Neural Network

Zexuan Zhao

2022-12-06

Data preparation for neural network

As shown in random forest, `n` and balance of data set are both important for data set, although setting `n` too high induces too much loss of data.

In this script, I try to find a more proper threshold of `n` based on distribution.

Load CSV data and exclude redundant variables

```
data <- read_csv("../data/AppendixS1.csv") %>%
# Exclude taxonomic variables
  select(-Species, -Genus, -Family, -Order, -Class, -Phylum, -Kingdom) %>%
# Exclude redundant statistical variables
  select(-pmtp, -pmtr, -dgr, -der, -ger, -gep) %>%
# Exclude variables of genetic sequence
  select(-bp, -pi) %>%
# Exclude variables generated by downstream analysis
  select(-r, -c, -t, -p,    -f, -fp,    -cgeo,  -tgeo,  -pgeo,  -cenv,  -tenv,  -penv) %>%
# Exclude gene, which is used to infer genetic distance.
# It has too many duplicated synonyms
  select(-Gene) %>%
# Exclude rows with NAs
  na.omit()
```

```
## Rows: 19197 Columns: 67
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (10): Species, Gene, Genus, Family, Order, Class, Phylum, Kingdom, Metab...
## dbl (57): n, pmtr, pmtp, dgr, dgp, der, dep, ger, gep, bp, pi, Postfloodingo...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

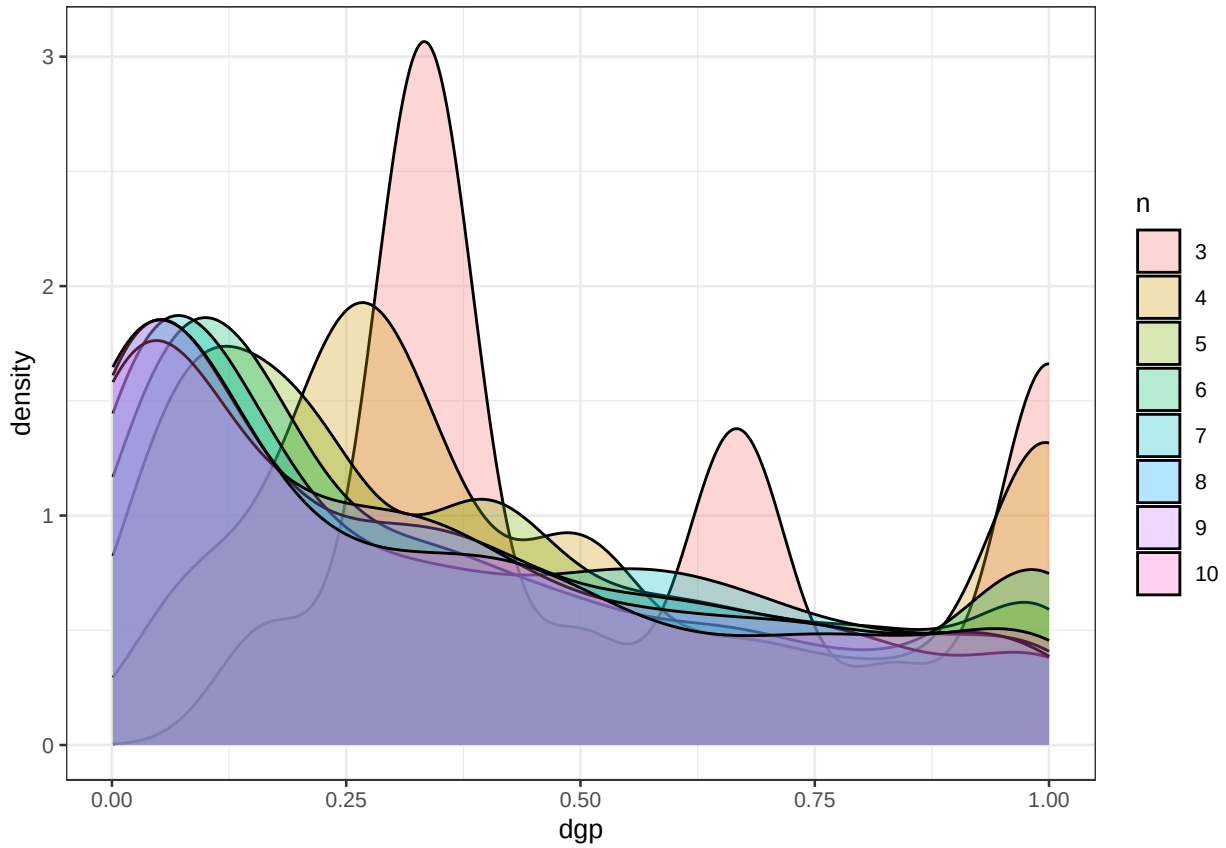
Distribution of `dgp` and `dep` depending on `n`

Instead of `pmtp`, which is the p-value of partial Mantel test, I kept `dgp` and `dep` as response variables for neural network. `dgp` is the p-value of correlation between genetic distance and geographical distance, while

`dep` is the p-value of correlation between genetic distance and environmental distance. Keeping both p-values is to consider 4 possible combinations of IBD and IBE, because they are compatible hypotheses about speciation.

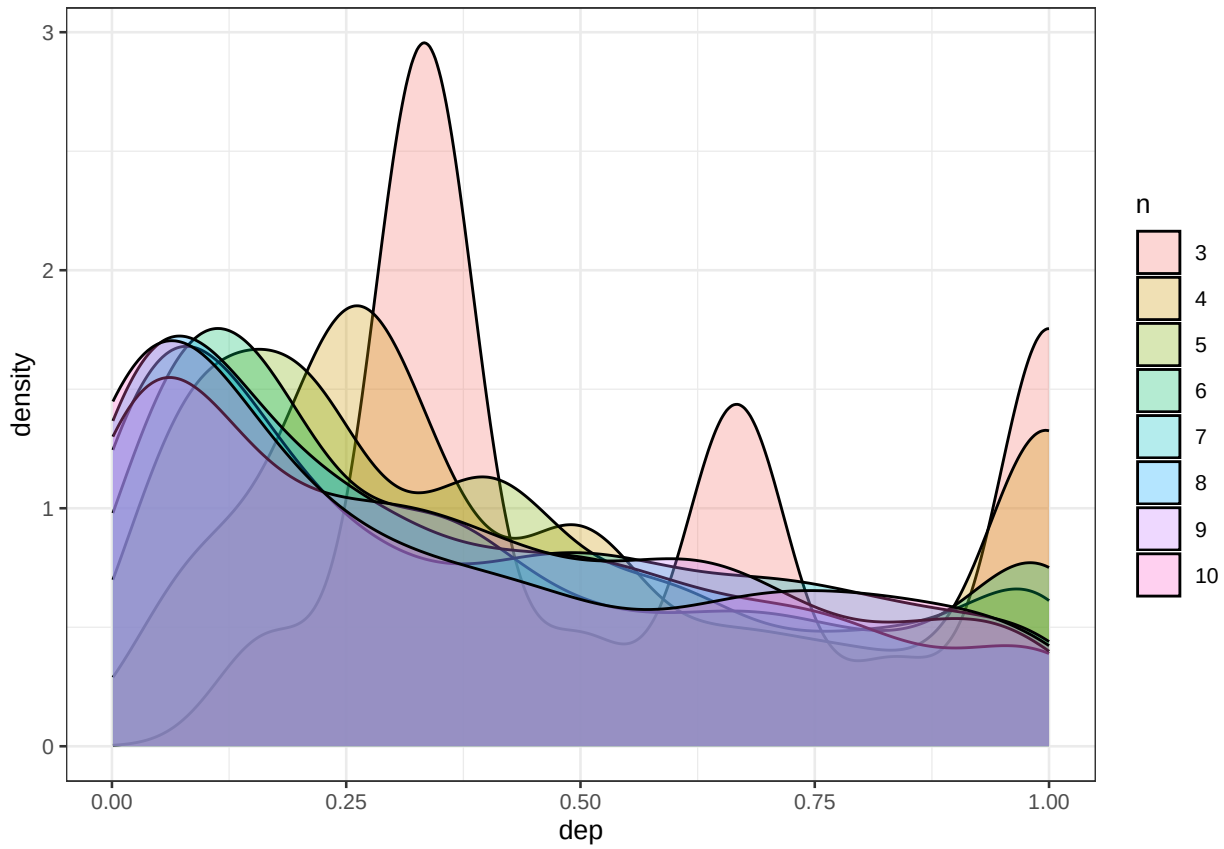
Here I plotted the histogram of `dgp` and `dep` by different `n`

```
dgp_hist <- data %>%
  filter(n <=10) %>%
  mutate(n = as.factor(n)) %>%
  ggplot(aes(x = dgp, fill = n)) +
    geom_density(alpha = 0.3) +
    theme_bw(base_size = 10)
dgp_hist
```




```
ggsave("../report/img/dgp_hist.png", plot = dgp_hist, width = 6, height = 4.5, units = "in")
```

```
dep_hist <- data %>%
  filter(n <=10) %>%
  mutate(n = as.factor(n)) %>%
  ggplot(aes(x = dep, fill = n)) +
    geom_density(alpha = 0.3) +
    theme_bw(base_size = 10)
dep_hist
```




```
ggsave("../report/img/dep_hist.png", plot = dep_hist, width = 4, height = 3, units = "in")
```

As shown, when `n=6`, the distribution converge.

```
data_n6 <- data %>%
  filter(n >=6) %>%
  select(-n)
dim(data_n6)
```

```
## [1] 8210   38
```

Also, after settiing `n>=6` I still have 60% of data.

Hot encoding string type variables

Before moving to Python for neural network, I hot encoded string type variables (metabolism and habit) in the data set.

```
metabolism_habit <- data_n6 %>%
  select(Metabolism, Habit) %>%
  mutate(Metabolism = as.factor(Metabolism),
         Habit = as.factor(Habit)) %>%
  as.data.table()
```

```r
metabolism_habit_hot_encoded <- one_hot(metabolism_habit) %>%
  as_tibble()

data_n6_hot_encoded <- data_n6 %>%
  select(-Metabolism, -Habit) %>%
  bind_cols(metabolism_habit_hot_encoded)
```

Saving data

```r
# Save X to X.csv and column names to X.colnames.txt
data_n6_hot_encoded %>%
  select(-dgp, -dep) %>%
  write_csv("../data/X.csv", col_names = FALSE)
data_n6_hot_encoded %>%
  select(-dgp, -dep) %>%
  filter(FALSE) %>%
  write_csv("../data/X.colnames.txt")
# Save Y to Y.csv
data_n6_hot_encoded %>%
  select(dgp, dep) %>%
  write_csv("../data/Y.csv", col_names = FALSE)
```