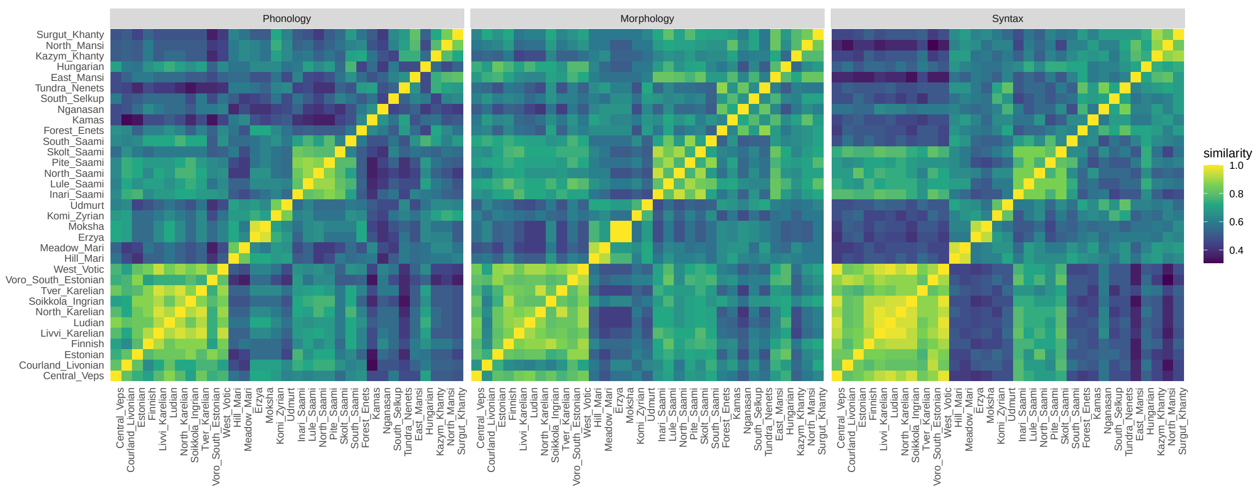


Figure 2: Typological similarities between Uralic languages across different areas in UT data

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()
```

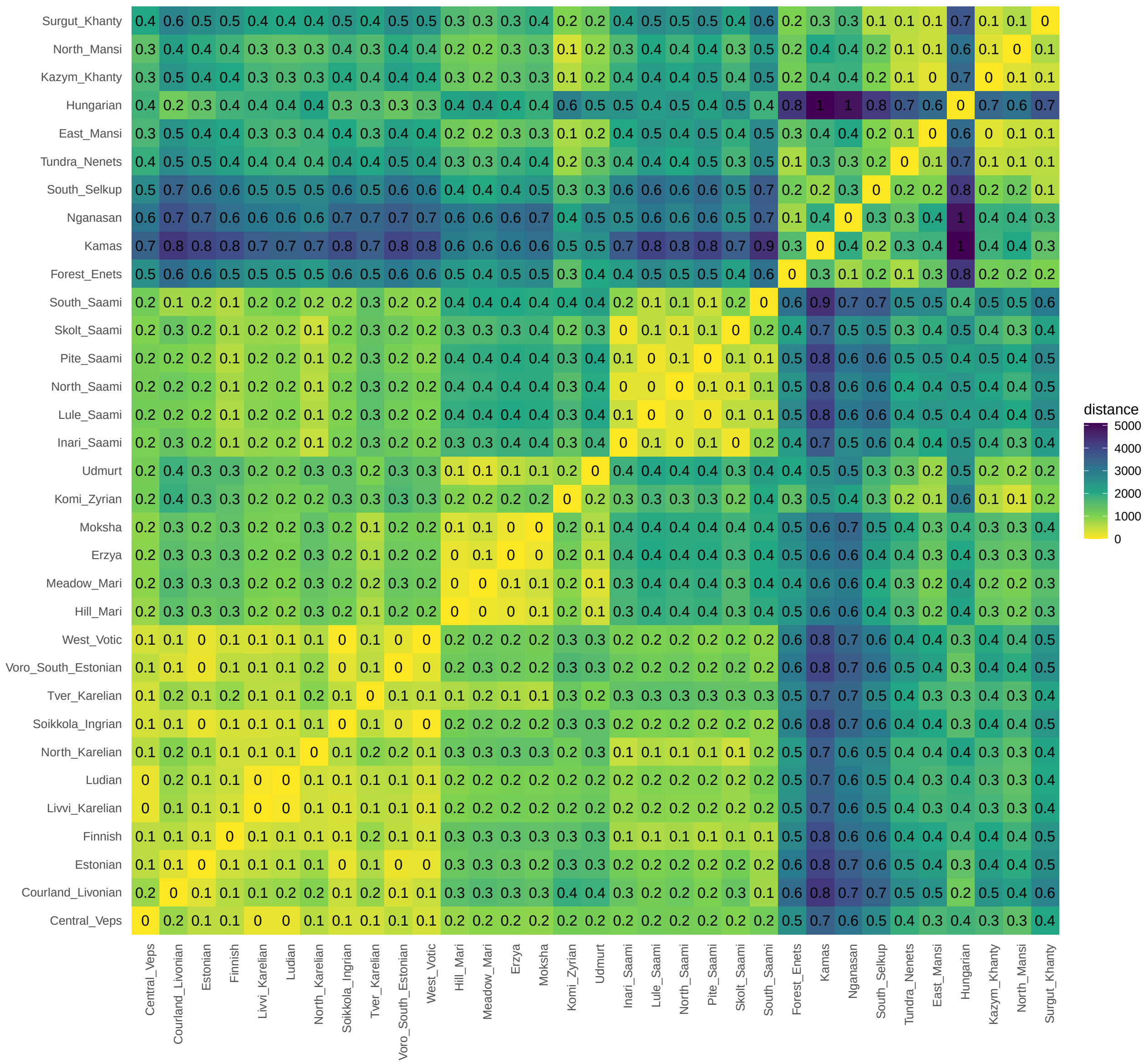


Figure 3: Geographical distances between Uralic languages