
```
mydata <- read.table(file = "mydata.txt",
header = TRUE,
na.strings = "n")
```

What was the sign for missing data in mydata.txt?

Answer: "n"

What is written in the first line of mydata.txt?

Answer: column names

Is the command correct?

Answer: YES!

An introduction to R
WS 2019/2020

Rearranging and manipulating data

Dr. Noémie Becker
Dr. Eliza Argyridou

Special thanks to:
Dr. Benedikt Holtmann and Dr. SOnja Grath for sharing slides for this lecture

What you should know after day 5

Rearranging and manipulating data

- Reshaping data
- Combining data sets
- Making new variables
- Subsetting data
- Summarizing data

We will work with two particular packages:

- `tidyr`
- `dplyr`

What do we have to do before we can work with a package in R? (2 things)

What you should know after day 5

Rearranging and manipulating data

- Reshaping data
- Combining data sets
- Making new variables
- Subsetting data
- Summarizing data

Reshaping data

We will use data on fish abundance.

- Download the file `Fish_survey.csv` from the course page.

Set directory, for example:

```
setwd("~/Desktop/Day_5")
```

- Import the sample data into a variable `Fish_survey`:

```
Fish_survey <- read.csv("Fish_survey.csv",
                        header = TRUE)
head(Fish_survey)
```

	Site	Month	Transect	Trout	Perch	Stickleback
1	River1	January	1	10	5	28
2	River1	January	2	0	13	42
3	River1	January	3	8	19	9
4	River2	January	1	3	5	72
5	River2	January	2	2	9	33
6	River2	January	3	15	24	65

Reshaping data

```
head(Fish_survey)
```

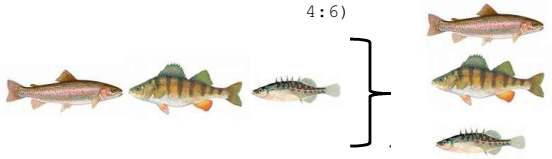
	Site	Month	Transect	Trout	Perch	Stickleback
1	River1	January	1	10	5	28
2	River1	January	2	0	13	42
3	River1	January	3	8	19	9
4	River2	January	1	3	5	72
5	River2	January	2	2	9	33
6	River2	January	3	15	24	65

Note:

- 3 species (trout, perch, stickleback)
- The numbers are abundance values for the species at specific sites

To combine the three columns into one column that contains all species you can use the function **gather()** from the **tidyr** package:

```
library(tidyr)
Fish_survey_long <- gather(Fish_survey,
                           Species,
                           Abundance,
                           4:6)
```



Reshaping data

```
Fish_survey_long <- gather(Fish_survey,
                           Species,
                           Abundance,
                           4:6)
head(Fish_survey_long)
```

```
    Site   Month Transect Species Abundance
1 River1 January        1   Trout        10
2 River1 January        2   Trout         0
3 River1 January        3   Trout         8
4 River2 January        1   Trout         3
5 River2 January        2   Trout         2
6 River2 January        3   Trout        15
```

```
tail(Fish_survey_long)
```

```
    Site Month Transect     Species Abundance
67 River1 April        1 Stickleback         5
68 River1 April        2 Stickleback        54
69 River1 April        3 Stickleback        31
70 River2 April        1 Stickleback        11
71 River2 April        2 Stickleback        31
72 River2 April        3 Stickleback        14
```


Reshaping data

To convert the data back into a format with separate columns for each species, you can use the function **spread()** from the **tidyr** package:

```
Fish_survey_wide <- spread(Fish_survey_long,
                           Species,
                           Abundance)
```

What you should know after day 5

Rearranging and manipulating data

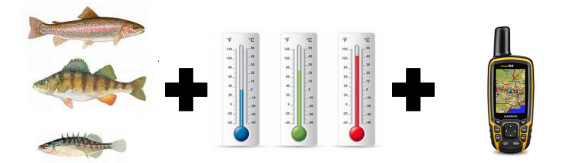
- Reshaping data
- Combining data sets
- Making new variables
- Subsetting data
- Summarizing data

Combining data

We now want to combine the information given by three different data sets.

To combine the data sets we will use the package **dplyr**:

```
library(dplyr)
```



Fish_survey.csv **Water_data.csv** **GPS_data.csv**

Combining data

We can join data sets by using the columns they share.

Fish survey
Site
Month
Transect
Species

+

Water characteristics
Site
Month
Water temp.
O_2 - content

+

GPS
Site
Transect
Latitude
Longitude

Which function could we use here?

Functions to combine data sets in dplyr	
left_join(a, b, by = "x1")	Joins matching rows from b to a
right_join(a, b, by = "x1")	Joins matching rows from a to b
inner_join(a, b, by = "x1")	Returns all rows from a where there are matching values in b
full_join(a, b, by = "x1")	Joins data and returns all rows and columns
semi_join(a, b, by = "x1")	All rows in a that have a match in b, keeping just columns from a.
anti_join(a, b, by = "x1")	All rows in a that do not have a match in b

Combining data

1) Join water characteristics to fish abundance data using inner_join()

```
Fish_and_Water <- inner_join(Fish_survey_long,
                             Water_data,
                             by = c("Site", "Month"))
```


Combining data

2) Add GPS locations to new Fish_and_Water data set using inner_join()

```
Fish_survey_combined <- inner_join(Fish_and_Water,
                                   GPS_location,
                                   by = c("Site", "Transect"))
```


What you should know after day 5

Rearranging and manipulating data

- Reshaping data
- Combining data sets
- Making new variables
- Subsetting data
- Summarizing data

Adding new variables

We will use data on bird behaviour.

```
Bird_Behaviour <- read.csv("Bird_Behaviour.csv",
                           header = TRUE,
                           stringsAsFactors=FALSE)
# Get an overview
str(Bird_Behaviour)
```

X1	X2
A	1
B	1
A	2
B	2

X1	X2	X3
A	1	T
B	1	F
A	2	T
B	2	F

We want to add the new variable (column) log_FID

Adding new variables

Three possibilities:

a) Using $

```
Bird_Behaviour$log_FID <- log(Bird_Behaviour$FID)
```

b) Using the [ ] - operator

```
Bird_Behaviour[ , "log_FID"] <- log(Bird_Behaviour$FID)
```

c) Using the function mutate() from dplyr package

```
Bird_Behaviour <- mutate(Bird_Behaviour,
                         log_FID = log(FID))
```


Adding new variables

The outcome:

```
head(Bird_Behaviour)
```

	Ind	Species	Sex	Year	FID	Disturbance	Fledglings	log_FID
1	PD1	Passer_domesticus	male	2013	5	8	1	1.6094379
2	PD1	Passer_domesticus	male	2014	2	40	4	0.6931472
3	PD1	Passer_domesticus	male	2015	8	30	4	2.0794415
4	PD2	Passer_domesticus	female	2013	10	35	3	2.3025851
5	PD2	Passer_domesticus	female	2014	10	15	0	2.3025851
6	PD2	Passer_domesticus	female	2015	6	6	2	1.7917595

Adding new variables

We can split one column into two using the function **separate()** from **dplyr** package:

```
Bird_Behaviour <- separate(Bird_Behaviour,
                           Species,
                           c("Genus","Species"),
                           sep = "_",
                           remove = TRUE)
```

X1	X2
A	1_1
B	1_2
A	2_1
B	2_2

X1	X2.1	X2.2
A	1	1
B	1	2
A	2	1
B	2	2

Combining variables

We can combine two columns into one using the function **unite()** from the **tidyr** package:

```
Bird_Behaviour <- unite(Bird_Behaviour,
                        "Genus_Species",
                        c(Genus, Species),
                        sep = "_",
                        remove = TRUE)
```

X1	X2.1	X2.2
A	1	1
B	1	2
A	2	1
B	2	2

X1	X2
A	1_1
B	1_2
A	2_1
B	2_2

What you should know after day 5

Rearranging and manipulating data

- Reshaping data
- Combining data sets
- Making new variables
- Subsetting data
- Summarizing data

Subsetting data

You can subset your data with:

- The [ ]-operator
- The function **subset()**
- With functions from the **dplyr** package
  - slice()
  - filter()
  - sample_frac()
  - sample_n()
  - select()

Subsetting data with the [ ]-operator

Examples:

```
# selects the first 4 columns
Bird_Behaviour[ , 1:4]

# selects rows 2 and 3
Bird_Behaviour[c(2,3), ]

# selects the rows 1 to 3 and columns 1 to 4
Bird_Behaviour[1:3, 1:4]

# selects the rows 1 to 3 and 6, and the columns 1 to 4
# and 8
Bird_Behaviour[c(1:3, 6), c(1:4, 8)]
```


Subsetting data with the [ ] and $-operators

Example:

```
# selects all rows with males
Bird_Behaviour[Bird_Behaviour$Sex == "male",  ]
```

Subsetting data with subset()

```
?subset()
```

Argument	Description
x	The object from which to extract subset
subset	A logical expression that describes the set of rows to return
select	An expression indicating which columns to return

Examples

What will R return in these cases?

```
subset(Bird_Behaviour, FID < 10)
# selects all rows with FID smaller than 10m

subset(Bird_Behaviour, FID < 10 & Sex == "male")
# selects all rows for males with FID smaller than
# 10

subset(Bird_Behaviour, FID > 10 | FID < 15,
       select = c(Ind, Sex, Year))
# selects all rows that have a value of FID
# greater than 10 or less than 15. We keep only
# the IND, Sex and Year column
```


Subsetting rows in dplyr

Subsetting by rows using slice() and filter()

Examples slice() and filter():

```
Bird_Behaviour.slice <- slice(Bird_Behaviour,
                              3:5)
# selects rows 3-5

Bird_Behaviour.filter <- filter(Bird_Behaviour,
                                FID < 5)
# selects rows that meet certain criteria
```


Subsetting rows in dplyr

You can take a random sample of rows with sample_frac() and sample_n()

Examples sample_frac() and sample_n():

```
Bird_Behaviour.50 <- sample_frac(Bird_Behaviour,
                                 size = 0.5,
                                 replace=FALSE)
# takes randomly 50% of the rows


Bird_Behaviour_50Rows <- sample_n(Bird_Behaviour,
                                  50,
                                  replace=FALSE)
# takes randomly 50 rows
```


Subsetting columns in dplyr

You can subset by columns with select()

Examples:

```
Bird_Behaviour_col <- select(Bird_Behaviour,
                             Ind,
                             Sex,
                             Fledglings)
# selects the columns Ind, Sex, and Fledglings

Bird_Behaviour_reduced <- select(Bird_Behaviour,
                                 -Disturbance)
# excludes the variable disturbance
```


What you should know after day 5

Rearranging and manipulating data

- Reshaping data
- Combining data sets
- Making new variables
- Subsetting data
- Summarizing data

Summarizing your data

You can summarize your data with **dplyr**

Example:
Get the overall mean for FID using **summarize() and mean()**

```
summarize(Bird_Behaviour,
          mean.FID = mean(FID))
```

```
  mean.FID
1 11.82639
```


Summarizing your data

We can add more measurements to our summary:

```
summarize(Bird_Behaviour,
          mean.FID = mean(FID),    # mean
          min.FID = min(FID),      # minimum
          max.FID = max(FID),      # maximum
          med.FID = median(FID),   # median
          sd.FID = sd(FID),        # standard deviation
          var.FID = var(FID),      # variance
          n.FID = n())             # sample size
```

```
   mean.FID max.FID med.FID   sd.FID var.FID n.FID
 1 11.82639      30      10 8.082036 65.3193   144
```


How can we get summaries for each species?

Before you can calculate these summaries, you have to apply the **group_by()** function from the **dplyr** package:

```
Bird_Behaviour_by_Species <- group_by(Bird_Behaviour,
                                      Genus_Species)
```


How can we get summaries for each species?

Now we can get summaries for each species:

```
Summary_species <-
summarize(Bird_Behaviour_by_Species,
          mean.FID=mean(FID), # mean
          min.FID=min(FID),   # minimum
          max.FID=max(FID),   # maximum
          med.FID=median(FID),# median
          sd.FID=sd(FID),     # standard deviation
          var.FID=var(FID),   # variance
          n.FID=n())          # sample size
Summary_species
```

```
# A tibble: 3 x 7
     Species  mean.FID max.FID med.FID   sd.FID   var.FID n.FID
       <chr>     <dbl>   <dbl>   <dbl>    <dbl>     <dbl> <int>
1    coelebs 20.437500      30      21 6.310768 39.825798    48
2 domesticus  6.104167      10       7 3.116460  9.712323    48
3   montanus  8.937500      20       8 5.613078 31.506649    48
```

We can make a data frame out of a tibble with:

```
as.data.frame(Summary_species)
```


Take-home message

- use `gather()` and `spread()` for combining or splitting columns
- use `XX_join()` for combining datasets
- functions from package dplyr allow for easy dataset splitting, modifying, summarizing etc...