An Introduction to R

2.5 A few data manipulation tricks!

Dan Navarro (daniel.navarro@adelaide.edu.au)
School of Psychology, University of Adelaide
ua.edu.au/ccs/people/dan
DSTO R Workshop, 29-Apr-2015

Reshaping data
(wide to long)

As always, R has lots of tools

- reshape()
  - pretty powerful
  - a little difficult to learn
- melt() and cast() [reshape package]
  - very powerful
  - very difficult to learn
- longToWide() and wideToLong() [lsr package]
  - handles the single most common problem
  - fairly easy to learn

Reminder: wide form...

```
> dwr.wide

   id gender wm_alc_t1 wm_caf_t1 wm_nil_t1 wm_alc_t2 wm_caf_t2 wm_nil_t2 rt_alc_t1
1   1 female       3.0       2.6       3.0       3.1       3.4       3.2       671
7   2 female       4.4       4.0       5.3       4.6       5.3       4.9       836
13  3 female       4.0       3.7       4.6       4.5       6.0       4.8       859
19  4   male       4.6       5.2       5.4       5.0       5.5       5.5       852
25  5 female       3.4       4.1       3.6       3.7       4.4       4.4       788
31  6   male       3.1       4.6       5.0       3.8       4.8       4.8       673
37  7   male       4.3       5.0       5.8       5.7       5.0       5.8       914
43  8   male       2.8       3.7       2.4       4.0       4.7       4.2       650
49  9 female       3.7       4.1       4.5       4.2       4.5       6.2       915
55 10 female       4.4       5.3       5.0       5.0       5.0       5.8       869

   rt_caf_t1 rt_nil_t1 rt_alc_t2 rt_caf_t2 rt_nil_t2
1        472       556       682       438       543
7        575       563       820       537       530
13       421       583       803       449       599
19       448       437       877       403       430
25       452       611       771       439       619
31       402       676       682       395       600
37       459       519       845       450       538
43       417       502       678       433       502
49       475       566       890       482       536
55       451       702       848       412       682
```

Each row corresponds to a specific subject

```
> dwr.wide
```

	id	gender	wm_alc_t1	wm_caf_t1	wm_nil_t1	wm_alc_t2	wm_caf_t2	wm_nil_t2	rt_alc_t1
1	1	female	3.0	2.6	3.0	3.1	3.4	3.2	671
7	2	female	4.4	4.0	5.3	4.6	5.3	4.9	836
13	3	female	4.0	3.7	4.6	4.5	6.0	4.8	859
19	4	male	4.6	5.2	5.4	5.0	5.5	5.5	852
25	5	female	3.4	4.1	3.6	3.7	4.4	4.4	788
31	6	male	3.1	4.6	5.0	3.8	4.8	4.8	673
37	7	male	4.3	5.0	5.8	5.7	5.0	5.8	914
43	8	male	2.8	3.7	2.4	4.0	4.7	4.2	650
49	9	female	3.7	4.1	4.5	4.2	4.5	6.2	915
55	10	female	4.4	5.3	5.0	5.0	5.0	5.8	869

	rt_caf_t1	rt_nil_t1	rt_alc_t2	rt_caf_t2	rt_nil_t2
1	472	556	682	438	543
7	575	563	820	537	530
13	421	583	803	449	599
19	448	437	877	403	430
25	452	611	771	439	619
31	402	676	682	395	600
37	459	519	845	450	538
43	417	502	678	433	502
49	475	566	890	482	536
55	451	702	848	412	682

Some variables are "id variables" or "between subjects" variables: things that aren't measured repeatedly

```
> dwr.wide
```

	id	gender	wm_alc_t1	wm_caf_t1	wm_nil_t1	wm_alc_t2	wm_caf_t2	wm_nil_t2	rt_alc_t1
1	1	female	3.0	2.6	3.0	3.1	3.4	3.2	671
7	2	female	4.4	4.0	5.3	4.6	5.3	4.9	836
13	3	female	4.0	3.7	4.6	4.5	6.0	4.8	859
19	4	male	4.6	5.2	5.4	5.0	5.5	5.5	852
25	5	female	3.4	4.1	3.6	3.7	4.4	4.4	788
31	6	male	3.1	4.6	5.0	3.8	4.8	4.8	673
37	7	male	4.3	5.0	5.8	5.7	5.0	5.8	914
43	8	male	2.8	3.7	2.4	4.0	4.7	4.2	650
49	9	female	3.7	4.1	4.5	4.2	4.5	6.2	915
55	10	female	4.4	5.3	5.0	5.0	5.0	5.8	869

	rt_caf_t1	rt_nil_t1	rt_alc_t2	rt_caf_t2	rt_nil_t2
1	472	556	682	438	543
7	575	563	820	537	530
13	421	583	803	449	599
19	448	437	877	403	430
25	452	611	771	439	619
31	402	676	682	395	600
37	459	519	845	450	538
43	417	502	678	433	502
49	475	566	890	482	536
55	451	702	848	412	682

Other variables are "within subjects" or "repeated" variables.

```
> dwr.wide
```

	id	gender	wm_alc_t1	wm_caf_t1	wm_nil_t1	wm_alc_t2	wm_caf_t2	wm_nil_t2	rt_alc_t1
1	1	female	3.0	2.6	3.0	3.1	3.4	3.2	671
7	2	female	4.4	4.0	5.3	4.6	5.3	4.9	836
13	3	female	4.0	3.7	4.6	4.5	6.0	4.8	859
19	4	male	4.6	5.2	5.4	5.0	5.5	5.5	852
25	5	female	3.4	4.1	3.6	3.7	4.4	4.4	788
31	6	male	3.1	4.6	5.0	3.8	4.8	4.8	673
37	7	male	4.3	5.0	5.8	5.7	5.0	5.8	914
43	8	male	2.8	3.7	2.4	4.0	4.7	4.2	650
49	9	female	3.7	4.1	4.5	4.2	4.5	6.2	915
55	10	female	4.4	5.3	5.0	5.0	5.0	5.8	869

	rt_caf_t1	rt_nil_t1	rt_alc_t2	rt_caf_t2	rt_nil_t2
1	472	556	682	438	543
7	575	563	820	537	530
13	421	583	803	449	599
19	448	437	877	403	430
25	452	611	771	439	619
31	402	676	682	395	600
37	459	519	845	450	538
43	417	502	678	433	502
49	475	566	890	482	536
55	451	702	848	412	682

This is "working memory" (wm), measured under the influence of "alcohol" (alc), measured the "first time" (t1)

```
> dwr.wide
```

	id	gender	wm_alc_t1	wm_caf_t1	wm_nil_t1	wm_alc_t2	wm_caf_t2	wm_nil_t2	rt_alc_t1
1	1	female	3.0	2.6	3.0	3.1	3.4	3.2	671
7	2	female	4.4	4.0	5.3	4.6	5.3	4.9	836
13	3	female	4.0	3.7	4.6	4.5	6.0	4.8	859
19	4	male	4.6	5.2	5.4	5.0	5.5	5.5	852
25	5	female	3.4	4.1	3.6	3.7	4.4	4.4	788
31	6	male	3.1	4.6	5.0	3.8	4.8	4.8	673
37	7	male	4.3	5.0	5.8	5.7	5.0	5.8	914
43	8	male	2.8	3.7	2.4	4.0	4.7	4.2	650
49	9	female	3.7	4.1	4.5	4.2	4.5	6.2	915
55	10	female	4.4	5.3	5.0	5.0	5.0	5.8	869

	rt_caf_t1	rt_nil_t1	rt_alc_t2	rt_caf_t2	rt_nil_t2
1	472	556	682	438	543
7	575	563	820	537	530
13	421	583	803	449	599
19	448	437	877	403	430
25	452	611	771	439	619
31	402	676	682	395	600
37	459	519	845	450	538
43	417	502	678	433	502
49	475	566	890	482	536
55	451	702	848	412	682

There are three different levels of "drug"... alcohol, caffeine, and no drug

```
> dwr.wide
```

	id	gender	wm_alc_t1	wm_caf_t1	wm_nil_t1	wm_alc_t2	wm_caf_t2	wm_nil_t2	rt_alc_t1
1	1	female	3.0	2.6	3.0	3.1	3.4	3.2	671
7	2	female	4.4	4.0	5.3	4.6	5.3	4.9	836
13	3	female	4.0	3.7	4.6	4.5	6.0	4.8	859
19	4	male	4.6	5.2	5.4	5.0	5.5	5.5	852
25	5	female	3.4	4.1	3.6	3.7	4.4	4.4	788
31	6	male	3.1	4.6	5.0	3.8	4.8	4.8	673
37	7	male	4.3	5.0	5.8	5.7	5.0	5.8	914
43	8	male	2.8	3.7	2.4	4.0	4.7	4.2	650
49	9	female	3.7	4.1	4.5	4.2	4.5	6.2	915
55	10	female	4.4	5.3	5.0	5.0	5.0	5.8	869

	rt_caf_t1	rt_nil_t1	rt_alc_t2	rt_caf_t2	rt_nil_t2
1	472	556	682	438	543
7	575	563	820	537	530
13	421	583	803	449	599
19	448	437	877	403	430
25	452	611	771	439	619
31	402	676	682	395	600
37	459	519	845	450	538
43	417	502	678	433	502
49	475	566	890	482	536
55	451	702	848	412	682

For all three "drugs", we take measurements at the start of the session and the end of the session

```
> dwr.wide
```

	id	gender	wm_alc_t1	wm_caf_t1	wm_nil_t1	wm_alc_t2	wm_caf_t2	wm_nil_t2	rt_alc_t1
1	1	female	3.0	2.6	3.0	3.1	3.4	3.2	671
7	2	female	4.4	4.0	5.3	4.6	5.3	4.9	836
13	3	female	4.0	3.7	4.6	4.5	6.0	4.8	859
19	4	male	4.6	5.2	5.4	5.0	5.5	5.5	852
25	5	female	3.4	4.1	3.6	3.7	4.4	4.4	788
31	6	male	3.1	4.6	5.0	3.8	4.8	4.8	673
37	7	male	4.3	5.0	5.8	5.7	5.0	5.8	914
43	8	male	2.8	3.7	2.4	4.0	4.7	4.2	650
49	9	female	3.7	4.1	4.5	4.2	4.5	6.2	915
55	10	female	4.4	5.3	5.0	5.0	5.0	5.8	869

	rt_caf_t1	rt_nil_t1	rt_alc_t2	rt_caf_t2	rt_nil_t2
1	472	556	682	438	543
7	575	563	820	537	530
13	421	583	803	449	599
19	448	437	877	403	430
25	452	611	771	439	619
31	402	676	682	395	600
37	459	519	845	450	538
43	417	502	678	433	502
49	475	566	890	482	536
55	451	702	848	412	682

And we measure two things: working memory capacity, and median response time in a simple 2-AFC task

```
> dwr.wide
```

	id	gender	wm_alc_t1	wm_caf_t1	wm_nil_t1	wm_alc_t2	wm_caf_t2	wm_nil_t2	rt_alc_t1
1	1	female	3.0	2.6	3.0	3.1	3.4	3.2	671
7	2	female	4.4	4.0	5.3	4.6	5.3	4.9	836
13	3	female	4.0	3.7	4.6	4.5	6.0	4.8	859
19	4	male	4.6	5.2	5.4	5.0	5.5	5.5	852
25	5	female	3.4	4.1	3.6	3.7	4.4	4.4	788
31	6	male	3.1	4.6	5.0	3.8	4.8	4.8	673
37	7	male	4.3	5.0	5.8	5.7	5.0	5.8	914
43	8	male	2.8	3.7	2.4	4.0	4.7	4.2	650
49	9	female	3.7	4.1	4.5	4.2	4.5	6.2	915
55	10	female	4.4	5.3	5.0	5.0	5.0	5.8	869

	rt_caf_t1	rt_nil_t1	rt_alc_t2	rt_caf_t2	rt_nil_t2
1	472	556	682	438	543
7	575	563	820	537	530
13	421	583	803	449	599
19	448	437	877	403	430
25	452	611	771	439	619
31	402	676	682	395	600
37	459	519	845	450	538
43	417	502	678	433	502
49	475	566	890	482	536
55	451	702	848	412	682

Variable name format matters...
[dependent variable]_[level of factor 1]_[level of factor 2]

```
> dwr.wide
```

	id	gender	wm_alc_t1	wm_caf_t1	wm_nil_t1	wm_alc_t2	wm_caf_t2	wm_nil_t2	rt_alc_t1
1	1	female	3.0	2.6	3.0	3.1	3.4	3.2	671
7	2	female	4.4	4.0	5.3	4.6	5.3	4.9	836
13	3	female	4.0	3.7	4.6	4.5	6.0	4.8	859
19	4	male	4.6	5.2	5.4	5.0	5.5	5.5	852
25	5	female	3.4	4.1	3.6	3.7	4.4	4.4	788
31	6	male	3.1	4.6	5.0	3.8	4.8	4.8	673
37	7	male	4.3	5.0	5.8	5.7	5.0	5.8	914
43	8	male	2.8	3.7	2.4	4.0	4.7	4.2	650
49	9	female	3.7	4.1	4.5	4.2	4.5	6.2	915
55	10	female	4.4	5.3	5.0	5.0	5.0	5.8	869

	rt_caf_t1	rt_nil_t1	rt_alc_t2	rt_caf_t2	rt_nil_t2
1	472	556	682	438	543
7	575	563	820	537	530
13	421	583	803	449	599
19	448	437	877	403	430
25	452	611	771	439	619
31	402	676	682	395	600
37	459	519	845	450	538
43	417	502	678	433	502
49	475	566	890	482	536
55	451	702	848	412	682

Important: The separator character "_" does not appear anywhere else in the names!

```
> dwr.wide

   id gender wm_alc_t1 wm_caf_t1 wm_nil_t1 wm_alc_t2 wm_caf_t2 wm_nil_t2 rt_alc_t1
1   1 female       3.0       2.6       3.0       3.1       3.4       3.2       671
7   2 female       4.4       4.0       5.3       4.6       5.3       4.9       836
13  3 female       4.0       3.7       4.6       4.5       6.0       4.8       859
19  4   male       4.6       5.2       5.4       5.0       5.5       5.5       852
25  5 female       3.4       4.1       3.6       3.7       4.4       4.4       788
31  6   male       3.1       4.6       5.0       3.8       4.8       4.8       673
37  7   male       4.3       5.0       5.8       5.7       5.0       5.8       914
43  8   male       2.8       3.7       2.4       4.0       4.7       4.2       650
49  9 female       3.7       4.1       4.5       4.2       4.5       6.2       915
55 10 female       4.4       5.3       5.0       5.0       5.0       5.8       869

   rt_caf_t1 rt_nil_t1 rt_alc_t2 rt_caf_t2 rt_nil_t2
1        472       556       682       438       543
7        575       563       820       537       530
13       421       583       803       449       599
19       448       437       877       403       430
25       452       611       771       439       619
31       402       676       682       395       600
37       459       519       845       450       538
43       417       502       678       433       502
49       475       566       890       482       536
55       451       702       848       412       682
```

The variable names (almost) perfectly describe the experimental design!

- The wideToLong() function relies on this:
  - It looks for variables whose names include "_", and assumes those correspond to levels of a within subjects factor
  - Variables without "_" are assumed to be between subjects variables...

```
> library(lsr)
> wideToLong( dwr.wide )
```

```
> library(lsr)
> wideToLong( dwr.wide )

   id gender  wm  rt within1 within2
1   1 female 3.0 671     alc      t1
2   2 female 4.4 836     alc      t1
3   3 female 4.0 859     alc      t1
4   4   male 4.6 852     alc      t1
5   5 female 3.4 788     alc      t1
6   6   male 3.1 673     alc      t1
7   7   male 4.3 914     alc      t1
8   8   male 2.8 650     alc      t1
9   9 female 3.7 915     alc      t1
10 10 female 4.4 869     alc      t1
11  1 female 2.6 472     caf      t1
12  2 female 4.0 575     caf      t1
13  3 female 3.7 421     caf      t1
14  4   male 5.2 448     caf      t1
15  5 female 4.1 452     caf      t1
16  6   male 4.6 402     caf      t1
17  7   male 5.0 459     caf      t1

etc...
```

```
> library(lsr)
> wideToLong( dwr.wide )

   id gender  wm  rt within1 within2
1   1 female 3.0 671    alc      t1
2   2 female 4.4 836    alc      t1
3   3 female 4.0 859    alc      t1
4   4   male 4.6 852    alc      t1
5   5 female 3.4 788    alc      t1
6   6   male 3.1 673    alc      t1
7   7   male 4.3 914    alc      t1
8   8   male 2.8 650    alc      t1
9   9 female 3.7 915    alc      t1
10 10 female 4.4 869    alc      t1
11  1 female 2.6 472    caf      t1
12  2 female 4.0 575    caf      t1
13  3 female 3.7 421    caf      t1
14  4   male 5.2 448    caf      t1
15  5 female 4.1 452    caf      t1
16  6   male 4.6 402    caf      t1
17  7   male 5.0 459    caf      t1

etc...
```

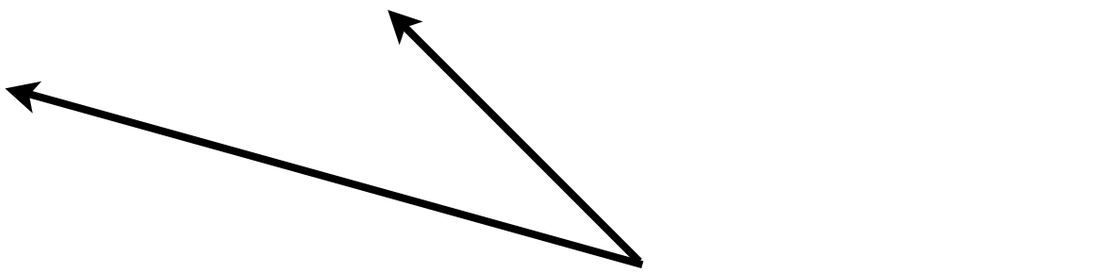
Rows are defined by a unique combination of subject (and hence between-subject variables) AND the within-subject factors

```
> library(lsr)
> wideToLong( dwr.wide )

   id gender  wm  rt within1 within2
1   1 female 3.0 671    alc      t1
2   2 female 4.4 836    alc      t1
3   3 female 4.0 859    alc      t1
4   4   male 4.6 852    alc      t1
5   5 female 3.4 788    alc      t1
6   6   male 3.1 673    alc      t1
7   7   male 4.3 914    alc      t1
8   8   male 2.8 650    alc      t1
9   9 female 3.7 915    alc      t1
10 10 female 4.4 869    alc      t1
11  1 female 2.6 472    caf      t1
12  2 female 4.0 575    caf      t1
13  3 female 3.7 421    caf      t1
14  4   male 5.2 448    caf      t1
15  5 female 4.1 452    caf      t1
16  6   male 4.6 402    caf      t1
17  7   male 5.0 459    caf      t1

etc...
```

By default, the within subject factors get generic names like this

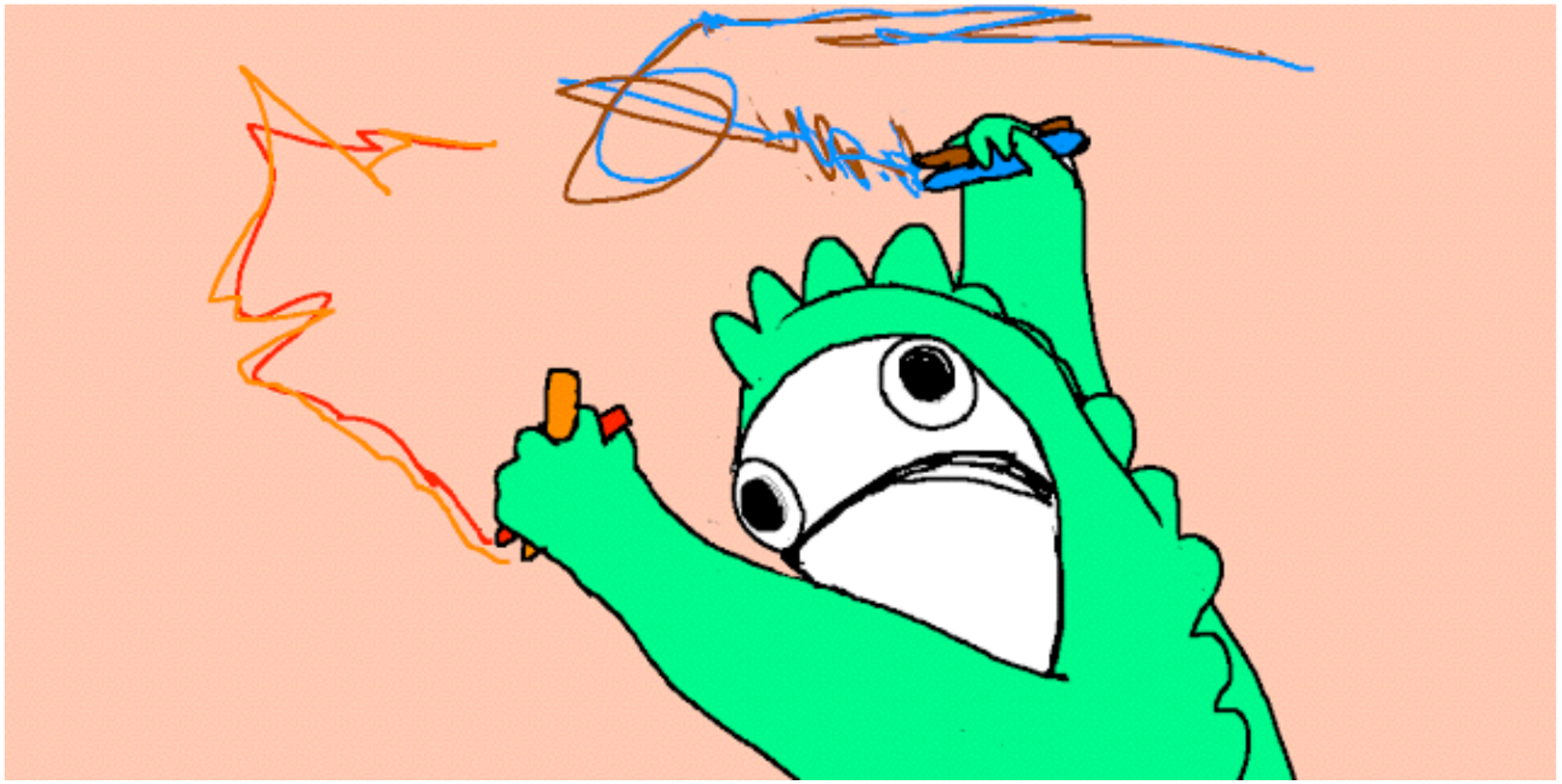
(because the wide form doesn't actually specify what they should be called)

```
> wideToLong( dwr.wide, within = c("drug","time") )

   id gender  wm  rt drug time
1   1 female 3.0 671  alc   t1
2   2 female 4.4 836  alc   t1
3   3 female 4.0 859  alc   t1
4   4   male 4.6 852  alc   t1
5   5 female 3.4 788  alc   t1
6   6   male 3.1 673  alc   t1
7   7   male 4.3 914  alc   t1
8   8   male 2.8 650  alc   t1
9   9 female 3.7 915  alc   t1
10 10 female 4.4 869  alc   t1
11  1 female 2.6 472  caf   t1
12  2 female 4.0 575  caf   t1
13  3 female 3.7 421  caf   t1
14  4   male 5.2 448  caf   t1
```



Easy to fix this though

Try it yourself (Exercise 2.5.1)



Reshaping data
(long to wide)

What if we want to go the other way?

```
> dwr.long

   id gender  rt  wm drug time
1   1 female 671 3.0  alc   t1
2   1 female 472 2.6  caf   t1
3   1 female 556 3.0  nil   t1
4   1 female 682 3.1  alc   t2
5   1 female 438 3.4  caf   t2
6   1 female 543 3.2  nil   t2
7   2 female 836 4.4  alc   t1
8   2 female 575 4.0  caf   t1
9   2 female 563 5.3  nil   t1
10  2 female 820 4.6  alc   t2
11  2 female 537 5.3  caf   t2
12  2 female 530 4.9  nil   t2

etc...
```

The long form is easier to reshape

- The longToWide() function needs you to tell it
  - (a) which variables define the within subjects factors (i.e., drug and time)
  - (b) and which variables are measured at the different levels of the within subjects factors (i.e., wm and rt)
  - Put these in a formula like this:

```
wm + rt ~ drug + time
```

dependent variables / measured variables

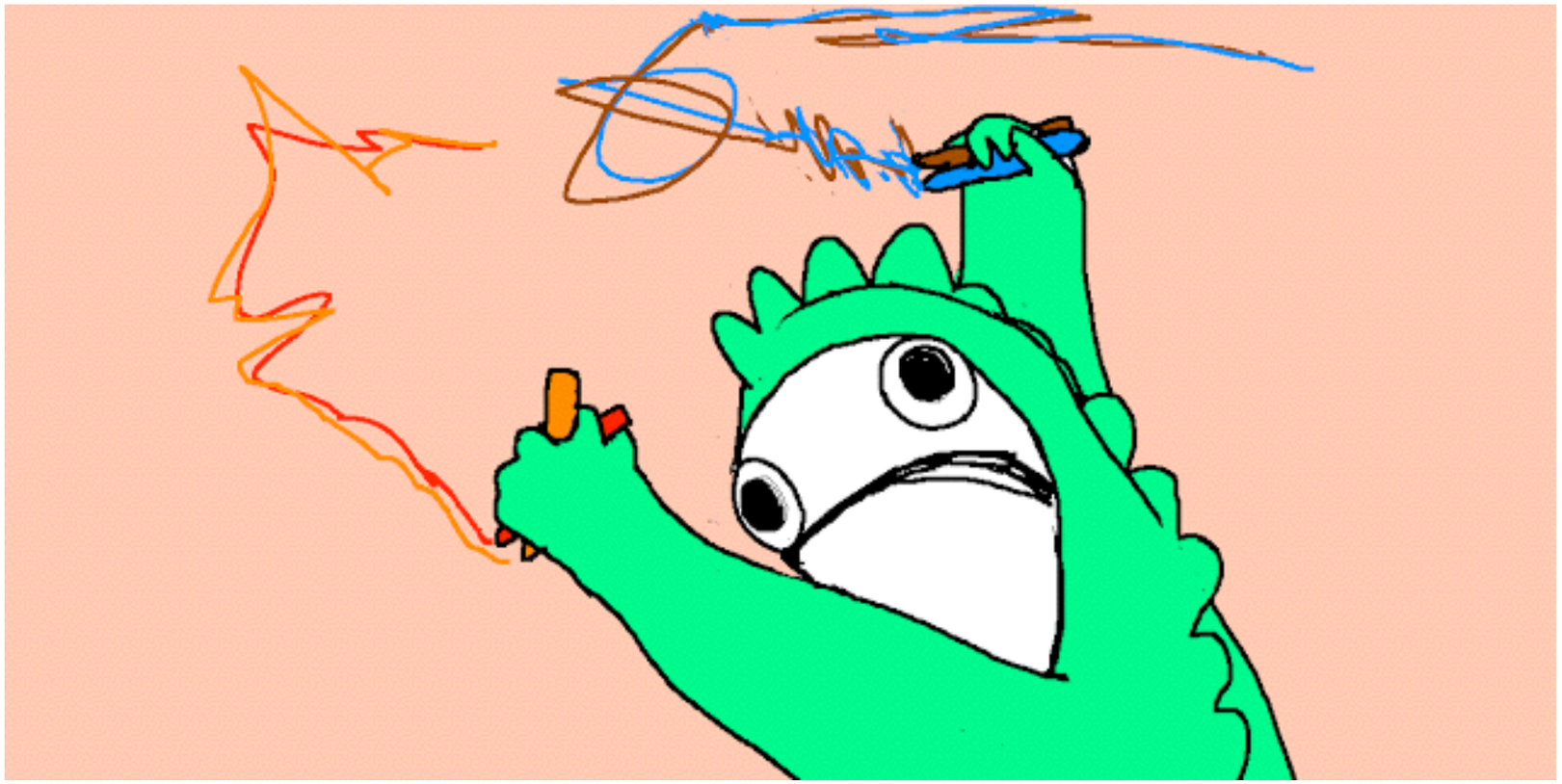
within subject factors / repeated factors

```
> longToWide( dwr.long, wm + rt ~ drug + time )
```

```
> longToWide( dwr.long, wm + rt ~ drug + time )

   id gender wm_alc_t1 rt_alc_t1 wm_caf_t1 rt_caf_t1 wm_nil_t1 rt_nil_t1 wm_alc_t2
1   1 female       3.0       671       2.6       472       3.0       556       3.1
2   2 female       4.4       836       4.0       575       5.3       563       4.6
3   3 female       4.0       859       3.7       421       4.6       583       4.5
4   4   male       4.6       852       5.2       448       5.4       437       5.0
5   5 female       3.4       788       4.1       452       3.6       611       3.7
6   6   male       3.1       673       4.6       402       5.0       676       3.8
7   7   male       4.3       914       5.0       459       5.8       519       5.7
8   8   male       2.8       650       3.7       417       2.4       502       4.0
9   9 female       3.7       915       4.1       475       4.5       566       4.2
10 10 female       4.4       869       5.3       451       5.0       702       5.0

   rt_alc_t2 wm_caf_t2 rt_caf_t2 wm_nil_t2 rt_nil_t2
1        682       3.4       438       3.2       543
2        820       5.3       537       4.9       530
3        803       6.0       449       4.8       599
4        877       5.5       403       5.5       430
5        771       4.4       439       4.4       619
6        682       4.8       395       4.8       600
7        845       5.0       450       5.8       538
8        678       4.7       433       4.2       502
9        890       4.5       482       6.2       536
10       848       5.0       412       5.8       682
```

Done!

Try it yourself (Exercise 2.5.2)



Coercing data from one type to another
(no exercises for this one!)

Data aren't always in the format you want

- Common problems:
  - Character vectors converted to factors?
  - Characters (e.g."10") converted to numbers (10)
  - Logicals to numbers?

Data aren't always in the format you want

- Common problems:
  - Character vectors converted to factors?
  - Characters (e.g."10") converted to numbers (10)
  - Logicals to numbers?

- Relevant R concept: coercion
  - "Coercion" is when R forces one data type into another one.
  - Use functions like as.numeric(), as.factor() etc...

Character to factor

```
> eyeColour <- c( "blue", "blue", "green", "brown", "green" )

> class( eyeColour )
[1] "character"
```

```
> eyeColour <- as.factor( eyeColour )
> eyeColour
[1] blue  blue  green brown green
Levels: blue brown green

> class( eyeColour )
[1] "factor"
```

Character to numeric

```
> age <- c("17", "19", "30", "37", "thirty")

> class(age)
[1] "character"
```

```
> age <- as.numeric( age )
Warning message:
NAs introduced by coercion

> age
[1] 17 19 30 37 NA

> class(age)
[1] "numeric"
```

Logical to numeric

```
> isYoung <- age < 30
> isYoung
[1]  TRUE  TRUE FALSE FALSE    NA

> class( isYoung )
[1] "logical"
```

```
> isYoung <- as.numeric( isYoung )
> isYoung
[1]  1  1  0  0 NA

> class( isYoung )
[1] "numeric"
```

“Cutting” a continuous variable into discrete categories

Binning variables

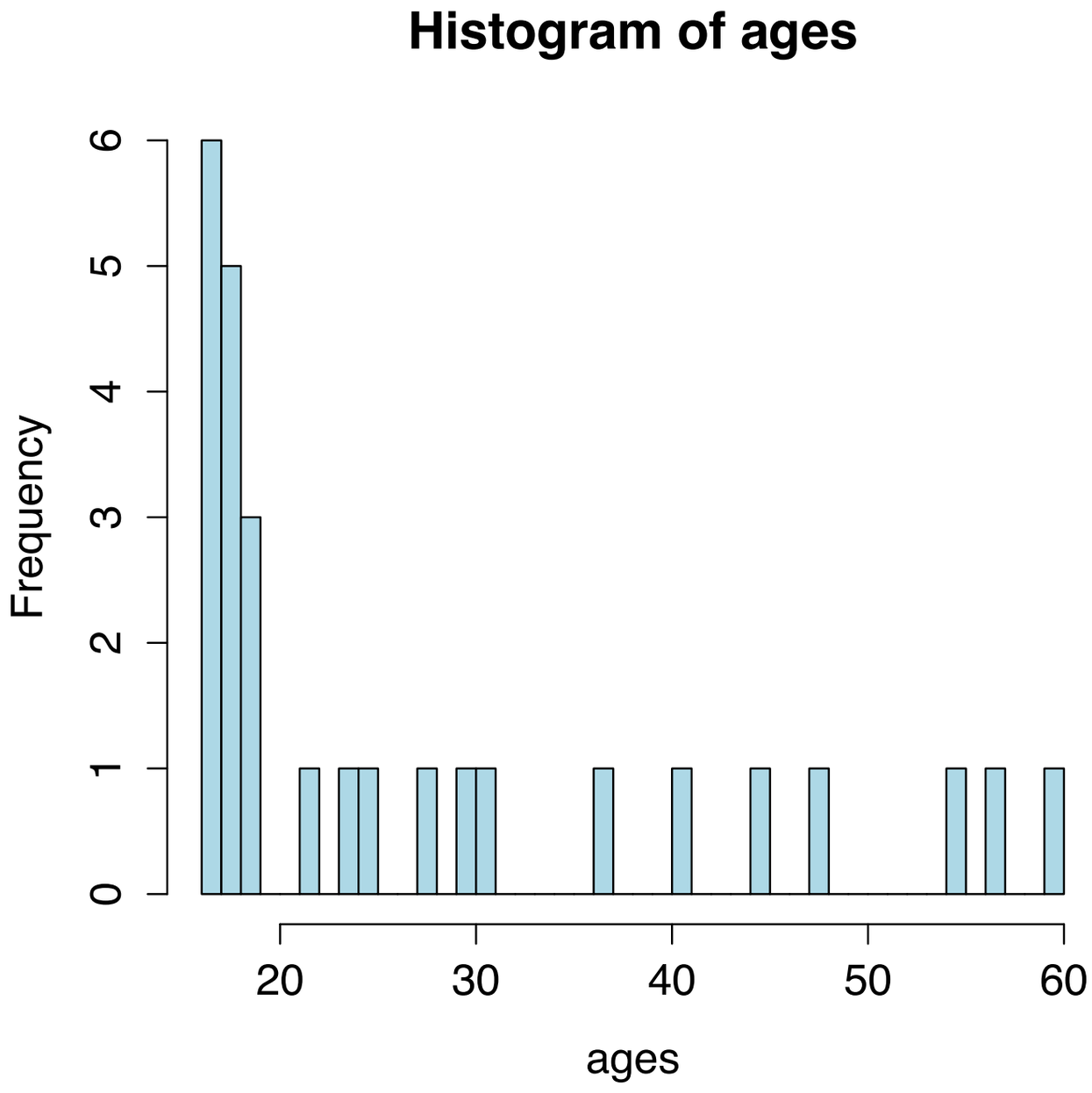
- Suppose I want to split "age" into three groups, "young", "middle", "old"

- Two different methods:
  - Equal ranges for each bin: (16-30), (31-45), (46-60)
  - Equal sample size in each bin: (16-18), (19-28), (29-60)

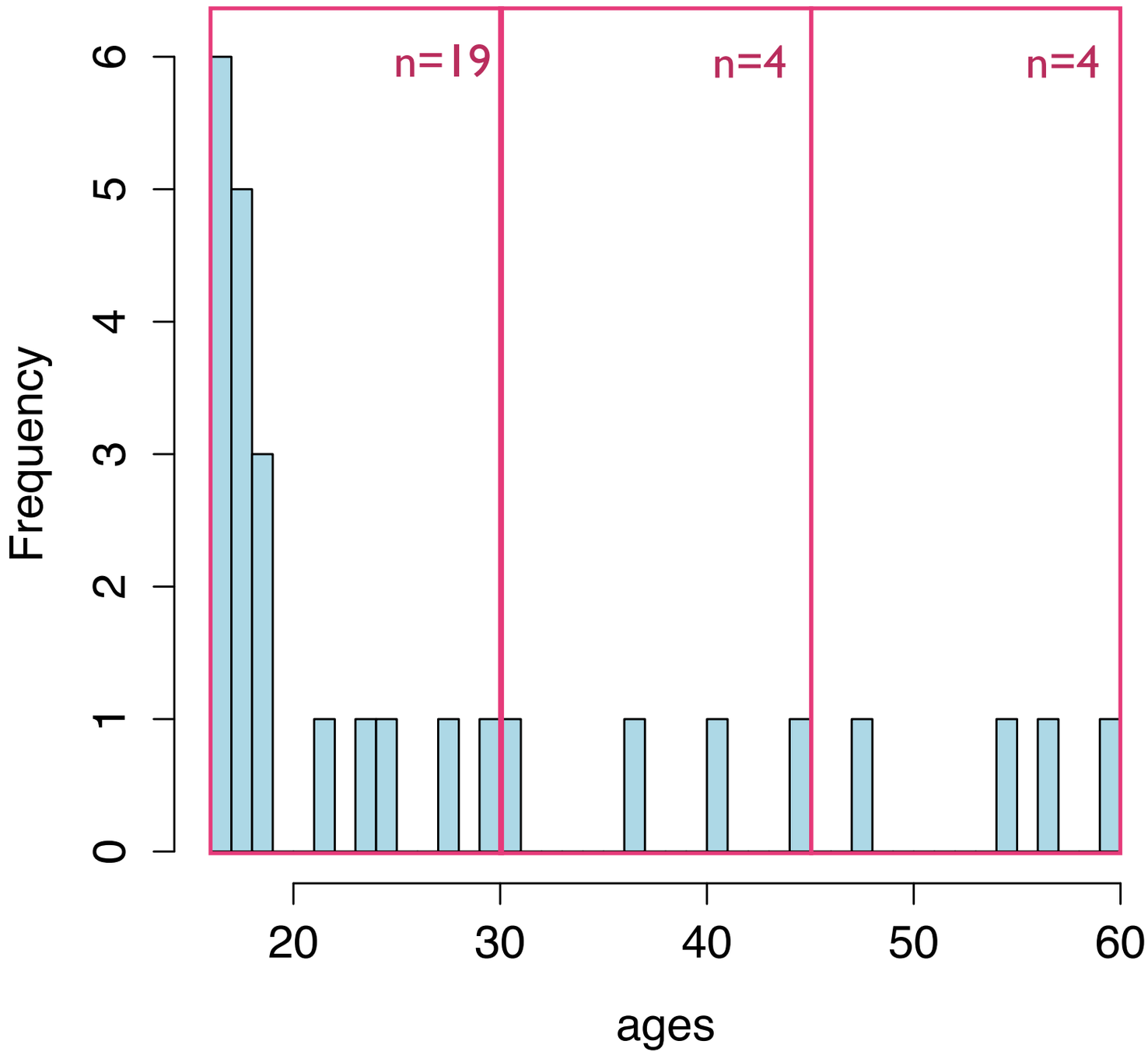
- Usual comment:
  - Equal ranges makes categories meaningful (important)
  - Equal numbers makes ANOVA easier (less important, I think)

Example

```
ages <- c( 16,16,16,17,17,17,18,18,18,18,18,19,
           19,19,22,24,25,28,30,31,37,41,45,48,
           55,57,60 )

hist( ages,
      col="light blue",
      breaks=16:60  )
```

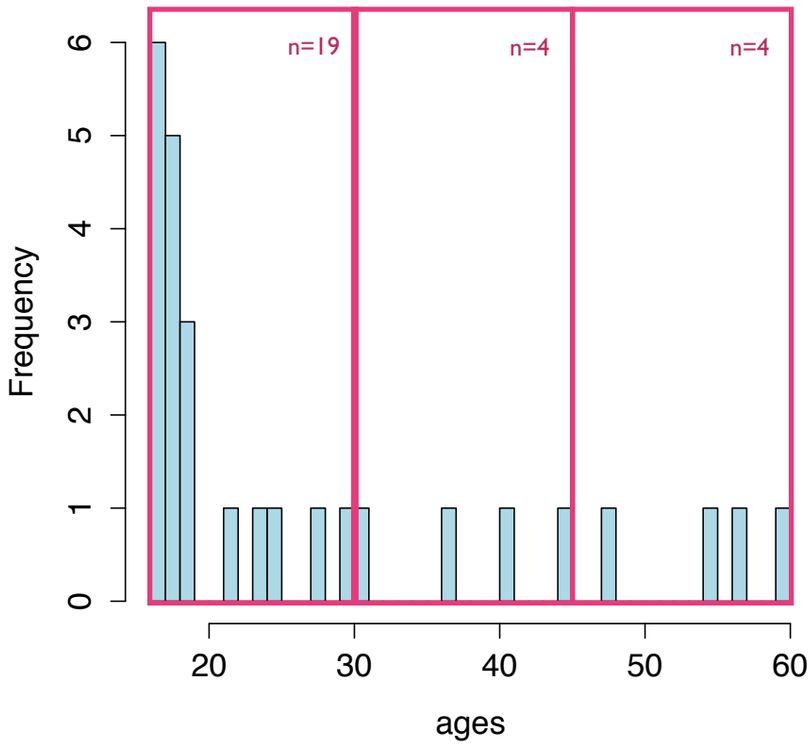


Three bins of equal range

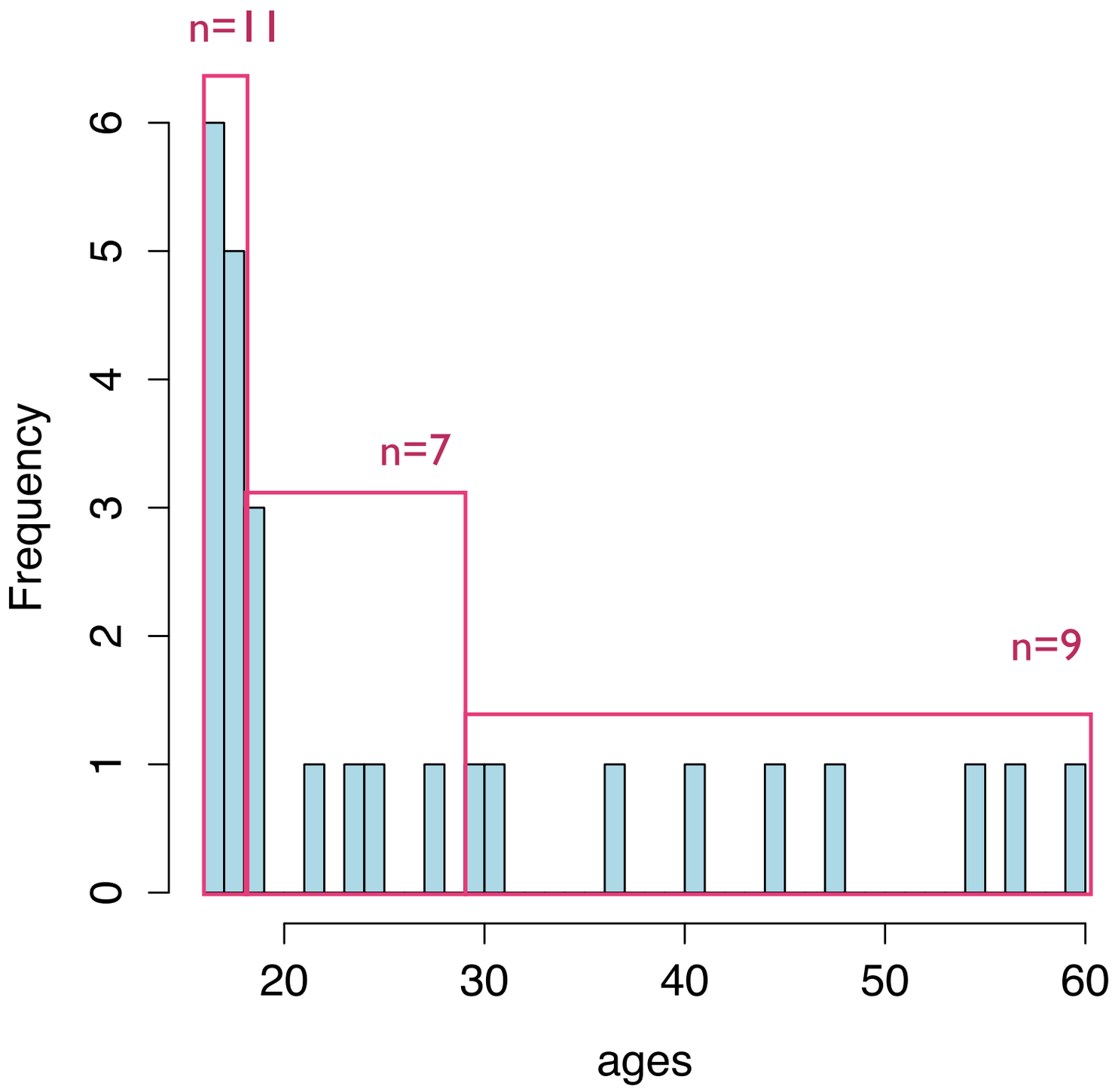


How to do it?

```
> cut( ages, 3 )

 [1] (16,30.7]   (16,30.7]   (16,30.7]
 [4] (16,30.7]   (16,30.7]   (16,30.7]
 [7] (16,30.7]   (16,30.7]   (16,30.7]
[10] (16,30.7]   (16,30.7]   (16,30.7]
[13] (16,30.7]   (16,30.7]   (16,30.7]
[16] (16,30.7]   (16,30.7]   (16,30.7]
[19] (16,30.7]   (30.7,45.3] (30.7,45.3]
[22] (30.7,45.3] (30.7,45.3] (45.3,60]
[25] (45.3,60]   (45.3,60]   (45.3,60]
Levels: (16,30.7] (30.7,45.3] (45.3,60]
```

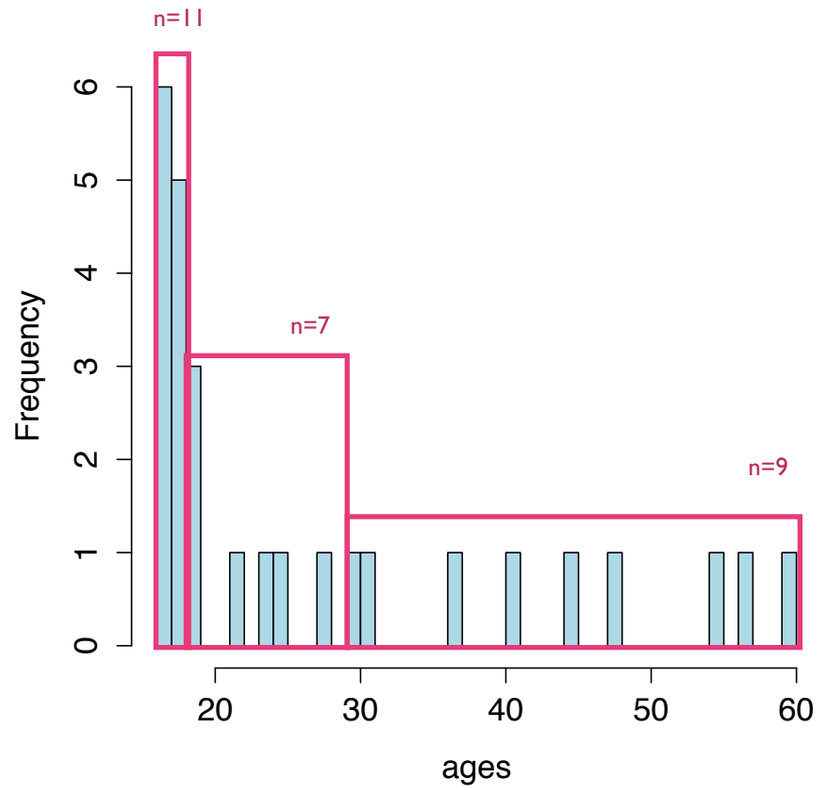


Three bins of "equal" sample size



How to do it?

```
> library(lsr)
> quantileCut( ages, 3 )
 [1] (16,18]   (16,18]   (16,18]   (16,18]
 [5] (16,18]   (16,18]   (16,18]   (16,18]
 [9] (16,18]   (16,18]   (16,18]   (18,28.7]
[13] (18,28.7] (18,28.7] (18,28.7] (18,28.7]
[17] (18,28.7] (18,28.7] (28.7,60] (28.7,60]
[21] (28.7,60] (28.7,60] (28.7,60] (28.7,60]
[25] (28.7,60] (28.7,60] (28.7,60]
Levels: (16,18] (18,28.7] (28.7,60]
```



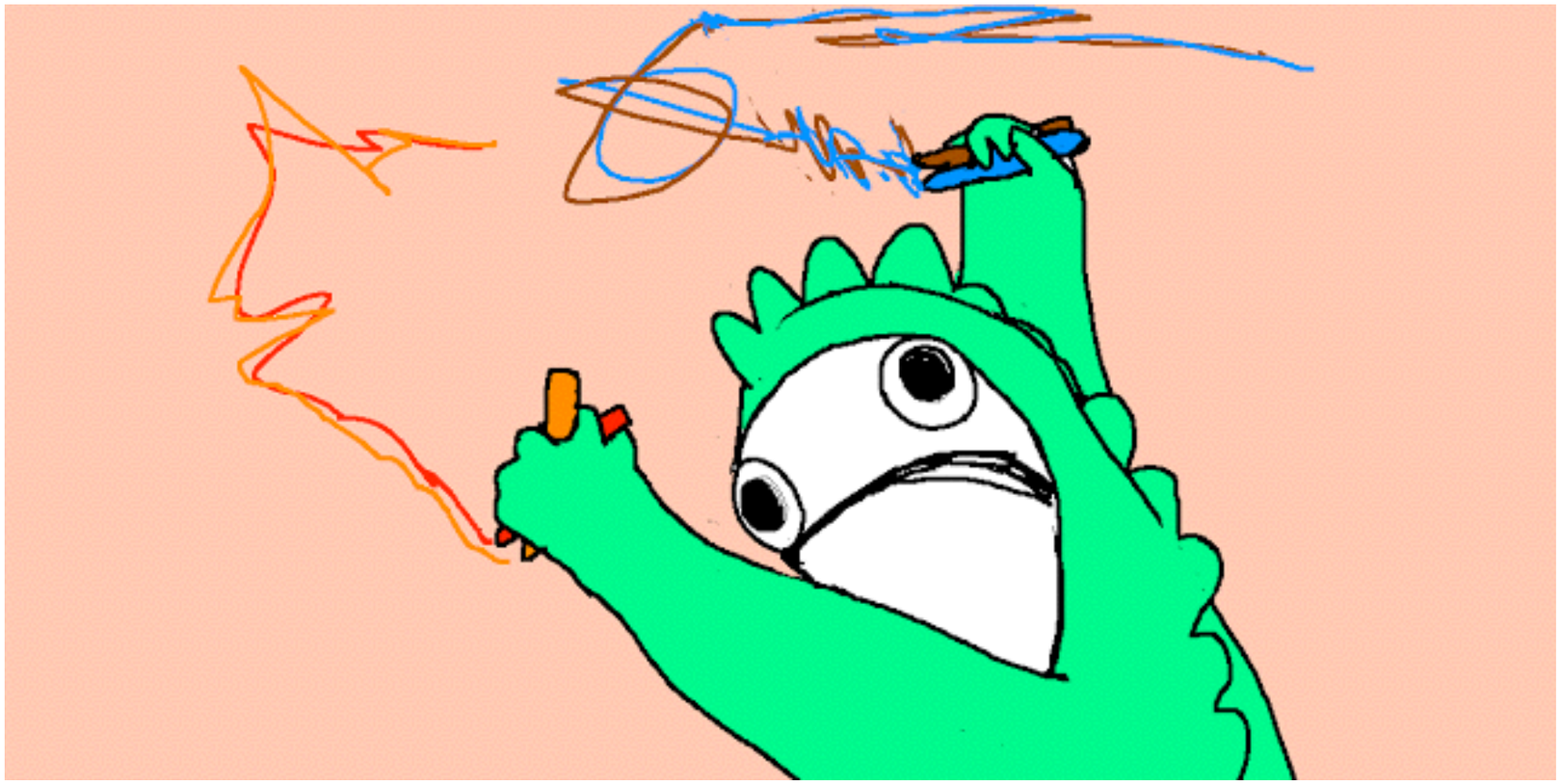
Manually specify the ranges:

```
> cut( ages, breaks=c(15,22,35,60))

 [1] (15,22] (15,22] (15,22] (15,22] (15,22]
 [6] (15,22] (15,22] (15,22] (15,22] (15,22]
[11] (15,22] (15,22] (15,22] (15,22] (15,22]
[16] (22,35] (22,35] (22,35] (22,35] (22,35]
[21] (35,60] (35,60] (35,60] (35,60] (35,60]
[26] (35,60] (35,60]
Levels: (15,22] (22,35] (35,60]
```

How to change the labels on a factor so that they're more meaningful...

```
> ageGroups <- cut(ages,3)
> levels( ageGroups ) <- c("young","middle","old")

> ageGroups
 [1] young  young  young  young  young  young
 [7] young  young  young  young  young  young
[13] young  young  young  young  young  young
[19] young  middle middle middle middle old
[25] old    old    old
Levels: young middle old
```

Try it yourself (Exercise 2.5.3)



Shuffling factor levels?

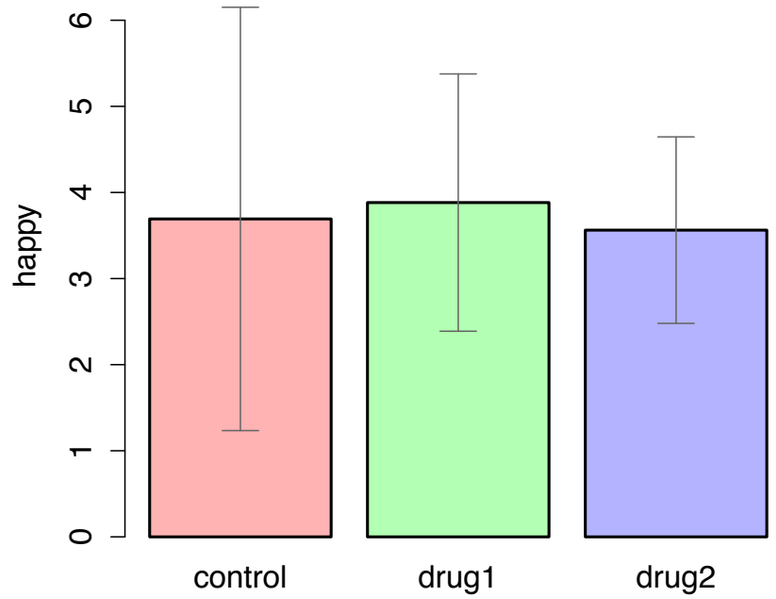
Categorical variables are fundamentally "unordered", but factor levels have to be specified one after the other...

```
> expt$treatment

 [1] control drug1   drug2   control drug1
 [6] drug2   control drug1   drug2   control
[11] drug1   drug2

Levels: control drug1 drug2
```

R "understands" that this is a nominal scale variable (that's what a factor does!) but sometimes the order does make a difference for some things...

Example: bar plots

```
bars(
 formula = happy ~ treatment,
 data = expt
)
```



The bars appear in the same order as the factor levels.

Factor levels can be reordered

- Two commands worth knowing:
  - relevel() -- moves a level to the "first" position
  - permuteLevels() [lsr package] -- arbitrary reshuffle

- Both are easy, but permuteLevels() is more powerful, so I'll talk about that one.

permuteLevels()

```
permuteLevels(
     expt$treatment,
     c(2,3,1)
)
```

The factor to be permuted

The permutation to be applied:

(a) take the current level 2, and move it to position 1

(b) take the current level 3, and move it to position 2

(c) take the current level 1, and move it to position 3

permuteLevels()

```
permuteLevels(
    expt$treatment,
    c(2,3,1)
)
```

```
 [1] control drug1   drug2   control drug1
 [6] drug2   control drug1   drug2   control
[11] drug1   drug2
Levels: drug1 drug2 control
```

None of the data values are affected, but the ordering of the levels has been changed

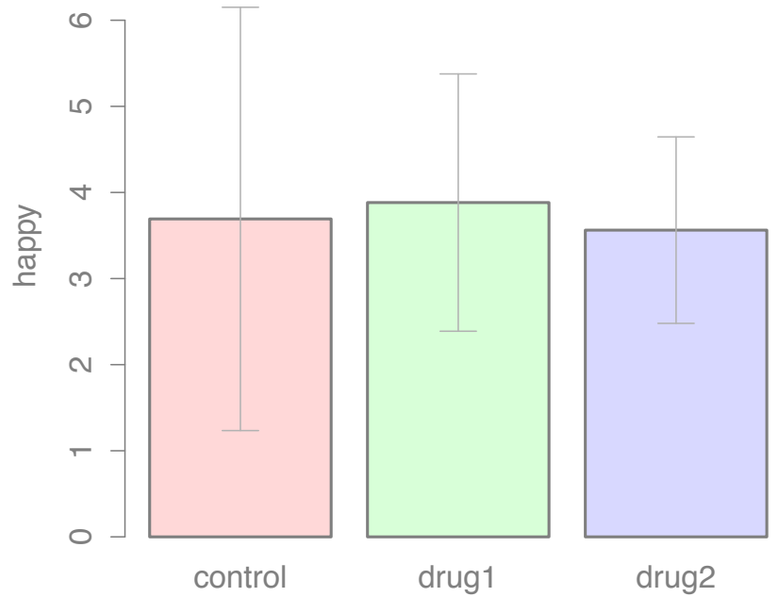
permuteLevels()

```
expt$treatment <- permuteLevels(
                      expt$treatment,
                      c(2,3,1)
                  )
```

Don’t forget that if you want to change expt$treatment itself, then you need to assign the results of the permutation to the expt$treatment variable! Otherwise all that happens is that R prints the results to screen and saves nothing!

Example:

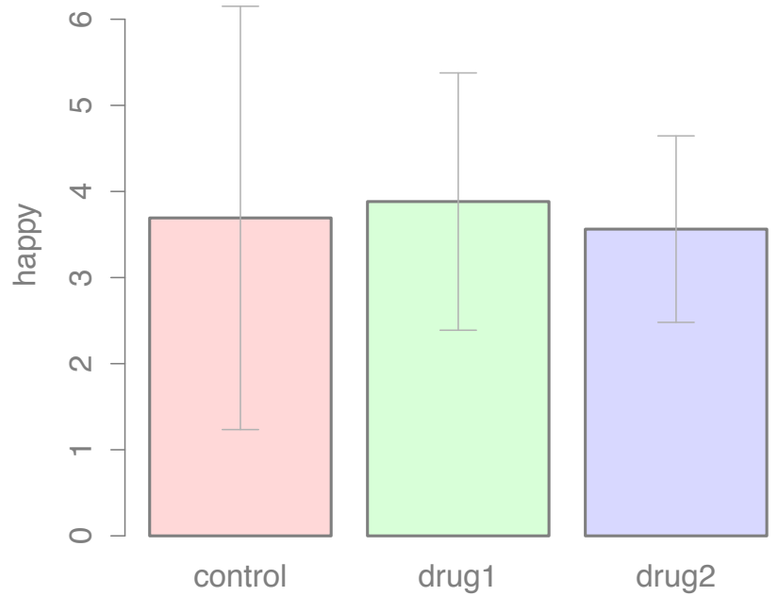
```
bars(
 formula = happy ~ treatment,
 data = expt
)
```




```
expt$treatment <- permuteLevels(
                    expt$treatment,
                    c(2,3,1)
                  )
```

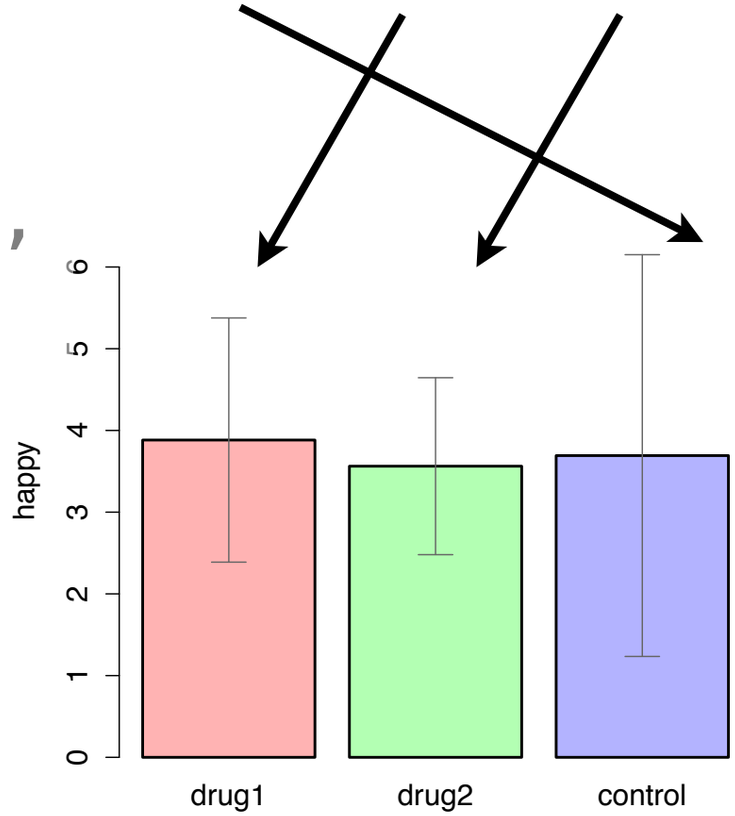
Example:

```
bars(
 formula = happy ~ treatment,
 data = expt
)
```

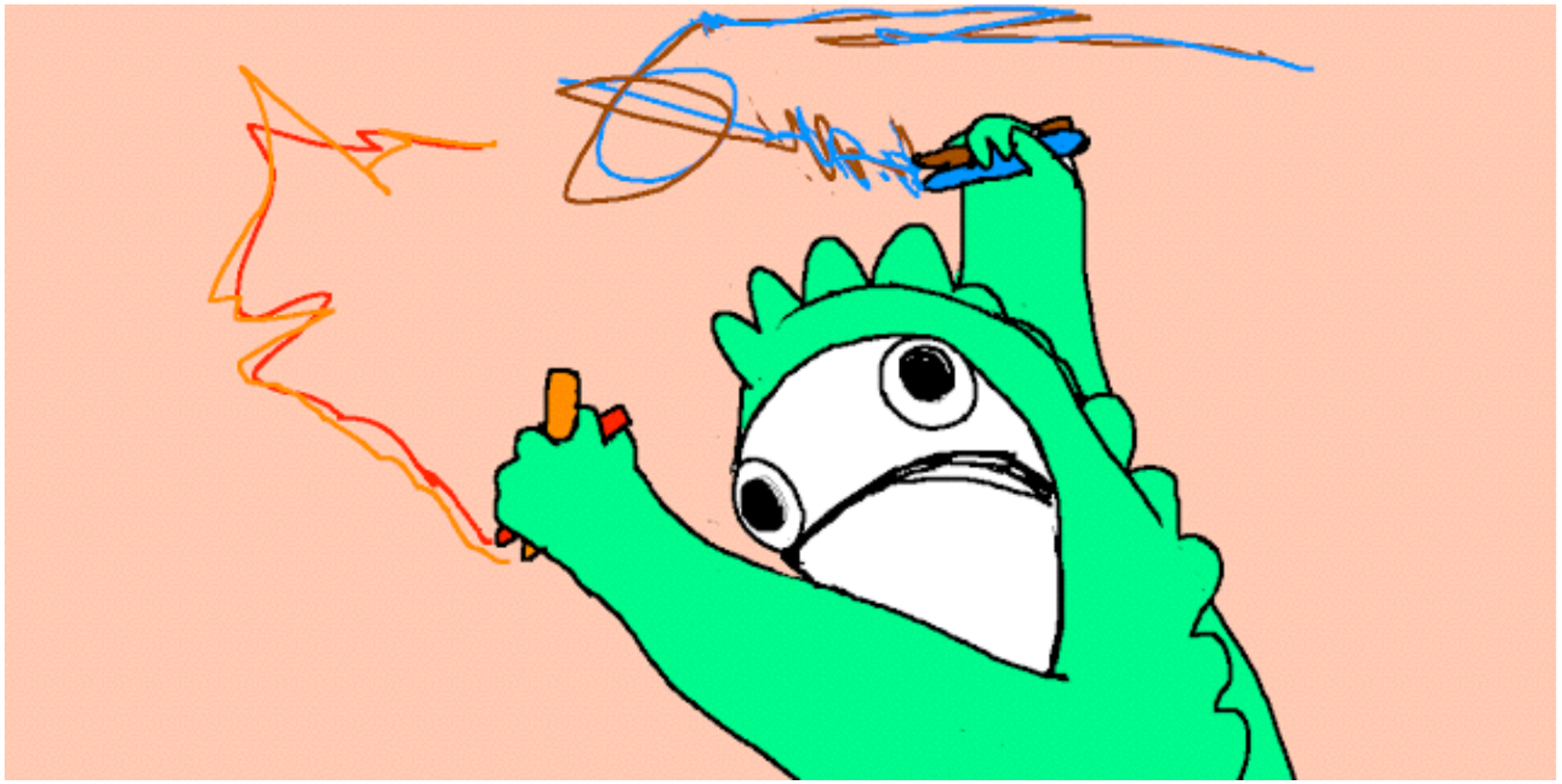



```
expt$treatment <- permuteLevels(
                    expt$treatment,
                    c(2,3,1)
                  )
```

```
bars(
 formula = happy ~ treatment,
 data = expt
)
```



Try it yourself (Exercise 2.5.4)



End of this section