Data Munging with R

Rob Kabacoff, Ph.D.

Topics

- Single dataset
  - subsetting data
  - sorting data
  - creating new variables
  - renaming variables
  - aggregating
- Multiple datasets
  - merging data
- Additional topics
  - reshaping
  - working with dates
  - cleaning text

Data Management with a single dataset

dplyr functions

- **filter** – select rows
- **select** – select columns
- **arrange** – reorder rows
- **mutate** – create new columns
- **rename** – rename columns
- **group_by** and **summarize** - aggregate

be sure to issue **library(dplyr)** to make these available

filter

subset data by selecting rows

```
df1 <- filter(mtcars, cyl==4, mpg > 20)

df2 <- filter(mtcars, cyl==4 & mpg > 20) # same

df3 <- filter(mtcars, cyl %in% c(4, 6) | am ==1)
```

Logical Operators

Operator	Description
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Exactly equal to
!=	Not equal to
!x	Not x
x | y	x or y
x & y	x and y
isTRUE(x)	Test if x is TRUE

select

subset data by selecting columns (variables)

```
df1 <- select(mtcars, mpg, cyl, wt)
df2 <- select(mtcars, mpg:qsec, carb)
df3 <- select(mtcars, -am, -carb)
```

arrange

reorder rows

```
df1 <- arrange(mtcars, cyl)
df2 <- arrange(mtcars, cyl, mpg)
df3 <- arrange(mtcars, cyl, desc(mpg))
```

mutate

create new variables (add new columns)

```
df1 <- mutate(mtcars,
              power = disp * hp,
              am = factor(am,
                      levels=c(0, 1),
                      labels =
                      c("automatic", "manual"))
 )
```

Arithmetic Operators

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
^	Exponentiation

rename

rename variables (columns)

```
df <- rename(mtcars,
             displacement = disp,
             transmission = am)
```

group_by and summarize

aggregate data by groups

```
df  <- group_by(mtcars, cyl, gear)
df2 <- summarise(df,
                 disp_n = n(),
                 disp_mean = mean(disp),
                 disp_sd = sd(disp)
)
```

group_by and summarize (2)

aggregate data by groups

```
df <- group_by(mtcars, cyl, gear)
df2 <- summarise_each(df, funs(mean))
df3 <- summarise_each(df, funs(min, max))
```

Putting it all together

```
df <- select(mtcars, cyl, disp, mpg)
df <- filter(df, mpg > 20)
df <- arrange(df, cyl, desc(mpg))

df <- select(mtcars, cyl, disp, mpg) %>%
             filter(mpg > 20) %>%
             arrange(cyl, desc(mpg))
```

Calculating percentages

```
mtcars %>% group_by(cyl) %>%
            summarise(n = n()) %>%
            mutate(pct = n/sum(n))
```

```
    cyl     n     pct
  <dbl> <int>   <dbl>
1     4    11 0.34375
2     6     7 0.21875
3     8    14 0.43750
```

```
as.data.frame(mtcars %>% group_by(cyl) %>%
    summarise(n = n()) %>%
    mutate(pct = paste0(round(100 * n/sum(n), 0), "%")))
```

```
  cyl  n pct
1   4 11 34%
2   6  7 22%
3   8 14 44%
```

Calculating percentages

```
as.data.frame(mtcars %>% group_by(cyl, gear) %>%
    summarise(n = n()) %>%
    mutate(pct = paste0(round(100 * n/sum(n), 0), "%")))
```

```
  cyl gear  n pct
1   4    3  1  9%
2   4    4  8 73%
3   4    5  2 18%
4   6    3  2 29%
5   6    4  4 57%
6   6    5  1 14%
7   8    3 12 86%
8   8    5  2 14%
```

Windows functions (min_rank)

```
# what are the 2 automatic transmission cars and
# 2 manual transmission cars that have the lowest gas mileage?

mtcars$name <- row.name(mtcars)
mtcars %>% group_by(am) %>%
           filter(min_rank(mpg) <= 2) %>%
           select(name, am, mpg)

# have the highest gas mileage?

mtcars %>% group_by(am) %>%
           filter(min_rank(desc(mpg)) <= 2) %>%
           select(name, am, mpg)
```

Merging Datasets

Start with some data

```
monitors <- read.table(header=TRUE, text='
  monitorid        lat       long
          1  42.467573  -87.810047
          2  42.049148  -88.273029
          3  39.110539  -90.324080
                        ')

pollutants <- read.table(header=TRUE, text='
  pollutant   duration    monitorid
      ozone         1h            1
        so2         1h            1
      ozone         8h            2
        no2         1h            4
                          ')
```

example from https://rpubs.com/NateByers/Merging

Inner join

```
library(dplyr)
inner_join(pollutants, monitors, by = "monitorid")
```

```
  pollutant duration monitorid      lat      long
1     ozone       1h         1 42.46757 -87.81005
2       so2       1h         1 42.46757 -87.81005
3     ozone       8h         2 42.04915 -88.27303
```

pollutants

```
  pollutant duration monitorid
1     ozone       1h         1
2       so2       1h         1
3     ozone       8h         2
4       no2       1h         4
```

monitors

```
  monitorid      lat      long
1         1 42.46757 -87.81005
2         2 42.04915 -88.27303
3         3 39.11054 -90.32408
```

Left join

```
library(dplyr)
left_join(pollutants, monitors, by = "monitorid")
```

```
  pollutant duration monitorid      lat      long
1     ozone       1h         1 42.46757 -87.81005
2       so2       1h         1 42.46757 -87.81005
3     ozone       8h         2 42.04915 -88.27303
4       no2       1h         4       NA        NA
```

pollutants

```
  pollutant duration monitorid
1     ozone       1h         1
2       so2       1h         1
3     ozone       8h         2
4       no2       1h         4
```

monitors

```
  monitorid      lat      long
1         1 42.46757 -87.81005
2         2 42.04915 -88.27303
3         3 39.11054 -90.32408
```

Full join

```
library(dplyr)
full_join(pollutants, monitors, by = "monitorid")
```

```
  pollutant duration monitorid      lat      long
1     ozone       1h         1 42.46757 -87.81005
2       so2       1h         1 42.46757 -87.81005
3     ozone       8h         2 42.04915 -88.27303
4       no2       1h         4       NA        NA
5      <NA>     <NA>         3 39.11054 -90.32408
```

pollutants

```
  pollutant duration monitorid
1     ozone       1h         1
2       so2       1h         1
3     ozone       8h         2
4       no2       1h         4
```

monitors

```
  monitorid      lat      long
1         1 42.46757 -87.81005
2         2 42.04915 -88.27303
3         3 39.11054 -90.32408
```

Filtering with semi_join

```
library(dplyr)
semi_join(pollutants, monitors, by = "monitorid")
```

```
  pollutant duration monitorid
1     ozone       1h         1
2       so2       1h         1
3     ozone       8h         2
```

keep pollutants rows that have a match in monitors

pollutants

```
  pollutant duration monitorid
1     ozone       1h         1
2       so2       1h         1
3     ozone       8h         2
4       no2       1h         4
```

monitors

```
  monitorid      lat      long
1         1 42.46757 -87.81005
2         2 42.04915 -88.27303
3         3 39.11054 -90.32408
```

Filtering with anti_join

```
library(dplyr)
anti_join(pollutants, monitors, by = "monitorid")
```

```
  pollutant duration monitorid
1       no2       1h         4
```

keep pollutants rows that don't have a match in monitors

pollutants

```
  pollutant duration monitorid
1     ozone       1h         1
2       so2       1h         1
3     ozone       8h         2
4       no2       1h         4
```

monitors

```
  monitorid      lat      long
1         1 42.46757 -87.81005
2         2 42.04915 -88.27303
3         3 39.11054 -90.32408
```

Reshaping Datasets

Wide to long

```
library(tidyr)
long <- wide %>% gather(Drug, Value, Heroin:Alcohol)
```

wide

ID	Sex	Heroin	Cocaine	Alcohol
1	M	1	0	0
2	F	0	1	1
3	M	1	1	0

long

ID	Sex	Drug	Value
1	M	Heroin	1
2	F	Heroin	0
3	M	Heroin	1
1	M	Cocaine	0
2	F	Cocaine	1
3	M	Cocaine	1
1	M	Alcohol	0
2	F	Alcohol	1
3	M	Alcohol	0

Wide to long

```
library(tidyr)
wide <- long %>% spread(Drug, Value)
```

long

ID	Sex	Drug	Value
1	M	Heroin	1
2	F	Heroin	0
3	M	Heroin	1
1	M	Cocaine	0
2	F	Cocaine	1
3	M	Cocaine	1
1	M	Alcohol	0
2	F	Alcohol	1
3	M	Alcohol	0

wide

ID	Sex	Alcohol	Cocaine	Heroin
1	M	0	0	1
2	F	1	1	0
3	M	0	1	1

Working with dates

Reading dates

- Dates come in as a character variable
- Convert to a date variable
- Use the lubridate package

Reading dates

- Dates come in as a character variable
- Convert to a date variable
- Use the lubridate package

example:

say date variable is stored as a character variable in the form "`mm-dd-yyyy`"

convert it to a date variable using function mdy( )

```
mdy("12-01-2010")
```

Reading dates

example:

```
data <- read.table(header=TRUE, text='
First    Last        birthday
John     Smith       12-01-2010
Bill     Doe         1/9/1963
Jane     Williams    05/19/08
                             ')

library(lubridate)
data$DOB <- mdy(dates$birthday)
```

R doesn't know these are dates

R knows these are dates

```
   First     Last   birthday        DOB
1   John    Smith 12-01-2010 2010-12-01
2   Bill      Doe   1/9/1963 1963-01-09
3   Jane Williams   05/19/08 2008-05-19
```

Reading dates

Order of elements in date-time	Parse function
year, month, day	`ymd()`
year, day, month	`ydm()`
month, day, year	`mdy()`
day, month, year	`dmy()`
hour, minute	`hm()`
hour, minute, second	`hms()`
year, month, day, hour, minute, second	`ymd_hms()`

Accessing data parts

Date component	Accessor
Year	`year()`
Month	`month()`
Week	`week()`
Day of year	`yday()`
Day of month	`mday()`
Day of week	`wday()`
Hour	`hour()`
Minute	`minute()`
Second	`second()`
Time zone	`tz()`

Accessing date parts

```
data$year    <- year(data$DOB)
data$month   <- month(data$DOB, label = TRUE)
data$day     <- day(data$DOB)
data$weekday <- wday(data$DOB, label=TRUE, abbr = FALSE)
```

```
  First     Last   birthday        DOB year month day   weekday
1  John    Smith 12-01-2010 2010-12-01 2010   Dec   1 Wednesday
2  Bill      Doe   1/9/1963 1963-01-09 1963   Jan   9 Wednesday
3  Jane Williams   05/19/08 2008-05-19 2008   May  19    Monday
```

Date arithmetic

```
data$age <- difftime(now(), data$DOB)
```

```
  First     Last            DOB              age
1  John     Smith   2010-12-01   2281.775 days
2  Bill       Doe   1963-01-09  19774.775 days
3  Jane  Williams   2008-05-19   3207.775 days
```

```
data$ageyrs <- as.numeric(data$age) / 365.25
```

```
  First     Last            DOB              age     ageyrs
1  John     Smith   2010-12-01   2281.775 days      6.247
2  Bill       Doe   1963-01-09  19774.775 days     54.140
3  Jane  Williams   2008-05-19   3207.775 days      8.782
```

Manipulating Text

Character functions

Function	Description

`substr(x, start = n1, stop = n2)`	Extract or replace substrings.
	
	`x <- "abcdef"`
	`substr(x, 2, 4) is "bcd"`
	`substr(x, 2, 4) <- "22222" is "a222ef"`

`grep(pattern, x , ignore.case = FALSE, fixed = FALSE)`	Search for pattern in x. Returns matching indices. `grep("A", c("b","A","c"), fixed=TRUE) returns 2`
`sub(pattern, replacement, x, ignore.case = FALSE, fixed = FALSE)`	Find pattern in x and replace with replacement text. `sub("\\s", ".", "Hello There") returns "Hello.There"`

If fixed=FALSE then pattern is a regular expression.
If fixed = TRUE then pattern is a text string.

Character functions

Function	Description
`strsplit(x, split)`	Split the elements of character vector x at split. `strsplit("abc", "")` returns 3 element vector "a","b","c"

`paste(..., sep="")`	Concatenate strings after using sep string to seperate them.
	`paste("x", 1:3, sep = "")` returns `c("x1","x2" "x3")`
	
	`paste("x",1:3, sep = "M")` returns `c("xM1","xM2" "xM3")`
	
	`paste("Today is", date())`

`toupper(x)`	Uppercase
`tolower(x)`	Lowercase

Recoding variables

```
df$gender <- ifelse(df$sex == 1, "Male", "Female")

df$ethn <- ifelse(df$race == 1, "Black",
             ifelse(df$race == 2, "White",
               ifelse(df$race == 3, "Asian", "Other")))
```

What about missing values?

Recoding variables

```
library(dplyr)
mtcars$cyl <- recode(mtcars$cyl, "4"=40, "6"=60, "8"=80)

mtcars$vs <- recode(mtcars$vs, "0"=2)

mtcars$gear <- factor(mtcars$gear)
mtcars$gear <- recode(mtcars$gear, "3"="3gears",
                      "4"="4gears", "5"="5gears")
```

Manipulating Text with stringr

stringr package

- Consistent functions for manipulating text
  - install.packages("stringr")
  - library(stringr)
  - each function starts with str_

string length

str_length(string) number of characters in each element of a string or character vector

```
str_length(c("Bill", "Bob", "William"))
4  3  7
```

combine strings

str_c(strings, sep="") combine strings

```
str_c("01", "15", "2015", sep="/")
"01/15/2015"

str_c("x", "y", "z")
"xyz"
```

subsetting strings

str_sub(strings, start, end) extract and replace substrings

```
x <- "baby"
str_sub(x, 1, 3)
"bab"

str_sub(x, 4, 4) <- "ies"
"babies"
```

subsetting strings

str_sub(strings, start, end) extract and replace substrings

negative numbers count from the right

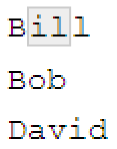
```
x <- "baby"
str_sub(x, -3, -1)
"aby"

str_sub(x, -1, -1) <- "ies"
"babies"
```

matching patterns with regular expressions

str_view(string, pattern) show first match

str_view_all(string, pattern) show all matches

```
x <- c("Bill", "Bob", "David")
str_view(x, "il")
```



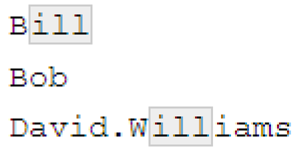
```
Bill
Bob
David
```

matching patterns with regular expressions

. matches any character

\\. matches the period

```
x <- c("Bill", "Bob", "David.Williams")
str_view(x, ".l.")
```



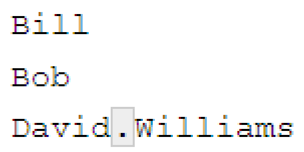
```
Bill
Bob
David.Williams
```

matching patterns with regular expressions

. matches any character

\\. matches the period

```
x <- c("Bill", "Bob", "David.Williams")
str_view(x, "\\.")
```



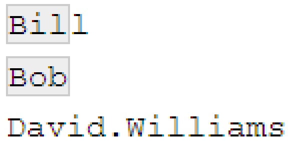
```
Bill
Bob
David.Williams
```

matching patterns with regular expressions

^ matches the start of a string

$ matches the end of a string

```
x <- c("Bill", "Bob", "David.Williams")
str_view(x, "^B..")
```



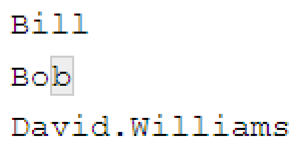
```
Bill
Bob
David.Williams
```

matching patterns with regular expressions

^ matches the start of a string

$ matches the end of a string

```
x <- c("Bill", "Bob", "David.Williams")
str_view(x, "b$")
```



```
Bill
Bob
David.Williams
```

matching patterns with regular expressions

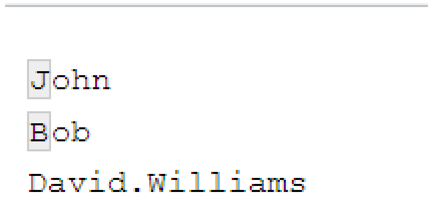
\\d matches any digit

\\s matches any whitespace

[abc] matches a, b, or c

[^abc] matches anything but a, b, or c

```
x <- c("John", "Bob", "David.Williams")
str_view(x, "^[JB]")
```



```
John
Bob
David.Williams
```

matching patterns with regular expressions

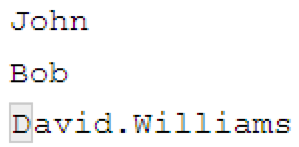
\\d matches any digit

\\s matches any whitespace

[abc] matches a, b, or c

[^abc] matches anything but a, b, or c

```
x <- c("John", "Bob", "David.Williams")
str_view(x, "^[^JB]")
```



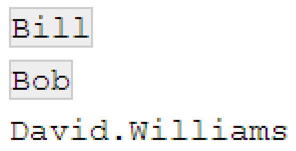
```
John
Bob
David.Williams
```

matching patterns with regular expressions

Repetition of characters

```
?: 0 or 1
+: 1 or more
*: 0 or more
```

```
x <- c("Bill", "Bob", "David.Williams")
str_view(x, "^B.+")
```



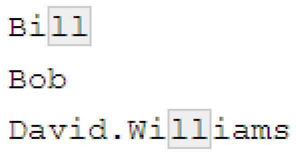
```
Bill
Bob
David.Williams
```

matching patterns with regular expressions

Repetition of characters

```
{n}: exactly n
{n,}: at least n
{,m}: at most m
{n,m}: between n and m
```

```
x <- c("Bill", "Bob", "David.Williams")
str_view(x, "l{2}")
```

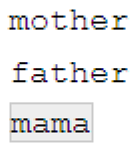


```
Bill
Bob
David.Williams
```

matching patterns with regular expressions

Back references

```
( ) plus \\1, \\2, \\3, etc.

x <- c("mother", "father", "mama")
str_view(x, "(..)\\1")
```



```
mother
father
mama
```

Detect matches

```
str_detect(strings, pattern)  returns T/F
```

```
x <- c("Bill", "Bob", "David.Williams")
str_detect(x, "il")

TRUE FALSE TRUE
```

Detect matches

detecting proper email addresses

```
x <- c("bsmith@wesleyan.edu",
        "jdoe@wesleyan",
        "jfwilliams.google.com")

str_detect(x,
      "(^.+)@(.+)(\\.edu$|\\.com$|\\.org$)")

TRUE FALSE TRUE
```