tidyr, dplyr and ggplot2

Ken Butler
Lecturer (Statistics), UTSC

Greater Toronto Area R Users Group

November 4, 2015

Does this give you a headache?

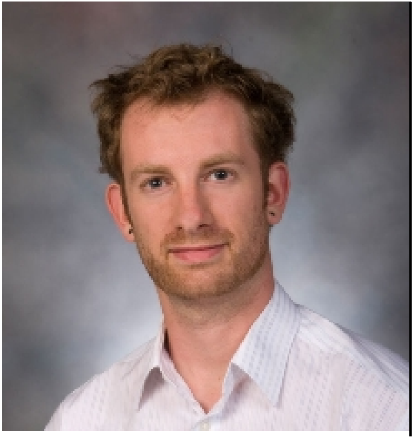
DrugA	DrugB	DrugC
4	6	6
5	8	7
4	4	6
3	5	6
2	4	7
4	6	5
3	5	6
4	8	5
4	6	5

- 27 migraine-suffering subjects, randomly chosen to receive *one* pain-killing drug
- On next migraine episode, take chosen drug, report pain 30 minutes after (1=no pain, 10=extreme pain)
- How do drugs compare?

The problem

- All numbers in data are pain scores, even though in 3 columns.
- “Wide format”.
- For eg. boxplots, ANOVA, want *2* columns:
    - one containing *all* scores
    - one identifying drug.
- How to get that?

The man behind the solution



- Hadley Wickham:
  - `tidyr`
  - `dplyr`
  - `ggplot2`
  - `stringr`
  - `readr`
  - `lubridate`
  - etc.

Tidy data (Wickham)

- Every value belongs to a *variable* and an *observation*.
- Variables in columns.
- Observations in rows.
- If this is done, data called “tidy”, ready for further analysis.
- If not, have “untidy” data, needs tidying.
- “Tidy” depends (somewhat) on kind of analysis you want to do.

Our data

```
> migraine=read.table("migraine.txt",header=T)
> migraine
  DrugA DrugB DrugC
1     4     6     6
2     5     8     7
3     4     4     6
4     3     5     6
5     2     4     7
6     4     6     5
7     3     5     6
8     4     8     5
9     4     6     5
```

- 3 columns all one variable. Not tidy!

gather: combining columns

- Combine columns that all measure same thing.
- Input: data frame, what makes columns different, what makes them same, columns to combine:

```
> library(tidyr)
> migraine2=gather(migraine,drug,pain,DrugA:DrugC)
```

The result

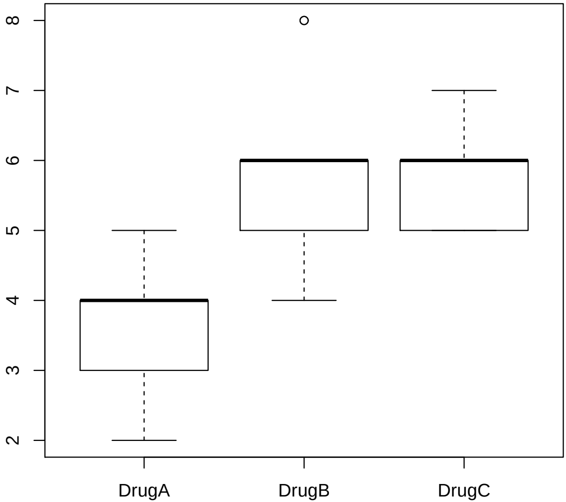
```
> migraine2

   drug pain
1  DrugA    4
2  DrugA    5
3  DrugA    4
4  DrugA    3
5  DrugA    2
6  DrugA    4
7  DrugA    3
8  DrugA    4
9  DrugA    4
10 DrugB    6
11 DrugB    8
12 DrugB    4
13 DrugB    5
14 DrugB    4
15 DrugB    6
16 DrugB    5
17 DrugB    8
18 DrugB    6
19 DrugC    6
20 DrugC    7
21 DrugC    6
22 DrugC    6
23 DrugC    7
24 DrugC    5
25 DrugC    6
26 DrugC    5
27 DrugC    5
```

- Tidy:
  - One column per variable (2 columns)
  - One row per observation (27 rows)
- “Long format”.
- Analysis eg. `aov(pain˜drug, data=migraine2)`.

Side-by-side boxplots

```
> boxplot(pain~drug,data=migraine2)
```



`spread`: the inverse of `gather`

- Our 27 migraine sufferers only tested one drug each (“between-subjects design”).
- But compare this (scores for 4 subjects in an ADHD study):

Subject	When	Score
1	t1	13
1	t2	10
1	t3	12
2	t1	30
2	t2	20
2	t3	26
3	t1	20
3	t2	13
3	t3	17
4	t1	26
4	t2	17
4	t3	20

- Tidy-ish (one variable per column).
- But each subject split over 3 rows.
- “Within-subject design”: repeated-measures analysis requires measurements for same subject in same line.
- Need to “undo” gather: `spread`.

Spreading these data

- `When` column is names of variables:

  ```
  > adhd=read.table("adhd.txt",header=T)
  > spread(adhd,When,Score)
  ```

  ```
    Subject t1 t2 t3
  1       1 13 10 12
  2       2 30 20 26
  3       3 20 13 17
  4       4 26 17 20
  ```

- Can turn this back into long format with `gather`.

Columns containing combined information

- Common way to display data is *contingency table*:

Species	Disease present		Disease absent	
	Location X	Location Y	Location X	Location Y
A	44	12	38	10
B	28	22	20	18

- File might be formatted like this:

Species	px	py	ax	ay
A	44	12	38	10
B	28	22	20	18

- Columns actually encode *two* variables: whether or not disease is present, and the location.

Tidying this table (1)

- Gather up columns that are all frequencies:

```
> disease=read.table("disease.txt",header=T)
> tmp=gather(disease,dis.loc,frequency,-Species)
> tmp
```

```
  Species dis.loc frequency
1       A      px        44
2       B      px        28
3       A      py        12
4       B      py        22
5       A      ax        38
6       B      ax        20
7       A      ay        10
8       B      ay        18
```

Tidying this table (2)

- Column now called `dis.loc` contains *two* variables: presence or absence of disease and location.
- `separate`:

```
> separate(tmp,dis.loc,c("disease","location"),1)
```

```
  Species disease location frequency
1       A       p        x        44
2       B       p        x        28
3       A       p        y        12
4       B       p        y        22
5       A       a        x        38
6       B       a        x        20
7       A       a        y        10
8       B       a        y        18
```

- Data frame, variable to split, what to split into, split after character 1.

`dplyr`: general data manipulation

- selecting rows from a data frame by value
- selecting columns from a data frame by name
- creating new variables from old ones
- summarizing variables, possibly by groups

Use (long) ADHD study results in `adhd` for example.

The ADHD data

```
> adhd

   Subject When Score
1        1   t1    13
2        1   t2    10
3        1   t3    12
4        2   t1    30
5        2   t2    20
6        2   t3    26
7        3   t1    20
8        3   t2    13
9        3   t3    17
10       4   t1    26
11       4   t2    17
12       4   t3    20

> library(dplyr)
```

Selecting rows (1)

- The rows for subject 2:

```
> filter(adhd,Subject==2)
```

```
  Subject When Score
1       2   t1    30
2       2   t2    20
3       2   t3    26
```

- The time-3 scores:

```
> filter(adhd,When=="t3")
```

```
  Subject When Score
1       1   t3    12
2       2   t3    26
3       3   t3    17
4       4   t3    20
```

Selecting rows (2)

- The scores bigger than 25:

```
> filter(adhd,Score>25)

  Subject When Score
1       2   t1    30
2       2   t3    26
3       4   t1    26
```

- Scores either for subject 2 or score 25+:

```
> filter(adhd,Subject==2 | Score>25)

  Subject When Score
1       2   t1    30
2       2   t2    20
3       2   t3    26
4       4   t1    26
```

Selecting columns

Name the ones you want to keep or to omit. Thus:

```
> select(adhd,c(Subject,
+   Score))
   Subject Score
1        1    13
2        1    10
3        1    12
4        2    30
5        2    20
6        2    26
7        3    20
8        3    13
9        3    17
10       4    26
11       4    17
12       4    20
```

```
> select(adhd,-When)

   Subject Score
1        1    13
2        1    10
3        1    12
4        2    30
5        2    20
6        2    26
7        3    20
8        3    13
9        3    17
10       4    26
11       4    17
12       4    20
```

Creating new variables

- Score was out of 30. Turn into a percentage:

```
> tmp=mutate(adhd,pct=Score/30*100)
> head(tmp)
  Subject When Score       pct
1       1   t1    13  43.33333
2       1   t2    10  33.33333
3       1   t3    12  40.00000
4       2   t1    30 100.00000
5       2   t2    20  66.66667
6       2   t3    26  86.66667
```

Create percent and get rid of old score

```
> tmp=mutate(adhd,pct=Score/30*100)
> tmp2=select(tmp,-Score)
> head(tmp2)
```

```
  Subject When       pct
1       1   t1  43.33333
2       1   t2  33.33333
3       1   t3  40.00000
4       2   t1 100.00000
5       2   t2  66.66667
6       2   t3  86.66667
```

- Created a lot of temporary variables. Can we do better?

The “pipe” operator

Same effect as previous, but “use output from last step as input to next one”:

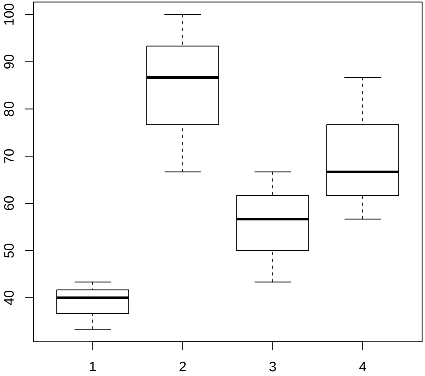
```
> adhd %>%
+   mutate(pct=Score/30*100) %>%
+   select(-Score) %>%
+   head()
```

```
  Subject When       pct
1       1   t1  43.33333
2       1   t2  33.33333
3       1   t3  40.00000
4       2   t1 100.00000
5       2   t2  66.66667
6       2   t3  86.66667
```

In a pipe, the initial data frame argument to any function disappears.

When the data frame isn't first

```
> adhd %>%
+   mutate(pct=Score/30*100) %>%
+   boxplot(pct~Subject,data=.)
```



Saving pipe output to a variable

I like this way:

```
> adhd %>%
+   mutate(pct=Score/30*100) %>%
+   select(-Score) -> adhd.2
> head(adhd.2)
```

```
  Subject When       pct
1       1   t1  43.33333
2       1   t2  33.33333
3       1   t3  40.00000
4       2   t1 100.00000
5       2   t2  66.66667
6       2   t3  86.66667
```

The “right-assignment” saves the data frame on the left (the output from `select`) into the variable on the *right*.

Summarizing variables

- Mean pct, well, duh:

```
> summarize(adhd.2,mean=mean(pct))
      mean
1 62.22222
```

- Summaries by *groups* takes an extra step:

```
> adhd.2 %>%
+   group_by(Subject) %>%
+   summarize( pct.mean=mean(pct),
+              pct.sd=sd(pct) )
Source: local data frame [4 x 3]

  Subject pct.mean    pct.sd
    (int)    (dbl)     (dbl)
1       1 38.88889  5.091751
2       2 84.44444 16.777410
3       3 55.55556 11.706282
4       4 70.00000 15.275252
```

dplyr and SQL

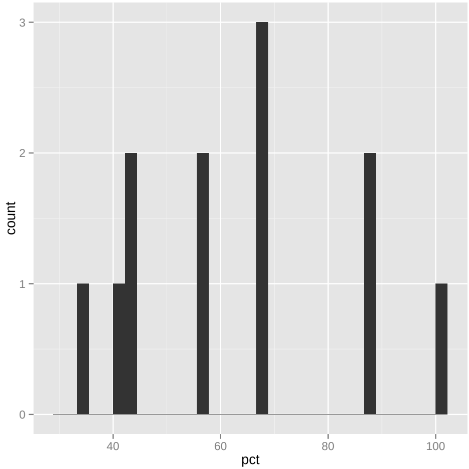
dplyr	SQL
select	select
filter	where
mutate	alter table
summarize	select mean(x) ...

`ggplot`: a grammar of graphics

- Base R graphics are functional, once you get used to the quirks.
- I learned base graphics by seeing a lot of examples.
- Hadley Wickham used a "grammar of graphics", implemented in `ggplot2`.
- Takes some getting used to, but once you do, *everything is consistent*.
- Separates:
  - what to plot
  - how to plot it
- Layers on plot constructed by *adding* (literally).

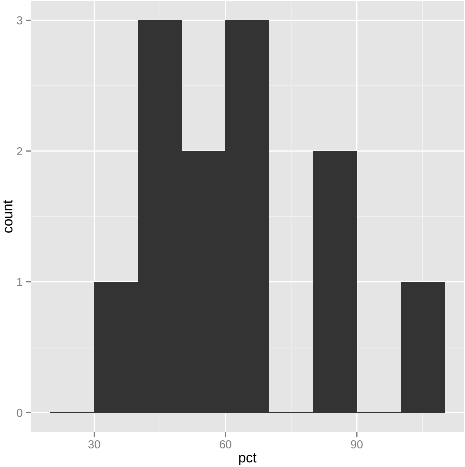
Histogram of ADHD percents

```
> library(ggplot2)
> p=ggplot(adhd.2,aes(x=pct))
> p+geom_histogram()
```



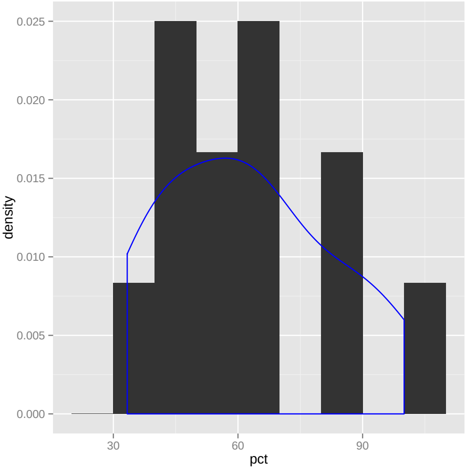
A better histogram

```
> p=ggplot(adhd.2,aes(x=pct))
> p+geom_histogram(binwidth=10)
```



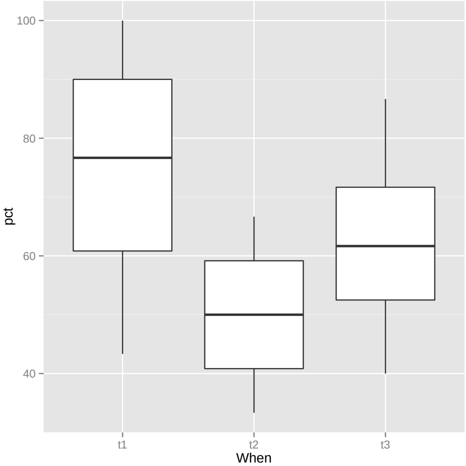
Histogram with density curve

```
> p=ggplot(adhd.2,aes(x=pct))
> p+geom_histogram(binwidth=10,aes(y=..density..))+
+   geom_density(col="blue")
```



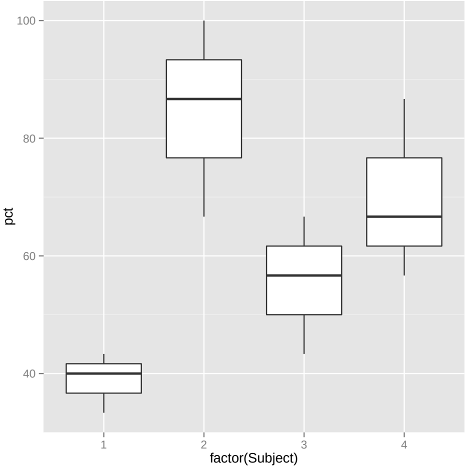
Boxplot of percents by When

```
> p=ggplot(adhd.2,aes(x=When,y=pct))
> p+geom_boxplot()
```



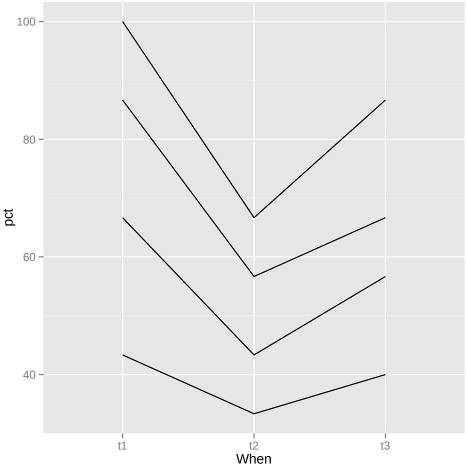
Boxplot of percents by `subject`

```
> p=ggplot(adhd.2,aes(x=factor(Subject),y=pct))
> p+geom_boxplot()
```



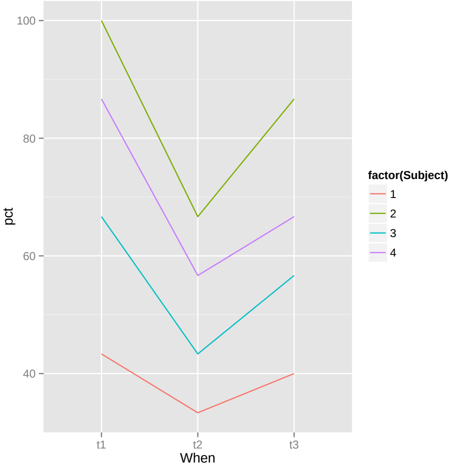
Line plot of subject's percents by When

```
> p=ggplot(adhd.2,aes(x=When,y=pct,group=Subject))
> p+geom_line()
```



Same line plot, with colours

```
> p=ggplot(adhd.2,aes(x=When,y=pct,group=Subject))
> p+geom_line(aes(colour=factor(Subject)))
```

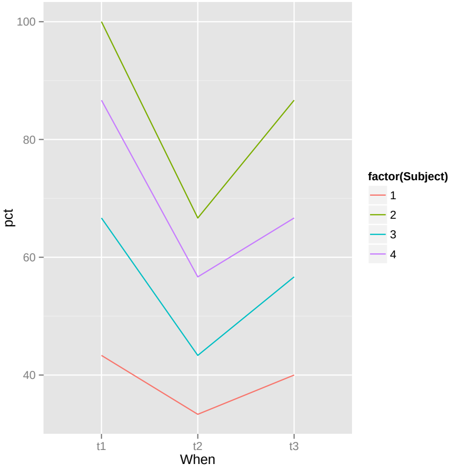


dplyr and ggplot2

- The first argument to ggplot is a data frame, so ggplot plays nicely with pipes.
- For example, to start from adhd, produce the percentages, get rid of the original scores, and then make the line plot, we can do all this:

```
> adhd %>%
+   mutate(pct=Score/30*100) %>%
+   select(-Score) %>%
+   ggplot(aes(x=When,y=pct,group=Subject)) +
+     geom_line(aes(colour=factor(Subject)))
```

Output



That contingency table

Species	Disease present		Disease absent	
	Location X	Location Y	Location X	Location Y
A	44	12	38	10
B	28	22	20	18

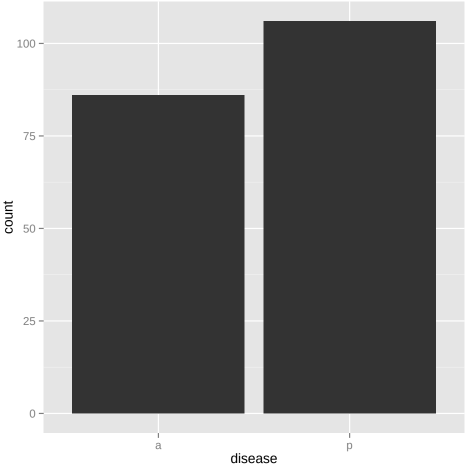
- Make a plot of disease present/absent by location and species.
- Technique in ggplot called **faceting**.
- Code we had before, rewritten with pipes:

```
> disease %>% gather(dis.loc,frequency,-Species) %>%
+   separate(dis.loc,c("disease","location"),1) -> dis.2
> head(dis.2,4)
```

```
  Species disease location frequency
1       A       p        x        44
2       B       p        x        28
3       A       p        y        12
4       B       p        y        22
```

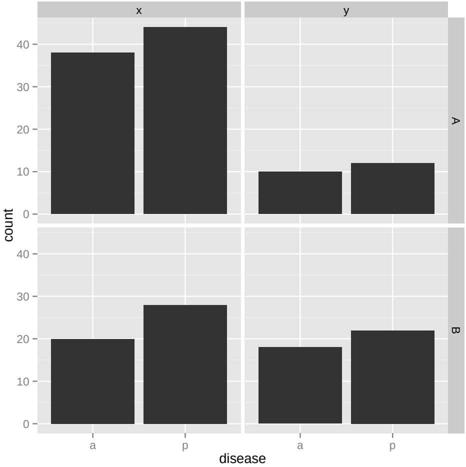
Without considering species and location

```
> dis.2 %>%
+   ggplot(aes(x=disease,weight=frequency)) +
+     geom_bar()
```



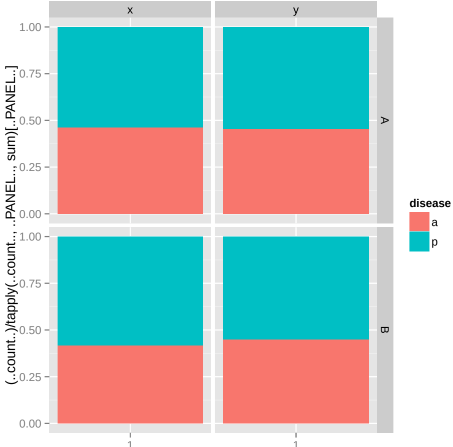
By species and location

```
> dis.2 %>%
+   ggplot(aes(x=disease,weight=frequency)) +
+   geom_bar() + facet_grid(Species ~ location)
```



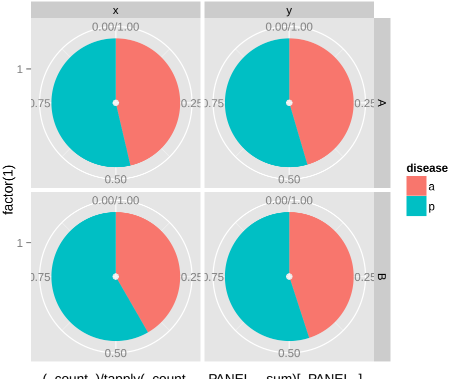
Stacked bars, with black magic

```
> dis.2 %>%
+   ggplot(aes(x=factor(1),weight=frequency,fill=disease))
+     geom_bar(aes(y=(..count..)/
+     tapply(..count..,..PANEL..,sum)[..PANEL..])) +
+     facet_grid(Species~location)
```



Or as pie charts, if you must

```
> dis.2 %>%
+   ggplot(aes(x=factor(1),weight=frequency,fill=disease)) +
+     geom_bar(aes(y=(..count..)/
+     tapply(..count..,..PANEL..,sum)[..PANEL..])) +
+     facet_grid(Species~location) +
+     coord_polar(theta="y")
```



Soccer goalscoring

- The English Premier league this year:

```
> premier=read.csv("premier.csv",header=T)
> tail(premier)
```

```
          date                 name            name.1 score
104 2015-10-31 West Bromwich Albion    Leicester City 2 - 3
105 2015-10-31       Crystal Palace Manchester United 0 - 0
106 2015-10-31              Watford   West Ham United 2 - 0
107 2015-10-31         Swansea City           Arsenal 0 - 3
108 2015-11-01          Southampton   AFC Bournemouth 2 - 0
109 2015-11-01              Everton        Sunderland 6 - 2
```

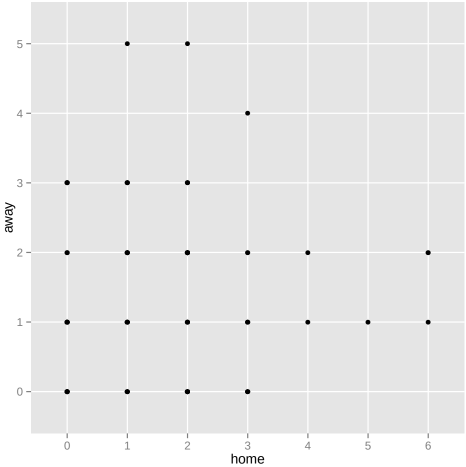
Sorting out those goals

```
> premier %>%
+   separate(score,c("home","away")," - ") -> goals
> tail(goals)
```

```
          date                 name            name.1 home away
104 2015-10-31 West Bromwich Albion    Leicester City    2    3
105 2015-10-31       Crystal Palace Manchester United    0    0
106 2015-10-31              Watford   West Ham United    2    0
107 2015-10-31         Swansea City           Arsenal    0    3
108 2015-11-01          Southampton   AFC Bournemouth    2    0
109 2015-11-01              Everton        Sunderland    6    2
```

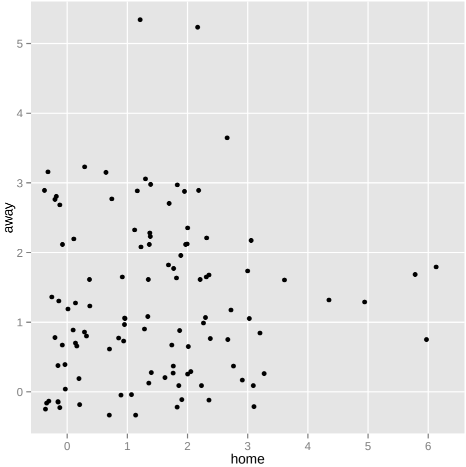
Scatterplot of goals

```
> goals %>%
+   ggplot(aes(x=home,y=away)) + geom_point()
```



Jittering to solve overplotting

```
> goals %>% ggplot(aes(x=home, y=away)) + geom_jitter()
```



Comments

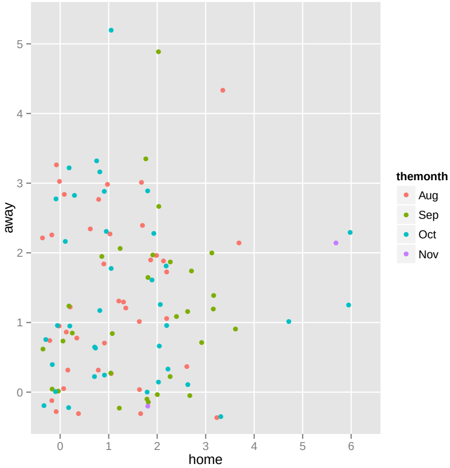
- Default jitter is “40% of resolution of data” in each direction, here, 0.4 goals.
- Can colour each point by the month in which the game was played.
- Alternative is to plot at actual points, but have size of symbol proportional to frequency.

Jittered plot coloured by month

- Uses another Wickham package `lubridate` to extract the month of a date.
- My dates are text strings, so convert to R `Date`s first.
- Add the colour to the `aes` for the `ggplot`.

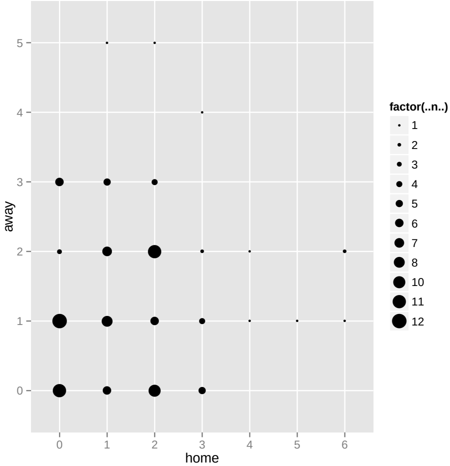
```
> library(lubridate)
> goals %>%
+   mutate(thedate=as.Date(date)) %>%
+   mutate(themonth=month(thedate,label=T)) %>%
+   ggplot(aes(x=home,y=away,colour=themonth)) +
+     geom_jitter()
```

The plot



Original points, plotted according to frequency

```
> goals %>% ggplot(aes(x=home,y=away)) +
+   stat_sum(aes(size = factor(..n..)), geom = "point")
```



Thank you

for your attention!

`http://www.utsc.utoronto.ca/~butler/rug/`