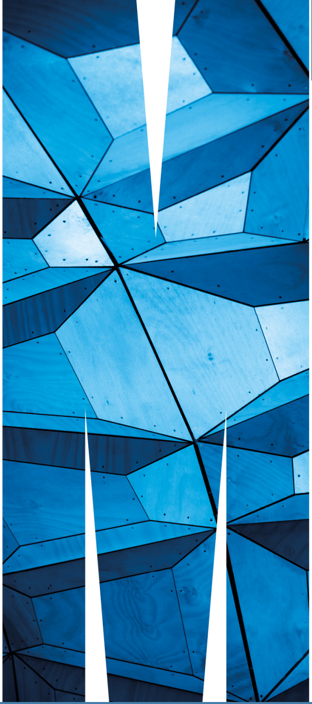
MONASH University

MONASH BUSINESS SCHOOL

Introduction to Data Analysis and Visualisation using R

Professor Di Cook, Econometrics and Business Statistics

Workshop for the Institute for Safety, Compensation and Recovery Research



GROUP OF EIGHT AUSTRALIA

AACSB ACCREDITED · EFMD EQUIS ACCREDITED · ASSOCIATION OF MBAS ACCREDITED

Session 3

Wrangling your data into shape for analysis
(If you re-started RStudio, be sure to re-open your project too.)

Using the packages tidyr, dplyr, purrr

- Writing readable code using **pipes**
- What is **tidy data**? Why do you want tidy data? Getting your data into tidy form using tidyr.
- **Summarise, mutate, filter, select, arrange** with dplyr
- Reading different **data formats**
- String operations, working with **text**
- Re-structuring **time** variables
- Computing on **lists** with purrr (not covered)
- Handling **missing values** (not covered)

Pipes %>%

Pipes allow the code to be read like a sequence of operations

```
# Instead of table(workers$Gender)
workers %>%
  group_by(Gender) %>%
  tally()
#> Source: local data frame [4 x 2]
#>
#>   Gender       n
#>    (chr)   (int)
#> 1      F  901075
#> 2      M 1510517
#> 3      U   36740
#> 4     NA     274
```

```r
# Instead of
# mean(workers$`Birth Year`, na.rm=TRUE)
# sd(workers$`Birth Year`, na.rm=TRUE)
workers %>%
  group_by(Gender) %>%
  filter(Gender %in% c("F", "M")) %>%
  summarise(m=mean(`Birth Year`, na.rm=TRUE),
            s=sd(`Birth Year`, na.rm=TRUE))
#> Source: local data frame [2 x 3]
#>
#>   Gender     m     s
#>    (chr) (dbl) (dbl)
#> 1      F  1964    13
#> 2      M  1965    13
```

Warmups - Problem 1

What are the variables?

Inst	AvNumPubs	AvNumCits	PctCompleti
ARIZONA STATE UNIVERSITY	0.90	1.57	3
AUBURN UNIVERSITY	0.79	0.64	4
BOSTON COLLEGE	0.51	1.03	4
BOSTON UNIVERSITY	0.49	2.66	3
BRANDEIS UNIVERSITY	0.30	3.03	4
BROWN UNIVERSITY	0.84	2.31	5

Warmups - Problem 2

What's in the column names of this data? What are the experimental units? What are the measured variables?

id	WI-6.R1	WI-6.R2	WI-6.R4	WM-6.R1	WM-6.R2	WI-12.R1
Gene 1	2.2	2.20	4.2	2.63	5.1	4.5
Gene 2	1.5	0.59	1.9	0.52	2.9	1.4
Gene 3	2.0	0.87	3.3	0.53	4.6	2.2

Warmups - Problem 3

How many ways can you write today's date?

Warmups - Problem 4

What are the variables? What are the records?

```
melbtemp <- read.fwf("data/ASN00086282.dly",
   c(11, 4, 2, 4, rep(c(5, 1, 1, 1), 31)), fill=T)
kable(head(melbtemp[,c(1,2,3,4,seq(5,128,4))]))
```

V1	V2	V3	V4	V5	V9	V13	V17	V21	V25
ASN00086282	1970	7	TMAX	141	124	113	123	148	149
ASN00086282	1970	7	TMIN	80	63	36	57	69	47
ASN00086282	1970	7	PRCP	3	30	0	0	36	3
ASN00086282	1970	8	TMAX	145	128	150	122	109	112
ASN00086282	1970	8	TMIN	50	61	75	67	41	51
ASN00086282	1970	8	PRCP	0	66	0	53	13	3

Warmups - Problem 5

What are the variables? What are the experimental units?

```
tb <- read_csv("data/tb.csv")
#tail(tb)
colnames(tb)
#>  [1] "iso2"   "year"   "m_04"   "m_514"  "m_014"  "m_1524"
#>  [8] "m_3544" "m_4554" "m_5564" "m_65"   "m_u"    "f_04"
#> [15] "f_014"  "f_1524" "f_2534" "f_3544" "f_4554" "f_5564"
#> [22] "f_u"
```


Warmups - Problem 6

What are the variables? What are the experimental units?

```
pew <- read.delim(
  file = "http://stat405.had.co.nz/data/pew.txt",
  header = TRUE,
  stringsAsFactors = FALSE,
  check.names = F
)
kable(pew[1:5, 1:5])
```

religion	<$10k	$10-20k	$20-30k	$30-40k
Agnostic	27	34	60	81
Atheist	12	27	37	52
Buddhist	27	21	30	34
Catholic	418	617	732	670
Don't know/refused	15	14	15	11

Warmups - Problem 7

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	painty
1	1	3	1	2.9	0.0	0.0	0.0	5.5
1	1	3	2	14.0	0.0	0.0	1.1	0.0
1	1	10	1	11.0	6.4	0.0	0.0	0.0
1	1	10	2	9.9	5.9	2.9	2.2	0.0

What is the experimental unit? What are the factors of the experiment? What was measured? What do you want to know?

Messy data patterns

There are various features of messy data that one can observe in practice. Here are some of the more commonly observed patterns.

- Column headers are values, not variable names
- Variables are stored in both rows and columns, contingency table format
- One type of experimental unit stored in multiple tables
- Dates in many different formats

What is tidy data?

- Each observation forms a row
- Each variable forms a column
- Contained in a single table
- Long form makes it easier to reshape in many different ways
- Wide form is common for analysis

Tidy data

Figure 1:

Messy data

Figure 2:

Tidy verbs

- **gather**: specify the **keys** (identifiers) and the **values** (measures) to make long form (used to be called melting)
- **spread**: variables in columns (used to be called casting)
- nest/unnest: working with lists
- separate/unite: split and combine columns

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	painty
1	1	3	1	2.9	0.0	0.0	0.0	5.5
1	1	3	2	14.0	0.0	0.0	1.1	0.0
1	1	10	1	11.0	6.4	0.0	0.0	0.0
1	1	10	2	9.9	5.9	2.9	2.2	0.0
1	1	15	1	1.2	0.1	0.0	1.1	5.1
1	1	15	2	8.8	3.0	3.6	1.5	2.3

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.

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

```
ff_long <- gather(french_fries, key = variable, value =
                    rating, potato:painty)
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 = rating)

head(french_fries_wide)
#>   time treatment subject rep buttery grassy painty potato r
#> 1    1         1       3   1     0.0    0.0    5.5    2.9
#> 2    1         1       3   2     0.0    0.0    0.0   14.0
#> 3    1         1      10   1     6.4    0.0    0.0   11.0
#> 4    1         1      10   2     5.9    2.9    0.0    9.9
#> 5    1         1      15   1     0.1    0.0    5.1    1.2
#> 6    1         1      15   2     3.0    3.6    2.3    8.8
```


Gather and spread to answer some questions

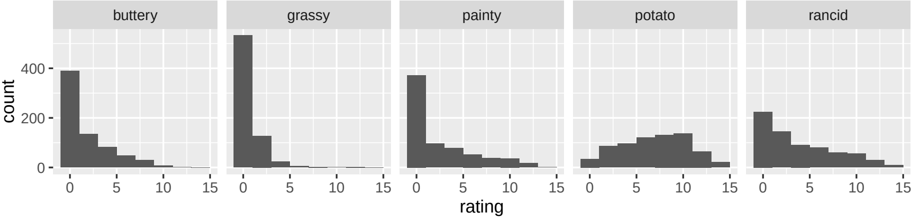
- Easiest question to start is whether the ratings are similar on the different scales, potato'y, buttery, grassy, rancid and painty.
- We need to gather the data into long form, and make plots facetted by the scale.

Ratings on the different scales

```
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
```

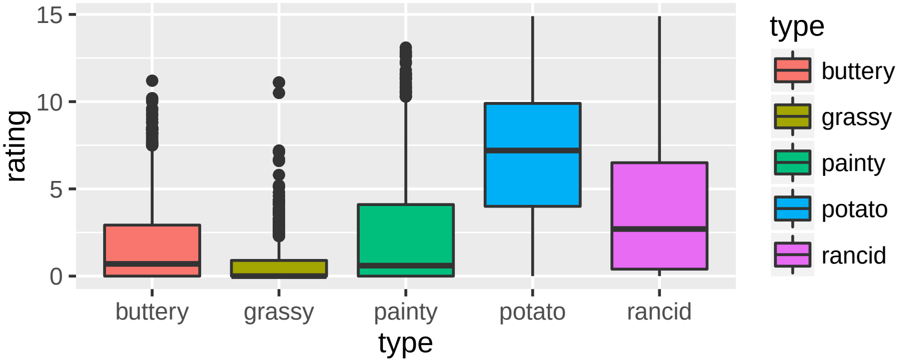


```
ggplot(data=ff.m, aes(x=rating)) + geom_histogram(binwidth=2) +
  facet_wrap(~type, ncol=5)
```



Side-by-side boxplots

```
ggplot(data=ff.m, aes(x=type, y=rating, fill=type)) +
  geom_boxplot()
```



Do the replicates look like each other?

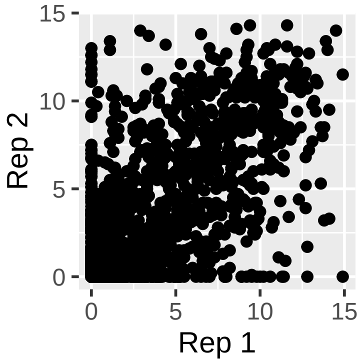
We will start to tackle this by plotting the replicates against each other using a scatterplot.
We need to gather the data into long form, and then get the replicates spread into separate columns.

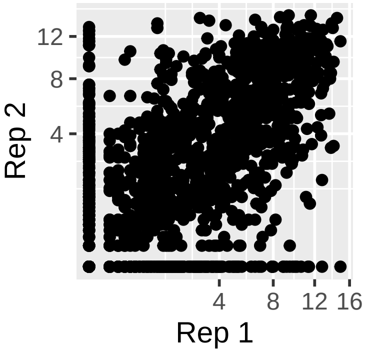
Check replicates

```
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 buttery 0.0  0.0
#> 2    1         1       3  grassy 0.0  0.0
#> 3    1         1       3  painty 5.5  0.0
#> 4    1         1       3  potato 2.9 14.0
#> 5    1         1       3  rancid 0.0  1.1
```

Check replicates

```
ggplot(data=ff.s, aes(x=`1`, y=`2`)) + geom_point() +
  theme(aspect.ratio=1) + xlab("Rep 1") + ylab("Rep 2")
ggplot(data=ff.s, aes(x=`1`, y=`2`)) + geom_point() +
  theme(aspect.ratio=1) + xlab("Rep 1") + ylab("Rep 2") +
  scale_x_sqrt() + scale_y_sqrt()
```





Your turn

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

Your turn

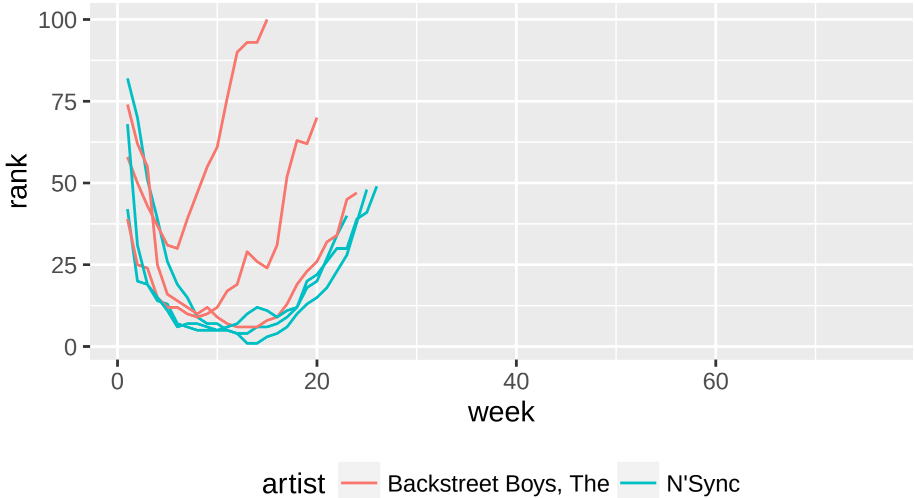
Read in the billboard top 100 music data, which contains N'Sync and Backstreet Boys songs that entered the billboard charts in the year 2000

```
billboard <- read.csv("data/billboard.csv")
```

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:



dplyr verbs

The package dplyr helps to make various summaries of the data. 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

The Split-Apply-Combine Approach

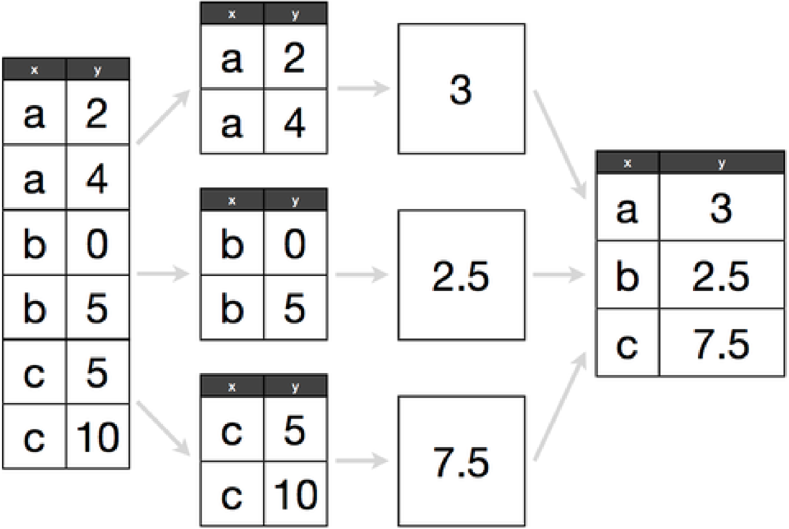


Figure 3:

Split-Apply-Combine in dplyr

```
french_fries_split <- group_by(ff_long, variable) # SPLIT
french_fries_apply <- summarise(french_fries_split,
     rating = mean(rating, na.rm = TRUE)) # APPLY + COMBINE
french_fries_apply
#> Source: local data frame [5 x 2]
#>
#>   variable rating
#>      (chr)  (dbl)
#> 1  buttery   1.82
#> 2   grassy   0.66
#> 3   painty   2.52
#> 4   potato   6.95
#> 5   rancid   3.85
```

Filter

```
french_fries %>%
    filter(subject == 3, time == 1)
#>   time treatment subject rep potato buttery grassy rancid p
#> 1    1         1       3   1    2.9     0.0    0.0    0.0
#> 2    1         1       3   2   14.0     0.0    0.0    1.1
#> 3    1         2       3   1   13.9     0.0    0.0    3.9
#> 4    1         2       3   2   13.4     0.1    0.0    1.5
#> 5    1         3       3   1   14.1     0.0    0.0    1.1
#> 6    1         3       3   2    9.5     0.0    0.6    2.8
```

Arrange

```
french_fries %>%
    arrange(desc(rancid)) %>%
    head
#>    time treatment subject rep potato buttery grassy rancid p
#> 1     9         2      51   1    7.3     2.3      0     15
#> 2    10         1      86   2    0.7     0.0      0     14
#> 3     5         2      63   1    4.4     0.0      0     14
#> 4     9         2      63   1    1.8     0.0      0     14
#> 5     5         2      19   2    5.5     4.7      0     13
#> 6     4         3      63   1    5.6     0.0      0     13
```

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(rancid))
#> 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
```


Answering remaining 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) %>% kable
```

subject	1	2	3	4	5	6	7	8	9	10
3	6	6	6	6	6	6	6	6	6	NA
10	6	6	6	6	6	6	6	6	6	6
15	6	6	6	6	6	6	6	6	6	6
16	6	6	6	6	6	6	6	6	6	6
19	6	6	6	6	6	6	6	6	6	6
31	6	6	6	6	6	6	6	6	NA	6
51	6	6	6	6	6	6	6	6	6	6
52	6	6	6	6	6	6	6	6	6	6
63	6	6	6	6	6	6	6	6	6	6

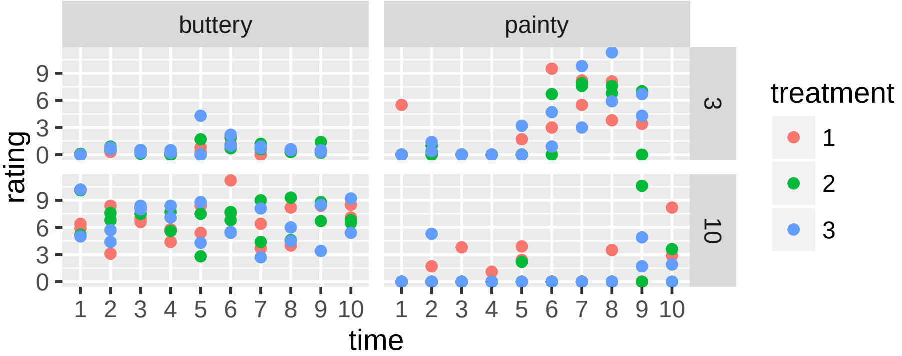
Check completeness with different scales, too

```
french_fries %>%
  gather(type, rating, -subject, -time, -treatment, -rep) %>%
  select(subject, time, treatment, type) %>%
  tbl_df() %>%
  count(subject, time) %>%
  spread(time, n) %>% kable
```

subject	1	2	3	4	5	6	7	8	9	10
3	30	30	30	30	30	30	30	30	30	NA
10	30	30	30	30	30	30	30	30	30	30
15	30	30	30	30	30	30	30	30	30	30
16	30	30	30	30	30	30	30	30	30	30
19	30	30	30	30	30	30	30	30	30	30
31	30	30	30	30	30	30	30	30	NA	30
51	30	30	30	30	30	30	30	30	30	30
52	30	30	30	30	30	30	30	30	30	30

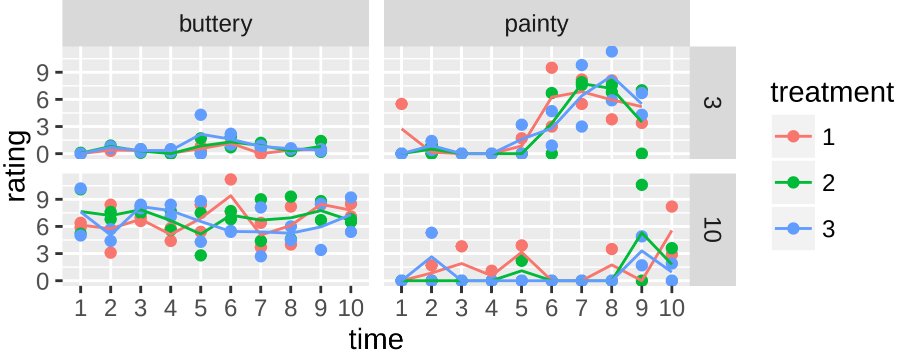
Change in ratings over weeks

```
ggplot(data=ff.m, aes(time, rating, colour=treatment)) +
  geom_point() +
  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))
ggplot(data=ff.m, aes(time, rating, colour=treatment)) +
  geom_point() +  facet_grid(subject~type) +
  geom_line(data=ff.m.av, aes(group=treatment))
```



String manipulation

When the experimental design is packed into column names, we need to extract it, and tidy it up.

```
genes <- read_csv("data/genes.csv")
kable(head(genes))
```

id	WI-6.R1	WI-6.R2	WI-6.R4	WM-6.R1	WM-6.R2	WI-12.R1
Gene 1	2.2	2.20	4.2	2.63	5.1	4.5
Gene 2	1.5	0.59	1.9	0.52	2.9	1.4
Gene 3	2.0	0.87	3.3	0.53	4.6	2.2

Gather column names into long form

```
gather(genes, variable, expr, -id) %>% kable
```

id	variable	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
Gene 1	WI-6.R4	4.20
Gene 2	WI-6.R4	1.86
Gene 3	WI-6.R4	3.28
Gene 1	WM-6.R1	2.63
Gene 2	WM-6.R1	0.52
Gene 3	WM-6.R1	0.53
Gene 1	WM-6.R2	5.06

Separate columns

```
genes %>%
  gather(variable, expr, -id) %>%
  separate(variable, c("trt", "leftover"), "-") %>%
  kable
```

id	trt	leftover	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
Gene 1	WI	6.R4	4.20
Gene 2	WI	6.R4	1.86
Gene 3	WI	6.R4	3.28
Gene 1	WM	6.R1	2.63

```r
genes %>%
  gather(variable, expr, -id) %>%
  separate(variable, c("trt", "leftover"), "-") %>%
  separate(leftover, c("time", "rep"), "\\.") %>% kable
```

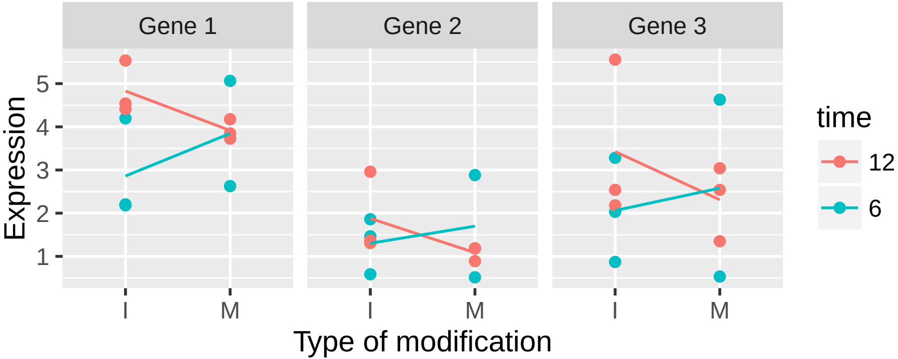
id	trt	time	rep	expr
Gene 1	WI	6	R1	2.18
Gene 2	WI	6	R1	1.46
Gene 3	WI	6	R1	2.03
Gene 1	WI	6	R2	2.20
Gene 2	WI	6	R2	0.59
Gene 3	WI	6	R2	0.87
Gene 1	WI	6	R4	4.20
Gene 2	WI	6	R4	1.86
Gene 3	WI	6	R4	3.28
Gene 1	WM	6	R1	2.63
Gene 2	WM	6	R1	0.52
Gene 3	WM	6	R1	0.53

```r
gtidy <- genes %>%
  gather(variable, expr, -id) %>%
  separate(variable, c("trt", "leftover"), "-") %>%
  separate(leftover, c("time", "rep"), "\\.") %>%
  mutate(trt = sub("W", "", trt)) %>%
  mutate(rep = sub("R", "", rep))
kable(head(gtidy))
```

id	trt	time	rep	expr
Gene 1	I	6	1	2.18
Gene 2	I	6	1	1.46
Gene 3	I	6	1	2.03
Gene 1	I	6	2	2.20
Gene 2	I	6	2	0.59
Gene 3	I	6	2	0.87

Your turn

1. Using the tidied dataset (`gtidy`), find the mean expression for each combination of id, trt, and time.
2. Use this tidied data to make this plot.



Re-structuring the temperature data

```
melbtemp.m <- melbtemp %>%
  select(num_range("V", c(1,2,3,4,seq(5,128,4)))) %>%
  filter(V4 %in% c("PRCP", "TMAX", "TMIN")) %>%
  gather(day, value, V5:V125, na.rm = TRUE) %>%
  spread(V4, value) %>%
  mutate(
    tmin = as.numeric(TMIN) / 10,
    tmax = as.numeric(TMAX) / 10,
    t_range = tmax - tmin,
    prcp = as.numeric(PRCP) / 10
  ) %>%
  rename(stn=V1, year=V2, month=V3)
```

```r
kable(head(melbtemp.m))
```

stn	year	month	day	PRCP	TMAX	TMIN	tmin	tm
ASN00086282	1970	7	V101	0	158	78	7.8	
ASN00086282	1970	7	V105	13	149	36	3.6	
ASN00086282	1970	7	V109	3	133	61	6.1	
ASN00086282	1970	7	V113	0	143	46	4.6	
ASN00086282	1970	7	V117	25	150	42	4.2	
ASN00086282	1970	7	V121	0	145	63	6.3	

```r
melbtemp.m$day <- factor(melbtemp.m$day,
  levels=c("V5","V9","V13","V17","V21","V25","V29",
           "V33","V37","V41","V45","V49","V53","V57",
           "V61","V65","V69","V73","V77","V81","V85",
           "V89","V93","V97","V101","V105","V109",
           "V113","V117","V121","V125"),
  labels=1:31)
melbtemp.m$date <- as.Date(paste(melbtemp.m$day,
     melbtemp.m$month, melbtemp.m$year, sep="-"),
     "%d-%m-%Y")
```

```r
kable(head(melbtemp.m))
```

stn	year	month	day	PRCP	TMAX	TMIN	tmin	tma
ASN00086282	1970	7	25	0	158	78	7.8	1
ASN00086282	1970	7	26	13	149	36	3.6	1
ASN00086282	1970	7	27	3	133	61	6.1	1
ASN00086282	1970	7	28	0	143	46	4.6	1
ASN00086282	1970	7	29	25	150	42	4.2	1
ASN00086282	1970	7	30	0	145	63	6.3	1

Re-structuring tuberculosis data

```
tb_tidy <- tb %>%
  gather(demographic, cases, m_04:f_u, na.rm = TRUE) %>%
  separate(demographic, c("sex", "age"), "_") %>%
  rename(country = iso2) %>%
  arrange(country, year, sex, age)
kable(head(tb_tidy))
```

country	year	sex	age	cases
AD	1996	f	014	0
AD	1996	f	1524	1
AD	1996	f	2534	1
AD	1996	f	3544	0
AD	1996	f	4554	0
AD	1996	f	5564	1

Dates and Times

Dates are deceptively hard to work with.
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**, and **timeDate** package helps tackle some of these issues.

Basic Lubridate Use

```r
now()
#> [1] "2016-05-18 14:17:37 AEST"
today()
#> [1] "2016-05-18"
now() + hours(4)
#> [1] "2016-05-18 18:17:37 AEST"
today() - days(2)
#> [1] "2016-05-16"
```

Parsing Dates

```
ymd("2013-05-14")
#> [1] "2013-05-14"
mdy("05/14/2013")
#> [1] "2013-05-14"
dmy("14052013")
#> [1] "2013-05-14"
ymd_hms("2013:05:14 14:5:30", tz = "Australia/Melbourne")
#> [1] "2013-05-14 14:05:30 AEST"
```

Extracting temporal elements

```
month(ymd("2013-05-14"))
#> [1] 5
year(ymd("2013-05-14"))
#> [1] 2013
wday(ymd("2013-05-14"), label=TRUE, abbr=TRUE)
#> [1] Tues
#> Levels: Sun < Mon < Tues < Wed < Thurs < Fri < Sat
isWeekday(ymd("2013-05-14"))
#> 2013-05-14
#>       TRUE
```

NY Workers Compensation

```
workers$`Accident Date` <- as.Date(workers$`Accident Date`,
                                   format="%m/%d/%Y")
summary(workers$`Accident Date`)
#>          Min.      1st Qu.       Median         Mean          3r
#> "1925-11-18" "2003-03-27" "2007-07-16" "2007-06-15" "2011-1
#>          Max.         NA's
#> "2015-01-30"      "16345"
```

```r
workers$year <- year(workers$`Accident Date`)
summary(workers$year)
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#>    1920    2000    2010    2010    2010    2020   16345
workers %>% group_by(year) %>% tally() %>% kable
```

year	n
1925	1
1929	1
1930	1
1931	10
1933	4
1937	1
1938	1
1940	6
1941	1
1944	1
1947	2

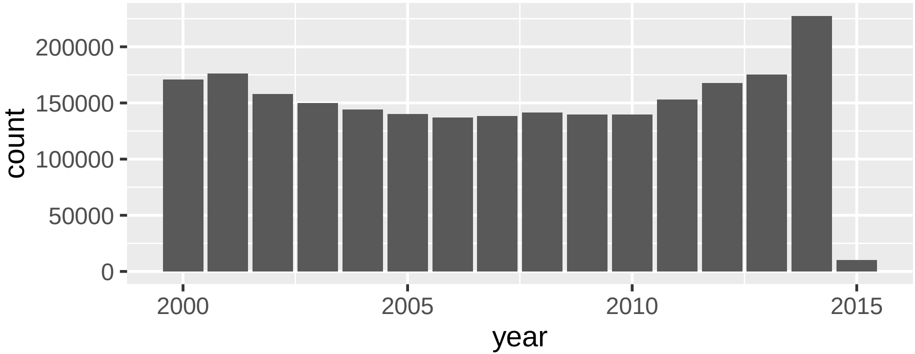
There should not be any claims before 2000, these are mistakes. Need to filter out the rows where `year < 2000`


```
workers$month <- month(workers$`Accident Date`,
                       label=TRUE, abbr=TRUE)
summary(workers$month)
#>     Jan     Feb     Mar     Apr     May     Jun     Jul     Aug
#> 227385 190013 200975 187979 202748 206868 206314 207985 21
#>     Nov     Dec    NA's
#> 182167 193598  16345
```

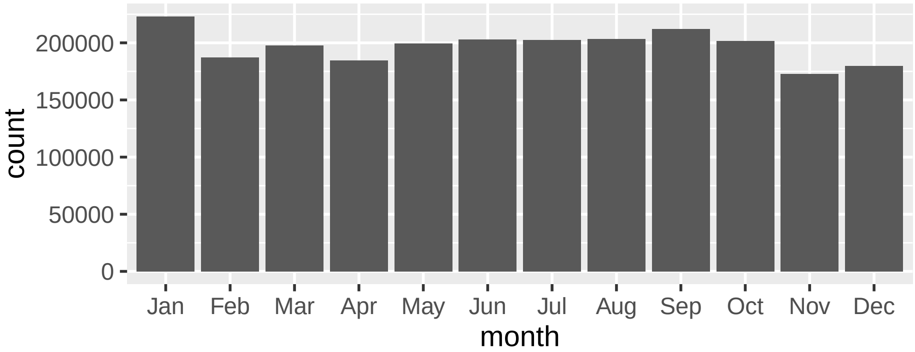
```
workers$wday <- wday(workers$`Accident Date`,
                     label=TRUE, abbr=TRUE)
summary(workers$wday)
#>     Sun    Mon   Tues    Wed  Thurs    Fri    Sat   NA's
#> 145609 432743 441741 426860 413289 387403 184616  16345
workers$timeindx <- as.numeric(workers$`Accident Date`-
        as.Date("01/01/2000", format="%m/%d/%Y"))
```

When do claims get made - by year

```r
ws <- workers %>% filter(year > 1999)
ggplot(ws, aes(x=year)) + geom_bar()
```

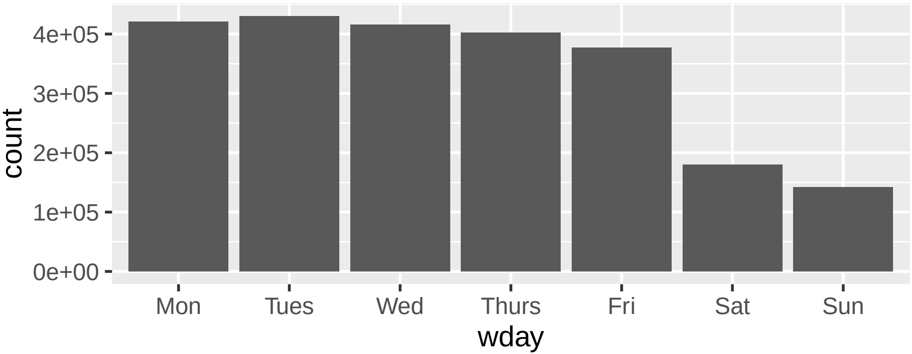



```
ggplot(ws, aes(x=month)) + geom_bar()
```



by day

```
ws$wday <- factor(ws$wday, levels=levels(ws$wday)[c(2:7,1)])
ggplot(ws, aes(x=wday)) + geom_bar()
```



Your turn

- Make a plot to examine the type of claim by year.
- Make a plot to examine the claims by gender by year.

Reading different file formats

- save, load: functions that allow you to save an R object, and load it back to R later
- fixed width fields: `read.fwf()`
- fixed width formats with SAS DATA script: `library(SAScii)`
- SPSS, Stata, SAS: `library(haven)`
- minitab, S, SAS, SPSS, systat, weka, dBase `library(foreign)`; SAS `library(SASxport)`
- googlesheets `library(googlesheets)`; html/web pages `library(rvest)`; images `library(EBImage)`; sound `library(tuneR)`; netCDF `library(ncdf)`; hdf5 `library(hdf5)`; json `library(jsonlite)`;
- economic data `library(quantmod)`

See MACHLIS MUSINGS for great info on reading different data into R, and other advice.

Credits

Notes prepared by Di Cook, building on joint workshops with Carson Sievert, Heike Hofmann, Eric Hare, Hadley Wickham.