Data Cleaning

Duncan Turnbull

19 December 2014

Swirl 3

Using dplyr

Useful tools in dplyr include ability to chain results together. We can alter and mutate data and filter using grep. Also can use lubridate for dates

Tidy data is formatted in a standard way that facilitates exploration and analysis and works seemlessly with other tidy data tools. Specifically, tidy data satisfies three conditions:

1) Each variable forms a column

2) Each observation forms a row

3) Each type of observational unit forms a table

Gather

```
students
  grade male female
1     A    1      5
2     B    5      0
3     C    5      2
4     D    5      5
5     E    7      4

| Using the help file as a guide, call gather() with the following arguments (in order):
| students, sex, count, -grade. Note the minus sign before grade, which says we want to
| gather all columns EXCEPT grade.

> gather(students ,sex,count, -grade)
   grade    sex count
1      A   male     1
2      B   male     5
3      C   male     5
4      D   male     5
5      E   male     7
6      A female     5
7      B female     0
8      C female     2
9      D female     5
10     E female     4
```

It's important to understand what each argument to gather() means. The data argument, students, gives the name of the original dataset. The key and value arguments – sex and count, respectively – give the column

names for our tidy dataset. The final argument, -grade, says that we want to gather all columns EXCEPT the grade column (since grade is already a proper column variable.)

```
> students2
  grade male_1 female_1 male_2 female_2
1     A      3        4      3        4
2     B      6        4      3        5
3     C      7        4      3        8
4     D      4        0      8        1
5     E      1        1      2        7

> res <- gather(students2 ,sex_class,count, -grade)

> res
   grade sex_class count
1      A    male_1     3
2      B    male_1     6
3      C    male_1     7
4      D    male_1     4
5      E    male_1     1
6      A  female_1     4
7      B  female_1     4
8      C  female_1     4
9      D  female_1     0
10     E  female_1     1
11     A    male_2     3
12     B    male_2     3
13     C    male_2     3
14     D    male_2     8
15     E    male_2     2
16     A  female_2     4
17     B  female_2     5
18     C  female_2     8
19     D  female_2     1
20     E  female_2     7
```

Call separate() on res to split the sex_class column into sex and class. You only need to specify the first three arguments: data = res, col = sex_class, into = c("sex", "class. You don't have to provide the argument names as long as they are in the correct order.

```
> separate(data= res,col=sex_class,into=c("sex","class"))
   grade    sex class count
1      A   male     1     3
2      B   male     1     6
3      C   male     1     7
4      D   male     1     4
5      E   male     1     1
6      A female     1     4
7      B female     1     4
8      C female     1     4
9      D female     1     0
10     E female     1     1
11     A   male     2     3
```

```
12     B   male     2     3
13     C   male     2     3
14     D   male     2     8
15     E   male     2     2
16     A female     2     4
17     B female     2     5
18     C female     2     8
19     D female     2     1
20     E female     2     7
```

Conveniently, separate() was able to figure out on its own how to separate the sex_class column. Unless you request otherwise with the 'sep' argument, it splits on non-alphanumeric values. In other words, it assumes that the values are separated by something other than a letter or number (in this case, an underscore.)

Summarize

Summarise is useful for seeing the relativities of observations relative to their grouping. By grouping first then you can create summaries

```
by_package <- group_by(cran, package)
pack_sum <- summarize(by_package,
                      count = n(),
                      unique = n_distinct(ip_id),
                      countries = n_distinct(country),
                      avg_bytes = mean(size))
```

Here's the new bit, but using the same approach we've been using this whole time.

```
top_countries <- filter(pack_sum, countries > 60)
```

Gather, spread, mutate & chains

Using the students3 messy dataset

```
##     name    test class1 class2 class3 class4 class5
## 1  Sally midterm      A   <NA>      B   <NA>   <NA>
## 2  Sally   final      C   <NA>      C   <NA>   <NA>
## 3   Jeff midterm   <NA>      D   <NA>      A   <NA>
## 4   Jeff   final   <NA>      E   <NA>      C   <NA>
## 5  Roger midterm   <NA>      C   <NA>   <NA>      B
## 6  Roger   final   <NA>      A   <NA>   <NA>      A
## 7  Karen midterm   <NA>   <NA>      C      A   <NA>
## 8  Karen   final   <NA>   <NA>      C      A   <NA>
## 9  Brian midterm      B   <NA>   <NA>   <NA>      A
## 10 Brian   final      B   <NA>   <NA>   <NA>      C
```

We want to split this into a much tidier result. Need to aggregate classes into 1-5, change test types to variables and get rid of NAs

Spread

Name the column which should be changed from factors to variables and then column that has the values those columns should have. In our case test disappears and becomes two new columns: midterm and final with the relevant grade

```
students3 %>%
  gather(class, grade, class1:class5, na.rm = TRUE) %>%
  spread(test, grade) %>%
  mutate(class =extract_numeric(class)) %>%
  print
```

looking at students4 - Splitting tables

```
> students4
    id  name sex class midterm final
1  168 Brian   F     1       B     B
2  168 Brian   F     5       A     C
3  588 Sally   M     1       A     C
4  588 Sally   M     3       B     C
5  710  Jeff   M     2       D     E
6  710  Jeff   M     4       A     C
7  731 Roger   F     2       C     A
8  731 Roger   F     5       B     A
9  908 Karen   M     3       C     C
10 908 Karen   M     4       A     A
```

In this we want to split this table into a key table (student info) with unique student info e.g. id, name, sex, and then the results table which has the results linked to id.

```
student_info <- students4 %>%
  select( id, name, sex ) %>%
  unique() %>%
  print

gradebook <- students4 %>%
  select(id,class, midterm, final) %>%
  print
```

More Messy data - joining tables

```
> passed
   name class final
1 Brian     1     B
2 Roger     2     A
3 Roger     5     A
4 Karen     4     A

> failed
   name class final
```

```
1 Brian     5     C
2 Sally     1     C
3 Sally     3     C
4  Jeff     2     E
5  Jeff     4     C
6 Karen     3     C
```

Add a status field of passed or failed respectively

```
passed <- mutate(passed,  status = "passed")
failed <- mutate(failed,  status = "failed")
```

Join the two tables into one

```
rbind_list(passed,failed)
```

Getting data and grouping and adding proportions

This is using sat data, we remove any total columns, then rearrange the sex and counts then break down that variable again into separate columns (it has a natural underscore separator), then group by to create groups which mutate can use to add proportions

```
sat %>% select( -contains("total")) %>% gather( part_sex,count, -score_range) %>% separate(part_sex, c(
```

Group By

sets grouping parameters to the data set such that other tidyr functions can use them

Mutate

Can add new data to a table or change existing data mutate(data, existingcol = existingcol /2 , newcol = existingcol4 * 5)

Separate

Can split fields into new columns separate(dataset , coltobeseparated , into = c(“newcolA”, :“newcolB”“))

Quiz 4

Question 1 using strsplit()

Apply strsplit() to split all the names of the data frame on the characters “wgtp”. What is the value of the 123 element of the resulting list?

```
data <- read.csv(file = "./Q3ACS2006.csv")
names(data)
```

```
##   [1] "RT"       "SERIALNO" "DIVISION" "PUMA"     "REGION"   "ST"
##   [7] "ADJUST"   "WGTP"     "NP"       "TYPE"     "ACR"      "AGS"
##  [13] "BDS"      "BLD"      "BUS"      "CONP"     "ELEP"     "FS"
##  [19] "FULP"     "GASP"     "HFL"      "INSP"     "KIT"      "MHP"
##  [25] "MRGI"     "MRGP"     "MRGT"     "MRGX"     "PLM"      "RMS"
##  [31] "RNTM"     "RNTP"     "SMP"      "TEL"      "TEN"      "VACS"
##  [37] "VAL"      "VEH"      "WATP"     "YBL"      "FES"      "FINCP"
##  [43] "FPARC"    "GRNTP"    "GRPIP"    "HHL"      "HHT"      "HINCP"
##  [49] "HUGCL"    "HUPAC"    "HUPAOC"   "HUPARC"   "LNGI"     "MV"
##  [55] "NOC"      "NPF"      "NPP"      "NR"       "NRC"      "OCPIP"
##  [61] "PARTNER"  "PSF"      "R18"      "R60"      "R65"      "RESMODE"
##  [67] "SMOCP"    "SMX"      "SRNT"     "SVAL"     "TAXP"     "WIF"
##  [73] "WKEXREL"  "WORKSTAT" "FACRP"    "FAGSP"    "FBDSP"    "FBLDP"
##  [79] "FBUSP"    "FCONP"    "FELEP"    "FFSP"     "FFULP"    "FGASP"
##  [85] "FHFLP"    "FINSP"    "FKITP"    "FMHP"     "FMRGIP"   "FMRGP"
##  [91] "FMRGTP"   "FMRGXP"   "FMVYP"    "FPLMP"    "FRMSP"    "FRNTMP"
##  [97] "FRNTP"    "FSMP"     "FSMXHP"   "FSMXSP"   "FTAXP"    "FTELP"
## [103] "FTENP"    "FVACSP"   "FVALP"    "FVEHP"    "FWATP"    "FYBLP"
## [109] "wgtp1"    "wgtp2"    "wgtp3"    "wgtp4"    "wgtp5"    "wgtp6"
## [115] "wgtp7"    "wgtp8"    "wgtp9"    "wgtp10"   "wgtp11"   "wgtp12"
## [121] "wgtp13"   "wgtp14"   "wgtp15"   "wgtp16"   "wgtp17"   "wgtp18"
## [127] "wgtp19"   "wgtp20"   "wgtp21"   "wgtp22"   "wgtp23"   "wgtp24"
## [133] "wgtp25"   "wgtp26"   "wgtp27"   "wgtp28"   "wgtp29"   "wgtp30"
## [139] "wgtp31"   "wgtp32"   "wgtp33"   "wgtp34"   "wgtp35"   "wgtp36"
## [145] "wgtp37"   "wgtp38"   "wgtp39"   "wgtp40"   "wgtp41"   "wgtp42"
## [151] "wgtp43"   "wgtp44"   "wgtp45"   "wgtp46"   "wgtp47"   "wgtp48"
## [157] "wgtp49"   "wgtp50"   "wgtp51"   "wgtp52"   "wgtp53"   "wgtp54"
## [163] "wgtp55"   "wgtp56"   "wgtp57"   "wgtp58"   "wgtp59"   "wgtp60"
## [169] "wgtp61"   "wgtp62"   "wgtp63"   "wgtp64"   "wgtp65"   "wgtp66"
## [175] "wgtp67"   "wgtp68"   "wgtp69"   "wgtp70"   "wgtp71"   "wgtp72"
## [181] "wgtp73"   "wgtp74"   "wgtp75"   "wgtp76"   "wgtp77"   "wgtp78"
## [187] "wgtp79"   "wgtp80"
```

```
strsplit( names(data), "wgtp")[123]
```

```
## [[1]]
## [1] ""   "15"
```

Q2 GDP Averaging out data by stripping ','s

Load the Gross Domestic Product data for the 190 ranked countries in this data set: https://d396qusza40orc.cloudfront.net/getdata%2Fdata%2FGDP.csv Remove the commas from the GDP numbers in millions of dollars and average them. What is the average?

```
data <- read.csv("./Q3GDP.csv", skip =5, header=F, nrows=190)
colnames(data) <- c("countrycode","ranking","x","LongName","GDP","y","z","za","zb")
data <- select(data, countrycode,ranking,LongName,GDP)
mean(as.numeric(gsub(",","",data$GDP)))
```

Q3 count the number of countries whose name begins with "United"

Use grep to return matching countries and length to sum them. It will raise warnings about 2 hex values in the data set. We could exclude them if we were more cunning Assumption we already have data loaded from

previous question

```
length(grep(pattern = "^United",x = data$LongName,ignore.case = TRUE))
```

```
## [1] 0
```

Q4 Which Counties FY ends in June

Took a little while but merging seemed unnecessary Needed to figure out which variable had the Fiscal Year in it which I hope is Special.Notes Assumption we already have data loaded from previous question Need to reload country

```
country <- read.csv( "./Q3Country.csv" )
merge(data,country, by.x = "countrycode", by.y="CountryCode")["Special.Notes"] %>%  filter(grepl("Fiscal
```

Q5 use the quantmod: How many values were collected in 2012? How many values were collected on Mondays in 2012?

```
library(quantmod)
amzn = getSymbols("AMZN",auto.assign=FALSE)
sampleTimes = index(amzn)
nrow(amzn[(year(ymd(sampleTimes))==2012)])
nrow(amzn[(year(ymd(sampleTimes))==2012) &(wday(ymd(sampleTimes))==2)])
```