Assignment 8.5

Nothing to hand in

The usual beginning:

```
library(tidyverse)
```

1. The data in `http://www.utsc.utoronto.ca/~butler/c32/calirain.txt` are rainfall and other measurements for 30 weather stations in California. Our aim is to understand how well the annual rainfall at these stations (measured in inches) can be predicted from the other measurements, which are the altitude (in feet above sea level), the latitude (degrees north of the equator) and the distance from the coast (in miles). Use R for this question.

   (a) Read in the data. You'll have to be careful here, since the values are space-delimited, but sometimes by more than one space, to make the columns line up. `read_table2`, with filename or url, will read it in.

   One of the variables is called `rainfall`, so as long as you *do not* call the data frame that, you should be safe.

Solution:

I used `rains` as the name of my data frame:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/calirain.txt"
rains=read_table2(my_url)
```

```
## Parsed with column specification:
## cols(
##  station = col_character(),
##  rainfall = col_double(),
##  altitude = col_integer(),
##  latitude = col_double(),
##  fromcoast = col_integer()
## )
```

I have the right number of rows and columns.

There is also `read_table`, but that requires *all* the columns, including the header row, to be lined up. You can try that here and see how it fails.

I don't need you to investigate the data yet (that happens in the next part), but this is interesting (to me):

```
rains
```

```
## # A tibble: 30 x 5
##          station rainfall altitude latitude fromcoast
##            <chr>    <dbl>    <int>    <dbl>     <int>
##  1       Eureka    39.57       43     40.8         1
##  2     RedBluff    23.27      341     40.2        97
##  3      Thermal    18.20     4152     33.8        70
##  4    FortBragg    37.48       74     39.4         1
##  5  SodaSprings    49.26     6752     39.3       150
##  6 SanFrancisco    21.82       52     37.8         5
##  7   Sacramento    18.07       25     38.5        80
##  8      SanJose    14.17       95     37.4        28
##  9  GiantForest    42.63     6360     36.6       145
## 10      Salinas    13.85       74     36.7        12
## # ... with 20 more rows
```

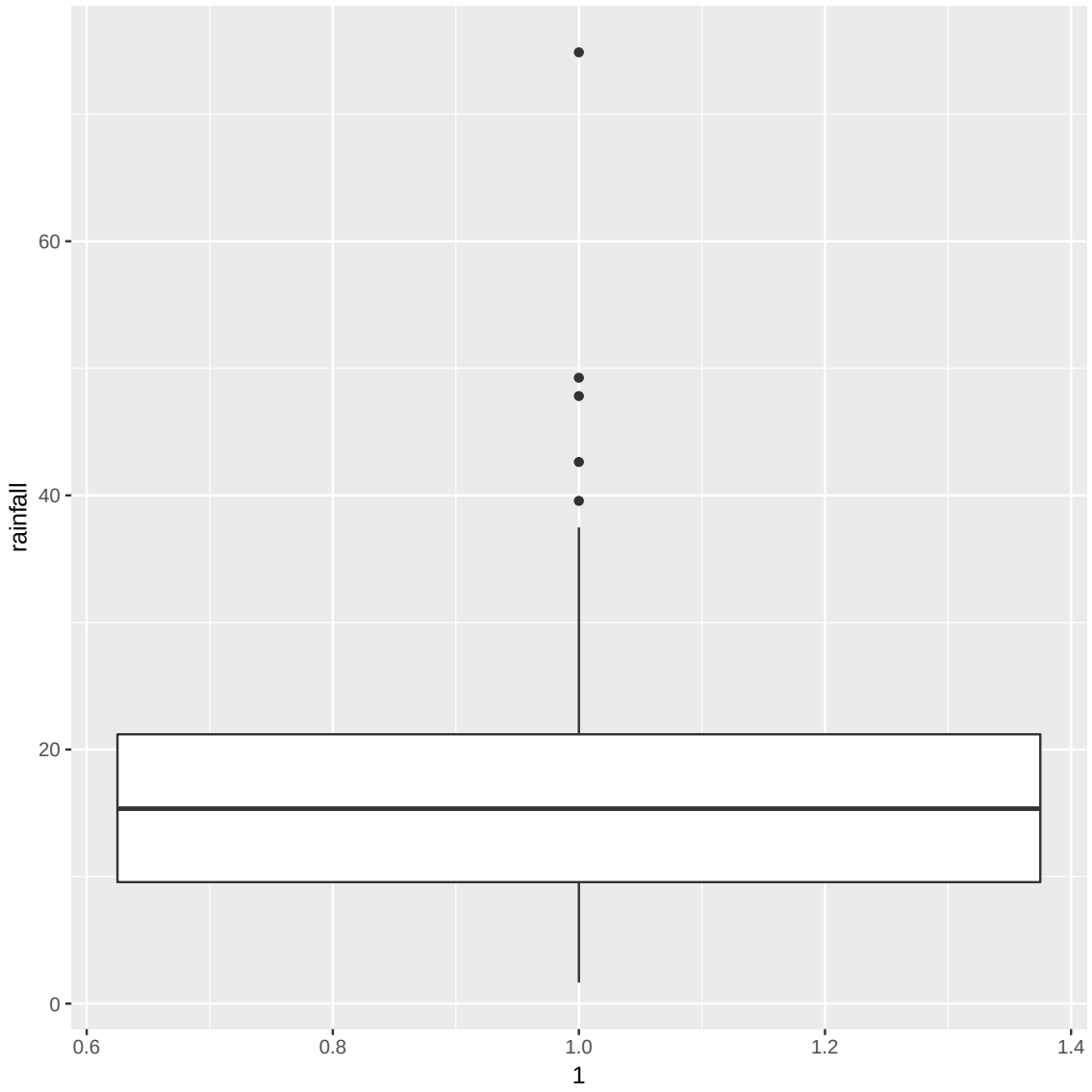
Some of the station names are two words, but they have been smooshed into one word, so that `read_table2` will recognize them as a single thing. Someone had already done that for us, so I didn't even have to do it myself.

If the station names had been two genuine words, a `.csv` would probably have been the best choice (the actual data values being separated by commas then, and not spaces).

(b) (2 marks) Make a boxplot of the rainfall figures, and explain why the values are reasonable. (A rainfall cannot be negative, and it is unusual for a annual rainfall to exceed 50 inches.) A `ggplot` boxplot needs *something* on the x-axis: the number 1 will do.

Solution:

```
ggplot(rains,aes(y=rainfall,x=1))+geom_boxplot()
```



There is only one rainfall over 50 inches, and the smallest one is close to zero but positive, so that is good.

What stations have those extreme values? `dplyr` is the tool for that, should you wish to find out (I didn't ask for this):

```
rains %>% filter(rainfall>50)
```

```
## # A tibble: 1 x 5
##        station rainfall altitude latitude fromcoast
##          <chr>    <dbl>    <int>    <dbl>     <int>
## 1 CrescentCity    74.87       35     41.7         1
```

This is a place right on the Pacific coast, almost up into Oregon (it's almost the northernmost of all the stations). So it makes sense that it would have a high rainfall, if anywhere does. (If you know anything about rainy places, you'll probably think of Vancouver and Seattle, in the Pacific Northwest.) Here

it is: https://www.google.ca/maps/place/Crescent+City,+CA,+USA/@41.7552589,-123.9652917,8.42z/data=!4m5!3m4!1s0x54d066375c6288db:0x76e89ab07375e62e!8m2!3d41.7557501!4d-124.2025913.

Which station has less than 2 inches of annual rainfall?

```
rains %>% filter(rainfall<2)
```

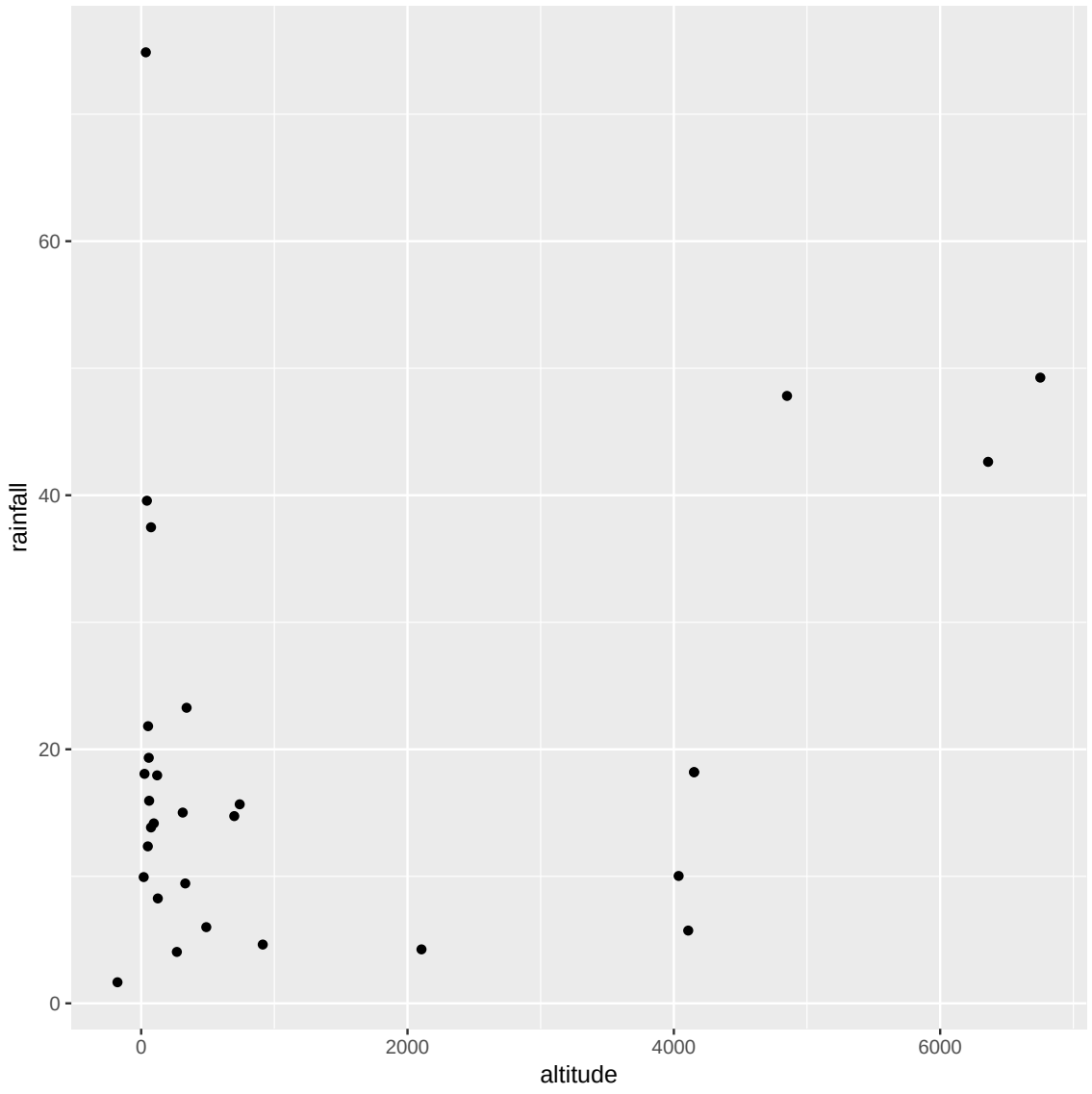
```
## # A tibble: 1 x 5
##       station rainfall altitude latitude fromcoast
##         <chr>    <dbl>    <int>    <dbl>     <int>
## 1 DeathValley     1.66     -178     36.5       194
```

The name of the station is a clue: this one is in the desert. So you'd expect very little rain. Its altitude is *negative*, so it's actually *below* sea level. This is correct. Here is where it is: https://www.google.ca/maps/place/Death+Valley,+CA,+USA/@36.6341288,-118.2252974,7.75z/data=!4m5!3m4!1s0x80c739a21e8fffb1:0x1c897383d723dd25!8m2!3d36.5322649!4d-116.9325408.

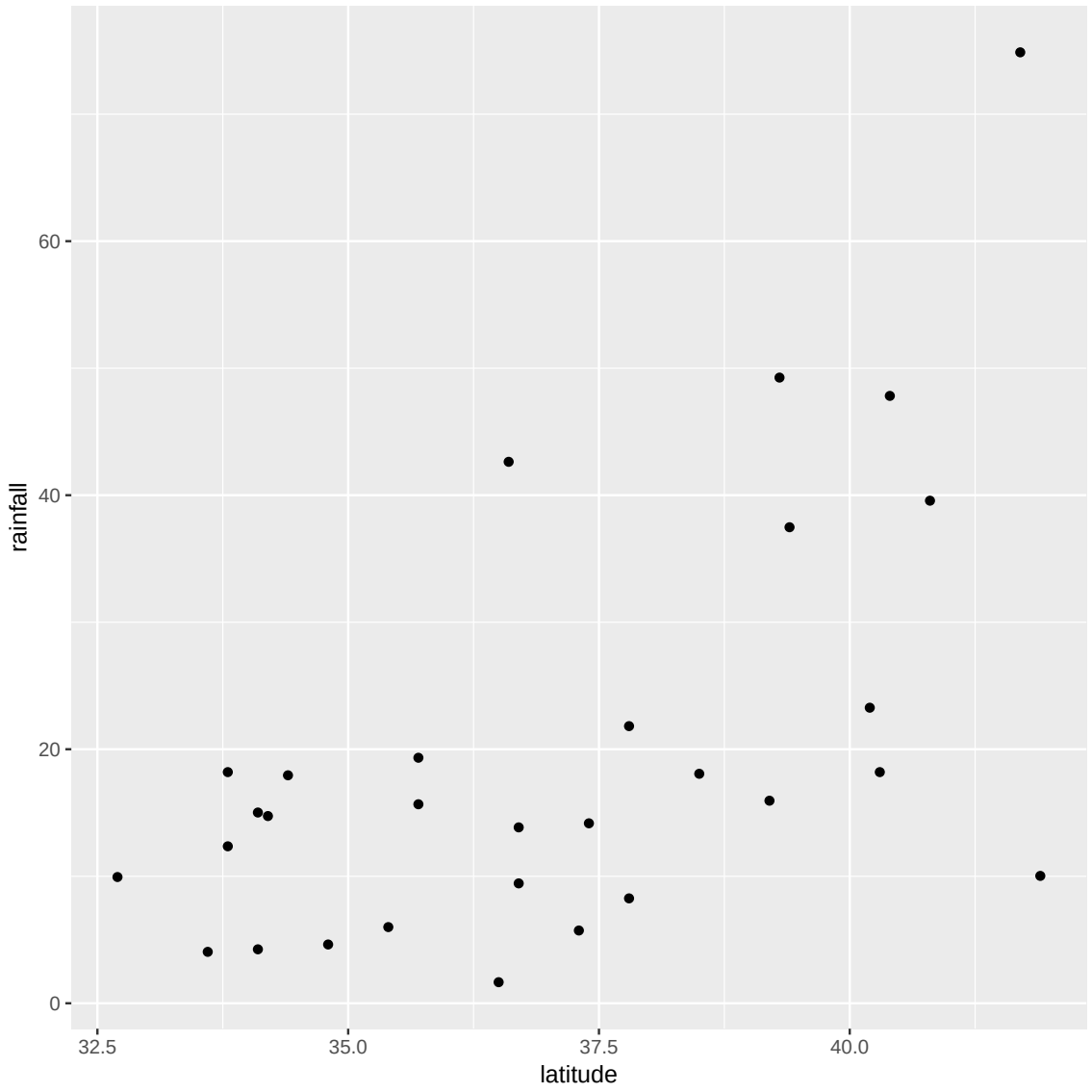
(c) Plot `rainfall` against each of the other quantitative variables (that is, not `station`).

Solution: That is, `altitude`, `latitude` and `fromcoast`. The obvious way to do this (perfectly acceptable) is one plot at a time:

```
ggplot(rains,aes(y=rainfall,x=altitude))+geom_point()
```

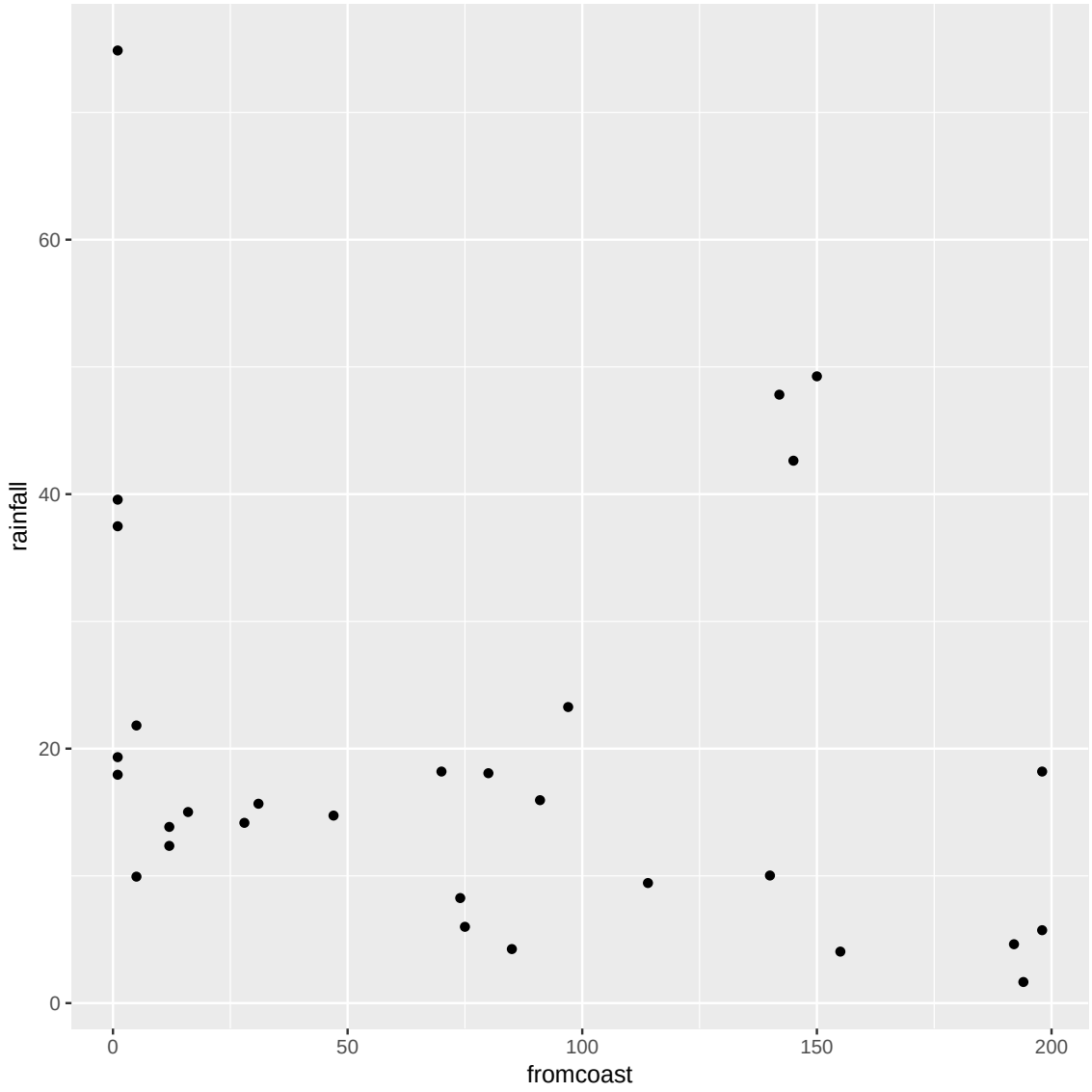



```
ggplot(rains,aes(y=rainfall,x=latitude))+geom_point()
```



and finally

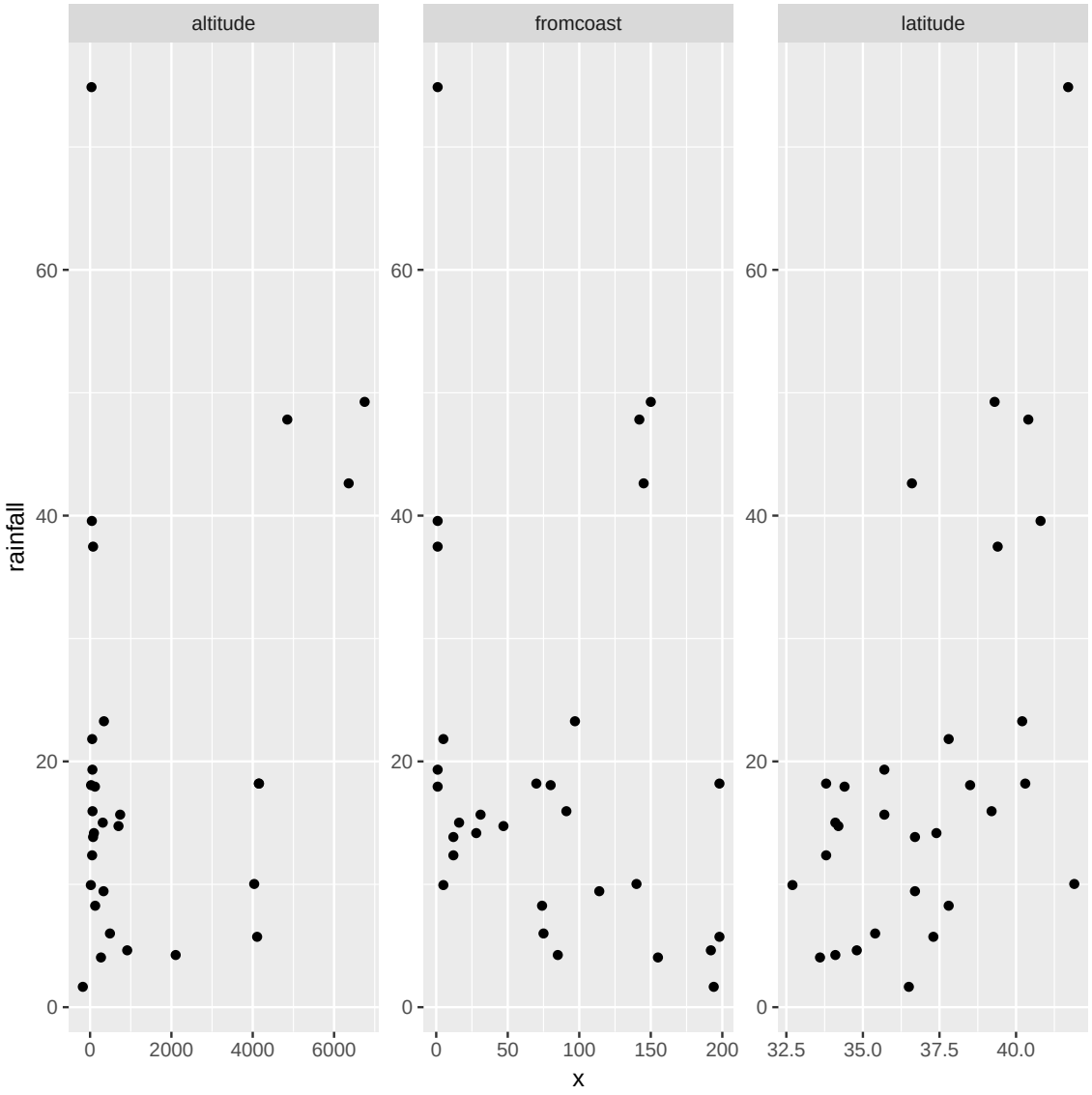
```
ggplot(rains,aes(y=rainfall,x=fromcoast))+geom_point()
```



You can add a smooth trend to these if you want. Up to you. Just the points is fine with me.

Here is a funky way to get all three plots in one shot:

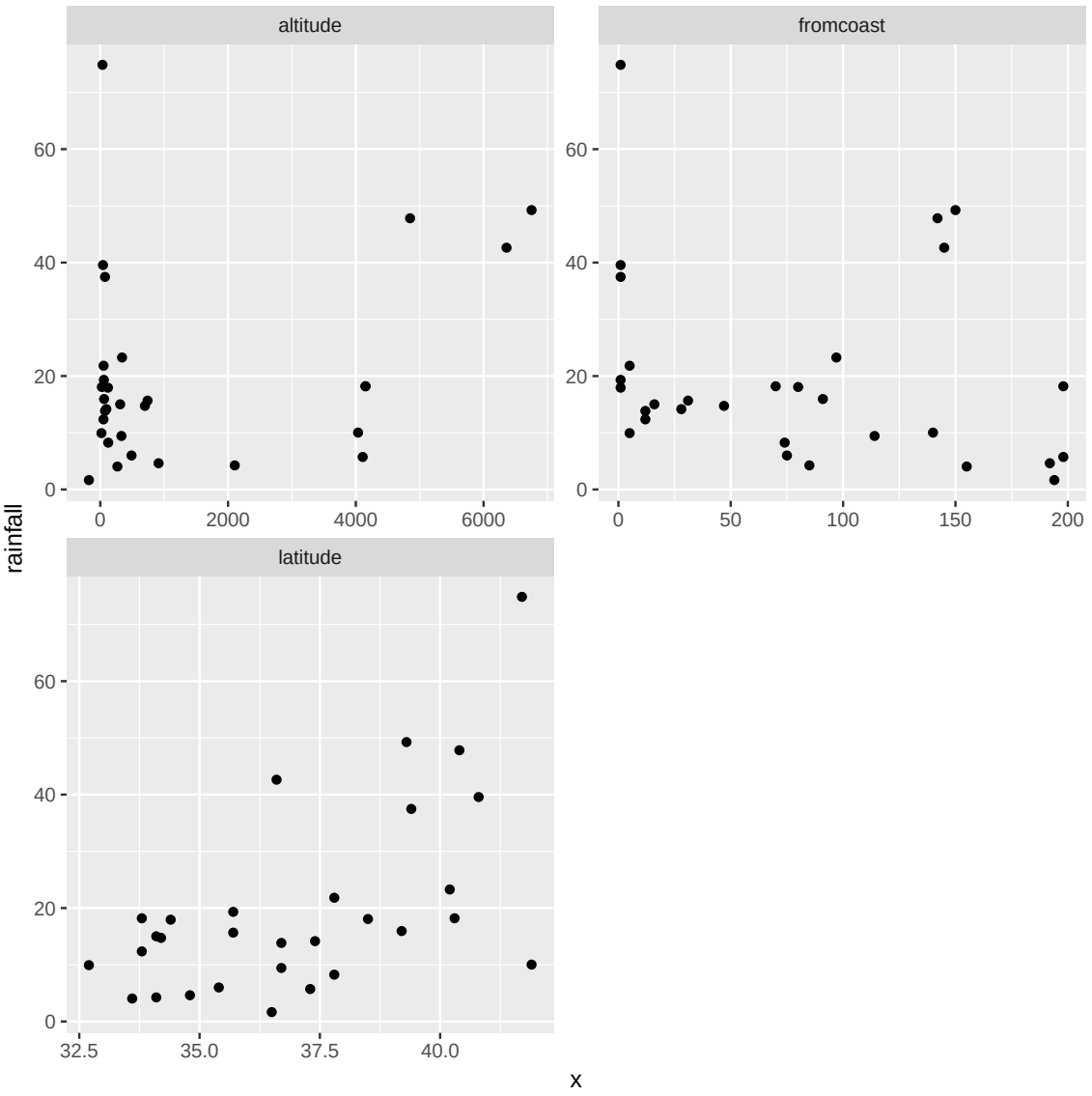
```
rains %>% gather(xname,x,altitude:fromcoast) %>%
  ggplot(aes(x=x,y=rainfall))+geom_point()+
    facet_wrap(~xname,scales="free")
```



This always seems extraordinarily strange if you haven't run into it before. The strategy is to put *all* the x-variables you want to plot into *one* column and then plot your y against the x-column. A nice side-effect of the way `gather` works is that what makes the x-columns different is that they are x-variables with different *names*, which is exactly what you want later for the facets. Thus: make a column of all the x's glued together, labelled by which x they are, then plot y against x but make a different sub-plot or "facet" for each different x-name. The last thing is that each x is measured on a different scale, and unless we take steps, all the sub-plots will have the *same* scale on each axis, which we don't want.

I'm not sure I like how it came out, with three very tall plots. `facet_wrap` can also take an `nrow` or an `ncol`, which tells it how many rows or columns to use for the display. Here, for example, two columns because I thought three was too many:

```
rains %>% gather(xname,x,altitude:fromcoast) %>%
  ggplot(aes(x=x,y=rainfall))+geom_point()+
    facet_wrap(~xname,scales="free",ncol=2)
```



Now, the three plots have come out about square, which I like a lot better.

(d) Look at the relationship of each other variable with `rainfall`. Justify the assertion that `latitude` seems most strongly related with `rainfall`. Is that relationship positive or negative? linear? Explain briefly.

Solution: Let's look at the three variables in turn:

- `altitude`: not much of anything. The stations near sea level have rainfall all over the place, though the three highest-altitude stations have the three highest rainfalls apart from Crescent City.
- `latitude`: there is a definite upward trend here, in that stations further north (higher latitude) are likely to have a higher rainfall. I'd call this trend linear (or, not obviously curved), though

the two most northerly stations have one higher and one much lower rainfall than you'd expect.

- `fromcoast`: this is a weak downward trend, though the trend is spoiled by those three stations about 150 miles from the coast that have more than 40 inches of rainfall.

Out of those, only `latitude` seems to have any meaningful relationship with `rainfall`.

(e) Fit a regression with `rainfall` as the response variable, and `latitude` as your explanatory variable. What are the intercept, slope and R-squared values? Is there a *significant* relationship between `rainfall` and your explanatory variable? What does that mean?

Solution: Save your `lm` into a variable, since it will get used again later:

```
rainfall.1=lm(rainfall~latitude,data=rains)
summary(rainfall.1)
```

```
##
## Call:
## lm(formula = rainfall ~ latitude, data = rains)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -27.297  -7.956  -2.103   6.082  38.262
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -113.3028    35.7210  -3.172  0.00366 **
## latitude       3.5950     0.9623   3.736  0.00085 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13.82 on 28 degrees of freedom
## Multiple R-squared:  0.3326,Adjusted R-squared:  0.3088
## F-statistic: 13.96 on 1 and 28 DF,  p-value: 0.0008495
```

My intercept is -113.3, slope is 3.6 and R-squared is 0.33 or 33%. (I want you to pull these numbers out of the output and round them off to something sensible.) The slope is significantly nonzero, its P-value being 0.00085: rainfall really does depend on latitude, although not strongly so.

Of course, I can easily do the others as well, though you don't have to:

```
rainfall.2=lm(rainfall~fromcoast,data=rains)
summary(rainfall.2)
```

```
##
## Call:
## lm(formula = rainfall ~ fromcoast, data = rains)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -15.240  -9.431  -6.603   2.871  51.147
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 23.77306    4.61296   5.154 1.82e-05 ***
## fromcoast   -0.05039    0.04431  -1.137    0.265
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.54 on 28 degrees of freedom
## Multiple R-squared:  0.04414,Adjusted R-squared:   0.01
## F-statistic: 1.293 on 1 and 28 DF,  p-value: 0.2651
```

Here, the intercept is 23.8, the slope is -0.05 and R-squared is a dismal 0.04 (4%). This is a way of seeing that this relationship is really weak, and it doesn't even have a curve to the trend or anything that would compensate for this. I looked at the scatterplot again and saw that if it were not for the point bottom right which is furthest from the coast and has almost no rainfall, there would be almost no trend at all. The slope here is not significantly different from zero, with a P-value of 0.265.

Finally:

```
rainfall.3=lm(rainfall~altitude,data=rains)
summary(rainfall.3)
```

```
##
## Call:
## lm(formula = rainfall ~ altitude, data = rains)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -20.620  -8.479  -2.729   4.555  58.271
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 16.514799   3.539141   4.666  6.9e-05 ***
## altitude     0.002394   0.001428   1.676    0.105
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.13 on 28 degrees of freedom
## Multiple R-squared:  0.09121,Adjusted R-squared:  0.05875
## F-statistic:  2.81 on 1 and 28 DF,  p-value: 0.1048
```

The intercept is 16.5, the slope is 0.002 and the R-squared is 0.09 or 9%, also terrible. The P-value is

0.105, which is not small enough to be significant.

So it looks as if it's only `latitude` that has any impact at all. This is the only explanatory variable with a significantly nonzero slope. On its own, at least.

(f) Fit a multiple regression predicting `rainfall` from all three of the other (quantitative) variables. Display the results. Comment is coming up later.

Solution: This, then:

```
rainfall.4=lm(rainfall~latitude+altitude+fromcoast,data=rains)
summary(rainfall.4)
```

```
##
## Call:
## lm(formula = rainfall ~ latitude + altitude + fromcoast, data = rains)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.722  -5.603  -0.531   3.510  33.317
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.024e+02  2.921e+01  -3.505 0.001676 **
## latitude     3.451e+00  7.949e-01   4.342 0.000191 ***
## altitude     4.091e-03  1.218e-03   3.358 0.002431 **
## fromcoast   -1.429e-01  3.634e-02  -3.931 0.000559 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.1 on 26 degrees of freedom
## Multiple R-squared:  0.6003,Adjusted R-squared:  0.5542
## F-statistic: 13.02 on 3 and 26 DF,  p-value: 2.205e-05
```

(g) What is the R-squared for the regression of the last part? How does that compare with the R-squared of your regression in part (e)?

Solution: The R-squared is 0.60 (60%), which is quite a bit bigger than the R-squared of 0.33 (33%) we got back in (e).

(h) What do you conclude about the importance of the variables that you did *not* include in your model in (e)? Explain briefly.

Solution: Both variables `altitude` and `fromcoast` are significant in this regression, so they have *something to add* over and above `latitude` when it comes to predicting rainfall, even though (and this seems odd) they have no apparent relationship with `rainfall` on their own.

Another way to say this is that the three variables work together as a team to predict rainfall, and together they do much better than any one of them can do by themselves.

This also goes to show that the scatterplots we began with don't get to the heart of multi-variable

relationships, because they are only looking at the variables two at a time.

(i) Make a suitable hypothesis test that the variables `altitude` and `fromcoast` significantly improve the prediction of `rainfall` over the use of `latitude` alone. What do you conclude?

Solution: This calls for `anova`. Feed this two fitted models, smaller (fewer explanatory variables) first. The null hypothesis is that the two models are equally good (so we should go with the smaller); the alternative is that the larger model is better, so that the extra complication is worth it:

```
anova(rainfall.1,rainfall.4)
```

```
## Analysis of Variance Table
##
## Model 1: rainfall ~ latitude
## Model 2: rainfall ~ latitude + altitude + fromcoast
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1     28 5346.8
## 2     26 3202.3  2    2144.5 8.7057 0.001276 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The P-value is small, so we reject the null in favour of the alternative: the regression with all three explanatory variables fits better than the one with just `latitude`, so the bigger model is the one we should go with.

If you have studied these things: this one is a "multiple-partial F-test", for testing the combined significance of more than one x but less than all the x's.[1]

2. A management consultancy obtained data on salaries and other work information of 100 company executives (from different companies). Their aim was to predict salary from some or all of the other variables (and to determine which of those other variables are important determinants of salary). The data are in `http://www.utsc.utoronto.ca/~butler/c32/execsal.xlsx`, as an Excel spreadsheet, with the columns being (respectively):

   - Row number (ignore)
   - Log of annual salary
   - experience (years)
   - education (years)
   - gender (1=male, 0=female)
   - number of employees supervised
   - corporate assets (millions of dollars)
   - board member (1=yes, 0=no)
   - age (years)
   - company profits (past 12 months, millions of dollars)
   - has international responsibility (1=yes, 0=no)
   - company's total sales (past 12 months, millions of dollars)

The consultancy used log of salary because the relationship with other variables (in previous studies) seemed to be straighter. (A consequence of using logs is that a one-unit increase in any of the other variables is associated with a certain *percentage* increase in annual salary, which often makes sense.) Note that the data set already contains a variable `logsal`, which is the log-salary, so you don't need to create one.

(a) Read the data into SAS, bearing in mind the format of the data. You'll need to know the name of the sheet you want to read in. Also, reading an Excel file only works "locally": that is, you'll need to grab your own copy of the spreadsheet and upload it to SAS Studio.

Solution: First, open the spreadsheet and take a look at it. The sheet you want is called `execsal2`. Save it somewhere on your computer and then upload it to SAS Studio.

Then, find a previous `proc import` with `dbms=xlsx`, and adapt it to what you need, replacing my username with yours:

```
proc import
  datafile='/home/ken/execsal.xlsx'
  dbms=xlsx
  out=salaries
  replace;
  sheet=execsal2;
  getnames=yes;
```

Or remember DODRG and this time note that you need an extra S for "sheet".

I ran that through `proc print` to check that I had the right thing, and I did. Or you can summarize:

```
proc means;
```

The MEANS Procedure

Variable	Label	N	Mean	Std Dev	Minimum	Maximum
row	row	100	50.5000000	29.0114920	1.0000000	100.0000000
logsal	logsal	100	11.4550180	0.2598109	10.6643000	12.0634000
exp	exp	100	13.0800000	7.3425287	1.0000000	26.0000000
educ	educ	100	16.0200000	2.3049354	12.0000000	20.0000000
gender	gender	100	0.6600000	0.4760952	0	1.0000000
sup	sup	100	340.1000000	167.1779733	60.0000000	600.0000000
cass	cass	100	175.1000000	15.4066102	150.0000000	200.0000000
board	board	100	0.4900000	0.5024184	0	1.0000000
age	age	100	42.8400000	9.0729034	23.0000000	64.0000000
profits	profits	100	7.7000000	1.5537508	5.0000000	10.0000000
int	int	100	0.1800000	0.3861229	0	1.0000000
sales	sales	100	24.8300000	2.7415803	20.0000000	30.0000000

As you see, there are 100 rows of data, which would be a lot for someone else to look at. The names (you can check) match up with what I said the variables were.

(b) Run a regression predicting log-salary from everything else, except row number. Show the text output (here and below).

Solution:

```
proc reg;
```

```
model logsal=exp educ gender sup cass board age profits int sales;
```

with output

The REG Procedure
Model: MODEL1
Dependent Variable: logsal logsal

Number of Observations Read 100
Number of Observations Used 100

Analysis of Variance

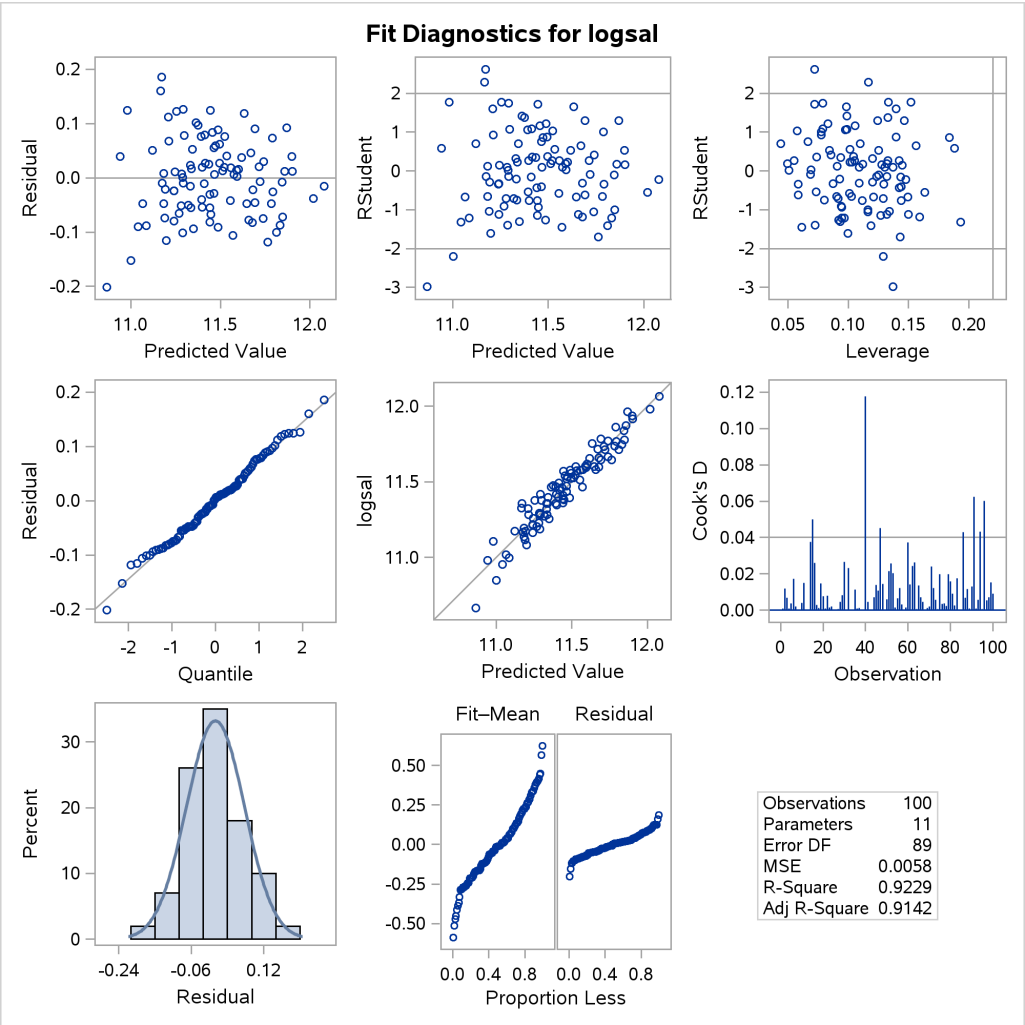
Source	DF	Sum of Squares	Mean Square	F Value	Pr > F
Model	10	6.16747	0.61675	106.54	<.0001
Error	89	0.51520	0.00579		
Corrected Total	99	6.68267			

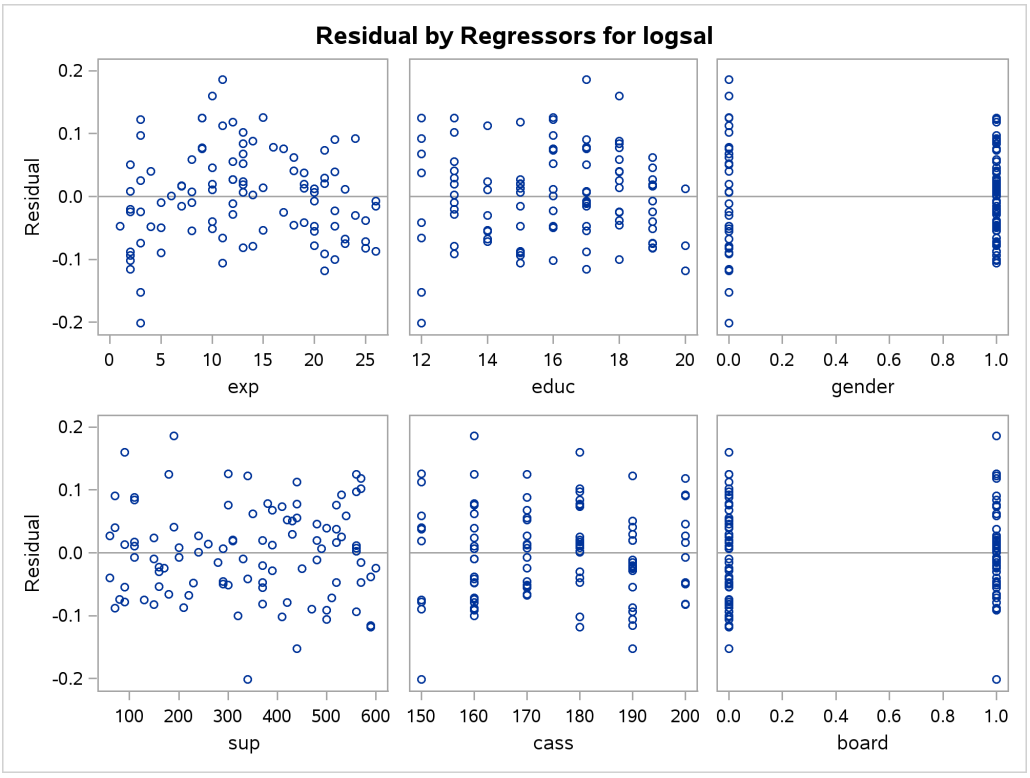
Root MSE	0.07608	R-Square	0.9229
Dependent Mean	11.45502	Adj R-Sq	0.9142
Coeff Var	0.66420		

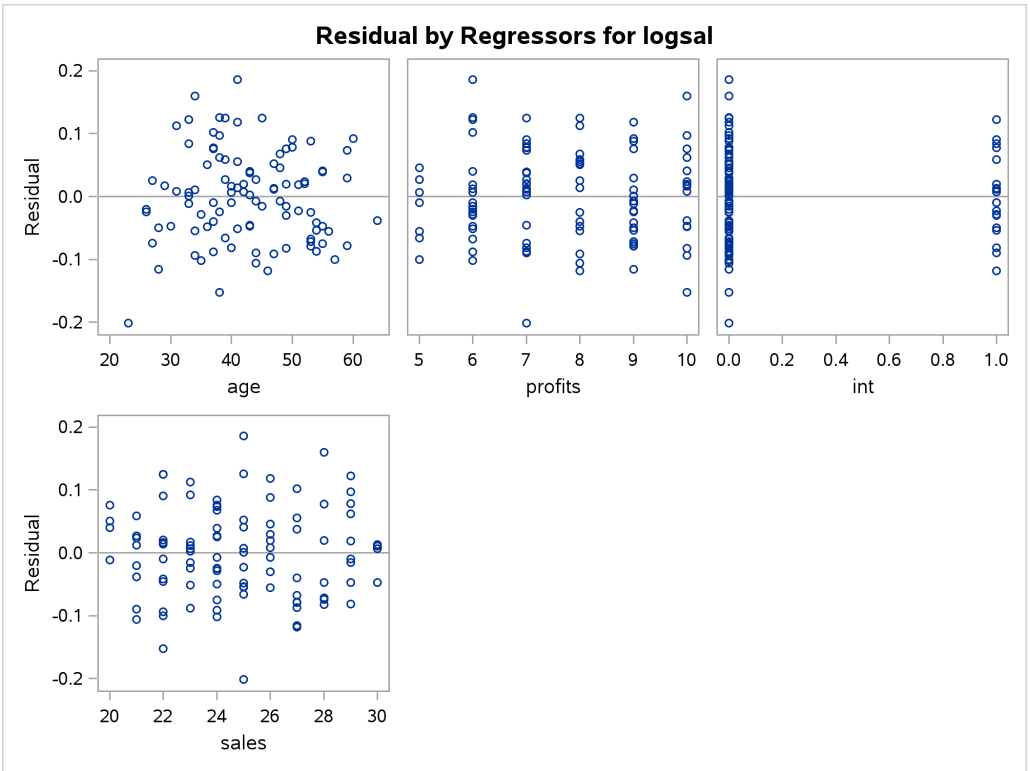
Parameter Estimates

Variable	Label	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	Intercept	1	10.02192	0.14805	67.69	<.0001
exp	exp	1	0.02792	0.00177	15.75	<.0001
educ	educ	1	0.02903	0.00343	8.48	<.0001
gender	gender	1	0.22434	0.01708	13.13	<.0001
sup	sup	1	0.00051397	0.00004922	10.44	<.0001
cass	cass	1	0.00205	0.00052499	3.90	0.0002
board	board	1	-0.01538	0.01686	-0.91	0.3641
age	age	1	-0.00050971	0.00144	-0.35	0.7238
profits	profits	1	-0.00263	0.00513	-0.51	0.6089
int	int	1	-0.02656	0.02037	-1.30	0.1956
sales	sales	1	-0.00097742	0.00296	-0.33	0.7420

and the graphs


Fit Diagnostics for logsal
Residual
Predicted Value
RStudent
Predicted Value
RStudent
Leverage
Residual
Quantile
logsal
Predicted Value
Cook's D
Observation
Percent
Residual
Fit–Mean
Residual
Proportion Less
Observations 100
Parameters 11
Error DF 89
MSE 0.0058
R-Square 0.9229
Adj R-Square 0.9142


Residual by Regressors for logsal
Residual
exp
educ
gender
sup
cass
board


Residual by Regressors for logsal
Residual
age
profits
int
sales

I didn't ask you to look at the plots, because I wanted you to do the variable-elimination (coming up). Normally, you would check that things are at least approximately OK, here and at the end. So I'll do it here, starting with the array of nine graphs of which I look at the usual two:

- residuals vs. fitted values, top left: a tiny bit of evidence of fanning-in, since the four residuals farthest from zero are all on the left. I'd really want more evidence of fanning-in than this, though.
- normal quantile plot of residuals: as straight as you could wish for.
- There are a lot of explanatory variables, and we get a plot of residuals against each one. These look pretty random and trend-free, so I don't think we need to be concerned. Note that some of the variables take only a few possible values (they are rather discrete), so you get stacks of points one above another, eg. for profits. Some of the explanatory variables are either 0 or 1 (these are "indicators" for categorical variables with two categories). For these, you want both categories to have average residual around zero with equal spread. For `gender` 1, the males, the residuals appear less spread out. I think we will have to live with that. (Another way would be to do the regression twice, for males and females separately.)

(c) Which explanatory variable is least significant? Run a regression without it.

Solution: This question is going to involve a great deal of copying and pasting. `sales` comes out first:

```
proc reg;
  model logsal=exp educ gender sup cass board age profits int;
```

with output

```
                          The REG Procedure
                            Model: MODEL1
                  Dependent Variable: logsal logsal

                Number of Observations Read         100
                Number of Observations Used         100
                           Analysis of Variance

                                    Sum of           Mean
Source                   DF        Squares         Square    F Value    Pr > F

Model                     9        6.16683        0.68520     119.55    <.0001
Error                    90        0.51583        0.00573
Corrected Total          99        6.68267

            Root MSE              0.07571    R-Square     0.9228
            Dependent Mean       11.45502    Adj R-Sq     0.9151
            Coeff Var             0.66090
                           Parameter Estimates

                                 Parameter      Standard
 Variable    Label       DF       Estimate         Error   t Value   Pr > |t|

 Intercept   Intercept    1        9.99498       0.12293     81.30     <.0001
 exp         exp          1        0.02776       0.00170     16.33     <.0001
 educ        educ         1        0.02904       0.00341      8.52     <.0001
 gender      gender       1        0.22525       0.01677     13.43     <.0001
 sup         sup          1     0.00051529    0.00004881     10.56     <.0001
 cass        cass         1        0.00204    0.00052207      3.91     0.0002
 board       board        1        -0.01472       0.01666     -0.88     0.3791
 age         age          1     -0.00041412       0.00140     -0.30     0.7683
 profits     profits      1        -0.00260       0.00510     -0.51     0.6118
 int         int          1        -0.02666       0.02027     -1.32     0.1916
```

(d) Continue removing the least significant variable until you need to stop, and explain briefly why you stopped.

Solution: You might guess that `board`, `age`, `profits` and `int` will need to come out, but take them one at a time, `age` first:

```
proc reg;
  model logsal=exp educ gender sup cass board profits int;
```

giving

```
                          The REG Procedure
                            Model: MODEL1
                  Dependent Variable: logsal logsal

                Number of Observations Read         100
                Number of Observations Used         100
                          Analysis of Variance

                                  Sum of           Mean
Source                    DF        Squares         Square    F Value    Pr > F

Model                      8        6.16633        0.77079     135.85    <.0001
Error                     91        0.51633        0.00567
Corrected Total           99        6.68267
             Root MSE              0.07533    R-Square     0.9227
             Dependent Mean       11.45502    Adj R-Sq     0.9159
             Coeff Var             0.65758
                          Parameter Estimates

                               Parameter      Standard
 Variable    Label       DF      Estimate         Error   t Value   Pr > |t|

 Intercept   Intercept    1       9.97787       0.10791     92.47     <.0001
 exp         exp          1       0.02737       0.00103     26.47     <.0001
 educ        educ         1       0.02915       0.00337      8.65     <.0001
 gender      gender       1       0.22451       0.01650     13.61     <.0001
 sup         sup          1    0.00051473    0.00004853     10.61     <.0001
 cass        cass         1       0.00206    0.00051702      3.98     0.0001
 board       board        1      -0.01336       0.01593     -0.84     0.4037
 profits     profits      1      -0.00259       0.00508     -0.51     0.6117
 int         int          1      -0.02617       0.02010     -1.30     0.1961
```

Then profits:

```
proc reg;
  model logsal=exp educ gender sup cass board int;
```

giving

```
                          The REG Procedure
                            Model: MODEL1
                  Dependent Variable: logsal logsal

                Number of Observations Read         100
                Number of Observations Used         100
                           Analysis of Variance

                                    Sum of           Mean
Source                   DF        Squares         Square    F Value    Pr > F

Model                     7        6.16486        0.88069     156.48    <.0001
Error                    92        0.51781        0.00563
Corrected Total          99        6.68267
             Root MSE              0.07502    R-Square     0.9225
             Dependent Mean       11.45502    Adj R-Sq     0.9166
             Coeff Var             0.65493
                           Parameter Estimates

                                  Parameter      Standard
 Variable    Label       DF      Estimate         Error   t Value   Pr > |t|

 Intercept   Intercept    1       9.96623       0.10503     94.88     <.0001
 exp         exp          1       0.02736       0.00103     26.58     <.0001
 educ        educ         1       0.02908       0.00335      8.67     <.0001
 gender      gender       1       0.22430       0.01643     13.65     <.0001
 sup         sup          1    0.00051576    0.00004829     10.68     <.0001
 cass        cass         1       0.00201    0.00050694      3.97     0.0001
 board       board        1      -0.01202       0.01564     -0.77     0.4444
 int         int          1      -0.02491       0.01986     -1.25     0.2130
```

Then board:

```
proc reg;
  model logsal=exp educ gender sup cass int;
```

```
                          The REG Procedure
                            Model: MODEL1
                  Dependent Variable: logsal logsal

                Number of Observations Read         100
                Number of Observations Used         100
                           Analysis of Variance

                                    Sum of           Mean
Source                    DF       Squares         Square    F Value    Pr > F

Model                      6       6.16154        1.02692     183.26    <.0001
Error                     93       0.52113        0.00560
Corrected Total           99       6.68267

              Root MSE              0.07486    R-Square     0.9220
              Dependent Mean       11.45502    Adj R-Sq     0.9170
              Coeff Var             0.65348
                           Parameter Estimates

                                 Parameter      Standard
 Variable    Label       DF       Estimate         Error   t Value   Pr > |t|

 Intercept   Intercept    1        9.94602       0.10146     98.03     <.0001
 exp         exp          1        0.02733       0.00103     26.62     <.0001
 educ        educ         1        0.02933       0.00333      8.81     <.0001
 gender      gender       1        0.22322       0.01633     13.67     <.0001
 sup         sup          1     0.00052305    0.00004724     11.07     <.0001
 cass        cass         1        0.00206    0.00050144      4.11     <.0001
 int         int          1       -0.02549       0.01980     -1.29     0.2011
```

Finally (we hope) `int`:

```
proc reg;
  model logsal=exp educ gender sup cass;
```

```
                          The REG Procedure
                            Model: MODEL1
                  Dependent Variable: logsal logsal

                Number of Observations Read         100
                Number of Observations Used         100
                           Analysis of Variance

                                    Sum of           Mean
Source                   DF        Squares         Square    F Value    Pr > F

Model                     5        6.15225        1.23045     218.06    <.0001
Error                    94        0.53041        0.00564
Corrected Total          99        6.68267
             Root MSE              0.07512    R-Square     0.9206
             Dependent Mean       11.45502    Adj R-Sq     0.9164
             Coeff Var             0.65576
                           Parameter Estimates

                                 Parameter      Standard
 Variable    Label       DF       Estimate         Error   t Value   Pr > |t|

 Intercept   Intercept    1        9.96193       0.10106     98.58     <.0001
 exp         exp          1        0.02728       0.00103     26.50     <.0001
 educ        educ         1        0.02909       0.00334      8.72     <.0001
 gender      gender       1        0.22469       0.01635     13.74     <.0001
 sup         sup          1     0.00052442    0.00004740     11.06     <.0001
 cass        cass         1        0.00196    0.00049718      3.95     0.0002
```

Yep, that's the end. Everything else is strongly significant and has to stay in the model.

Also note that R-squared began and also ended around 92%: taking out those variables has had only a tiny effect on the fit of the model.

(e) Which explanatory variables are in your final model? Name them in full. That is, don't just list the names of the variables.

Solution: These ones:

- Years of experience
- Years of education
- Gender
- Number of employees supervised
- Corporate assets

(f) Look at each of your slope coefficients. Are they positive or negative? Does that make sense in the context of this problem?

Solution: Mine are all positive. That is, someone who has more years of experience, more education, supervises more employees or works in a company with more corporate assets would be expected to receive a higher salary. We'd expect all of these variables to have this kind of effect.

The one I didn't talk about was `gender`. This is also positive. Since males were 1 and females 0, according to the question, this means that males are expected to make more than females, *all else being equal*. This may not make you happy, but it's what the data are saying. (And note the strength of the conclusion: it's *after adjusting for any other differences between males and females*.)

The right thing to do next is to look at residual plots for your final model. The *right* thing to do is to split your data into a "training set" with which you build your model, and a separate "test set" on which you see how well it works. But that's farther than we go now.

SAS also contains a procedure called `glmselect`, which automates this process. Here's how it looks for this dataset:

```
proc glmselect;
  model logsal=exp educ gender sup cass board age profits int sales
    / selection=backward;
```

with output

```
                    The GLMSELECT Procedure

                Data Set                     WORK.SALARIES
                Dependent Variable                 logsal
                Selection Method                 Backward
                Select Criterion                      SBC
                Stop Criterion                        SBC
                Effect Hierarchy Enforced            None
                 Number of Observations Read          100
                 Number of Observations Used          100
                              Dimensions

                       Number of Effects       11
                       Number of Parameters    11
                        The GLMSELECT Procedure

                       Backward Selection Summary

                      Effect         Number
               Step   Removed    Effects In           SBC

                  0                      11     -476.1799
              ------------------------------------------
                  1   sales              10     -480.6625
                  2   age                 9     -485.1707
                  3   profits             8     -489.4911
                  4   board               7     -493.4571
                  5   int                 6     -496.2957*

                       * Optimal Value of Criterion
```

Selection stopped at a local minimum of the SBC criterion.

Stop Details

Candidate For	Effect	Candidate SBC		Compare SBC
Removal	cass	-485.5667	>	-496.2957

The GLMSELECT Procedure

Selected Model

The selected model is the model at the last step (Step 5).

Effects: Intercept exp educ gender sup cass

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Value
Model	5	6.15225	1.23045	218.06
Error	94	0.53041	0.00564	
Corrected Total	99	6.68267		

Root MSE	0.07512
Dependent Mean	11.45502
R-Square	0.9206
Adj R-Sq	0.9164
AIC	-409.92676
AICC	-408.70937
SBC	-496.29574

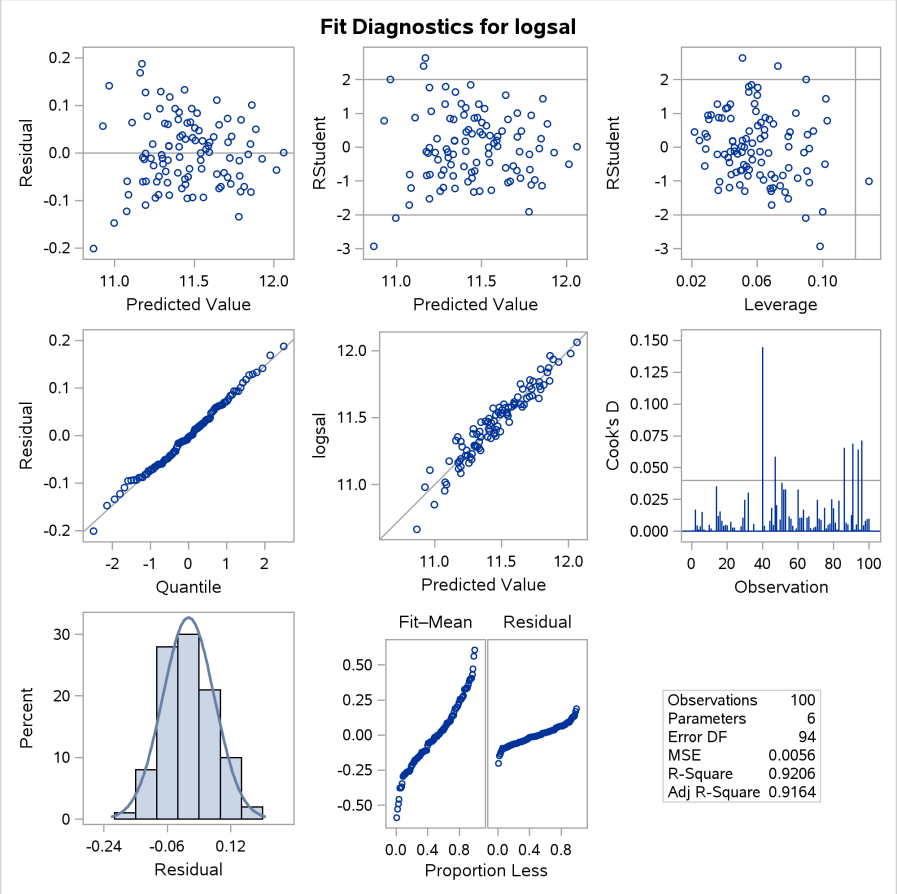
Parameter Estimates

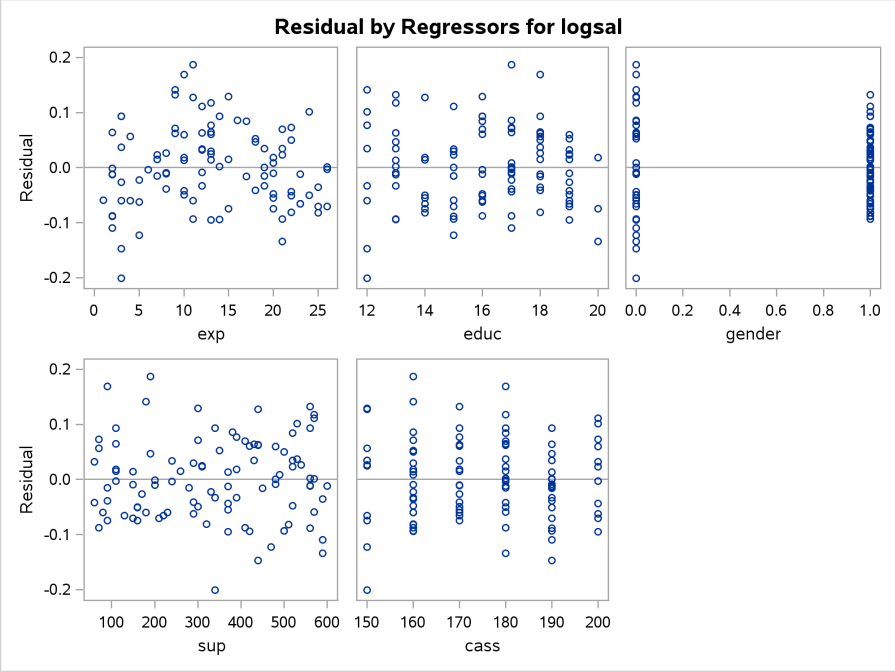
Parameter	DF	Estimate	Standard Error	t Value
Intercept	1	9.961935	0.101057	98.58
exp	1	0.027276	0.001029	26.50
educ	1	0.029092	0.003337	8.72
gender	1	0.224693	0.016350	13.74
sup	1	0.000524	0.000047398	11.06
cass	1	0.001962	0.000497	3.95

You can read through the output to see which variables were removed at each step, and which ones were left at the end: the same five as we found, since the procedure is supposed to be identical.

`proc glmselect` can also produce plots. See the baseball example at `https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_glmselect_sect030.htm` for illustrations.

We should probably look at our residual plots (from our last regression) just to make sure that all is OK:





The few issues we have are the same as before, which we decided to live with.

3. The (US) Federal Trade Commission assesses cigarettes according to their tar, nicotine and carbon monoxide contents. In a particular year, 25 brands were assessed. For each brand, the tar, nicotine and carbon monoxide (all in milligrams) were measured, along with the weight in grams. Our aim is to predict carbon monoxide from any or all of the other variables. The data are in `http://www.utsc.utoronto.ca/~butler/c32/ftccigar.txt`. These are aligned by column (except for the variable names), with more than one space between each column of data.

   (a) Read the data into R, and check that you have 25 observations and 4 variables.

Solution: This specification calls for `read_table2`:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/ftccigar.txt"
cigs=read_table2(my_url)
```

```
## Parsed with column specification:
## cols(
##   tar = col_double(),
##   nicotine = col_double(),
##   weight = col_double(),
##   co = col_double()
## )
```

```
cigs
```

```
## # A tibble: 25 x 4
##      tar nicotine weight    co
##    <dbl>    <dbl>  <dbl> <dbl>
##  1  14.1     0.86 0.9853  13.6
##  2  16.0     1.06 1.0938  16.6
##  3  29.8     2.03 1.1650  23.5
##  4   8.0     0.67 0.9280  10.2
##  5   4.1     0.40 0.9462   5.4
##  6  15.0     1.04 0.8885  15.0
##  7   8.8     0.76 1.0267   9.0
##  8  12.4     0.95 0.9225  12.3
##  9  16.6     1.12 0.9372  16.3
## 10  14.9     1.02 0.8858  15.4
## # ... with 15 more rows
```

Yes, I have 25 observations on 4 variables indeed.

`read_delim` won't work (try it and see what happens), because that would require the values to be separated by *exactly one* space.

(b) Run a regression to predict carbon monoxide from the other variables, and obtain a summary of the output.

Solution: The word "summary" is meant to be a big clue that `summary` is what you need:

```
cigs.1=lm(co~tar+nicotine+weight,data=cigs)
summary(cigs.1)
```

```
##
## Call:
## lm(formula = co ~ tar + nicotine + weight, data = cigs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.89261 -0.78269  0.00428  0.92891  2.45082
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.2022     3.4618   0.925 0.365464
## tar           0.9626     0.2422   3.974 0.000692 ***
## nicotine     -2.6317     3.9006  -0.675 0.507234
## weight       -0.1305     3.8853  -0.034 0.973527
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.446 on 21 degrees of freedom
## Multiple R-squared:  0.9186,Adjusted R-squared:  0.907
## F-statistic: 78.98 on 3 and 21 DF,  p-value: 1.329e-11
```

(c) Which one of your explanatory variables would you remove from this regression? Explain (very) briefly. Go ahead and fit the regression without it, and describe how the change in R-squared from the regression in (b) was entirely predictable.

Solution: First, the x-variable to remove. The obvious candidate is `weight`, since it has easily the highest, and clearly non-significant, P-value. So, out it comes:

```
cigs.2=lm(co~tar+nicotine,data=cigs)
summary(cigs.2)
```

```
##
## Call:
## lm(formula = co ~ tar + nicotine, data = cigs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.89941 -0.78470 -0.00144  0.91585  2.43064
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0896     0.8438   3.662 0.001371 **
## tar           0.9625     0.2367   4.067 0.000512 ***
## nicotine     -2.6463     3.7872  -0.699 0.492035
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.413 on 22 degrees of freedom
## Multiple R-squared:  0.9186,Adjusted R-squared:  0.9112
## F-statistic: 124.1 on 2 and 22 DF,  p-value: 1.042e-12
```

R-squared has dropped from 0.9186 to . . . 0.9186! That is, taking out `weight` has not just had a minimal effect on R-squared; it's not changed R-squared at all. This is because `weight` was so far from being significant: it literally had *nothing* to add.

Another way of achieving the same thing is via the function `update`, which takes a fitted model object and describes the *change* that you want to make:

```
cigs.2a=update(cigs.1,.~.-weight)
summary(cigs.2a)
```

```
##
## Call:
## lm(formula = co ~ tar + nicotine, data = cigs)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.89941 -0.78470 -0.00144  0.91585  2.43064
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.0896     0.8438   3.662 0.001371 **
## tar           0.9625     0.2367   4.067 0.000512 ***
## nicotine     -2.6463     3.7872  -0.699 0.492035
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.413 on 22 degrees of freedom
## Multiple R-squared:  0.9186,Adjusted R-squared:  0.9112
## F-statistic: 124.1 on 2 and 22 DF,  p-value: 1.042e-12
```

This can be shorter than describing the whole model again, as you do with the `cigs.2` version of `lm`. The syntax is that you first specify a "base" fitted model object that you're going to update. Because the model `cigs.1` contains all the information about the kind of model it is, and which data frame the data come from, R already knows that this is a linear multiple regression and which x's it contains. The second thing to describe is the change from the "base". In this case, we want to use the same response variable and all the same explanatory variables that we had before, except for `weight`. This is specified by a special kind of model formula where `.` means "whatever was there before": in English, "same response and same explanatories except take out `weight`".

(d) Fit a regression predicting carbon monoxide from `nicotine` *only*, and display the summary.

Solution: As you would guess:

```
cigs.3=lm(co~nicotine,data=cigs)
summary(cigs.3)
```

```
##
## Call:
## lm(formula = co ~ nicotine, data = cigs)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.3273 -1.2228  0.2304  1.2700  3.9357
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.6647     0.9936   1.675    0.107
## nicotine     12.3954     1.0542  11.759 3.31e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.828 on 23 degrees of freedom
## Multiple R-squared:  0.8574,Adjusted R-squared:  0.8512
## F-statistic: 138.3 on 1 and 23 DF,  p-value: 3.312e-11
```

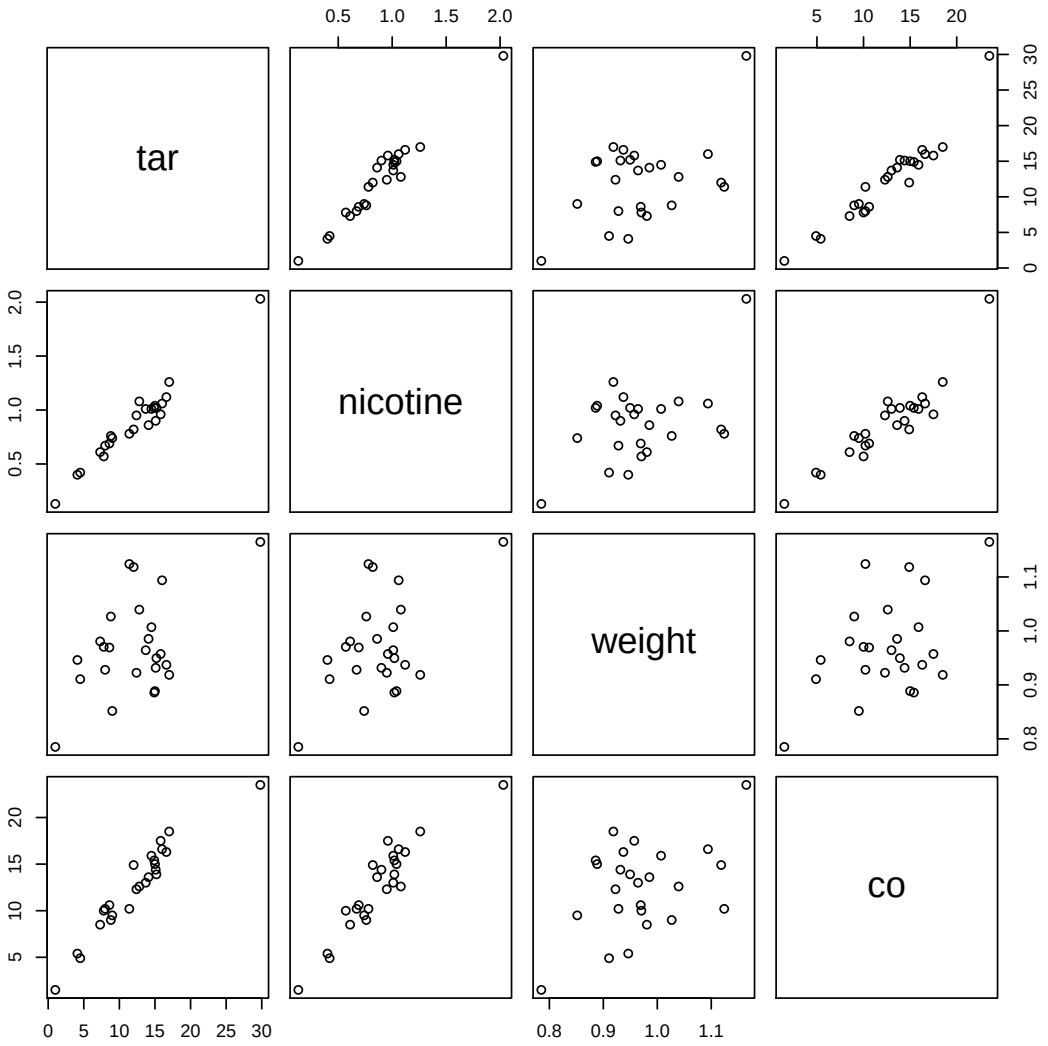
(e) `nicotine` was far from being significant in the model of (c), and yet in the model of (d), it was *strongly* significant, and the R-squared value of (d) was almost as high as that of (c). What does this say about the importance of `nicotine` as an explanatory variable? Explain, as briefly as you can manage.

Solution: What this says is that you *cannot* say anything about the "importance" of `nicotine` without also describing the context that you're talking about. *By itself*, `nicotine` is important, but *when you have `tar` in the model*, `nicotine` is not important: precisely, it now has nothing to add over and above the predictive value that `tar` has. You might guess that this is because `tar` and `nicotine` are "saying the same thing" in some fashion. We'll explore that in a moment.

(f) Make a "pairs plot": that is, scatter plots between all pairs of variables. This can be done by feeding the whole data frame into `plot`.[2] Do you see any strong relationships that do *not* include `co`? Does that shed any light on the last part? Explain briefly (or "at length" if that's how it comes out).

Solution: Plot the entire data frame:

```
plot(cigs)
```



We're supposed to ignore `co`, but I comment that strong relationships between `co` and *both* of `tar` and `nicotine` show up here, along with `weight` being at most weakly related to anything else.

That leaves the relationship of `tar` and `nicotine` with each other. That also looks like a strong linear trend. When you have correlations between explanatory variables, it is called "multicollinearity".

I mentioned a while back (in class) that having correlated x's was trouble. Here is where we find out why. The problem is that when `co` is large, `nicotine` is large, and a large value of `tar` will come along with it. So we don't know whether a large value of `co` is caused by a large value of `tar` or a large value of `nicotine`: there is no way to separate out their effects because in effect they are "glued together".

You might know of this effect (in an experimental design context) as "confounding": the effect of `tar` on `co` is confounded with the effect of `nicotine` on `co`, and you can't tell which one deserves the credit

for predicting `co`.

If you were able to design an experiment here, you could (in principle) manufacture a bunch of cigarettes with high tar; some of them would have high nicotine and some would have low. Likewise for low tar. Then the correlation between `nicotine` and `tar` would go away, their effects on `co` would no longer be confounded, and you could see unambiguously which one of the variables deserves credit for predicting `co`. Or maybe it depends on both, genuinely, but at least then you'd know.

We, however, have an observational study, so we have to make do with the data we have. Confounding is one of the risks we take when we work with observational data.

This was a "base graphics" plot. There is a way of doing a `ggplot`-style "pairs plot", as this is called, thus:

```
library(GGally)
```

```
## Warning: replacing previous import by 'utils::capture.output' when loading 'GGally'
```

```
## Warning: replacing previous import by 'utils::head' when loading 'GGally'
```

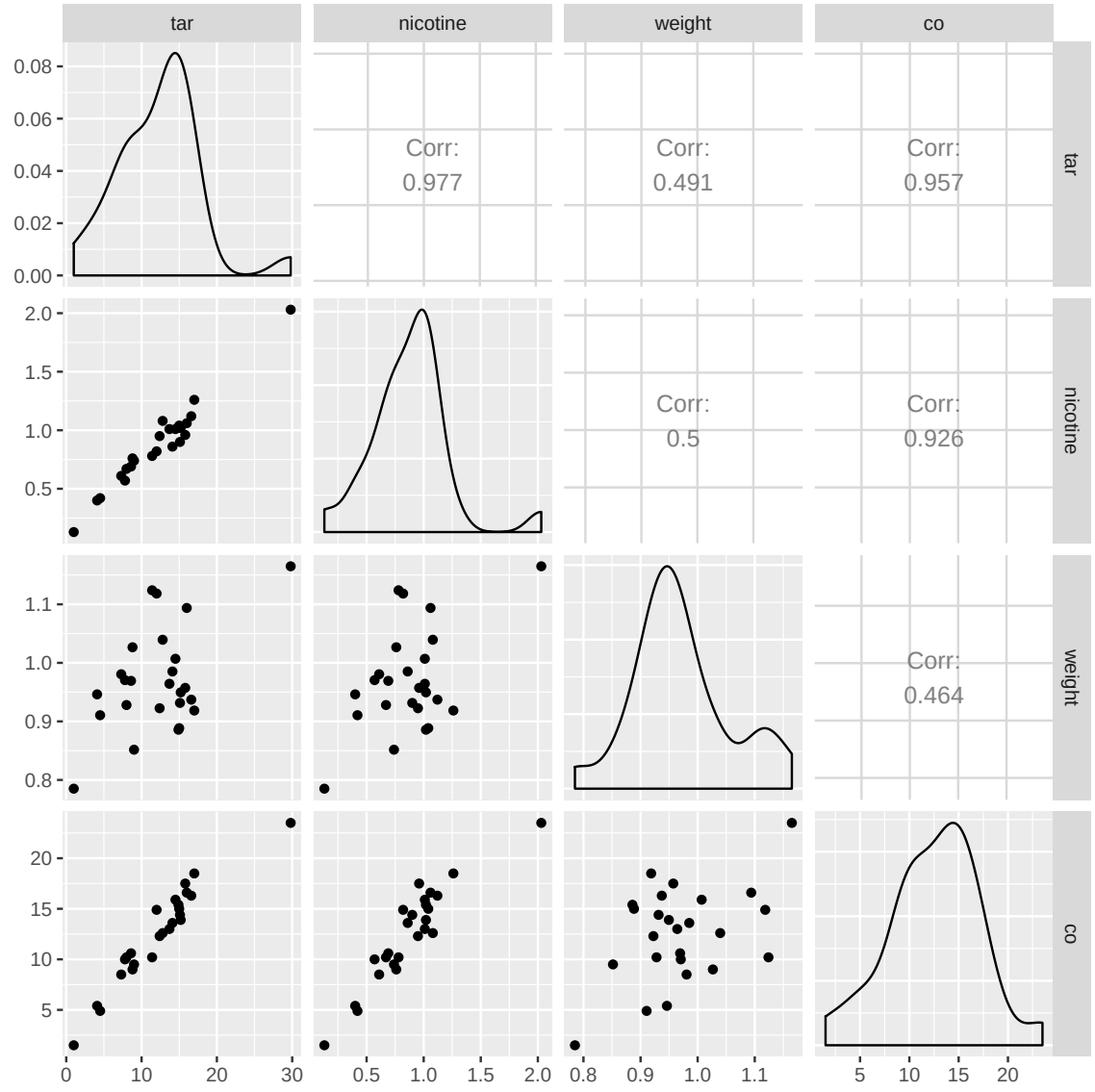
```
## Warning: replacing previous import by 'utils::installed.packages' when loading 'GGally'
```

```
## Warning: replacing previous import by 'utils::str' when loading 'GGally'
```

```
##
## Attaching package: 'GGally'
```

```
## The following object is masked from 'package:dplyr':
##
##     nasa
```

```
cigs %>% ggpairs()
```

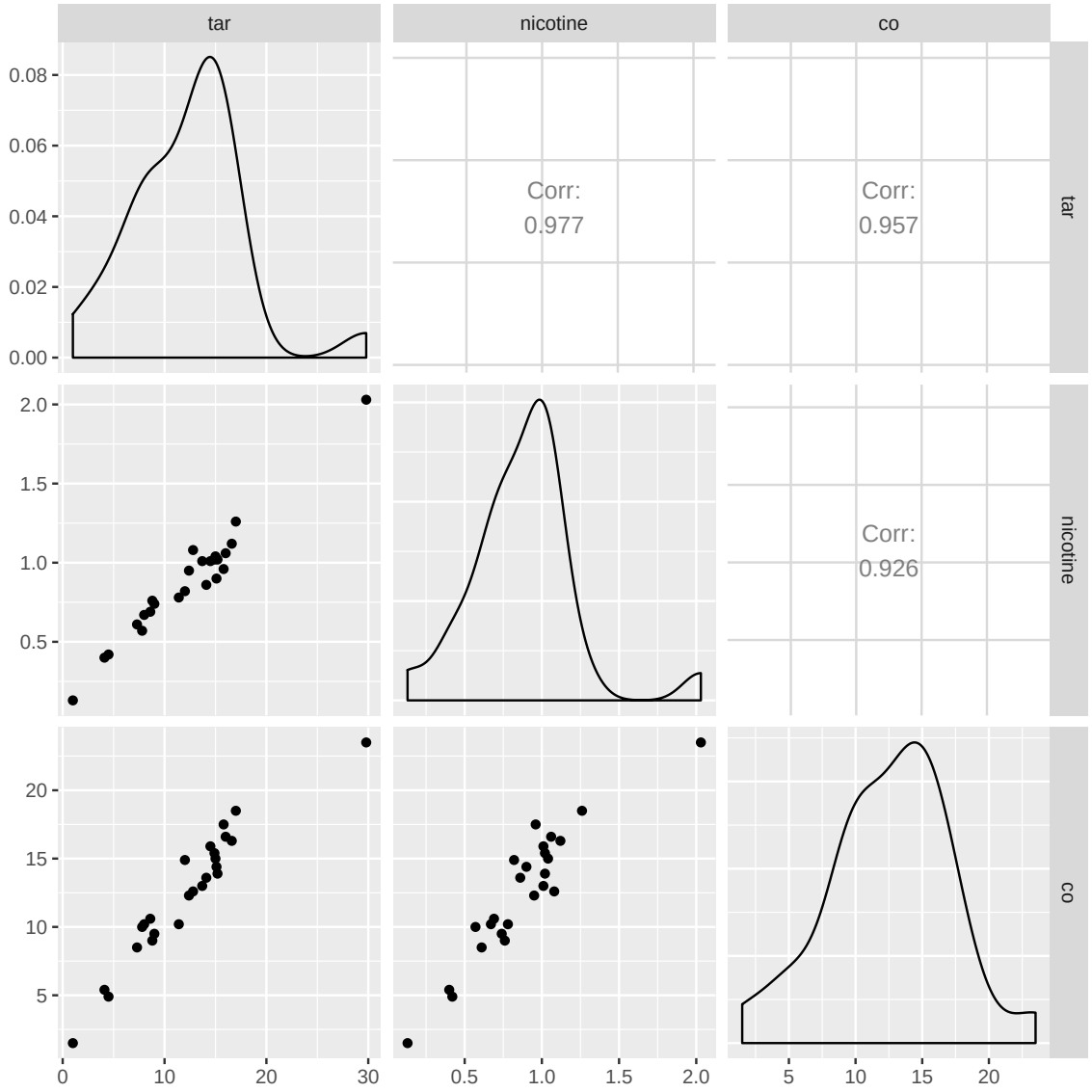


As ever, `install.packages` first, in the likely event that you don't have this package installed yet. Once you do, though, I think this is a nicer way to get a pairs plot.

This plot is a bit more sophisticated: instead of just having the scatterplots of the pairs of variables in the row and column, it uses the diagonal to show a "kernel density" (a smoothed-out histogram), and upper-right it shows the correlation between each pair of variables. The three correlations between `co`, `tar` and `nicotine` are clearly the highest.

If you want only some of the columns to appear in your pairs plot, `select` them first, and then pass that data frame into `ggpairs`. Here, we found that `weight` was not correlated with anything much, so we can take it out and then make a pairs plot of the other variables:

```
cigs %>% select(-weight) %>% ggpairs()
```



4. The United States is divided into a large number of counties: areas larger than a city but much smaller than a state. This question will work with a data set of the 440 largest counties, which can be found in `http://www.utsc.utoronto.ca/~butler/c32/smsa.txt`.

   The variables in the data set are:

   - an ID number of the county
   - the name of the county (text)
   - the state in which the county is located (text)
   - land area of the county (square miles)
   - total population

- Percent of population aged 18–34
- Percent of population aged 65 or older
- Number of active physicians
- Number of hospital beds
- Total number of serious crimes
- Percent high school graduates (percent of all adults aged 25 or older that completed grade 12)
- Percent of population with bachelor's degrees (B. Sc. or BA)
- Percent of population below poverty level
- Percent of labour force that is unemployed (labour force includes those who could be employed, and excludes university/college students, those serving in military, those who cannot work for health reasons).
- Per capita (mean) income of entire population
- Total personal income of entire population (millions of dollars)
- Region of the US (1=northeast, 2=north central, 3=south, 4=west)

Our aim in this question is to understand the factors affecting the number of active physicians (family doctors) in a county.

(a) Read the data into SAS, giving the variables suitable names. (You will have to read them all.)

Solution: This is tedious but not difficult. The hard part is to keep the variable names matched with the columns they represent. The data file starts right away with the data (no column headings):

```
filename myurl url "http://www.utsc.utoronto.ca/~butler/c32/smsa.txt";
proc import
  datafile=myurl
  dbms=dlm
  out=county
  replace;
  delimiter=' ';
  getnames=yes;
```

You ought to run this with `proc print` until you are happy that you have it right, *but if you were to hand in 440 lines of* `proc print` *output, you would deserve to lose as many marks as the grader decides to deduct.* Or more.

All the variables that are percentages had names starting with `pct`. This makes it easier to find them below.

(b) List the first 10 observations of your data set, and check that the columns that should be percentages actually look like percentages. Hint: to display a certain number of rows, *specify a data set name with* `data=` and put `obs=` and a number *in brackets* on the end of the line.

Solution: The hint suggests this (I was trying not to give it away completely). You have to specify a name for your data set; it doesn't work otherwise:

```
proc print data=county (obs=10);
```

Obs	id	name	state	area	pop
1	1	Los_Angeles	CA	4060	8863164
2	2	Cook	IL	946	5105067
3	3	Harris	TX	1729	2818199
4	4	San_Diego	CA	4205	2498016
5	5	Orange	CA	790	2410556
6	6	Kings	NY	71	2300664
7	7	Maricopa	AZ	9204	2122101
8	8	Wayne	MI	614	2111687
9	9	Dade	FL	1945	1937094
10	10	Dallas	TX	880	1852810

Obs	pct1834	pct65	physicians	beds	crimes
1	32.1	9.7	23677	27700	688936
2	29.2	12.4	15153	21550	436936
3	31.3	7.1	7553	12449	253526
4	33.5	10.9	5905	6179	173821
5	32.6	9.2	6062	6369	144524
6	28.3	12.4	4861	8942	680966
7	29.2	12.5	4320	6104	177593
8	27.4	12.5	3823	9490	193978
9	27.1	13.9	6274	8840	244725
10	32.6	8.2	4718	6934	214258

Obs	pcthighsch	pctbachelor	pctpverty	pctunemp	meaninc
1	70	22.3	11.6	8	20786
2	73.4	22.8	11.1	7.2	21729
3	74.9	25.4	12.5	5.7	19517
4	81.9	25.3	8.1	6.1	19588
5	81.2	27.8	5.2	4.8	24400
6	63.7	16.6	19.5	9.5	16803
7	81.5	22.1	8.8	4.9	18042
8	70	13.7	16.9	10	17461
9	65	18.8	14.2	8.7	17823
10	77.1	26.3	10.4	6.1	21001

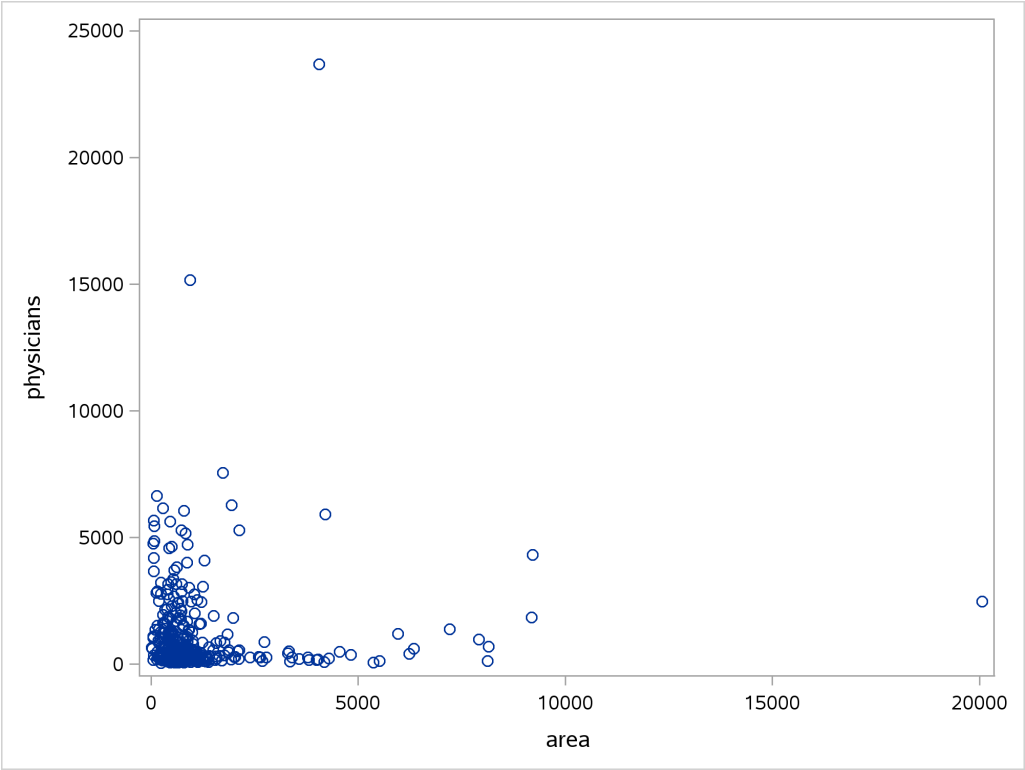
Obs	totalinc	region
1	184230	4
2	110928	2
3	55003	3
4	48931	4
5	58818	4
6	38658	1
7	38287	4
8	36872	2
9	34525	3
10	38911	3

This displays all the many variables for the first 10 observations. Now, because I named the "percent" variables beginning with `pct`, I can easily check that the percentage of people: aged 18–34, aged over 65, completing high school, with a bachelor's degree, in poverty and unemployed, look like percentages, and these are the only ones that do. (You should be checking six variables altogether.)

(c) We are going to predict the number of physicians in a county from some of the other variables. Start by obtaining a scatterplot of the number of physicians against the land area. What do you see, and what potential problems might this cause for a regression?

Solution:

```
proc sgplot;
  scatter x=area y=physicians;
```



Almost all the data points are at the bottom left of the picture, with only a few elsewhere. There are a few counties with very big land area (one especially big), and a few counties with a lot of physicians (not always the ones with large land area). As a result, the relationship is not at all clear.

One of the problems with regression is "influential points", observations that are very different from the others. We seem to have a few of them here. The problem with influential points is that they can (as their name implies) influence where the regression goes, even though there are only a few of them.

This is more discussion than you need, but I want you to observe two things: (i) that the majority of the points are bottom left (or that only a few are elsewhere), to answer "what do you see", and (ii) the points far away from the others can have a big influence over where the regression line goes, to answer "potential problems".

I guess this plot also shows a non-linear relationship, but that's not the best answer because the evidence for non-linearity is in those (relatively few) points off by themselves, not in the big mass of points bottom left, for which it's very unclear what kind of trend there is.

(d) One way to solve the problems unearthed in the previous part is to transform the variables that can be very large. Create a new data set with log-transformed number of physicians, land area and population.

Solution: This is `data` and `set`:

```
data county2;
  set county;
  logphys=log(physicians);
  logpop=log(pop);
  logarea=log(area);
```

If you like, print out the first few lines to check that the new values look sensible. Or you can summarize, eg. like this:

```
proc means;
  var physicians logphys pop logpop area logarea;
```

The MEANS Procedure

Variable	N	Mean	Std Dev	Minimum	Maximum
physicians	440	987.9977273	1789.75	39.0000000	23677.00
logphys	440	6.1517531	1.1440522	3.6635616	10.0722594
pop	440	393010.92	601987.02	100043.00	8863164.00
logpop	440	12.4759757	0.7903838	11.5133554	15.9974144
area	440	1041.41	1549.92	15.0000000	20062.00
logarea	440	6.5174458	0.8717066	2.7080502	9.9065828

The minimum and maximum of the logged variables should be the (natural) logs of the original values:

```
log(39)
```

```
## [1] 3.663562
```

```
log(23677)
```

```
## [1] 10.07226
```

```
log(100043)
```

```
## [1] 11.51336
```

```
log(8863164)
```

```
## [1] 15.99741
```

```
log(15)
```

```
## [1] 2.70805
```

```
log(20062)
```

```
## [1] 9.906583
```

That appears to check out.

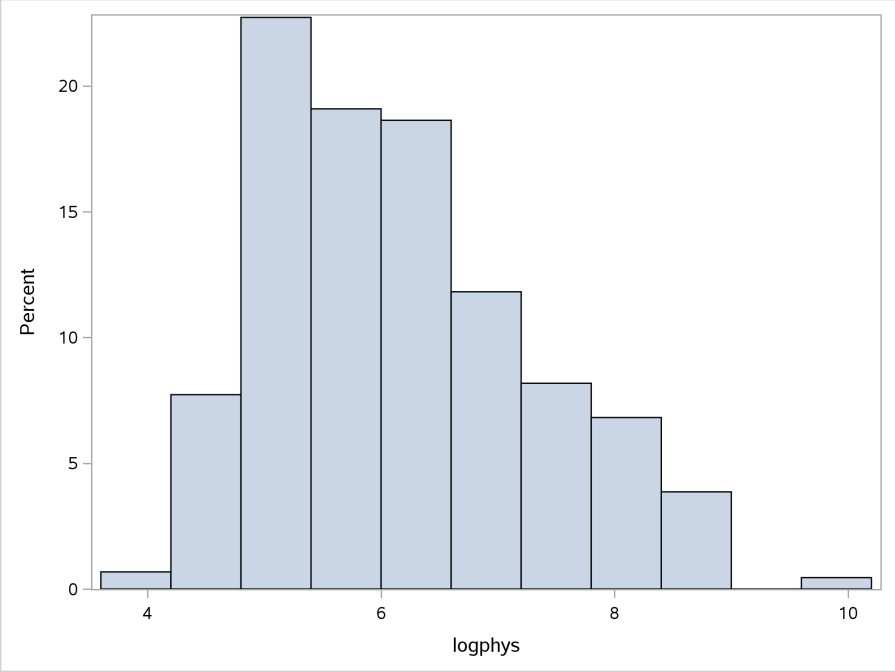
Note that taking logs has made the very big values not so very big. There is a county with over 8 million people in it! But the log of that is only about 16.

The log of the mean (population, say) is not the same as the mean of the log-population. You might like to think about why not.

(e) Draw histograms of your three new variables. Do they have something like normal distributions?

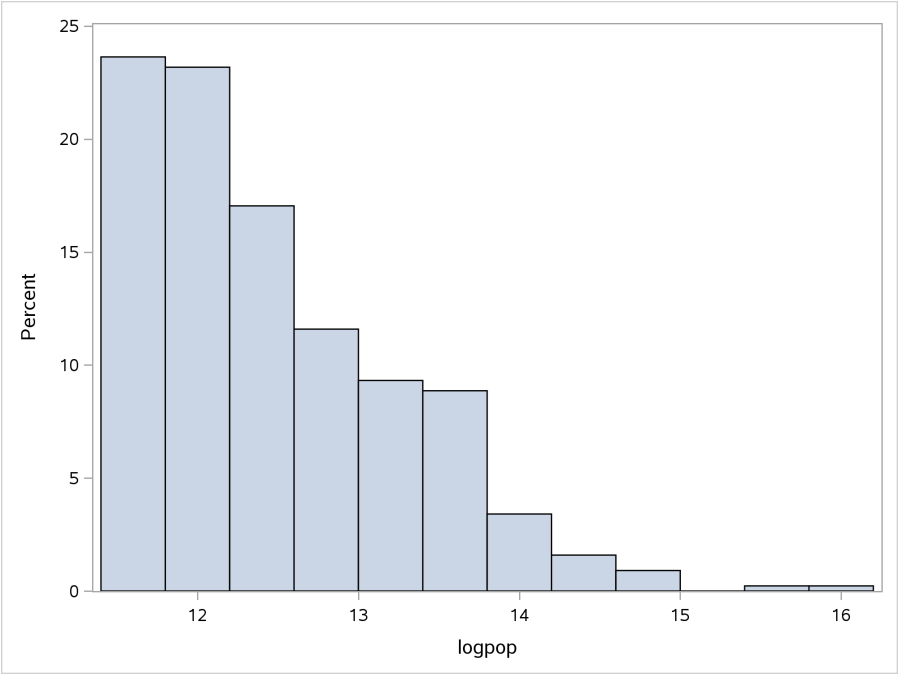
Solution: The obvious thing is to draw the histograms one at a time, copying and pasting your code. Log-physicians:

```
proc sgplot;
  histogram logphys;
```



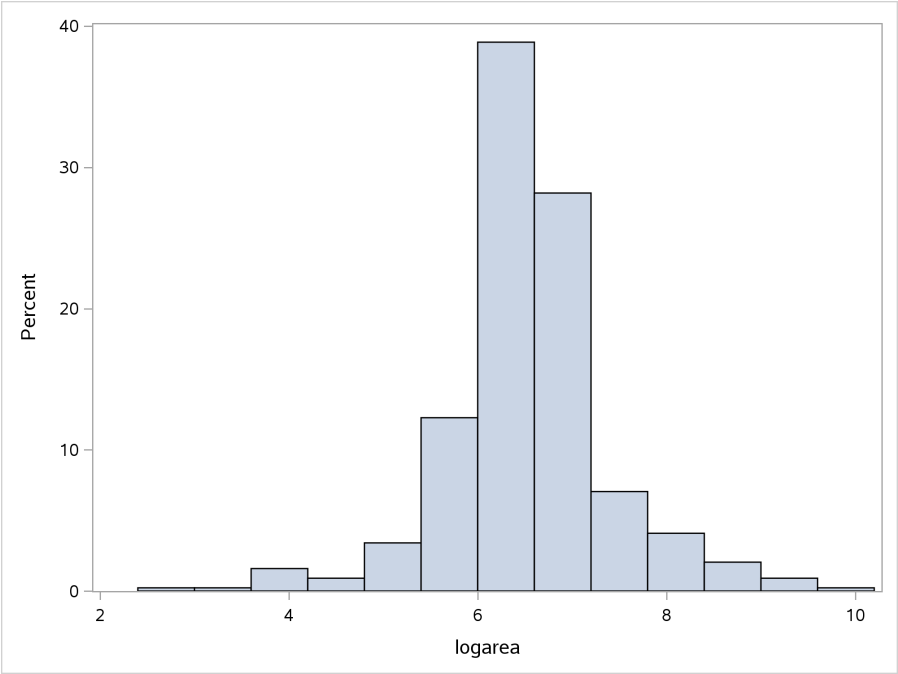
Log-population:

```
proc sgplot;
  histogram logpop;
```



Log-area:

```
proc sgplot;
  histogram logarea;
```



Log-area is nice and symmetric. Log-population is still a bit skewed, and log-physicians is a bit skewed with an outlier. But if you compare the histograms of the original variables, things are a lot better than they were.

I wanted to say something about normal distributions and regression at this point, since that often seems misunderstood. What you actually *need* is for the "errors" to be normally distributed, and since you never actually observe the errors themselves, you look at the residuals, and if they are approximately normal, with no patterns in relation to anything else, you are good. There is *no* need for the y values or the x values to be normally distributed; in fact, the theory of regression says only that the x's are *given* (not random at all), or, if you prefer, you work *conditional* on the x's you observed.

A little bit of the theory, for those who care: you assume that the model (one x) is $y_i = \alpha + \beta x_i + e_i$, where the errors e_i are the only random thing, and they have independent normal distributions with mean 0 and variance σ^2. The x_i are fixed, and the intercept α and slope β are constant parameters to be estimated (which is done by maximum likelihood or least squares). Another way to look at this, because of properties of the normal distribution, is that the y_i have independent normal distributions with mean $\alpha + \beta x_i$ and constant variance σ^2. (If you didn't have a normal distribution, this wouldn't work.) I actually like this way better, because it transfers over to generalized linear models, which you might see later.

As I said, you never actually observe the e_i; the best you can do is *estimate* them, using the residuals. The independence of the errors plays out in the need for randomness in any graphs involving residuals; the normality of the errors plays out in wanting the normal quantile plot of the residuals to be straight, and the constant variance plays out in wanting no fanning-out.

Having said all of that, if the distribution of your x's has outliers, so (probably) will the distribution of your y's, and then you will be dealing with influential points. It is not *necessary* for the distribution of your x's to be even approximately normal, but it generally makes your life easier if it is.

So that's why I had you do the transformations and look at the histograms afterwards.

(f) Do a regression predicting the log-number of physicians from the log-population and log-area. Display and comment on the results (the printed part, not the graphs, yet).

Solution: Nothing terribly surprising in the code. I forgot that you separate explanatory variables in SAS by a space, not a plus, so I had to do it twice:

```
proc reg;
  model logphys=logpop logarea;
```

```
                          The REG Procedure
                            Model: MODEL1
                      Dependent Variable: logphys

                 Number of Observations Read         440
                 Number of Observations Used         440
                           Analysis of Variance

                                    Sum of           Mean
Source                   DF        Squares         Square    F Value    Pr > F

Model                     2      472.58876      236.29438    1012.37    <.0001
Error                   437      101.99878        0.23341
Corrected Total         439      574.58754

            Root MSE              0.48312    R-Square     0.8225
            Dependent Mean        6.15175    Adj R-Sq     0.8217
            Coeff Var             7.85340
```

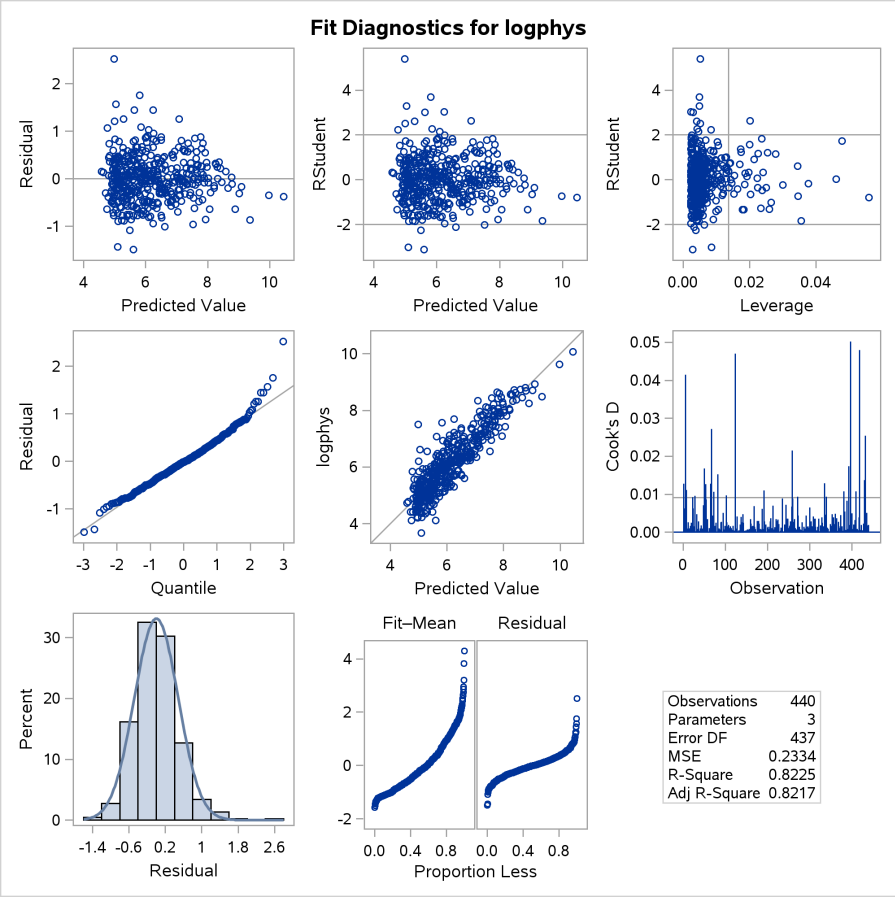
Parameter Estimates

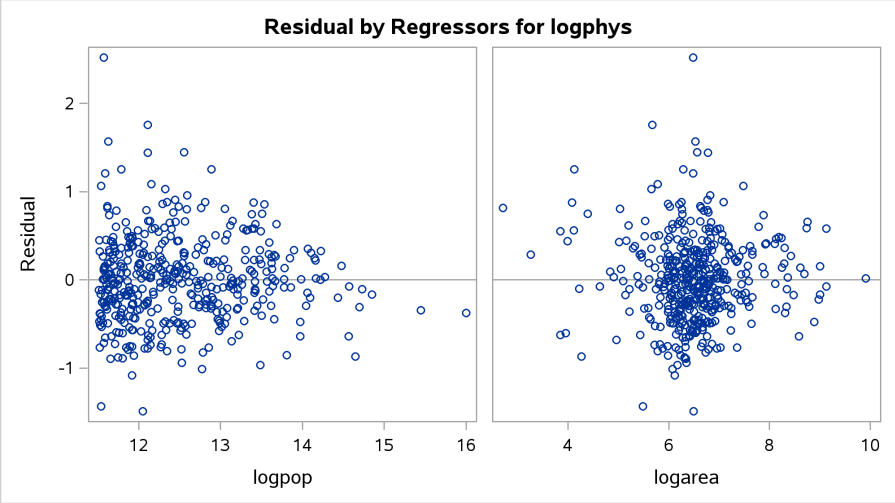
Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|
Intercept	1	-9.04884	0.39932	-22.66	<.0001
logpop	1	1.30489	0.02918	44.71	<.0001
logarea	1	-0.16557	0.02646	-6.26	<.0001

A nice high R-squared (for this kind of thing) of 82.25%. Both explanatory variables are strongly significant. Log-population has a positive slope and log-area has a negative one. That means that counties with a higher population have more physicians (no surprise there!). Counties with a larger area have fewer physicians, even after accounting for population. That is to say, you can't just say that larger counties are likely to be more sparsely populated and *that's* the reason they have fewer doctors. I think you have to say something along the lines of cities having to be big enough to support having a physician, and a county with large area might have a decent-sized population but not very many cities of any size and therefore not many places where it is profitable for a doctor to be. Something like that.

(g) Check the residual plots for the regression you just did. Do you see anything unacceptable?

Solution: Here is that array of graphs:





Residuals vs. fitted values top left looks pretty much like a random cloud (a couple of outliers); normal quantile plot looks pretty straight, with maybe a few outliers at the top; residuals against log-area (on the right) a nice random scatter; residuals vs. log-population has some fanning in. This might be because even the distribution of log-population was still skewed and we ought to have gone further in our transformation (something like reciprocal, maybe, instead of log).

I might think about some other way of transforming population, but overall I think this is not too bad.

(h) The regression you just did predicts log-physicians from log-population and log-area. Do a little algebra to get a relationship predicting the actual number of physicians from (functions of) the other variables. Simplify your result as far as you can.

Solution: I can't remember whether I promised "no math" or "very little math" at the start of the course, but anyway. Let's define some symbols to make our lives easier: let d be the number of physicians ("d" for "doctor"), p be the population and a the area of a county. Then our regression says (rounding things off a bit):

$$\log d = -9.05 + 1.30 \log p - 0.17 \log a$$

Take e-to-the-power-of both sides, which I'll write exp:

$$d = \exp(-9.05 + 1.30 \log p - 0.17 \log a)$$

Adding things inside exp means multiplying the separate exp's:

$$d = \exp(-9.05) \exp(1.30 \log p) \exp(-0.17 \log a)$$

A piece of math: $\exp(a \log x) = \{\exp(\log x)\}^a = x^a$:

$$d = e^{-9.05} p^{1.30} a^{-0.17}$$

and you can work out the first exp if you like (it's a very small number).

This is a multiplicative model: the contribution of increasing population is to *multiply* predicted number of physicians by something. You can even work out what: if you multiply the population by 2, leaving everything else fixed, you get this:

$$\begin{aligned}\frac{d(2p)}{d(p)} &= \frac{e^{-9.05}(2p)^{1.30}a^{-0.17}}{e^{-9.05}p^{1.30}a^{-0.17}} \\ &= 2^{1.3} = 2.46\end{aligned}$$

since almost everything cancels: that is, doubling the population slightly more than doubles the number of physicians, if the area of a county is held constant. Doubling the area while holding the population constant, by the same logic, changes the number of physicians by a factor of $2^{-0.17} = 0.89$; that is, making it about 90% of what it was before.

Clearly there is an effect of population density at work here.

(i) To satisfy the curiosity that you probably have, find the ten largest counties by population and then list them. Where do you think the second-largest county is?

Solution: This is actually the same strategy that you would use in R, but implemented differently: make a new data set that is the old one sorted (in descending order) by population, and then display its first ten lines. SAS has a `proc sort` that does this:

```
proc sort;
  by descending pop;

proc print data=county2 (obs=10);
  var name state area pop;
```

Obs	name	state	area	pop
1	Los_Angeles	CA	4060	8863164
2	Cook	IL	946	5105067
3	Harris	TX	1729	2818199
4	San_Diego	CA	4205	2498016
5	Orange	CA	790	2410556
6	Kings	NY	71	2300664
7	Maricopa	AZ	9204	2122101
8	Wayne	MI	614	2111687
9	Dade	FL	1945	1937094
10	Dallas	TX	880	1852810

What `proc sort` does is to sort the data set by the variable(s) requested and *save it back* in a data set of the same name. That's why my `proc print` worked. (If you don't like that, you put an `out=` on the `proc sort` line with a new data set name.)

All of Los Angeles is in one county, which makes it the biggest one in the entire country. You might not know where Cook County is, but it's in Illinois, and the biggest city in Illinois is Chicago, so you might guess that it includes Chicago. If you look it up on Google Maps, you'll see that you were exactly right.[3]

5. A physiologist wanted to understand the relationship between physical characteristics of pre-adolescent

boys and their maximal oxygen uptake (millilitres of oxygen per kilogram of body weight). The data are in `http://www.utsc.utoronto.ca/~butler/c32/youngboys.txt` for a random sample of 10 pre-adolescent boys. The variables are (with units):

- `uptake`: Oxygen uptake (millitres of oxygen per kilogram of body weight)
- `age`: boy's age (years)
- `height`: boy's height (cm)
- `weight`: boy's weight (kg)
- `chest`: chest depth (cm).

(a) Read the data into R and confirm that you do indeed have 10 observations.

Solution:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/youngboys.txt"
boys=read_delim(my_url," ")
```

```
## Parsed with column specification:
## cols(
##  uptake = col_double(),
##  age = col_double(),
##  height = col_double(),
##  weight = col_double(),
##  chest = col_double()
## )
```

```
glimpse(boys)
```

```
## Observations: 10
## Variables: 5
## $ uptake <dbl> 1.54, 1.74, 1.32, 1.50, 1.46, 1.35, 1.53, 1.71, 1.27, 1.50
## $ age    <dbl> 8.4, 8.7, 8.9, 9.9, 9.0, 7.7, 7.3, 9.9, 9.3, 8.1
## $ height <dbl> 132.0, 135.5, 127.7, 131.1, 130.0, 127.6, 129.9, 138.1,...
## $ weight <dbl> 29.1, 29.7, 28.4, 28.8, 25.9, 27.6, 29.0, 33.6, 27.7, 30.8
## $ chest  <dbl> 14.4, 14.5, 14.0, 14.2, 13.6, 13.9, 14.0, 14.6, 13.9, 14.5
```

This is the best way, though the data set is small enough that you could reasonably list all the values:

```
boys
```

```
## # A tibble: 10 x 5
##    uptake   age height weight chest
##     <dbl> <dbl>  <dbl>  <dbl> <dbl>
##  1   1.54   8.4  132.0   29.1  14.4
##  2   1.74   8.7  135.5   29.7  14.5
##  3   1.32   8.9  127.7   28.4  14.0
##  4   1.50   9.9  131.1   28.8  14.2
##  5   1.46   9.0  130.0   25.9  13.6
##  6   1.35   7.7  127.6   27.6  13.9
##  7   1.53   7.3  129.9   29.0  14.0
##  8   1.71   9.9  138.1   33.6  14.6
##  9   1.27   9.3  126.6   27.7  13.9
## 10   1.50   8.1  131.8   30.8  14.5
```

(b) Fit a regression predicting oxygen uptake from all the other variables, and display the results.

Solution:

```
boys.1=lm(uptake~age+height+weight+chest,data=boys)
summary(boys.1)
```

```
##
## Call:
## lm(formula = uptake ~ age + height + weight + chest, data = boys)
##
## Residuals:
##         1         2         3         4         5         6         7
## -0.020697  0.019741 -0.003649  0.038470 -0.023639 -0.026026  0.050459
##         8         9        10
## -0.014380  0.004294 -0.024573
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4.774739   0.862818  -5.534 0.002643 **
## age         -0.035214   0.015386  -2.289 0.070769 .
## height       0.051637   0.006215   8.308 0.000413 ***
## weight      -0.023417   0.013428  -1.744 0.141640
## chest        0.034489   0.085239   0.405 0.702490
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.03721 on 5 degrees of freedom
## Multiple R-squared:  0.9675,Adjusted R-squared:  0.9415
## F-statistic:  37.2 on 4 and 5 DF,  p-value: 0.0006513
```

(c) (A one-mark question.) Would you say, on the evidence so far, that the regression fits well or badly? Explain (very) briefly.

Solution: R-squared of 0.97 (97%) is very high, so I'd say this regression fits very well. That's all.

I said "on the evidence so far" to dissuade you from overthinking this, or thinking that you needed to produce some more evidence. That, plus the fact that this was only one mark.

(d) It seems reasonable that an older boy should have a greater oxygen uptake, all else being equal. Is this supported by your output? Explain briefly.

Solution: If an older boy has greater oxygen uptake (the "all else equal" was a hint), the slope of `age` should be positive. It is not: it is -0.035, so it is suggesting (all else equal) that a greater age goes with a *smaller* oxygen uptake.

The reason why this happens (which you didn't need, but you can include it if you like) is that `age` has a non-small P-value of 0.07, so that the `age` slope is not significantly different from zero. With all the other variables, `age` has nothing to *add* over and above them, and we could therefore remove it.

(e) It seems reasonable that a boy with larger weight should have larger lungs and thus a *statistically significantly* larger oxygen uptake. Is that what happens here? Explain briefly.

Solution: Look at the P-value for `weight`. This is 0.14, not small, and so a boy with larger weight does not have a significantly larger oxygen uptake, all else equal. (The slope for `weight` is not significantly different from zero either.)

I emphasized "statistically significant" to remind you that this means to do a test and get a P-value.

(f) Fit a model that contains only the significant explanatory variables from your first regression. How do the R-squared values from the two regressions compare? (The last sentence asks for more or less the same thing as the next part. Answer it either here or there. Either place is good.)

Solution: Only `height` is significant, so that's the only explanatory variable we need to keep. I would just do the regression straight rather than using `update` here:

```
boys.2=lm(uptake~height,data=boys)
summary(boys.2)
```

```
##
## Call:
## lm(formula = uptake ~ height, data = boys)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.069879 -0.033144  0.001407  0.009581  0.084012
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.843326   0.609198  -6.309 0.000231 ***
## height       0.040718   0.004648   8.761 2.26e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05013 on 8 degrees of freedom
## Multiple R-squared:  0.9056,Adjusted R-squared:  0.8938
## F-statistic: 76.75 on 1 and 8 DF,  p-value: 2.258e-05
```

If you want, you can use `update` here, which looks like this:

```
boys.2a=update(boys.1,.~.-age-weight-chest)
summary(boys.2a)
```

```
##
## Call:
## lm(formula = uptake ~ height, data = boys)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.069879 -0.033144  0.001407  0.009581  0.084012
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.843326   0.609198  -6.309 0.000231 ***
## height       0.040718   0.004648   8.761 2.26e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05013 on 8 degrees of freedom
## Multiple R-squared:  0.9056,Adjusted R-squared:  0.8938
## F-statistic: 76.75 on 1 and 8 DF,  p-value: 2.258e-05
```

This doesn't go quite so smoothly here because there are three variables being removed, and it's a bit of work to type them all.

(g) How has R-squared changed between your two regressions? Describe what you see in a few words.

Solution: R-squared has dropped by a bit, from 97% to 91%. (Make your own call: pull out the two R-squared numbers, and say a word or two about how they compare. I don't much mind what you say: "R-squared has decreased (noticeably)", "R-squared has hardly changed". But say something.)

(h) Carry out a test comparing the fit of your two regression models. What do you conclude, and therefore what recommendation would you make about the regression that would be preferred?

Solution: The word "test" again implies something that produces a P-value with a null hypothesis that you might reject. In this case, the test that compares two models differing by more than one x uses `anova`, testing the null hypothesis that the two regressions are equally good, against the alternative that the bigger (first) one is better. Feed `anova` two fitted model objects, smaller first:

```
anova(boys.2, boys.1)
```

```
## Analysis of Variance Table
##
## Model 1: uptake ~ height
## Model 2: uptake ~ age + height + weight + chest
##   Res.Df       RSS Df Sum of Sq      F Pr(>F)
## 1      8 0.0201016
## 2      5 0.0069226  3  0.013179 3.1729  0.123
```

This P-value of 0.123 is not small, so we do not reject the null hypothesis. There is not a significant difference in fit between the two models. Therefore, we should go with the smaller model `boys.2` because

it is simpler.

That drop in R-squared from 97% to 91% was, it turns out, *not* significant: the three extra variables could have produced a change in R-squared like that, *even if they were worthless.* (Recall that adding x's to a regression will always make R-squared go up, even if they are just random noise.)

If you have learned about "adjusted R-squared", you might recall that this is supposed to go down *only* if the variables you took out should not have been taken out. But adjusted R-squared goes down here as well, from 94% to 89% (not quite as much, therefore). What happens is that adjusted R-squared is rather more relaxed about keeping variables than the `anova` F-test is; if we had used an α of something like 0.10, the decision between the two models would have been a lot closer, and this is reflected in the adjusted R-squared values.

(i) Obtain a table of correlations between all the variables in the data frame. Do this by feeding the whole data frame into `cor`. We found that a regression predicting oxygen uptake from just `height` was acceptably good. What does your table of correlations say about why that is? (Hint: look for all the correlations that are *large*.)

Solution: Correlations first:

```
cor(boys)
```

```
##           uptake       age    height    weight     chest
## uptake 1.0000000 0.1361907 0.9516347 0.6576883 0.7182659
## age    0.1361907 1.0000000 0.3274830 0.2307403 0.1657523
## height 0.9516347 0.3274830 1.0000000 0.7898252 0.7909452
## weight 0.6576883 0.2307403 0.7898252 1.0000000 0.8809605
## chest  0.7182659 0.1657523 0.7909452 0.8809605 1.0000000
```

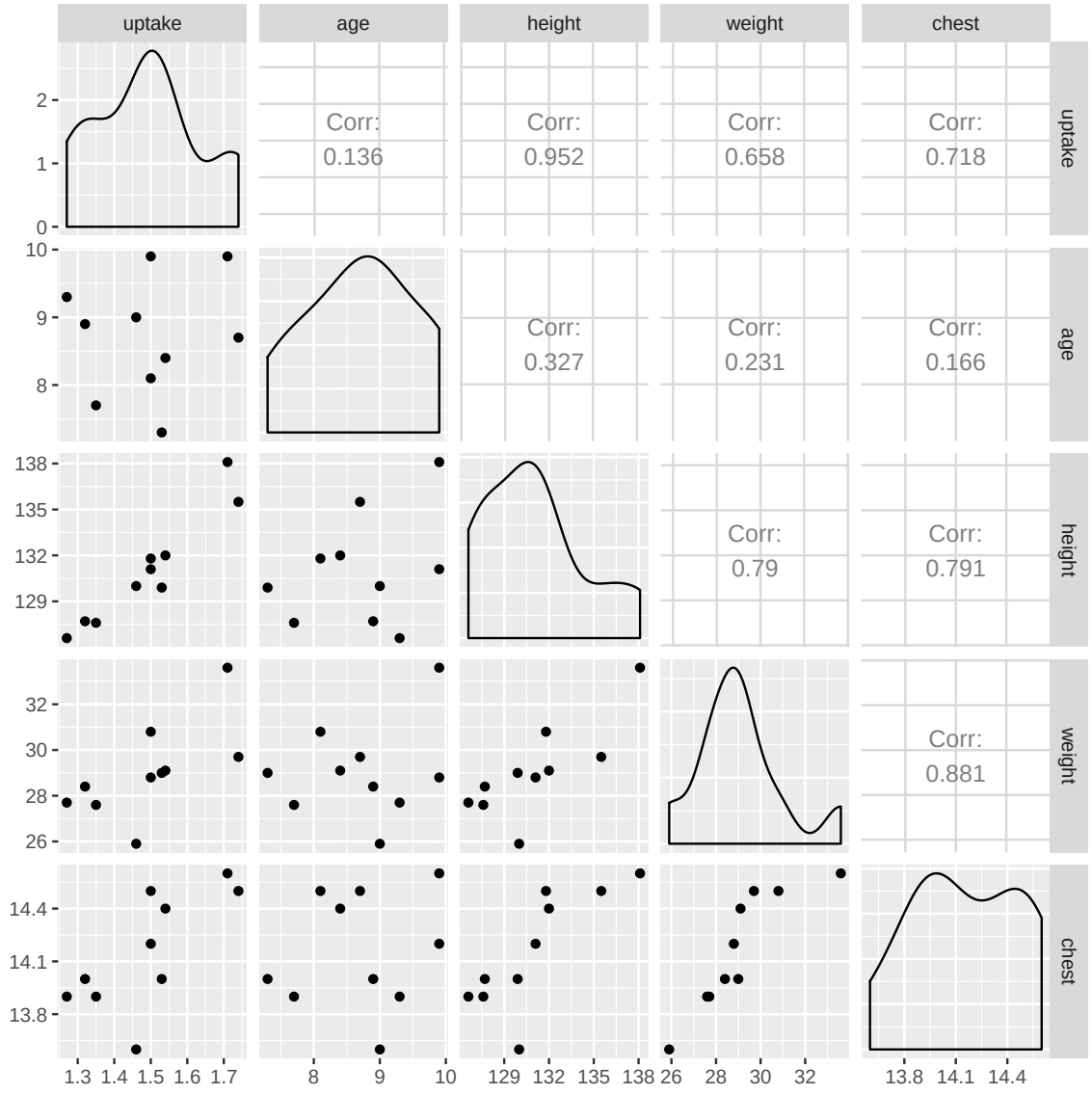
The correlations with `age` are all on the low side, but all the other correlations are high, not just between `uptake` and the other variables, but between the explanatory variables as well.

Why is this helpful in understanding what's going on? Well, imagine a boy with large height (a tall one). The regression `boys.2` says that this alone is enough to predict that such a boy's oxygen uptake is likely to be large, since the slope is positive. But the correlations tell you more: a boy with large height is also (somewhat) likely to be older (have large age), heavier (large weight) and to have larger `chest` cavity. So oxygen uptake does depend on those other variables as well, but once you know `height` you can make a good guess at their values; you don't need to know them.

Further remarks: `age` has a low correlation with `uptake`, so its non-significance earlier appears to be "real": it really does have nothing extra to say, because the other variables have a stronger link with `uptake` than `age`. Height, however, seems to be the best way of relating oxygen uptake to any of the other variables. I think the suppositions from earlier about relating oxygen uptake to "bigness"[4] in some sense are actually sound, but age and weight and `chest` capture "bigness" worse than height does. Later, when you learn about Principal Components, you will see that the first principal component, the one that best captures how the variables vary together, is often "bigness" in some sense.

Another way to think about these things is via pairwise scatterplots. The nicest way to produce these is via `ggpairs` from package `ggally`:

```
boys %>% ggpairs()
```



A final remark: with five variables, we really ought to have more than ten observations (something like 50 would be better). But with more observations and the same correlation structure, the same issues would come up again, so the question would not be materially changed.

6. Male tree crickets produce "mating songs" by rubbing their wings together to produce a chirping sound. It is hypothesized that female tree crickets identify males of the correct species by how fast (in chirps per second) the male's mating song is. This is called the "pulse rate". Some data for two species of crickets are in `http://www.utsc.utoronto.ca/~butler/c32/crickets.txt`. The columns, which are unlabelled, are temperature and pulse rate (respectively) for *Oecanthus exclamationis* (first two columns) and *Oecanthus niveus* (third and fourth columns). The columns are separated by tabs. There are some missing values in the first two columns because fewer *exclamationis* crickets than *niveus* crickets were measured.

The research question is whether males of the different species have different average pulse rates. It is also of interest to see whether temperature has an effect, and if so, what.

(a) Read in the data, allowing for the fact that you have no column names. You'll see that the columns have names `X1` through `X4`. This is OK.

Solution: Tab-separated, so `read_tsv`; no column names, so `col_names=F`:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/crickets.txt"
crickets=read_tsv(my_url,col_names=F)
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   X2 = col_double(),
##   X3 = col_double(),
##   X4 = col_double()
## )
```

```
crickets
```

```
## # A tibble: 17 x 4
##       X1    X2    X3    X4
##    <dbl> <dbl> <dbl> <dbl>
##  1  20.8  67.9  17.2  44.3
##  2  20.8  65.1  18.3  47.2
##  3  24.0  77.3  18.3  47.6
##  4  24.0  78.7  18.3  49.6
##  5  24.0  79.4  18.9  50.3
##  6  24.0  80.4  18.9  51.8
##  7  26.2  85.8  20.4  60.0
##  8  26.2  86.6  21.0  58.5
##  9  26.2  87.5  21.0  58.9
## 10  26.2  89.1  22.1  60.7
## 11  28.4  98.6  23.5  69.8
## 12  29.0 100.8  24.2  70.9
## 13  30.4  99.3  25.9  76.2
## 14  30.4 101.7  26.5  76.1
## 15    NA    NA  26.5  77.0
## 16    NA    NA  26.5  77.7
## 17    NA    NA  28.6  84.7
```

As promised.

If you didn't catch the tab-separated part, this probably happened to you:

```
d=read_delim(my_url," ",col_names=F)
```

```
## Parsed with column specification:
## cols(
##  X1 = col_character()
## )
```

```
## Warning in rbind(names(probs), probs_f): number of columns of result is not a multiple
of vector length (arg 1)
```

```
## Warning: 3 parsing failures.
## row # A tibble: 3 x 5 col    row  col  expected    actual expected  <int> <chr>
<chr>    <chr> actual 1    15  <NA> 1 columns 2 columns file 2    16  <NA> 1 columns
2 columns row 3    17  <NA> 1 columns 2 columns col # ... with 1 more variables: file
<chr>
```

This doesn't look good:

```
problems(d)
```

```
## # A tibble: 3 x 5
##     row   col  expected    actual
##   <int> <chr>     <chr>     <chr>
## 1    15  <NA> 1 columns 2 columns
## 2    16  <NA> 1 columns 2 columns
## 3    17  <NA> 1 columns 2 columns
## # ... with 1 more variables: file <chr>
```

The "expected columns" being 1 should bother you, since we know there are supposed to be 4 columns. At this point, we take a look at what got read in:

```
d
```

```
## # A tibble: 17 x 1
##                            X1
##                         <chr>
##  1  "20.8\t67.9\t17.2\t44.3"
##  2  "20.8\t65.1\t18.3\t47.2"
##  3  "24.0\t77.3\t18.3\t47.6"
##  4  "24.0\t78.7\t18.3\t49.6"
##  5  "24.0\t79.4\t18.9\t50.3"
##  6  "24.0\t80.4\t18.9\t51.8"
##  7  "26.2\t85.8\t20.4\t60.0"
##  8  "26.2\t86.6\t21.0\t58.5"
##  9  "26.2\t87.5\t21.0\t58.9"
## 10  "26.2\t89.1\t22.1\t60.7"
## 11  "28.4\t98.6\t23.5\t69.8"
## 12 "29.0\t100.8\t24.2\t70.9"
## 13  "30.4\t99.3\t25.9\t76.2"
## 14 "30.4\t101.7\t26.5\t76.1"
## 15                 "NA\tNA"
## 16                 "NA\tNA"
## 17                 "NA\tNA"
```

and there you see the `\t` or "tab" characters separating the values, instead of spaces. (This is what I

tried first, and once I looked at this, I realized that `read_tsv` was what I needed.)

(b) These data are rather far from being tidy. There need to be three variables, temperature, pulse rate and species, and there are $14 + 17 = 31$ observations altogether. This one is tricky in that there are temperature and pulse rate for each of two levels of a factor, so I'll suggest combining the temperature and chirp rate together into one thing for each species, then gathering them, then splitting them again. Create new columns, named for each species, that contain the temperature and pulse rate for that species in that order, `unite`d together.

For the rest of this question, start from the data frame you read in, and build a pipe, one or two steps at a time, to save creating a lot of temporary data frames.

Solution: Breathe, and then begin. `unite` creates new columns by joining together old ones (as `paste` does, actually):

```
crickets %>%
  unite(exclamationis,X1:X2) %>%
  unite(niveus,X3:X4)
```

```
## # A tibble: 17 x 2
##    exclamationis    niveus
##  *         <chr>     <chr>
##  1     20.8_67.9 17.2_44.3
##  2     20.8_65.1 18.3_47.2
##  3       24_77.3 18.3_47.6
##  4       24_78.7 18.3_49.6
##  5       24_79.4 18.9_50.3
##  6       24_80.4 18.9_51.8
##  7     26.2_85.8   20.4_60
##  8     26.2_86.6   21_58.5
##  9     26.2_87.5   21_58.9
## 10     26.2_89.1 22.1_60.7
## 11     28.4_98.6 23.5_69.8
## 12      29_100.8 24.2_70.9
## 13     30.4_99.3 25.9_76.2
## 14    30.4_101.7 26.5_76.1
## 15         NA_NA   26.5_77
## 16         NA_NA 26.5_77.7
## 17         NA_NA 28.6_84.7
```

Note that the original columns `X1:X4` are *gone*, which is fine, because the information we needed from them is contained in the two new columns. `unite` by default uses an underscore to separate the joined-together values, which is generally safe since you won't often find those in data.

Digression: `unite`-ing with a space could cause problems if the data values have spaces in them already. Consider this list of names:

```
names=c("Cameron McDonald","Durwin Yang","Ole Gunnar Solskjaer","Mahmudullah")
```

Two very former students of mine, a Norwegian soccer player, and a Bangladeshi cricketer. Only one of these has played for Manchester United:

```
manu=c(F,F,T,F)
```

and let's make a data frame:

```
d=tibble(name=names,manu=manu)
d
```

```
## # A tibble: 4 x 2
##                    name  manu
##                   <chr> <lgl>
## 1     Cameron McDonald FALSE
## 2          Durwin Yang FALSE
## 3 Ole Gunnar Solskjaer  TRUE
## 4          Mahmudullah FALSE
```

Now, what happens if we `unite` those columns, separating them by a space?

```
d %>% unite(joined,name:manu,sep=" ")
```

```
## # A tibble: 4 x 1
##                       joined
## *                      <chr>
## 1    Cameron McDonald FALSE
## 2         Durwin Yang FALSE
## 3 Ole Gunnar Solskjaer TRUE
## 4         Mahmudullah FALSE
```

If we then try to separate them again, what happens?

```
d %>% unite(joined,name:manu,sep=" ") %>%
  separate(joined,c("one","two")," ")
```

```
## Warning:  Too many values at 3 locations:  1, 2, 3
```

```
## # A tibble: 4 x 2
##           one      two
## *       <chr>    <chr>
## 1     Cameron McDonald
## 2      Durwin     Yang
## 3         Ole   Gunnar
## 4 Mahmudullah    FALSE
```

Things have gotten lost: most of the original values of `manu` and some of the names. If we use a different separator character, either choosing one deliberately or going with the default underscore, everything works swimmingly:

```
d %>% unite(joined,name:manu,sep=":") %>%
  separate(joined,c("one","two"),":")
```

```
## # A tibble: 4 x 2
##                    one   two
## *                  <chr> <chr>
## 1     Cameron McDonald FALSE
## 2          Durwin Yang FALSE
## 3 Ole Gunnar Solskjaer  TRUE
## 4          Mahmudullah FALSE
```

and we are back to where we started.

If you run just the `unite` line (move the pipe symbol to the next line so that the `unite` line is complete as it stands), you'll see what happened.

(c) The two columns `exclamationis` and `niveus` that you just created are both temperature-pulse rate combos, but for different species. `gather` them together into one column, labelled by species. (This is a straight `tidyr gather`, even though they contain something odd-looking.)

Solution: Thus, this, naming the new column `temp_pulse` since it contains both of those things. Add to the end of the pipe you started building in the previous part:

```
crickets %>%
  unite(exclamationis,X1:X2) %>%
  unite(niveus,X3:X4) %>%
  gather(species,temp_pulse,exclamationis:niveus)
```

```
## # A tibble: 34 x 2
##          species temp_pulse
##            <chr>      <chr>
##  1 exclamationis  20.8_67.9
##  2 exclamationis  20.8_65.1
##  3 exclamationis    24_77.3
##  4 exclamationis    24_78.7
##  5 exclamationis    24_79.4
##  6 exclamationis    24_80.4
##  7 exclamationis  26.2_85.8
##  8 exclamationis  26.2_86.6
##  9 exclamationis  26.2_87.5
## 10 exclamationis  26.2_89.1
## # ... with 24 more rows
```

Yep.

This is going to get rather long, but don't fret: we debugged the two `unite` lines before, so if you get any errors, they must have come from the `gather`. So that would be the place to check.

(d) Now split up the temperature-pulse combos at the underscore, into two separate columns. This is `separate`. When specifying what to separate by, you can use a number ("split after this many characters") or a piece of text, in quotes ("when you see this text, split at it").

Solution: The text to split by is an underscore (in quotes), since `unite` by default puts an underscore in between the values it pastes together. Glue the `separate` onto the end. We are creating two new variables `temperature` and `pulse_rate`:

```
crickets %>%
  unite(exclamationis,X1:X2) %>%
  unite(niveus,X3:X4) %>%
  gather(species,temp_pulse,exclamationis:niveus) %>%
  separate(temp_pulse,c("temperature","pulse_rate"),"_")
```

```
## # A tibble: 34 x 3
##          species temperature pulse_rate
##  *         <chr>       <chr>      <chr>
##  1 exclamationis        20.8       67.9
##  2 exclamationis        20.8       65.1
##  3 exclamationis          24       77.3
##  4 exclamationis          24       78.7
##  5 exclamationis          24       79.4
##  6 exclamationis          24       80.4
##  7 exclamationis        26.2       85.8
##  8 exclamationis        26.2       86.6
##  9 exclamationis        26.2       87.5
## 10 exclamationis        26.2       89.1
## # ... with 24 more rows
```

(e) Almost there. Temperature and pulse rate are still text (because `unite` turned them into text), but they should be numbers. Create new variables that are numerical versions of temperature and pulse rate (using `as.numeric`). Check that you have no extraneous variables (and, if necessary, get rid of the ones you don't want). (Species is also text and really ought to be a factor, but having it as text doesn't seem to cause any problems.)

Solution: `mutate`-ing into a column that already exists overwrites the variable that's already there (which saves us some effort here).

```
crickets.1 = crickets %>%
  unite(exclamationis,X1:X2) %>%
  unite(niveus,X3:X4) %>%
  gather(species,temp_pulse,exclamationis:niveus) %>%
  separate(temp_pulse,c("temperature","pulse_rate"),"_") %>%
  mutate(temperature=as.numeric(temperature)) %>%
  mutate(pulse_rate=as.numeric(pulse_rate))
```

```
## Warning in eval(substitute(expr), envir, enclos):  NAs introduced by coercion
```

```
## Warning in eval(substitute(expr), envir, enclos):  NAs introduced by coercion
```

```
crickets.1
```

```
## # A tibble: 34 x 3
##          species temperature pulse_rate
##            <chr>       <dbl>      <dbl>
##  1 exclamationis        20.8       67.9
##  2 exclamationis        20.8       65.1
##  3 exclamationis        24.0       77.3
##  4 exclamationis        24.0       78.7
##  5 exclamationis        24.0       79.4
##  6 exclamationis        24.0       80.4
##  7 exclamationis        26.2       85.8
##  8 exclamationis        26.2       86.6
##  9 exclamationis        26.2       87.5
## 10 exclamationis        26.2       89.1
## # ... with 24 more rows
```

I saved the data frame this time, since this is the one we will use for our analysis.

The warning message tells us that we got genuine missing-value NAs back, which is probably what we want. (We'll ignore them and see if they cause us any trouble. The same warning messages will show up on graphs later.) So I have 34 rows (including three rows of missings) instead of the 31 rows I would have liked. Otherwise, success.

There is (inevitably) another way to do this. We are doing the `as.numeric` twice, exactly the same on two different columns, and when you are doing the same thing on a number of columns, here a `mutate` with the same function, you have the option of using `mutate_if` or `mutate_at`. These are like `summarize_if` and `summarize_at` that we used way back to compute numerical summaries of a bunch of columns: the `if` variant works on columns that share a property, like being numeric, and the `at` variant works on columns whose names have something in common or that we can list, which is what we want here:

```
crickets %>%
  unite(exclamationis,X1:X2) %>%
  unite(niveus,X3:X4) %>%
  gather(species,temp_pulse,exclamationis:niveus) %>%
  separate(temp_pulse,c("temperature","pulse_rate"),"_") %>%
  mutate_at(vars(temperature:pulse_rate),funs(as.numeric))
```

```
## Warning in eval(substitute(expr), envir, enclos):  NAs introduced by coercion
```

```
## Warning in eval(substitute(expr), envir, enclos):  NAs introduced by coercion
```

```
## # A tibble: 34 x 3
##          species temperature pulse_rate
##            <chr>       <dbl>      <dbl>
##  1 exclamationis        20.8       67.9
##  2 exclamationis        20.8       65.1
##  3 exclamationis        24.0       77.3
##  4 exclamationis        24.0       78.7
##  5 exclamationis        24.0       79.4
##  6 exclamationis        24.0       80.4
##  7 exclamationis        26.2       85.8
##  8 exclamationis        26.2       86.6
##  9 exclamationis        26.2       87.5
## 10 exclamationis        26.2       89.1
## # ... with 24 more rows
```

Can't I just say that these are columns 2 and 3?

```
crickets %>%
  unite(exclamationis,X1:X2) %>%
  unite(niveus,X3:X4) %>%
  gather(species,temp_pulse,exclamationis:niveus) %>%
  separate(temp_pulse,c("temperature","pulse_rate"),"_") %>%
  mutate_at(vars(2:3),funs(as.numeric))
```

```
## Warning in eval(substitute(expr), envir, enclos):  NAs introduced by coercion
```

```
## Warning in eval(substitute(expr), envir, enclos):  NAs introduced by coercion
```

```
## # A tibble: 34 x 3
##          species temperature pulse_rate
##            <chr>       <dbl>      <dbl>
##  1 exclamationis        20.8       67.9
##  2 exclamationis        20.8       65.1
##  3 exclamationis        24.0       77.3
##  4 exclamationis        24.0       78.7
##  5 exclamationis        24.0       79.4
##  6 exclamationis        24.0       80.4
##  7 exclamationis        26.2       85.8
##  8 exclamationis        26.2       86.6
##  9 exclamationis        26.2       87.5
## 10 exclamationis        26.2       89.1
## # ... with 24 more rows
```

Yes. Equally good. What goes into the `vars` is the same as can go into a `select`: column numbers,

names, or any of those "select helpers" like `starts_with`.

Check that the temperature and pulse rate columns are now labelled `dbl`, which means they are decimal numbers (and don't just look like decimal numbers).

Either way, using `unite` and then `separate` means that all the columns we created we want to keep (or, all the ones we would have wanted to get rid of have already been gotten rid of).

Now we can actually do some statistics.

(f) Do a two-sample t-test to see whether the mean pulse rates differ between species. What do you conclude?

Solution: Drag your mind way back to this:

```
t.test(pulse_rate~species,data=crickets.1)
```

```
##
##  Welch Two Sample t-test
##
## data:  pulse_rate by species
## t = 5.2236, df = 28.719, p-value = 1.401e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  14.08583 32.22677
## sample estimates:
## mean in group exclamationis        mean in group niveus
##                    85.58571                    62.42941
```

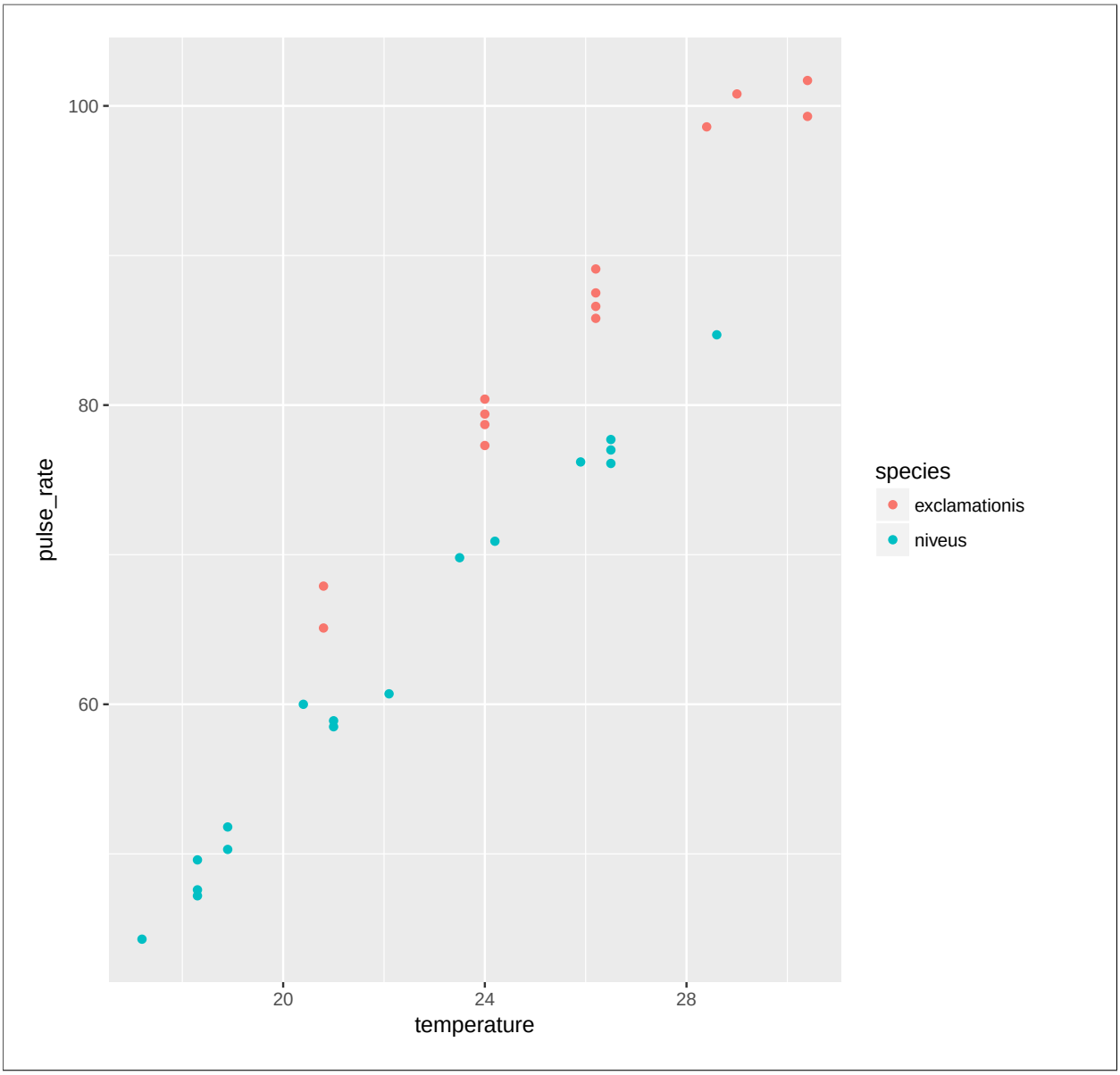
There is strong evidence of a difference in means (a P-value around 0.00001), and the confidence interval says that the mean chirp rate is higher for *exclamationis*. That is, not just for the crickets that were observed here, but for *all* crickets of these two species.

(g) The analysis in the last part did not use temperature, however. Is it possible that temperature also has an effect? To assess this, draw a scatterplot of pulse rate against temperature, with the points distinguished, somehow, by the species they are from.[5]

Solution: One of the wonderful things about `ggplot` is that doing the obvious thing works:

```
ggplot(crickets.1,aes(x=temperature,y=pulse_rate,colour=species))+
  geom_point()
```

```
## Warning:  Removed 3 rows containing missing values (geom_point).
```



(h) What does the plot tell you that the t-test doesn't? How would you describe differences in pulse rates between species now?

Solution: The plot tells you that (for both species) as temperature goes up, pulse rate goes up as well. *Allowing for that*, the difference in pulse rates between the two species is even clearer than it was before. To see an example, pick a temperature, and note that the mean pulse rate at that temperature seems to be at least 10 higher for exclamationis, with a high degree of consistency.

The t-test mixed up all the pulse rates at all the different temperatures. Even though the conclusion was clear enough, it could be clearer if we incorporated temperature into the analysis.

There was also a potential source of unfairness in that the exclamationis crickets tended to be observed at higher temperatures than niveus crickets; since pulse rates increase with temperature, the apparent difference in pulse rates between the species might have been explainable by one species being observed

mainly in higher temperatures. This was *utterly invisible* to us when we did the t-test, but it shows the importance of accounting for all the relevant variables when you do your analysis.[6] If the species had been observed at opposite temperatures, we might have concluded[7] that niveus have the higher pulse rates on average. I come back to this later when I discuss the confidence interval for species difference that comes out of the regression model with temperature.

(i) Fit a regression predicting pulse rate from species and temperature. Compare the P-value for species in this regression to the one from the t-test. What does that tell you?

Solution: This is actually a so-called "analysis of covariance model", which properly belongs in D29, but it's really just a regression:

```
pulse.1=lm(pulse_rate~species+temperature,data=crickets.1)
summary(pulse.1)
```

```
##
## Call:
## lm(formula = pulse_rate ~ species + temperature, data = crickets.1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.0128 -1.1296 -0.3912  0.9650  3.7800
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -7.21091    2.55094  -2.827  0.00858 **
## speciesniveus -10.06529    0.73526 -13.689 6.27e-14 ***
## temperature     3.60275    0.09729  37.032  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.786 on 28 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.9896,Adjusted R-squared:  0.9888
## F-statistic:  1331 on 2 and 28 DF,  p-value: < 2.2e-16
```

The P-value for species is now 6.27×10^{-14} or 0.00000000000006, which is even less than the P-value of 0.00001 that came out of the t-test. That is to say, when you know temperature, you can be even more sure of your conclusion that there is a difference between the species.

The R-squared for this regression is almost 99%, which says that if you know both temperature and species, you can predict the pulse rate almost exactly.

In the regression output, the slope for species is about -10. It is labelled `speciesniveus`. Since species is categorical, `lm` uses the first category, exclamationis, as the baseline and expresses each other species relative to that. Since the slope is about -10, it says that at any given temperature, the mean pulse rate for niveus is about 10 less than for exclamationis. This is pretty much what the scatterplot told us.

We can go a little further here:

```
confint(pulse.1)
```

```
##                     2.5 %    97.5 %
## (Intercept)   -12.436265 -1.985547
## speciesniveus -11.571408 -8.559175
## temperature     3.403467  3.802038
```

The second line says that the pulse rate for *niveus* is between about 8.5 and 11.5 less than for *exclamationis*, at any given temperature (comparing the two species at the same temperature as each other, but that temperature could be anything). This is a lot shorter than the CI that came out of the t-test, that went from 14 to 32. This is because we are now accounting for temperature, which also makes a difference. (In the t-test, the temperatures were all mixed up). What we also see is that the t-interval is shifted up compared to the one from the regression. This is because the t-interval conflates[8] two things: the *exclamationis* crickets do have a higher pulse rate, but they were also observed at higher temperatures, which makes it look as if their pulse rates are more higher[9] than they really are, when you account for temperature.

This particular model constrains the slope with temperature to be the same for both species (just the intercepts differ). If you want to allow the slopes to differ between species, you add an interaction between temperature and species:

```
pulse.2=lm(pulse_rate~species*temperature,data=crickets.1)
summary(pulse.2)
```

```
##
## Call:
## lm(formula = pulse_rate ~ species * temperature, data = crickets.1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7031 -1.3417 -0.1235  0.8100  3.6330
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)                -11.0408     4.1515  -2.659    0.013 *
## speciesniveus               -4.3484     4.9617  -0.876    0.389
## temperature                  3.7514     0.1601  23.429   <2e-16 ***
## speciesniveus:temperature   -0.2340     0.2009  -1.165    0.254
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.775 on 27 degrees of freedom
##   (3 observations deleted due to missingness)
## Multiple R-squared:  0.9901,Adjusted R-squared:  0.989
## F-statistic: 898.9 on 3 and 27 DF,  p-value: < 2.2e-16
```

To see whether adding the interaction term added anything to the prediction,[10] compare the model with and without using `anova`:

```
anova(pulse.1,pulse.2)
```

```
## Analysis of Variance Table
##
## Model 1: pulse_rate ~ species + temperature
## Model 2: pulse_rate ~ species * temperature
##   Res.Df    RSS Df Sum of Sq     F Pr(>F)
## 1     28 89.350
## 2     27 85.074  1    4.2758 1.357 0.2542
```

There's no significant improvement by adding the interaction, so there's no evidence that having different slopes for each species is necessary. Note that `anova` gave the same P-value as did the t-test for the slope coefficient for the interaction in `summary`, 0.254 in both cases. This is because there were only two species and therefore only one slope coefficient was required to distinguish them. If there had been three species, we would have had to look at the `anova` output to hunt for a difference among species, since there would have been two slope coefficients, each with its own P-value.[11]

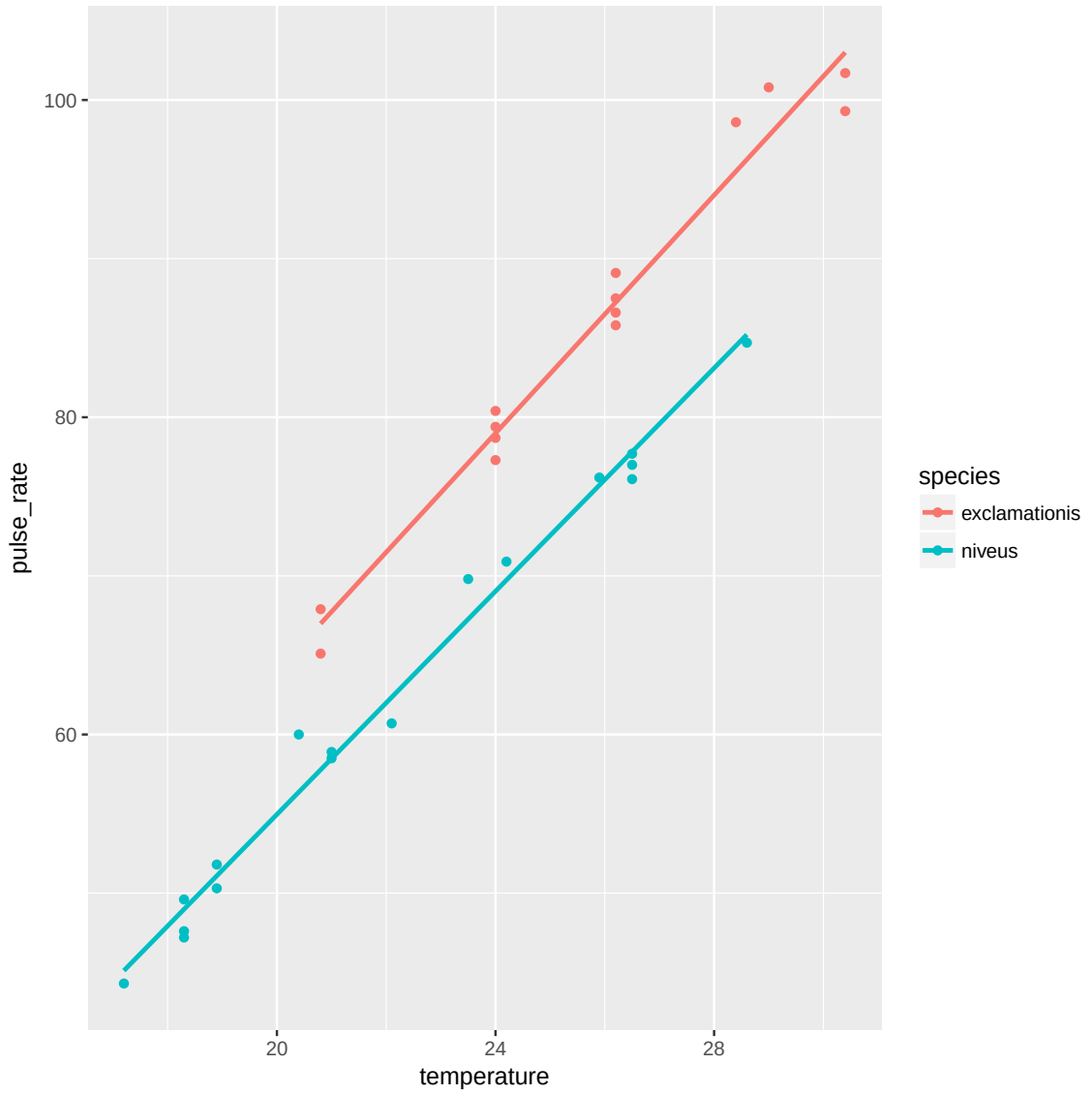
The upshot is that we do not need different slopes; the model `pulse.1` with the same slope for each species describes what is going on.

`ggplot` makes it almost laughably easy to add regression lines for each species to our plot, thus:

```
ggplot(crickets.1,aes(x=temperature,y=pulse_rate,colour=species))+
  geom_point()+geom_smooth(method="lm",se=F)
```

```
## Warning:  Removed 3 rows containing non-finite values (stat_smooth).
```

```
## Warning:  Removed 3 rows containing missing values (geom_point).
```



The lines are almost exactly parallel, so having the same slope for each species makes perfect sense.

Notes

[1]If you had just one x, you'd use a t-test for its slope, and if you were testing all the x's, you'd use the global F-test that appears in the regression output.

[2]This is a "base graphics" graph rather than a "ggplot" one, but it will do for our purposes.

[3]If you're as old as I am, you'll remember a TV series called *ER* starring a very young George Clooney, which was set in Cook County Hospital, in Chicago.

[4]This is not, I don't think, a real word, but I mean "size" emphasizing how big a boy is generally, rather than how small.

[5]This was the actual reason I gave you this question: I wanted you to do this. It sort of morphed into all the other stuff as well.

[6]And it shows the value of looking at relevant plots.

[7]Mistakenly.

[8]Mixes up.

[9]This is actually grammatically correct.

[10]Though it's hard to imagine being able to improve on an R-squared of 99%.

[11]This wouldn't have told us about the overall effect of `species`.