Lecture 9

BIOF 339

November 7, 2017

Goals for today

- Understanding tidy data
- Using packages in the `tidyverse`
- Stringing together actions using pipes

Tidy data

Tidy data

Tidy data is a concept explicitly stated by Hadley Wickham in this paper. It has three essential characteristics

1. Each variable forms a column
2. Each observation forms a row
3. Each type of observational unit forms a table.

Ways to have messy (i.e. not tidy) data

1. Column headers contain values

Country	< $10K	$10-20K	$20-50K	$50-100K	> $100K
India	40	25	25	9	1
USA	20	20	20	30	10

Ways to have messy (i.e. not tidy) data

1. Column headers contain values

Country	Income	Percentage
India	< $10K	40
USA	< $10K	20

This is a case of reshaping or melting

Ways to have messy (i.e. not tidy) data

1. Multiple variables in one column

Country	Year	M_0-14	F_0-14	M_ 15-60	F_15-60	M_60+	F_60+
UK	2010						
UK	2011						

Country	Year	Gender	Age	Count

Separating columns into different variables

Ways to have messy (i.e. not tidy) data

1. Variables stored in both rows and columns
2. Multiple types (or multiple levels) of data in one table
3. One type of data in multiple tables

Why tidy data

1. Information is captured in usable form
2. Tidy data is most amenable to modeling

In R, tidy data will most often be stored in a data.frame object

The tidyverse

The `tidyverse` package is a meta-package bundling several packages commonly used to analyze tidy data. The core packages are

1. ggplot2 : Visualization
2. tibble : A modern iteration of a data.frame
3. tidyr : Easy reshaping of tidy data
4. readr : Reading files
5. dplyr : Manipulating data frames
6. purrr : Functional programming in R

The tidyverse

Optional related packages are (among several others)

1. stringr : Easy string manipulation
2. broom : Tidying the results of models
3. lubridate: Easy date manipulation
4. forcats : Manipulating factors
5. DBI : Working with databases (SQL)

```
library(tidyverse)
library(stringr)
library(broom)
```

The tidyverse

These packages are syntactically consistent and operationally pretty fast:

- Their functions all typically take a vector or data.frame as the first argument
- Imposes good practices
- Reduces ambiguity about data types
- Wraps common operations into single functions
- Typically runs C++ code underneath via `Rcpp`

The tibble

```
library(tidyverse)
tdf <- tibble(x=1:1e4, y = rnorm(1e4))
tdf
```

```
#  # A tibble: 10,000 x 2
#         x         y
#     <int>     <dbl>
#  1      1  2.471461
#  2      2 -1.278678
#  3      3  0.375232
#  4      4 -1.264910
#  # ... with 9,996 more rows
```

The tibble

```
tdf <- tibble(x=1:1e4, y = rnorm(1e4))
options( tibble.print_min=5)
tdf
```

```
#  # A tibble: 10,000 x 2
#        x          y
#    <int>      <dbl>
#  1     1  1.4356251
#  2     2  1.8744543
#  3     3 -0.1820764
#  4     4  1.2006233
#  5     5  1.0259309
#  # ... with 9,995 more rows
```

The tibble

All subsets of tibbles are also tibbles

Tibbles never convert characters into factors implicitly

Reading data into a tibble (using read_csv or the like) doesn't change the names into weird strings

Pipes

Pipes are a relatively new concept in R (about 2 years)

For some, pipes are a more natural way of implementing processes to be done to a data frame

The pipe operator is `%>%`, which originally is from the packages `magrittr`

Pipes

Pipes take an object on the left side and pass it to the first argument of a function on the right side

```
1:10 %>% sqrt()
```

```
#   [1] 1.000000 1.414214 1.732051 2.000000 2.236068 2.449490 2.645751
#   [8] 2.828427 3.000000 3.162278
```

However, it's really useful in the tidyverse for working on data frames (or tibbles)

dplyr

There are 4 core verbs in `dplyr`:

1. mutate : create new variables in columns
2. filter : create subsets based on variable characteristics
3. select : extract particular columns
4. group_by : Group by levels of a variable

dplyr

```
library(dplyr)
mtcars <- as_tibble(mtcars)
options(tibble.print_min=4)
mtcars %>%
  mutate(kmpg = mpg * 1.6)
```

```
#  # A tibble: 32 x 12
#      mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb  kmpg
#    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#  1  21.0     6   160   110  3.90 2.620 16.46     0     1     4     4 33.60
#  2  21.0     6   160   110  3.90 2.875 17.02     0     1     4     4 33.60
#  3  22.8     4   108    93  3.85 2.320 18.61     1     1     4     1 36.48
#  4  21.4     6   258   110  3.08 3.215 19.44     1     0     3     1 34.24
#  # ... with 28 more rows
```

dplyr

```
mtcars %>%
  mutate(kmpg = mpg * 1.6) %>%
  filter(cyl == 4)
```

```
#  # A tibble: 11 x 12
#       mpg   cyl  disp    hp  drat    wt  qsec    vs    am  gear  carb  kmpg
#     <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#   1  22.8     4 108.0    93  3.85 2.320 18.61     1     1     4     1 36.48
#   2  24.4     4 146.7    62  3.69 3.190 20.00     1     0     4     2 39.04
#   3  22.8     4 140.8    95  3.92 3.150 22.90     1     0     4     2 36.48
#   4  32.4     4  78.7    66  4.08 2.200 19.47     1     1     4     1 51.84
#   5  30.4     4  75.7    52  4.93 1.615 18.52     1     1     4     2 48.64
#   6  33.9     4  71.1    65  4.22 1.835 19.90     1     1     4     1 54.24
#   7  21.5     4 120.1    97  3.70 2.465 20.01     1     0     3     1 34.40
#   8  27.3     4  79.0    66  4.08 1.935 18.90     1     1     4     1 43.68
#   9  26.0     4 120.3    91  4.43 2.140 16.70     0     1     5     2 41.60
#  10  30.4     4  95.1   113  3.77 1.513 16.90     1     1     5     2 48.64
#  11  21.4     4 121.0   109  4.11 2.780 18.60     1     1     4     2 34.24
```

dplyr

```
mtcars %>%
  mutate(kmpg = mpg * 1.6) %>%
  filter(cyl == 4) %>%
  select(disp, kmpg)
```

```
#  # A tibble: 11 x 2
#      disp  kmpg
#     <dbl> <dbl>
#   1 108.0 36.48
#   2 146.7 39.04
#   3 140.8 36.48
#   4  78.7 51.84
#   5  75.7 48.64
#   6  71.1 54.24
#   7 120.1 34.40
#   8  79.0 43.68
#   9 120.3 41.60
#  10  95.1 48.64
#  11 121.0 34.24
```

dplyr

If you want to pipe the data frame on the left to a non-first argument of a function, you can use `.`

```
mtcars %>%
  mutate(kmpg = mpg * 1.6) %>%
  filter(cyl == 4) %>%
  select(disp, kmpg) %>%
  lm(kmpg ~ disp, data = .)


#
#  Call:
#  lm(formula = kmpg ~ disp, data = .)
#
#  Coefficients:
#  (Intercept)         disp
#      65.3951      -0.2162
```

broom

The package `broom` has a function `tidy` that will make the output of models into tidy data sets

```
library(broom)
mtcars %>%
  mutate(kmpg = mpg * 1.6) %>%
  filter(cyl == 4) %>%
  select(disp, kmpg) %>%
  lm(kmpg ~ disp, data = .) %>%
  tidy()
```

```
#          term   estimate  std.error statistic      p.value
#  1 (Intercept) 65.3951285 5.74336864 11.386197 1.202715e-06
#  2        disp -0.2162269 0.05307457 -4.074021 2.782827e-03
```

tidyr

The `tidyr` package reshapes data from long to wide, much like `reshape2`. It has two core functions:

1. `gather` : Gather multiple columns into two columns
2. `spread` : Opposite of gather

It also has a function, `separate`, which will separate a composite column out into separate columns.

tidyr

I use `gather` a lot to prep for ggplot panels

```
mtcars %>%
  gather(variable, value, disp:qsec) %>% head()
```

```
#  # A tibble: 6 x 8
#      mpg   cyl    vs    am  gear  carb variable value
#    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>    <chr> <dbl>
#  1  21.0     6     0     1     4     4     disp   160
#  2  21.0     6     0     1     4     4     disp   160
#  3  22.8     4     1     1     4     1     disp   108
#  4  21.4     6     1     0     3     1     disp   258
#  5  18.7     8     0     0     3     2     disp   360
#  6  18.1     6     1     0     3     1     disp   225
```

Using the tidyverse to run multiple univariate models

Suppose I want to run a series of univariate regressions on the mtcars dataset, seeing how mpg is related to each of the continuous variables.

Let's build this

Many models

```
mtcars %>% select(mpg, disp:qsec)
```

```
#  # A tibble: 32 x 6
#      mpg  disp    hp  drat    wt  qsec
#  * <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
#  1  21.0   160   110  3.90 2.620 16.46
#  2  21.0   160   110  3.90 2.875 17.02
#  3  22.8   108    93  3.85 2.320 18.61
#  4  21.4   258   110  3.08 3.215 19.44
#  # ... with 28 more rows
```

Many models

```
mtcars %>% select(mpg, disp:qsec) %>%
  gather(variable, value, -mpg)
```

```
#  # A tibble: 160 x 3
#      mpg variable value
#    <dbl>    <chr> <dbl>
#  1  21.0     disp   160
#  2  21.0     disp   160
#  3  22.8     disp   108
#  4  21.4     disp   258
#  # ... with 156 more rows
```

Many models

```
mtcars %>% select(mpg, disp:qsec) %>%
  gather(variable, value, -mpg) %>%
  group_by(variable) %>%
  lm(mpg~value, data=.)


#
#  Call:
#  lm(formula = mpg ~ value, data = .)
#
#  Coefficients:
#  (Intercept)        value
#     21.28328     -0.01483
```

Many models

```
mtcars %>% select(mpg, disp:qsec) %>%
  gather(variable, value, -mpg) %>%
  nest(-variable)
```

```
#  # A tibble: 5 x 2
#    variable             data
#       <chr>           <list>
#  1     disp <tibble [32 x 2]>
#  2       hp <tibble [32 x 2]>
#  3     drat <tibble [32 x 2]>
#  4       wt <tibble [32 x 2]>
#  5     qsec <tibble [32 x 2]>
```

Many models

```
bl <- mtcars %>% select(mpg, disp:qsec) %>%
  gather(variable, value, -mpg) %>%
  nest(-variable)
bl$data[[1]]
```

```
#  # A tibble: 32 x 2
#      mpg value
#    <dbl> <dbl>
#  1  21.0   160
#  2  21.0   160
#  3  22.8   108
#  4  21.4   258
#  # ... with 28 more rows
```

Many models

```
mtcars %>% select(mpg, disp:qsec) %>%
  gather(variable, value, -mpg) %>%
  nest(-variable) %>%
  mutate(models = map(data, ~lm(mpg~value, data=.)))
```

```
#  # A tibble: 5 x 3
#    variable             data   models
#       <chr>           <list>   <list>
#  1     disp <tibble [32 x 2]> <S3: lm>
#  2       hp <tibble [32 x 2]> <S3: lm>
#  3     drat <tibble [32 x 2]> <S3: lm>
#  4       wt <tibble [32 x 2]> <S3: lm>
#  5     qsec <tibble [32 x 2]> <S3: lm>
```

Many models

```
mtcars %>% select(mpg, disp:qsec) %>%
 gather(variable, value, -mpg) %>%
 nest(-variable) %>%
 mutate(models = map(data, ~lm(mpg~value, data=.)),
        outputs = map(models, ~tidy(.)))
```

```
#  # A tibble: 5 x 4
#    variable              data   models              outputs
#       <chr>            <list>   <list>               <list>
#  1     disp <tibble [32 x 2]> <S3: lm> <data.frame [2 x 5]>
#  2       hp <tibble [32 x 2]> <S3: lm> <data.frame [2 x 5]>
#  3     drat <tibble [32 x 2]> <S3: lm> <data.frame [2 x 5]>
#  4       wt <tibble [32 x 2]> <S3: lm> <data.frame [2 x 5]>
#  5     qsec <tibble [32 x 2]> <S3: lm> <data.frame [2 x 5]>
```

Many models

```
mtcars %>% select(mpg, disp:qsec) %>%
 gather(variable, value, -mpg) %>%
 nest(-variable) %>%
 mutate(models = map(data, ~lm(mpg~value, data=.)),
        outputs = map(models, ~tidy(.))) %>%
 select(variable, outputs)
```

```
#  # A tibble: 5 x 2
#    variable             outputs
#       <chr>              <list>
#  1     disp <data.frame [2 x 5]>
#  2       hp <data.frame [2 x 5]>
#  3     drat <data.frame [2 x 5]>
#  4       wt <data.frame [2 x 5]>
#  5     qsec <data.frame [2 x 5]>
```

Many models

```
mtcars %>% select(mpg, disp:qsec) %>%
 gather(variable, value, -mpg) %>%
 nest(-variable) %>%
 mutate(models = map(data, ~lm(mpg~value, data=.)),
        outputs = map(models, ~tidy(.))) %>%
 select(variable, outputs) %>%
 unnest()
```

```
#  # A tibble: 10 x 6
#     variable        term    estimate    std.error  statistic      p.value
#        <chr>       <chr>       <dbl>        <dbl>      <dbl>        <dbl>
#   1     disp (Intercept) 29.59985476  1.229719515 24.0704115 3.576586e-21
#   2     disp       value -0.04121512  0.004711833 -8.7471515 9.380327e-10
#   3       hp (Intercept) 30.09886054  1.633920950 18.4212465 6.642736e-18
#   4       hp       value -0.06822828  0.010119304 -6.7423885 1.787835e-07
#   5     drat (Intercept) -7.52461844  5.476662574 -1.3739423 1.796391e-01
#   6     drat       value  7.67823260  1.506705108  5.0960421 1.776240e-05
#   7       wt (Intercept) 37.28512617  1.877627337 19.8575753 8.241799e-19
#   8       wt       value -5.34447157  0.559101045 -9.5590441 1.293959e-10
```

Many models

```
mtcars %>% select(mpg, disp:qsec) %>%
  gather(variable, value, -mpg) %>%
  nest(-variable) %>%
  mutate(models = map(data, ~lm(mpg~value, data=.)),
         outputs = map(models, ~tidy(.))) %>%
  select(variable, outputs) %>%
  unnest() %>%
  filter(term=='value')
```

```
#  # A tibble: 5 x 6
#    variable  term    estimate   std.error statistic      p.value
#       <chr> <chr>       <dbl>       <dbl>     <dbl>        <dbl>
#  1     disp value -0.04121512 0.004711833 -8.747152 9.380327e-10
#  2       hp value -0.06822828 0.010119304 -6.742389 1.787835e-07
#  3     drat value  7.67823260 1.506705108  5.096042 1.776240e-05
#  4       wt value -5.34447157 0.559101045 -9.559044 1.293959e-10
#  5     qsec value  1.41212484 0.559210130  2.525213 1.708199e-02
```

Many models

```
mtcars %>% select(mpg, disp:qsec) %>%
  gather(variable, value, -mpg) %>%
  nest(-variable) %>%
  mutate(models = map(data, ~lm(mpg~value, data=.)),
         outputs = map(models, ~tidy(.))) %>%
  select(variable, outputs) %>%
  unnest() %>%
  filter(term=='value') %>%
  mutate_if(is.numeric, funs(round(., 3)))
```

```
#  # A tibble: 5 x 6
#    variable  term estimate std.error statistic p.value
#       <chr> <chr>    <dbl>     <dbl>     <dbl>   <dbl>
#  1     disp value   -0.041     0.005    -8.747   0.000
#  2       hp value   -0.068     0.010    -6.742   0.000
#  3     drat value    7.678     1.507     5.096   0.000
#  4       wt value   -5.344     0.559    -9.559   0.000
#  5     qsec value    1.412     0.559     2.525   0.017
```

Another example (from here)

Grab the raw data

```
url <- "http://varianceexplained.org/files/Brauer2008_DataSet1.tds"
raw_data <- read_delim(url, delim='\t')
head(raw_data)
```

```
#  # A tibble: 6 x 40
#          GID       YORF
#        <chr>      <chr>
#  1 GENE1331X A_06_P5820
#  2 GENE4924X A_06_P5866
#  3 GENE4690X A_06_P1834
#  4 GENE1177X A_06_P4928
#  5  GENE511X A_06_P5620
#  6 GENE2133X A_06_P5307
#  # ... with 38 more variables: NAME <chr>, GWEIGHT <int>, G0.05 <dbl>,
#  #   G0.1 <dbl>, G0.15 <dbl>, G0.2 <dbl>, G0.25 <dbl>, G0.3 <dbl>,
#  #   N0.05 <dbl>, N0.1 <dbl>, N0.15 <dbl>, N0.2 <dbl>, N0.25 <dbl>,
#  #   N0.3 <dbl>, P0.05 <dbl>, P0.1 <dbl>, P0.15 <dbl>, P0.2 <dbl>,
#  #   P0.25 <dbl>, P0.3 <dbl>, S0.05 <dbl>, S0.1 <dbl>, S0.15 <dbl>,
#  #   S0.2 <dbl>, S0.25 <dbl>, S0.3 <dbl>, L0.05 <dbl>, L0.1 <dbl>,
```

Look at the annotation data

```
head(raw_data$NAME)
```

```
#  [1] "SFB2       || ER to Golgi transport || molecular function unknown || YNL049C
#  [2] "          || biological process unknown || molecular function unknown || YNL
#  [3] "QRI7       || proteolysis and peptidolysis || metalloendopeptidase activity
#  [4] "CFT2       || mRNA polyadenylylation* || RNA binding || YLR115W || 1081958"
#  [5] "SSO2       || vesicle fusion* || t-SNARE activity || YMR183C || 1081214"
#  [6] "PSP2       || biological process unknown || molecular function unknown || YM
```

Separate annotation into columns

```
cleaned_data <- raw_data %>%
  separate(NAME, c("name", "BP", "MF", "systematic_name", "number"),
           sep = "\\|\\|")
head(cleaned_data)
```

```
#  # A tibble: 6 x 44
#          GID       YORF        name                                BP
#        <chr>      <chr>       <chr>                             <chr>
#  1 GENE1331X A_06_P5820 SFB2                ER to Golgi transport
#  2 GENE4924X A_06_P5866                biological process unknown
#  3 GENE4690X A_06_P1834 QRI7          proteolysis and peptidolysis
#  4 GENE1177X A_06_P4928 CFT2               mRNA polyadenylylation*
#  5  GENE511X A_06_P5620 SSO2                        vesicle fusion*
#  6 GENE2133X A_06_P5307 PSP2           biological process unknown
#  # ... with 40 more variables: MF <chr>, systematic_name <chr>,
#  #   number <chr>, GWEIGHT <int>, G0.05 <dbl>, G0.1 <dbl>, G0.15 <dbl>,
#  #   G0.2 <dbl>, G0.25 <dbl>, G0.3 <dbl>, N0.05 <dbl>, N0.1 <dbl>,
#  #   N0.15 <dbl>, N0.2 <dbl>, N0.25 <dbl>, N0.3 <dbl>, P0.05 <dbl>,
#  #   P0.1 <dbl>, P0.15 <dbl>, P0.2 <dbl>, P0.25 <dbl>, P0.3 <dbl>,
#  #   S0.05 <dbl>, S0.1 <dbl>, S0.15 <dbl>, S0.2 <dbl>, S0.25 <dbl>,
```

Get rid of padding in annotation

```
cleaned_data <- raw_data %>%
  separate(NAME, c("name", "BP", "MF", "systematic_name", "number"),
           sep = "\\|\\|") %>%
  mutate_at(vars(name:systematic_name), funs(stringr::str_trim))
head(cleaned_data)
```

```
#  # A tibble: 6 x 44
#          GID       YORF  name                              BP
#        <chr>      <chr> <chr>                           <chr>
#  1 GENE1331X A_06_P5820  SFB2        ER to Golgi transport
#  2 GENE4924X A_06_P5866         biological process unknown
#  3 GENE4690X A_06_P1834  QRI7 proteolysis and peptidolysis
#  4 GENE1177X A_06_P4928  CFT2      mRNA polyadenylylation*
#  5  GENE511X A_06_P5620  SSO2                vesicle fusion*
#  6 GENE2133X A_06_P5307  PSP2   biological process unknown
#  # ... with 40 more variables: MF <chr>, systematic_name <chr>,
#  #   number <chr>, GWEIGHT <int>, G0.05 <dbl>, G0.1 <dbl>, G0.15 <dbl>,
#  #   G0.2 <dbl>, G0.25 <dbl>, G0.3 <dbl>, N0.05 <dbl>, N0.1 <dbl>,
#  #   N0.15 <dbl>, N0.2 <dbl>, N0.25 <dbl>, N0.3 <dbl>, P0.05 <dbl>,
#  #   P0.1 <dbl>, P0.15 <dbl>, P0.2 <dbl>, P0.25 <dbl>, P0.3 <dbl>,
```

Get rid of some columns

```
cleaned_data <- raw_data %>%
  separate(NAME, c("name", "BP", "MF", "systematic_name", "number"),
           sep = "\\|\\|") %>%
  mutate_at(vars(name:systematic_name), funs(stringr::str_trim)) %>%
  select(-number, -GID, -YORF, -GWEIGHT)
head(cleaned_data)
```

```
#  # A tibble: 6 x 40
#     name                              BP                              MF
#    <chr>                           <chr>                           <chr>
#  1  SFB2              ER to Golgi transport    molecular function unknown
#  2         biological process unknown    molecular function unknown
#  3  QRI7 proteolysis and peptidolysis metalloendopeptidase activity
#  4  CFT2      mRNA polyadenylylation*                     RNA binding
#  5  SSO2                 vesicle fusion*                t-SNARE activity
#  6  PSP2   biological process unknown    molecular function unknown
#  # ... with 37 more variables: systematic_name <chr>, G0.05 <dbl>,
#  #   G0.1 <dbl>, G0.15 <dbl>, G0.2 <dbl>, G0.25 <dbl>, G0.3 <dbl>,
#  #   N0.05 <dbl>, N0.1 <dbl>, N0.15 <dbl>, N0.2 <dbl>, N0.25 <dbl>,
#  #   N0.3 <dbl>, P0.05 <dbl>, P0.1 <dbl>, P0.15 <dbl>, P0.2 <dbl>,
```

Make data tidy

```
cleaned_data <- raw_data %>%
  separate(NAME, c("name", "BP", "MF", "systematic_name", "number"),
           sep = "\\|\\|") %>%
  mutate_at(vars(name:systematic_name), funs(stringr::str_trim)) %>%
  select(-number, -GID, -YORF, -GWEIGHT) %>%
  gather(sample, expression, G0.05:U0.3)
head(cleaned_data)
```

```
#  # A tibble: 6 x 6
#     name                          BP                              MF
#    <chr>                       <chr>                           <chr>
#  1  SFB2        ER to Golgi transport    molecular function unknown
#  2         biological process unknown    molecular function unknown
#  3  QRI7 proteolysis and peptidolysis metalloendopeptidase activity
#  4  CFT2      mRNA polyadenylylation*                 RNA binding
#  5  SSO2                vesicle fusion*             t-SNARE activity
#  6  PSP2   biological process unknown    molecular function unknown
#  # ... with 3 more variables: systematic_name <chr>, sample <chr>,
#  #   expression <dbl>
```

Split columns

```
cleaned_data <- raw_data %>%
  separate(NAME, c("name", "BP", "MF", "systematic_name", "number"),
           sep = "\\|\\|") %>%
  mutate_at(vars(name:systematic_name), funs(stringr::str_trim)) %>%
  select(-number, -GID, -YORF, -GWEIGHT) %>%
  gather(sample, expression, G0.05:U0.3) %>%
  separate(sample, c("nutrient", "rate"), sep=1, convert = TRUE)
head(cleaned_data)
```

```
#  # A tibble: 6 x 7
#     name                              BP                              MF
#    <chr>                           <chr>                           <chr>
#  1  SFB2        ER to Golgi transport    molecular function unknown
#  2         biological process unknown    molecular function unknown
#  3  QRI7 proteolysis and peptidolysis metalloendopeptidase activity
#  4  CFT2      mRNA polyadenylylation*                  RNA binding
#  5  SSO2                vesicle fusion*              t-SNARE activity
#  6  PSP2   biological process unknown    molecular function unknown
#  # ... with 4 more variables: systematic_name <chr>, nutrient <chr>,
#  #   rate <dbl>, expression <dbl>
```

Get rid of rows with missing expression or name

```
cleaned_data <- raw_data %>%
  separate(NAME, c("name", "BP", "MF", "systematic_name", "number"),
           sep = "\\|\\|") %>%
  mutate_at(vars(name:systematic_name), funs(stringr::str_trim)) %>%
  select(-number, -GID, -YORF, -GWEIGHT) %>%
  gather(sample, expression, G0.05:U0.3) %>%
  separate(sample, c("nutrient", "rate"), sep=1, convert = TRUE) %>%
  filter(!is.na(expression), systematic_name != '')
head(cleaned_data)
```

```
#  # A tibble: 6 x 7
#     name                          BP                              MF
#    <chr>                       <chr>                           <chr>
#  1  SFB2        ER to Golgi transport    molecular function unknown
#  2         biological process unknown    molecular function unknown
#  3  QRI7 proteolysis and peptidolysis metalloendopeptidase activity
#  4  CFT2      mRNA polyadenylylation*                 RNA binding
#  5  SSO2                vesicle fusion*              t-SNARE activity
```

Visualize

```
cleaned_data %>%
  filter(BP == "leucine biosynthesis") %>%
  ggplot(aes(rate, expression, color = nutrient)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(~name + systematic_name, nrow=1) +
  theme(legend.position='bottom')
```