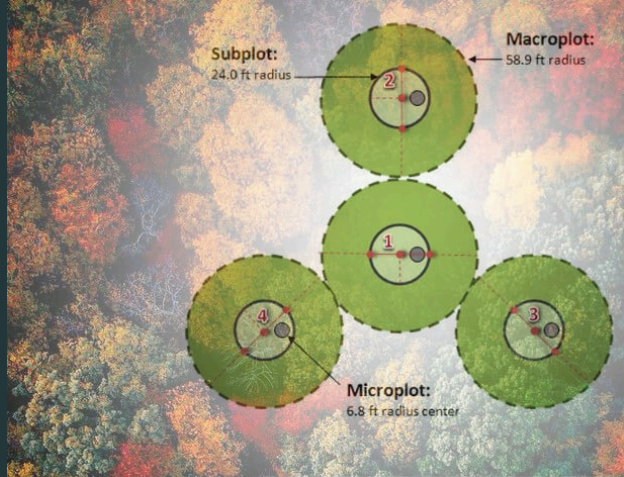
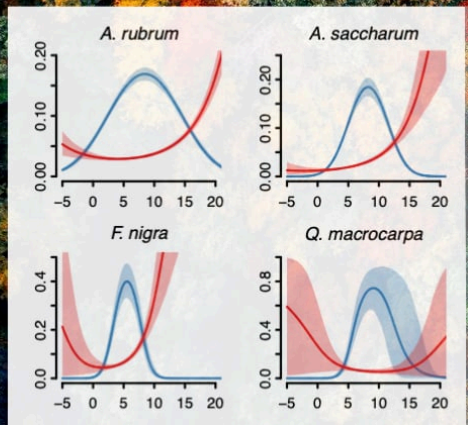
Lecture 4

Intermediate Data Manipulation

Introduction to R for Biologists - Lauren Talluto





Tall vs wide data

From the first day:

- Arrange your data so that each row is a single observation, each column is a variable ("tidy" data)

A wide data example

```
##   pid year Abies.balsamifera Acer.saccharum Betula.papyrifera
## 1   1 1980                 4              0                 0
## 2   2 2006                40              0                 1
## 3   3 2006                36              0                 9
## 4   4 1980                 3              0                 0
## 5   5 1980                 4              0                 0
## 6   6 1980                16              0                 0
##   Populus.tremuloides Tsuga.canadensis
## 1                   0                0
## 2                   0                0
## 3                   0                0
## 4                   0                0
## 5                   0                0
## 6                   0                0
```

- Abundances of five tree species
- Note that the format of the data makes some analyses easy
  - What is the average abundance for *Abies balsamifera* across all years?

A wide data example

```
##   pid year Abies.balsamifera Acer.saccharum Betula.papyrifera
## 1   1 1980                 4              0                 0
## 2   2 2006                40              0                 1
## 3   3 2006                36              0                 9
## 4   4 1980                 3              0                 0
## 5   5 1980                 4              0                 0
## 6   6 1980                16              0                 0
##   Populus.tremuloides Tsuga.canadensis
## 1                   0                0
## 2                   0                0
## 3                   0                0
## 4                   0                0
## 5                   0                0
## 6                   0                0
```

- But other things are harder
  - What is the average of all species combined for each year?
  - What is the average for each species among years?

A wide data example

```
##   pid year Abies.balsamifera Acer.saccharum Betula.papyrifera
## 1   1 1980                 4              0                 0
## 2   2 2006                40              0                 1
## 3   3 2006                36              0                 9
## 4   4 1980                 3              0                 0
## 5   5 1980                 4              0                 0
## 6   6 1980                16              0                 0
##   Populus.tremuloides Tsuga.canadensis
## 1                   0                0
## 2                   0                0
## 3                   0                0
## 4                   0                0
## 5                   0                0
## 6                   0                0
```

- Arrange your data so that each row is a single observation, each column is a variable ("tidy" data)
- This dataset violates this principle!
  - `abundance` is spread across five colums.
  - `species` is a variable but not a column!

Converting from wide to tall

- An add-on package, `reshape2`, can help us convert between **wide** and **tall** data frames with two functions:
  - From wide => tall: `melt`
  - From tall => wide: `dcast`

Converting from wide to tall

```
# install.packages("reshape2") # run this once, to install the package
library("reshape2")
trees_tall = melt(trees, id.vars = c("pid", "year"),
                  variable.name = "species",
                  value.name = "abundance")
head(trees_tall)
##   pid year           species abundance
## 1   1 1980 Abies.balsamifera         4
## 2   2 2006 Abies.balsamifera        40
## 3   3 2006 Abies.balsamifera        36
## 4   4 1980 Abies.balsamifera         3
## 5   5 1980 Abies.balsamifera         4
## 6   6 1980 Abies.balsamifera        16
```

Converting back to wide

```
# install.packages("reshape2") # run this once, to install the package
library("reshape2")
trees_wide = dcast(pid ~ year+species, data = trees_tall, value = "abundance", fill
## Using abundance as value column: use value.var to override.
```

```
trees_wide[1:7, 1:7]
##   pid 1975_Abies.balsamifera 1975_Acer.saccharum 1975_Betula.papyrifera
## 1   1                      0                   0                      0
## 2   2                      0                   0                      0
## 3   3                      0                   0                      0
## 4   4                      0                   0                      0
## 5   5                      0                   0                      0
## 6   6                      0                   0                      0
## 7   7                      0                   0                      0
##   1975_Populus.tremuloides 1975_Tsuga.canadensis 1980_Abies.balsamifera
## 1                        0                      0                      4
## 2                        0                      0                      0
## 3                        0                      0                      0
## 4                        0                      0                      3
## 5                        0                      0                      4
## 6                        0                      0                     16
## 7                        0                      0                      0
```

String manipulation: substr

`substr(x, i, j)`: extract a piece of a string `x`, starting in position `i` and ending in position `j`.

```
substr("here is a string", 6, 12)
## [1] "is a st"
```

String manipulation: substr

`substr(x, i, j)`: extract a piece of a string `x`, starting in position `i` and ending in position `j`.

```
vel = read.csv("../data/ex4/velocity.csv")
vel[1:3, 1:7]
##   location_id v_m_per_s_01 v_m_per_s_02 v_m_per_s_03 v_m_per_s_04 v_m_per_s_05
## 1          37        0.002         0.79         1.14         1.43         1.51
## 2          43        0.100         0.35         0.46         0.59         0.49
## 3          30        0.440         0.60         0.90         1.01         1.12
##   v_m_per_s_06
## 1         1.38
## 2         0.72
## 3         1.35
```

```
substr(colnames(vel)[-1], 11, 12)
##  [1] "01" "02" "03" "04" "05" "06" "07" "08" "09" "10" "11" "12" "13" "14" "15"
## [16] "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30"
## [31] "31" "32" "33" "34" "35" "36" "37" "38" "39" "40" "41" "42" "43" "44" "45"
```

String manipulation: paste

`paste(x, y, sep = "-")`: combine x and y together into 1 string, separated by the "-" character

```
x = "hello"
y = "world"
paste(x, y, sep = " ")
## [1] "hello world"
```

String manipulation: pattern matching

Complex string manipulations are possible using **regular expressions**.

Useful functions: `grep`, `grepl`, `sub`, `gsub`

Combining datasets: merge

Two tables can be combined based on matching **keys**

```
head(velocity_clean)
##    location_id vel_m_per_s measurement_number
## 4           10        0.34                 01
## 20          10        0.28                 02
## 36          10        0.19                 03
## 52          10        0.05                 04
## 68          10        0.07                 05
## 84          10        0.10                 06
```

```
head(width_clean)
##   location_id width_m
## 1          40  28.800
## 2          44  11.400
## 3          37  42.700
## 4          69   3.600
## 5          56  18.425
## 6          65  13.400
```

Combining datasets: merge

Two tables can be combined based on matching **keys**

Options to check in the help: `all.x`, `all.y`

```
merged1 = merge(velocity_clean, width_clean, by = "location_id")
head(merged1)
##   location_id vel_m_per_s measurement_number  width_m
## 1          10        0.34                 01 11.05714
## 2          10        0.28                 02 11.05714
## 3          10        0.19                 03 11.05714
## 4          10        0.05                 04 11.05714
## 5          10        0.07                 05 11.05714
## 6          10        0.10                 06 11.05714
```

Complex aggregation: aggregate

Velocity comes with multiple measurements per location ID. `aggregate` is like a supersized `tapply` for computing statistics across different categories in a table.

```
head(velocity_clean)
##    location_id vel_m_per_s measurement_number
## 4           10        0.34                 01
## 20          10        0.28                 02
## 36          10        0.19                 03
## 52          10        0.05                 04
## 68          10        0.07                 05
## 84          10        0.10                 06
```

Complex aggregation: aggregate

Velocity comes with multiple measurements per location ID. `aggregate` is like a supersized `tapply` for computing statistics across different categories in a table.

```
vel_agg = aggregate(vel_m_per_s ~ location_id, data = velocity_clean, FUN = mean, na
head(vel_agg)
##   location_id vel_m_per_s
## 1          10   0.2426316
## 2          30   1.0631818
## 3          37   1.0569091
## 4          40   0.6736000
## 5          41   0.1518182
## 6          43   0.7695556
```