Data wRangling

Monash University

Di Cook (dicook@monash.edu, @visnut) and Carson Sievert
(cpsievert1@gmail.com, @cpsievert)

2015-11-25

Warmups - Problem 1

10 week sensory experiment, 12 individuals assessed taste of french fries on several scales (how potato-y, buttery, grassy, rancid, paint-y do they taste?), fried in one of 3 different oils, replicated twice.
First few rows:

time	treatment	subject	rep	potato	buttery	grassy	rancid	pa
1	1	3	1	2.9	0.0	0.0	0.0	
1	1	3	2	14.0	0.0	0.0	1.1	
1	1	10	1	11.0	6.4	0.0	0.0	
1	1	10	2	9.9	5.9	2.9	2.2	

What do you want to know?

Warmups - Problem 2

What's in the column names of this data?

id	WI-6.R1	WI-6.R2	WI-6.R4	WM-6.R1	WM-6.R2	WI-12
Gene 1	2.2	2.20	4.2	2.63	5.1	
Gene 2	1.5	0.59	1.9	0.52	2.9	
Gene 3	2.0	0.87	3.3	0.53	4.6	

Warmups - Problem 3

How many ways can you write today's date?

What we are going to cover today

- Reading different data formats
- Tidying data
- Split - apply - combine
- Pipes
- Joins
- Working with dates
- Splitting strings

Reading different data formats

- images: `library(EBImage)`
- sound: `library(tuneR)`
- fixed width fields: `read.fwf()`
- netCDF: `library(ncdf)`
- hdf5: `library(hdf5)`
- json: `library(jsonlite)`

XML/HTML

```
library(XML)
src ="http://www.realclearpolitics.com/epolls/2012/presider
tables <- readHTMLTable(src)
polls <- tables[[1]]
head(polls)
#>                    Poll        Date Sample MoE Romney  Sa
#> 1          RCP Average  4/9 - 4/17      --  --    52.8
#> 2    CBS News/NY Times 4/13 - 4/17 268 RV 6.0      54
#> 3 CNN/Opinion Research 4/13 - 4/15  473 A 4.5      57
#> 4               PPP (D) 4/12 - 4/15 742 RV 3.6      54
#> 5             FOX News  4/9 - 4/11 376 RV 5.0      46
#>   Paul  Perry  Huntsman  Bachmann  Cain          Spread
#> 1  15.0                                  Romney +33.8
#> 2    12     --        --        --    --   Romney +34
#> 3    18     --        --        --    --   Romney +38
#> 4    14     --        --        --    --   Romney +30
#> 5    16     --        --        --    --   Romney +30
```

GIS

This code is a bit slow to run, but it draws all the electoral districts of Australia.

```
library(maptools)
xx <- readShapeSpatial("http://dicook.github.io/Monash-R/da
object.size(as(xx, "SpatialPolygons"))
xxx <- thinnedSpatialPoly(as(xx, "SpatialPolygons"),
  tolerance=0.5, minarea=0.001, topologyPreserve=TRUE)
object.size(as(xxx, "SpatialPolygons"))
qplot(long, lat, data=xx, group=group) + geom_path() + coor
```

French fries - hot chips

10 week sensory experiment, 12 individuals assessed taste of french fries on several scales (how potato-y, buttery, grassy, rancid, paint-y do they taste?), fried in one of 3 different oils, replicated twice.
First few rows:

time	treatment	subject	rep	potato	buttery	grassy	rancid	pa
1	1	3	1	2.9	0.0	0.0	0.0	
1	1	3	2	14.0	0.0	0.0	1.1	
1	1	10	1	11.0	6.4	0.0	0.0	
1	1	10	2	9.9	5.9	2.9	2.2	
1	1	15	1	1.2	0.1	0.0	1.1	
1	1	15	2	8.8	3.0	3.6	1.5	

What would we like to know?

- Is the design complete?
- Are replicates like each other?
- How do the ratings on the different scales differ?
- Are raters giving different scores on average?
- Do ratings change over the weeks?

Each of these questions involves different summaries of the data.

What we have and what we want

Gathering

- When gathering, you need to specify the **keys** (identifiers) and the **values** (measures).

Keys/Identifiers: - Identify a record (must be unique) - Example: Indices on an random variable - Fixed by design of experiment (known in advance) - May be single or composite (may have one or more variables)

Values/Measures: - Collected during the experiment (not known in advance) - Usually numeric quantities

Gathering the French Fries

```
library(tidyr)
ff_long <- gather(french_fries, key = variable, value = rat
head(ff_long)
#>   time treatment subject rep variable rating
#> 1    1         1       3   1   potato    2.9
#> 2    1         1       3   2   potato   14.0
#> 3    1         1      10   1   potato   11.0
#> 4    1         1      10   2   potato    9.9
#> 5    1         1      15   1   potato    1.2
#> 6    1         1      15   2   potato    8.8
```

Long to Wide

In certain applications, we may wish to take a long dataset and convert it to a wide dataset (Perhaps displaying in a table).

This is called “spreading” the data.

Spread

We use the **spread** function from tidyr to do this:

```
french_fries_wide <- spread(ff_long, key = variable, value

head(french_fries_wide)
#>   time treatment subject rep potato buttery grassy ranc
#> 1    1         1       3   1    2.9     0.0    0.0    0
#> 2    1         1       3   2   14.0     0.0    0.0    1
#> 3    1         1      10   1   11.0     6.4    0.0    0
#> 4    1         1      10   2    9.9     5.9    2.9    2
#> 5    1         1      15   1    1.2     0.1    0.0    1
#> 6    1         1      15   2    8.8     3.0    3.6    1
```

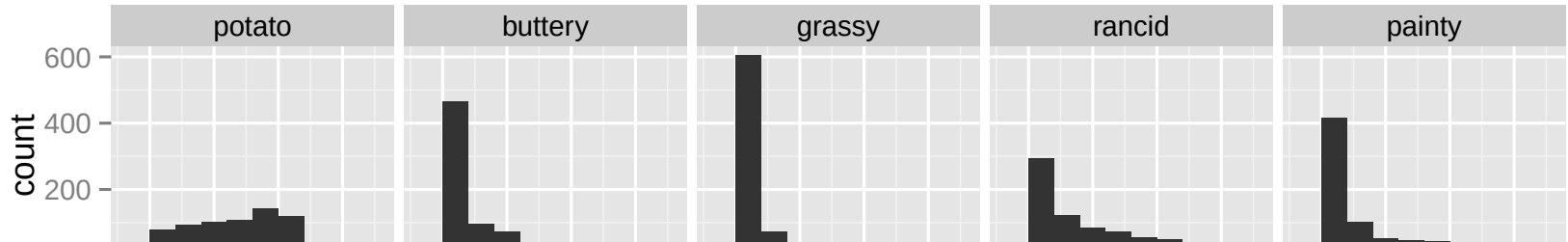
Lets use gather and spread to answer some questions

Easiest question to start is whether the ratings are similar on the different scales, potato'y, buttery, grassy, rancid and painty.

We need to gather the data into long form, and make plots facetted by the scale.

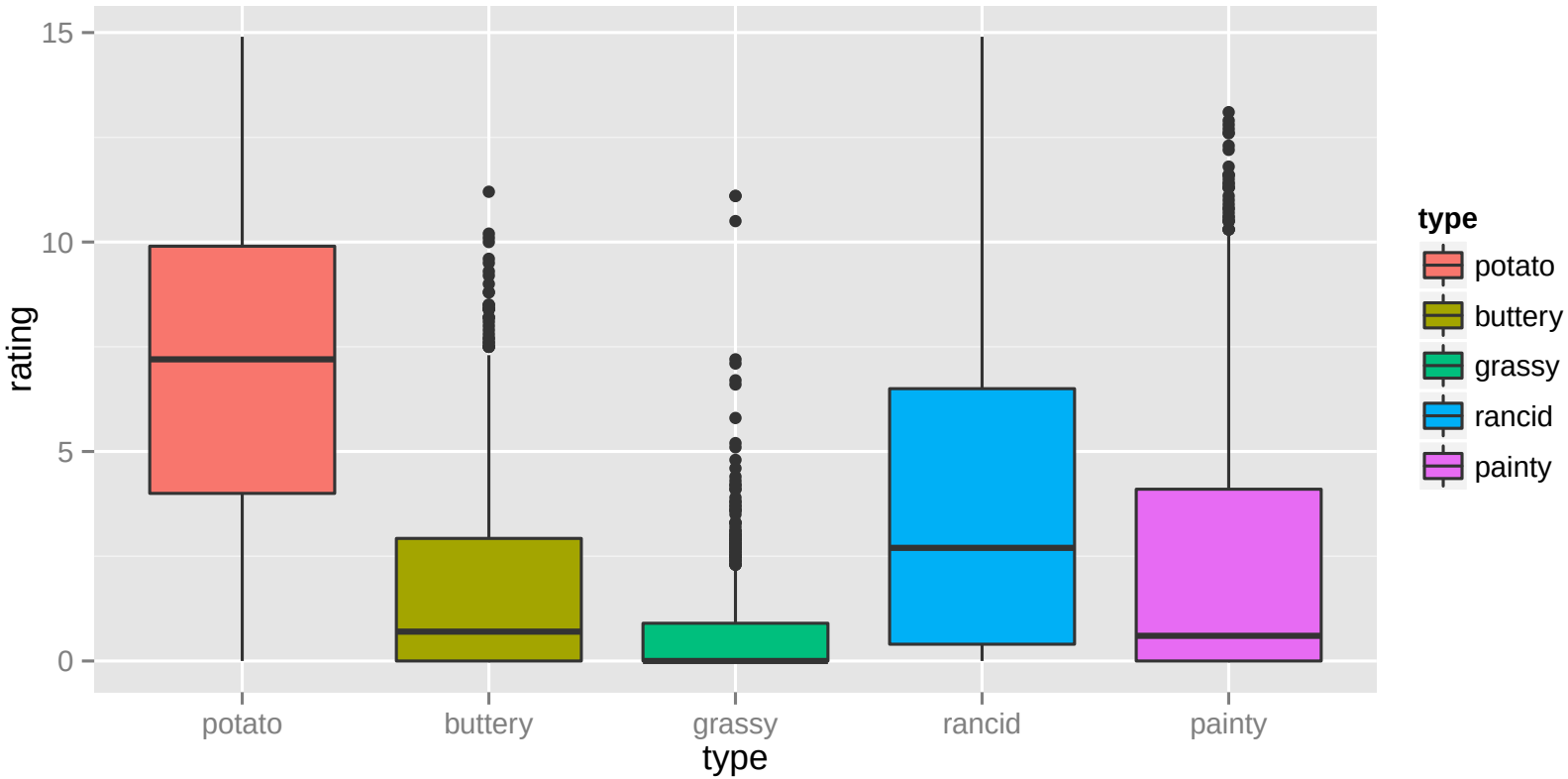
Ratings on the different scales

```
library(ggplot2)
ff.m <- french_fries %>%
  gather(type, rating, -subject, -time, -treatment, -rep)
head(ff.m)
#>   time treatment subject rep    type rating
#> 1    1         1       3   1 potato    2.9
#> 2    1         1       3   2 potato   14.0
#> 3    1         1      10   1 potato   11.0
#> 4    1         1      10   2 potato    9.9
#> 5    1         1      15   1 potato    1.2
#> 6    1         1      15   2 potato    8.8
qplot(rating, data=ff.m, binwidth=2) +
  facet_wrap(~type, ncol=5)
```



Side-by-side boxplots

```
qplot(type, rating, data = ff.m, fill = type, geom = "boxpl
```



Do the replicates look like each other?

We will start to tackle this by plotting the replicates against each other using a scatterplot.

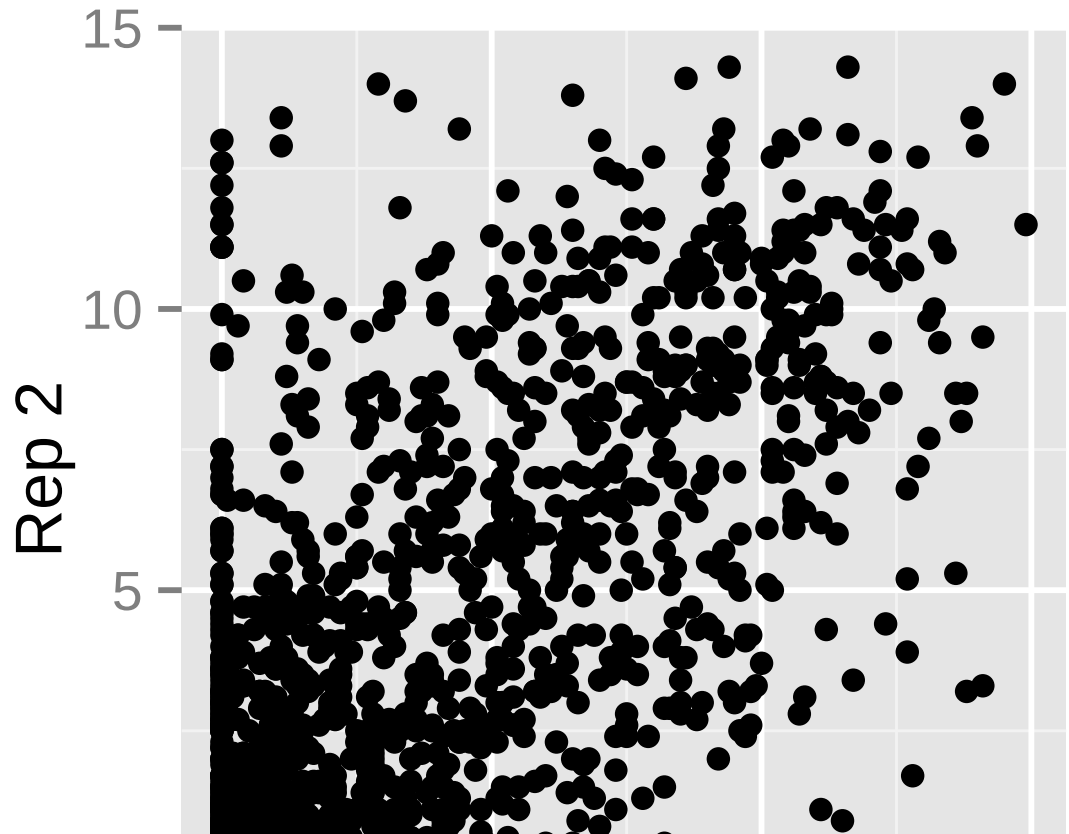
We need to gather the data into long form, and then get the replicates spread into separate columns.

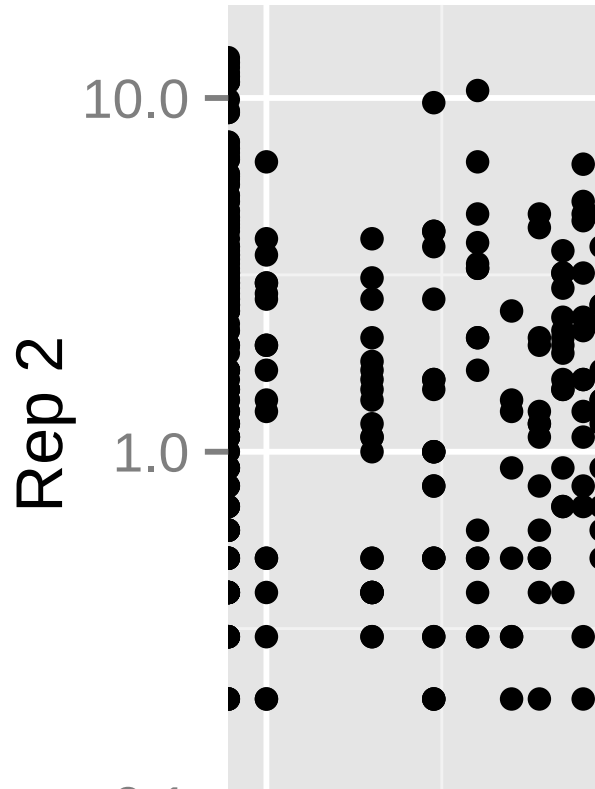
Check replicates

```
head(ff.m)
#>   time treatment subject rep    type rating
#> 1    1         1       3   1 potato    2.9
#> 2    1         1       3   2 potato   14.0
#> 3    1         1      10   1 potato   11.0
#> 4    1         1      10   2 potato    9.9
#> 5    1         1      15   1 potato    1.2
#> 6    1         1      15   2 potato    8.8
ff.s <- ff.m %>% spread(rep, rating)
head(ff.s)
#>   time treatment subject    type    1    2
#> 1    1         1       3  potato  2.9 14.0
#> 2    1         1       3 buttery  0.0  0.0
#> 3    1         1       3  grassy  0.0  0.0
#> 4    1         1       3  rancid  0.0  1.1
#> 5    1         1       3  painty  5.5  0.0
#> 6    1         1      10  potato 11.0  9.9
```

Check replicates

```
qplot(`1`, `2`, data=ff.s) + theme(aspect.ratio=1) +
  xlab("Rep 1") + ylab("Rep 2")
qplot(`1`, `2`, data=ff.s) + theme(aspect.ratio=1) +
  xlab("Rep 1") + ylab("Rep 2") +
  scale_x_log10() + scale_y_log10()
```

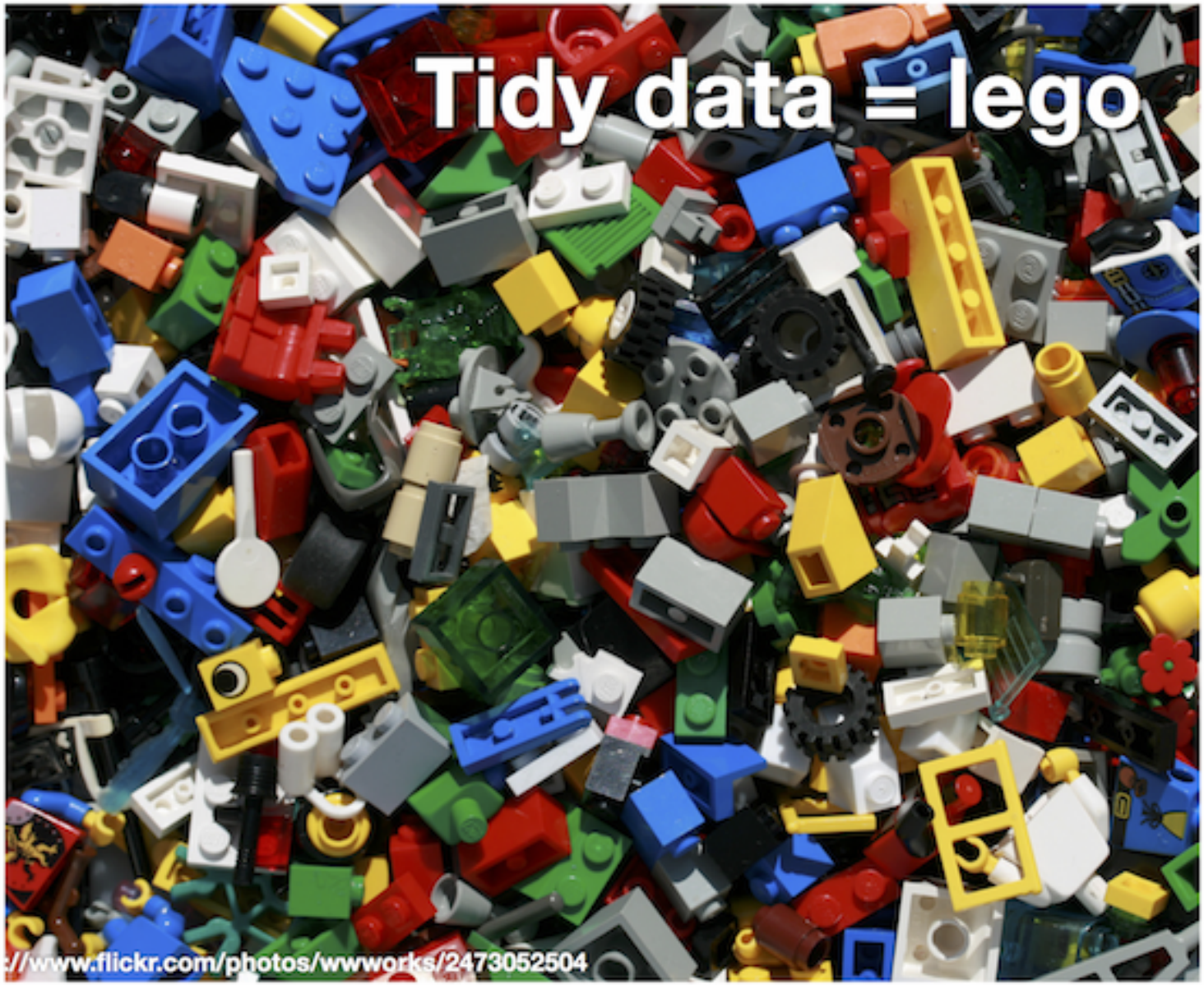




Your turn

Make the scatterplots of reps against each other separately for scales, and treatment.

Legos = tidy data



(Courtesy of Hadley Wickham)

Play mobile = messy data

(Courtesy of Hadley Wickham)

Your turn

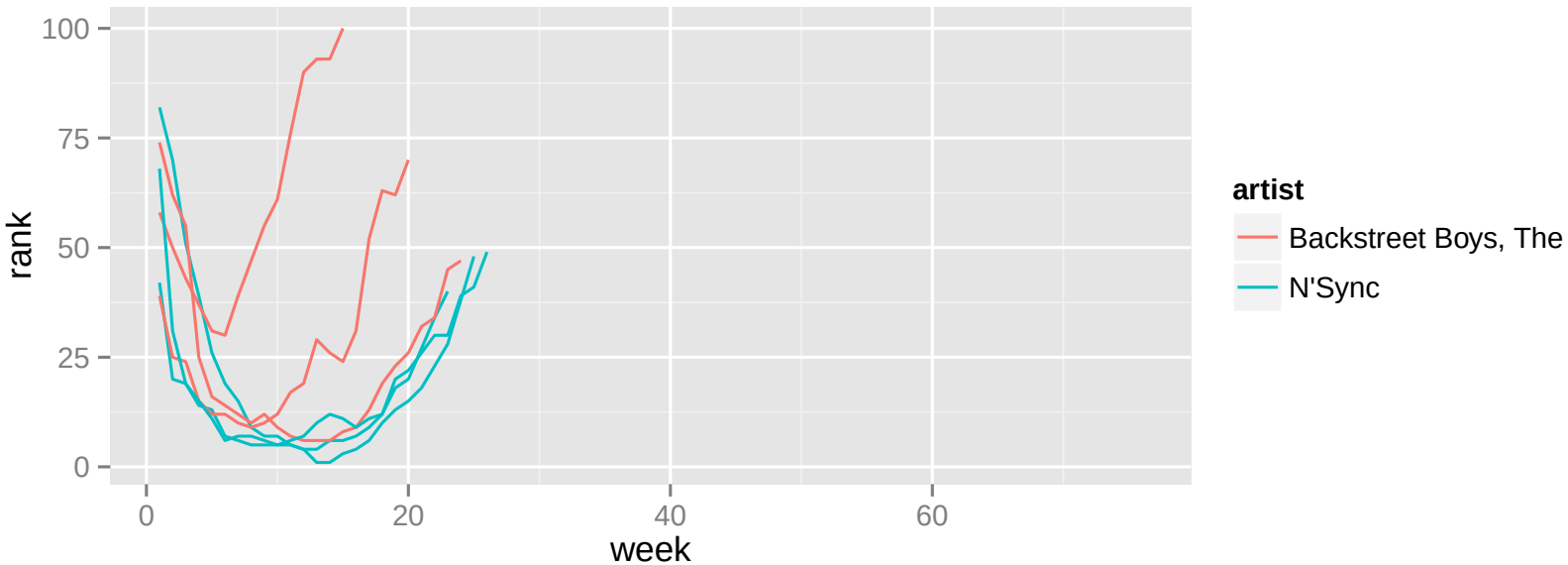
Read in the billboard top 100 music data, which contains N'Sync and Backstreet Boys songs that entered the billboard charts in the year 2000

```
billboard <- read.csv("http://dicook.github.io/Monash-R/dat
```

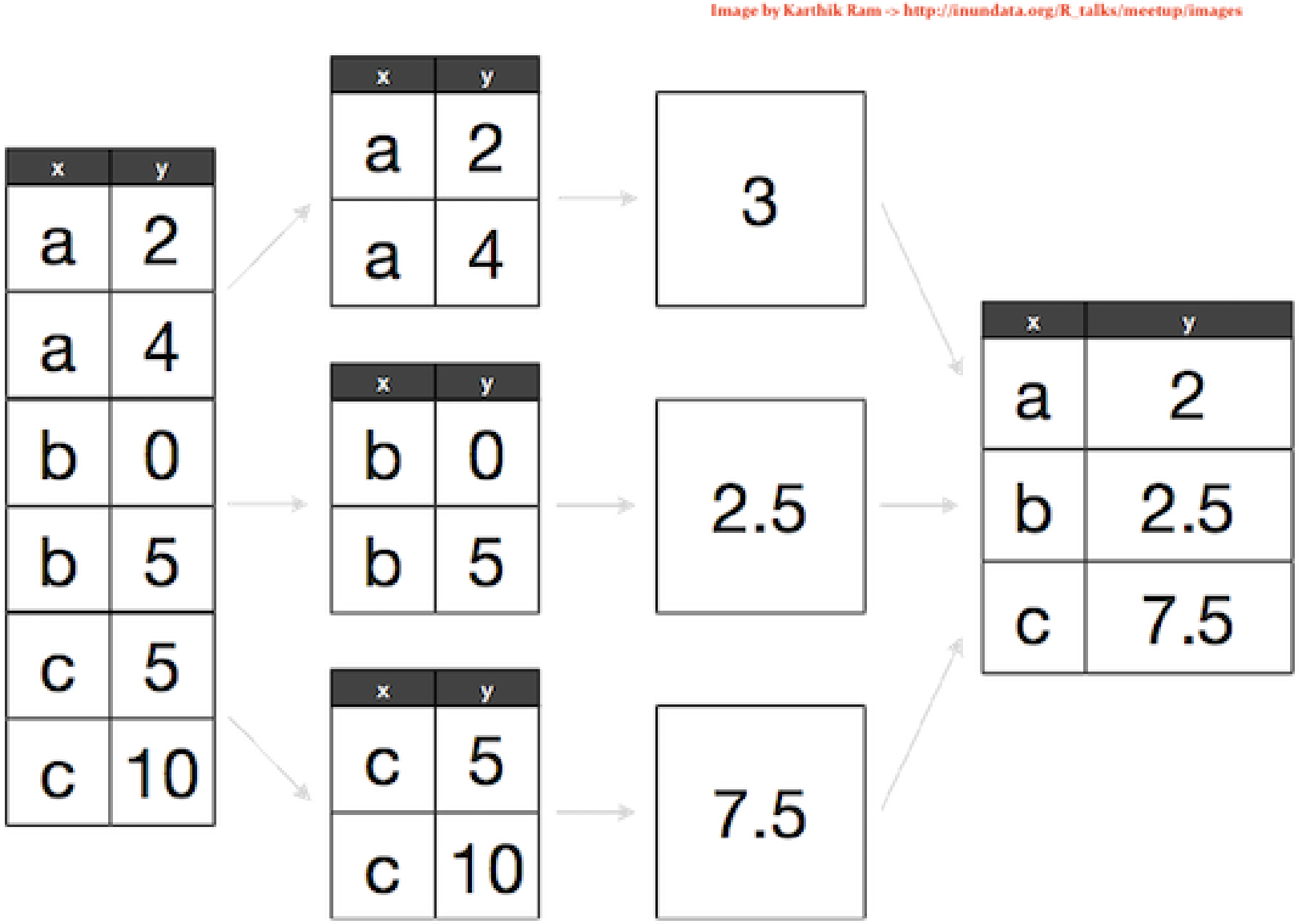
What's in this data? What's X1-X76?

Your turn

1. Use `tidyr` to convert this data into a long format appropriate for plotting a time series (date on the x axis, chart position on the y axis)
2. Use `ggplot2` to create this time series plot:



The Split-Apply-Combine Approach



(Diagram originally from Hadley Wickham)

Split-Apply-Combine in dplyr

```
library(dplyr)
french_fries_split <- group_by(ff_long, variable) # SPLIT
french_fries_apply <- summarise(french_fries_split, rating
french_fries_apply
#> Source: local data frame [5 x 2]
#>
#>    variable rating
#>      (fctr)  (dbl)
#> 1    potato   6.95
#> 2   buttery   1.82
#> 3    grassy   0.66
#> 4    rancid   3.85
#> 5    painty   2.52
```

The pipe operator

- dplyr allows us to chain together these data analysis tasks using the %>% (pipe) operator
- `x %>% f(y)` is shorthand for `f(x, y)`
- Example:

```
french_fries %>%
    gather(key = variable, value = rating, potato:painty) %
    group_by(variable) %>%
    summarise(rating = mean(rating, na.rm = TRUE))
#> Source: local data frame [5 x 2]
#>
#>   variable rating
#>      (fctr)  (dbl)
#> 1   potato   6.95
#> 2  buttery   1.82
#> 3   grassy   0.66
#> 4   rancid   3.85
#> 5   painty   2.52
```

dplyr verbs

There are five primary dplyr **verbs**, representing distinct data analysis tasks:

- Filter: Remove the rows of a data frame, producing subsets
- Arrange: Reorder the rows of a data frame
- Select: Select particular columns of a data frame
- Mutate: Add new columns that are functions of existing columns
- Summarise: Create collapsed summaries of a data frame

Filter

```
french_fries %>%
    filter(subject == 3, time == 1)
#>   time treatment subject rep potato buttery grassy ranc
#> 1    1         1       3   1    2.9     0.0    0.0    0
#> 2    1         1       3   2   14.0     0.0    0.0    1
#> 3    1         2       3   1   13.9     0.0    0.0    3
#> 4    1         2       3   2   13.4     0.1    0.0    1
#> 5    1         3       3   1   14.1     0.0    0.0    1
#> 6    1         3       3   2    9.5     0.0    0.6    2
```

Arrange

```
french_fries %>%
    arrange(desc(rancid)) %>%
    head
#>    time treatment subject rep potato buttery grassy ranc
#> 1     9         2      51   1    7.3     2.3      0    .
#> 2    10         1      86   2    0.7     0.0      0    .
#> 3     5         2      63   1    4.4     0.0      0    .
#> 4     9         2      63   1    1.8     0.0      0    .
#> 5     5         2      19   2    5.5     4.7      0    .
#> 6     4         3      63   1    5.6     0.0      0    .
```

Select

```
french_fries %>%
    select(time, treatment, subject, rep, potato) %>%
    head
#>     time treatment subject rep potato
#> 61     1         1       3   1    2.9
#> 25     1         1       3   2   14.0
#> 62     1         1      10   1   11.0
#> 26     1         1      10   2    9.9
#> 63     1         1      15   1    1.2
#> 27     1         1      15   2    8.8
```

Summarise

```
french_fries %>%
    group_by(time, treatment) %>%
    summarise(mean_rancid = mean(rancid), sd_rancid = sd(ra
#> Source: local data frame [30 x 4]
#> Groups: time [?]
#>
#>      time treatment mean_rancid sd_rancid
#>    (fctr)    (fctr)       (dbl)     (dbl)
#> 1       1         1         2.8       3.2
#> 2       1         2         1.7       2.7
#> 3       1         3         2.6       3.2
#> 4       2         1         3.9       4.4
#> 5       2         2         2.1       3.1
#> 6       2         3         2.5       3.4
#> 7       3         1         4.7       3.9
#> 8       3         2         2.9       3.8
#> 9       3         3         3.6       3.6
#> 10      4         1         2.1       2.4
```

Let's use these tools to answer the rest of the french fries questions

If the data is complete it should be 12 x 10 x 3 x 2, that is, 6 records for each person. (Assuming that each person rated on all scales.)

To check this we want to tabulate the number of records for each subject, time and treatment. This means select appropriate columns, tabulate, count and spread it out to give a nice table.

Check completeness

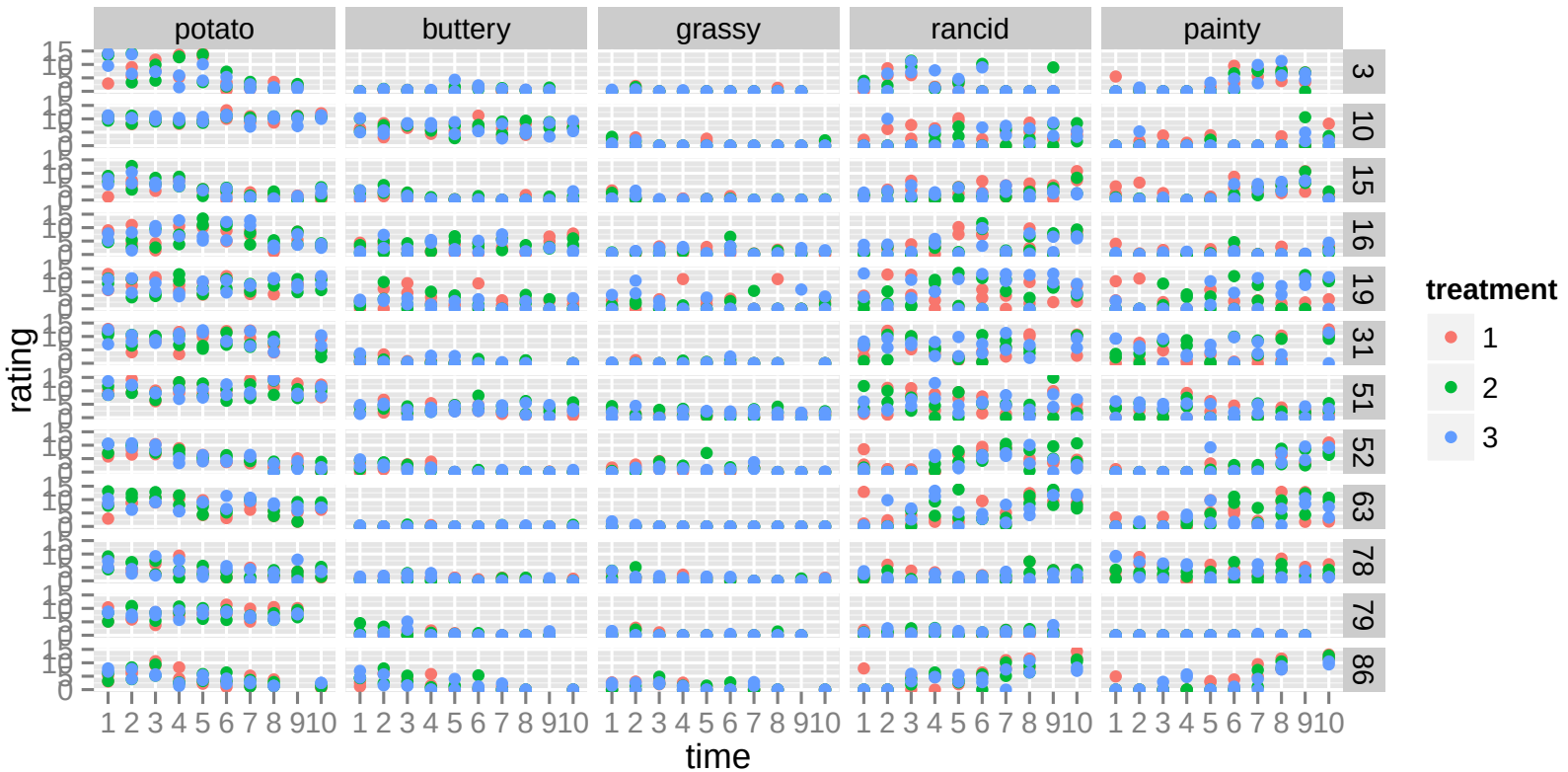
```
french_fries %>%
  select(subject, time, treatment) %>%
  tbl_df() %>%
  count(subject, time) %>%
  spread(time, n)
#> Source: local data frame [12 x 11]
#>
#>    subject     1     2     3     4     5     6     7
#>     (fctr) (int) (int) (int) (int) (int) (int) (int) (i
#> 1       3     6     6     6     6     6     6     6
#> 2      10     6     6     6     6     6     6     6
#> 3      15     6     6     6     6     6     6     6
#> 4      16     6     6     6     6     6     6     6
#> 5      19     6     6     6     6     6     6     6
#> 6      31     6     6     6     6     6     6     6
#> 7      51     6     6     6     6     6     6     6
#> 8      52     6     6     6     6     6     6     6
#> 9      63     6     6     6     6     6     6     6
```

Check completeness with different scales, too

```
french_fries %>%
  gather(type, rating, -subject, -time, -treatment, -rep) %
  select(subject, time, treatment, type) %>%
  tbl_df() %>%
  count(subject, time) %>%
  spread(time, n)
#> Source: local data frame [12 x 11]
#>
#>    subject     1     2     3     4     5     6     7
#>     (fctr) (int) (int) (int) (int) (int) (int) (int) (i
#> 1       3    30    30    30    30    30    30    30
#> 2      10    30    30    30    30    30    30    30
#> 3      15    30    30    30    30    30    30    30
#> 4      16    30    30    30    30    30    30    30
#> 5      19    30    30    30    30    30    30    30
#> 6      31    30    30    30    30    30    30    30
#> 7      51    30    30    30    30    30    30    30
#> 8      52    30    30    30    30    30    30    30
```

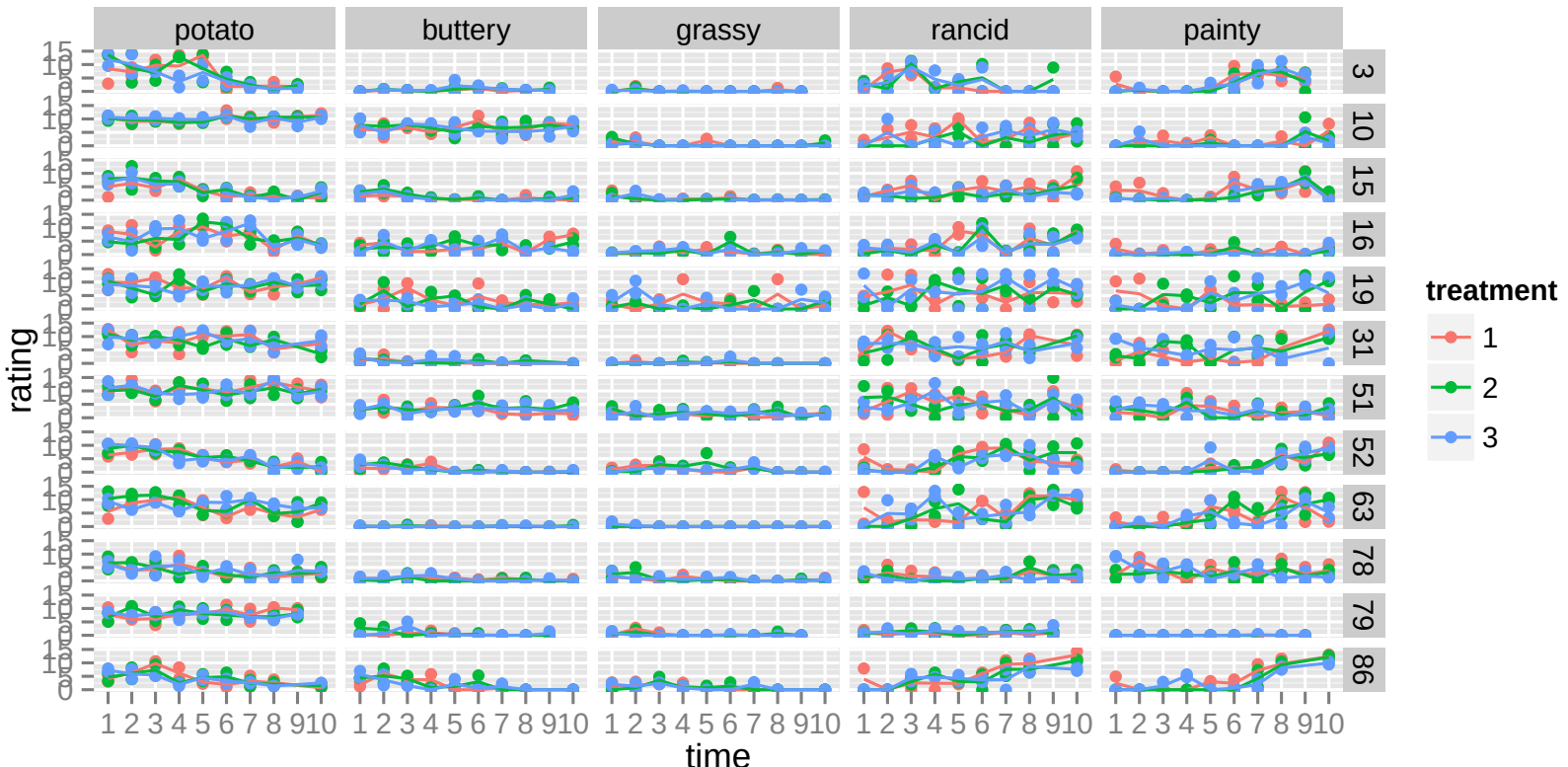
Change in ratings over weeks, relative to experimental design

```
qplot(time, rating, data=ff.m, colour=treatment) +
  facet_grid(subject~type)
```



Add means over reps, and connect the dots

```
ff.m.av <- ff.m %>%
  group_by(subject, time, type, treatment) %>%
  summarise(rating=mean(rating))
qplot(time, rating, data=ff.m, colour=treatment) +
  facet_grid(subject~type) +
  geom_line(data=ff.m.av, aes(group=treatment))
```



Your turn

Read in the flights data:

```
library(nycflights13)
flights
#> Source: local data frame [336,776 x 16]
#>
#>      year month   day dep_time dep_delay arr_time arr_de
#>     (int) (int) (int)    (int)     (dbl)    (int)     (d
#> 1   2013     1     1      517         2      830
#> 2   2013     1     1      533         4      850
#> 3   2013     1     1      542         2      923
#> 4   2013     1     1      544        -1     1004
#> 5   2013     1     1      554        -6      812
#> 6   2013     1     1      554        -4      740
```

Your turn

This dataset contains information on over 300,000 flights that departed from New York City in the year 2013.

1. Using dplyr and the pipe operator, create a data frame consisting of the average arrival delay (`arr_delay`) based on the destination airport (`dest`). Sort this data frame in descending order, so the destination airport with the largest delay is first.
2. Find out the most used airports for each airline carrier.

Dates and Times

Dates are deceptively hard to work with in R.

Example: 02/05/2012. Is it February 5th, or May 2nd?

Other things are difficult too:

- Time zones
- POSIXct format in base R is challenging

The **lubridate** package helps tackle some of these issues.

Basic Lubridate Use

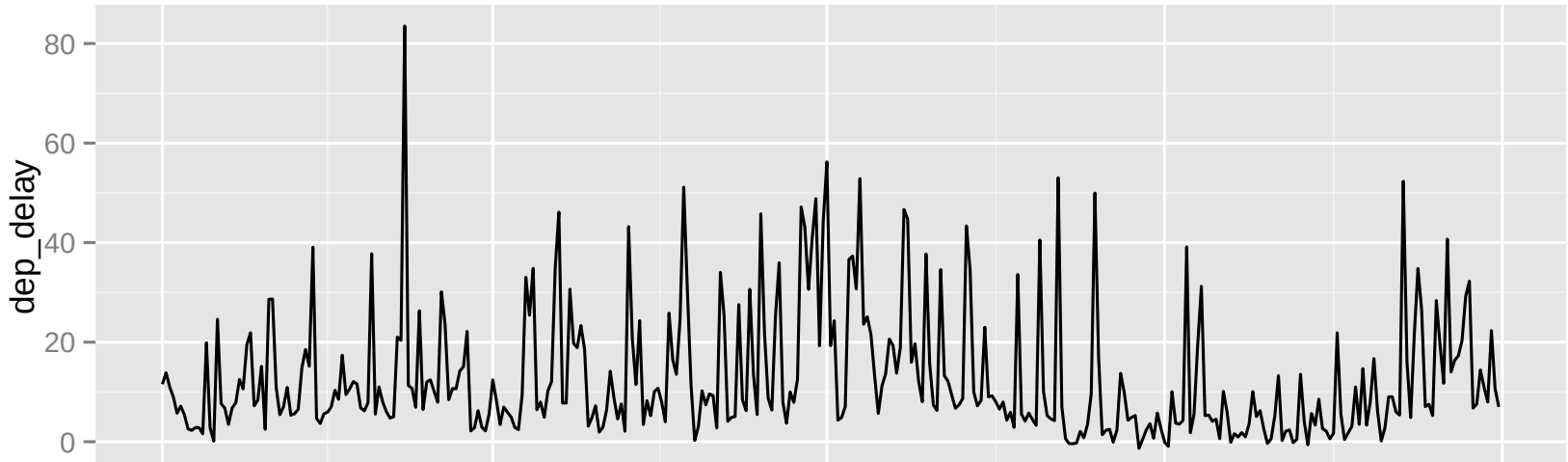
```
library(lubridate)

now()
today()
now() + hours(4)
today() - days(2)
#> [1] "2015-11-25 08:04:45 AEDT"
#> [1] "2015-11-25"
#> [1] "2015-11-25 12:04:45 AEDT"
#> [1] "2015-11-23"
```

Parsing Dates

```
ymd("2013-05-14")
mdy("05/14/2013")
dmy("14052013")
ymd_hms("2013:05:14 14:50:30", tz = "America/Chicago")
#> [1] "2013-05-14 UTC"
#> [1] "2013-05-14 UTC"
#> [1] "2013-05-14 UTC"
#> [1] "2013-05-14 14:50:30 CDT"
```

Your turn

1. Using the `flights` data, create a new column Date using lubridate. You will need to paste together the columns year, month, and day in order to do this. See the `paste` function.
2. Use dplyr to calculate the average departure delay for each date.
3. Plot a time series of the date versus the average departure delay



String manipulation

When the experimental design is packed into the column name, we need to be able to unpack it.

id	WI-6.R1	WI-6.R2	WI-6.R4	WM-6.R1	WM-6.R2	WI-12
Gene 1	2.2	2.20	4.2	2.63	5.1	
Gene 2	1.5	0.59	1.9	0.52	2.9	
Gene 3	2.0	0.87	3.3	0.53	4.6	

Gather column names into long form

id	treatment	expr
Gene 1	WI-6.R1	2.18
Gene 2	WI-6.R1	1.46
Gene 3	WI-6.R1	2.03
Gene 1	WI-6.R2	2.20
Gene 2	WI-6.R2	0.59
Gene 3	WI-6.R2	0.87

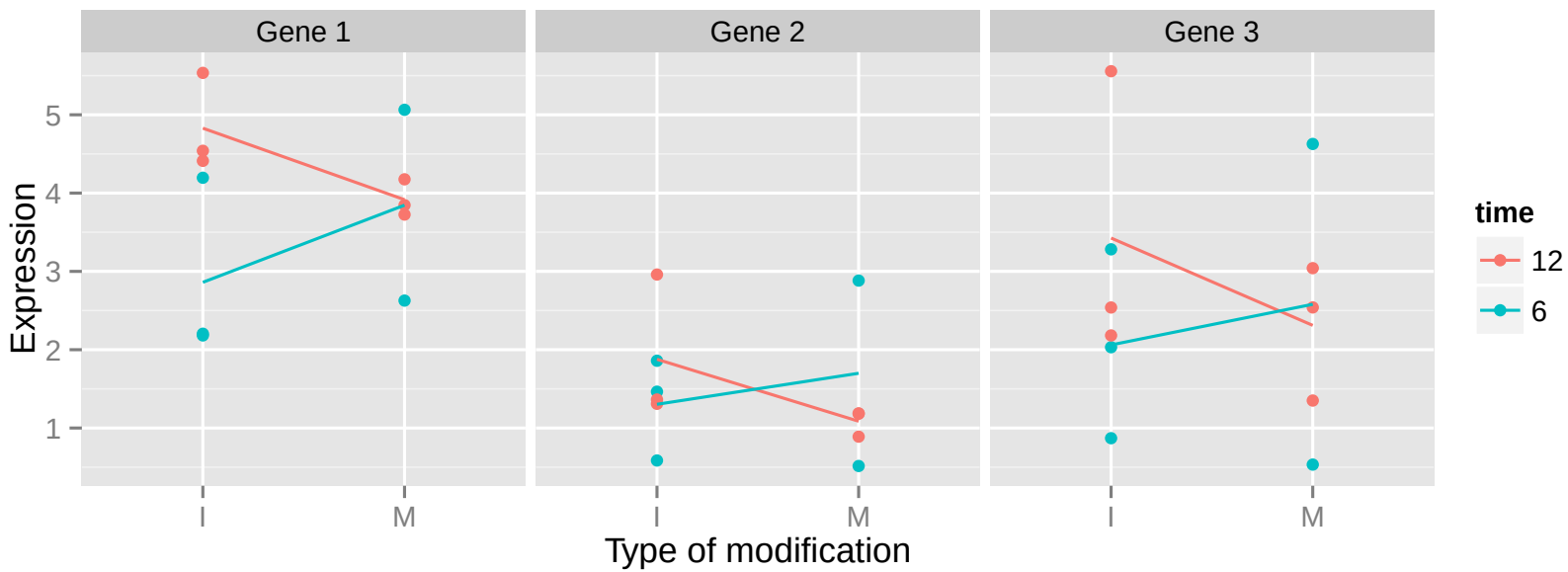
Separate columns

```
genes.m %>%
  separate(treatment, c("trt", "leftover"), "-")
#> Source: local data frame [33 x 4]
#>
#>         id   trt leftover  expr
#>      (chr) (chr)    (chr) (dbl)
#> 1  Gene 1    WI      6.R1  2.18
#> 2  Gene 2    WI      6.R1  1.46
#> 3  Gene 3    WI      6.R1  2.03
#> 4  Gene 1    WI      6.R2  2.20
#> 5  Gene 2    WI      6.R2  0.59
#> 6  Gene 3    WI      6.R2  0.87
#> 7  Gene 1    WI      6.R4  4.20
#> 8  Gene 2    WI      6.R4  1.86
#> 9  Gene 3    WI      6.R4  3.28
#> 10 Gene 1    WM      6.R1  2.63
#> ..    ...   ...       ...   ...
```

```r
genes.m2 <- genes.m %>%
  separate(treatment, c("trt", "leftover"), "-") %>%
  separate(leftover, c("time", "rep"), "\\.") %>%
  mutate(trt = sub("W", "", trt))
genes.m2
#> Source: local data frame [33 x 5]
#>
#>         id   trt  time   rep  expr
#>      (chr) (chr) (chr) (chr) (dbl)
#> 1  Gene 1     I     6    R1  2.18
#> 2  Gene 2     I     6    R1  1.46
#> 3  Gene 3     I     6    R1  2.03
#> 4  Gene 1     I     6    R2  2.20
#> 5  Gene 2     I     6    R2  0.59
#> 6  Gene 3     I     6    R2  0.87
#> 7  Gene 1     I     6    R4  4.20
#> 8  Gene 2     I     6    R4  1.86
#> 9  Gene 3     I     6    R4  3.28
#> 10 Gene 1     M     6    R1  2.63
#>
```

Now we can look it, in various ways

```
genes.m.av <- genes.m2 %>%
  group_by(id, trt, time) %>%
  summarise(expr = mean(expr))
qplot(trt, expr, data = genes.m2, colour = time) +
  xlab("Type of modification") + ylab("Expression") +
  facet_wrap(~id) +
  geom_line(data = genes.m.av, aes(group = time))
```



TODO

- SQL/JOINS with dplyr and flight data? (Carson)
- dplyr do() + tidyr::unnest() (Carson)
- reading different data formats: image, json, audio, xml/html, ncdf, excel/sas/stata, GIS files, (Di)