SI 618 – Week 10

Instructor: Ceren Budak

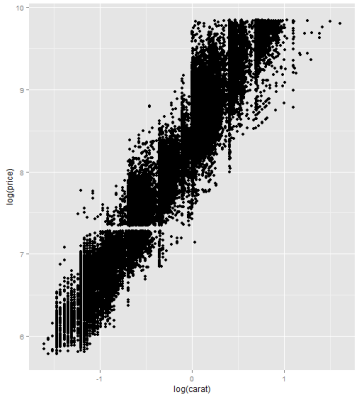
A note about homeworks

Today's roadmap: relationships between two variables

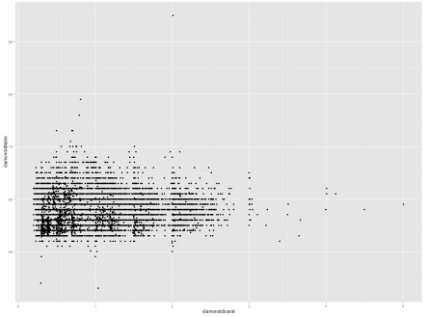
- Relationships between continuous variables
  - Correlation vs causation
  - Linear models and residuals in R and ggplot2
  - Time series and autocorrelation
- Relationships between categorical variables
  - Contingency tables
- Case study: Texas housing prices

Correlation: a simple test of linear relationship

- Suppose our dataset has two variables X, Y
- Plot how X and Y change together
- Construct a rule to predict Y based on an observation of X
- If X and Y are correlated, our prediction of Y should be better than just guessing
- Linear rule: X & Y tend to fall on a sloping line



log(price) and log(carat) are highly correlated: cor(log price, log carat) = 0.92



carat and table size are weakly correlated. cor(carat, table) = 0.18

The Pearson correlation function cor() indicates the strength of a linear relationship between two variables

- cor (X, Y) has a value between -1 and +1
- Strong correlations are close to -1 or +1
- You can compute correlations between sets of variables

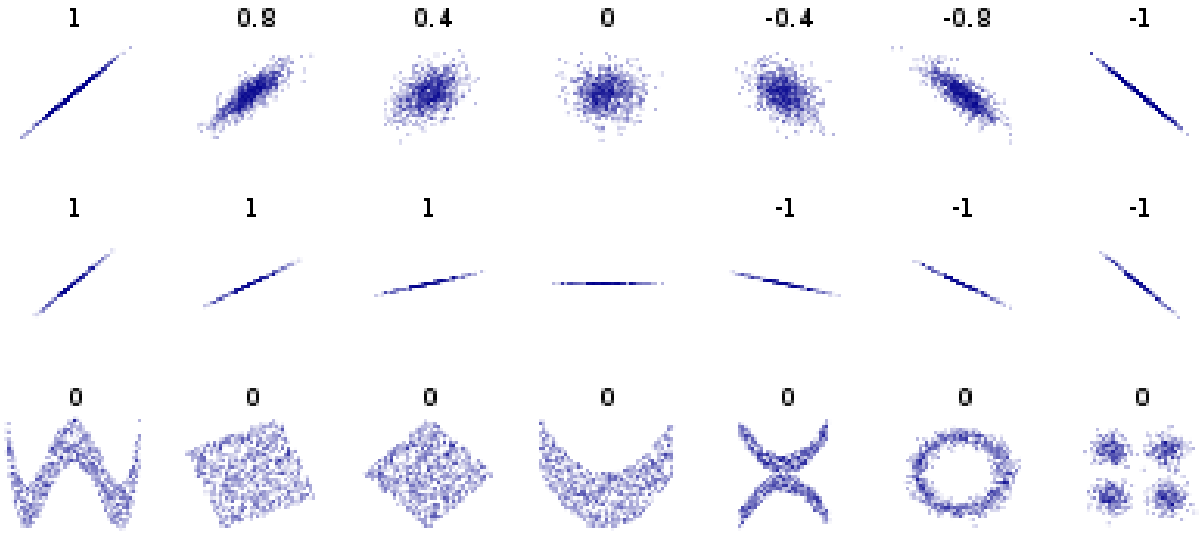
```
> cor(mtcars[1:3], mtcars[4:6])
             hp       drat         wt
mpg  -0.7761684  0.6811719 -0.8676594
cyl   0.8324475 -0.6999381  0.7824958
disp  0.7909486 -0.7102139  0.8879799
```

Horsepower and miles-per-gallon are negatively correlated
Horsepower and number of cylinders are (highly) positively correlated

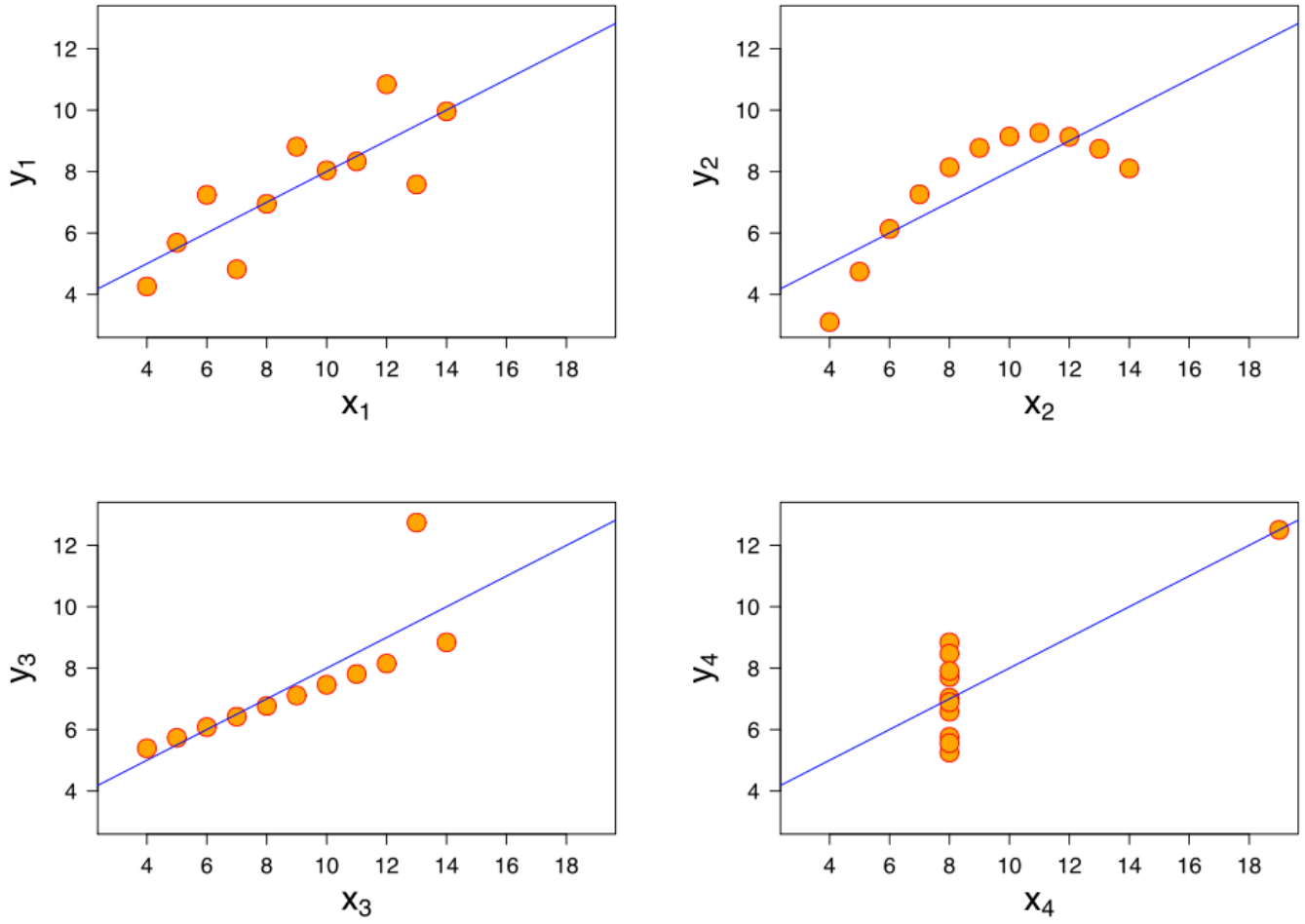
http://www.statmethods.net/stats/correlations.html
http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient

Different sets of (x,y) points with Pearson correlation per set

- Correlation reflects the noisiness and direction of a linear relationship
- But not the slope or non-linear relationships



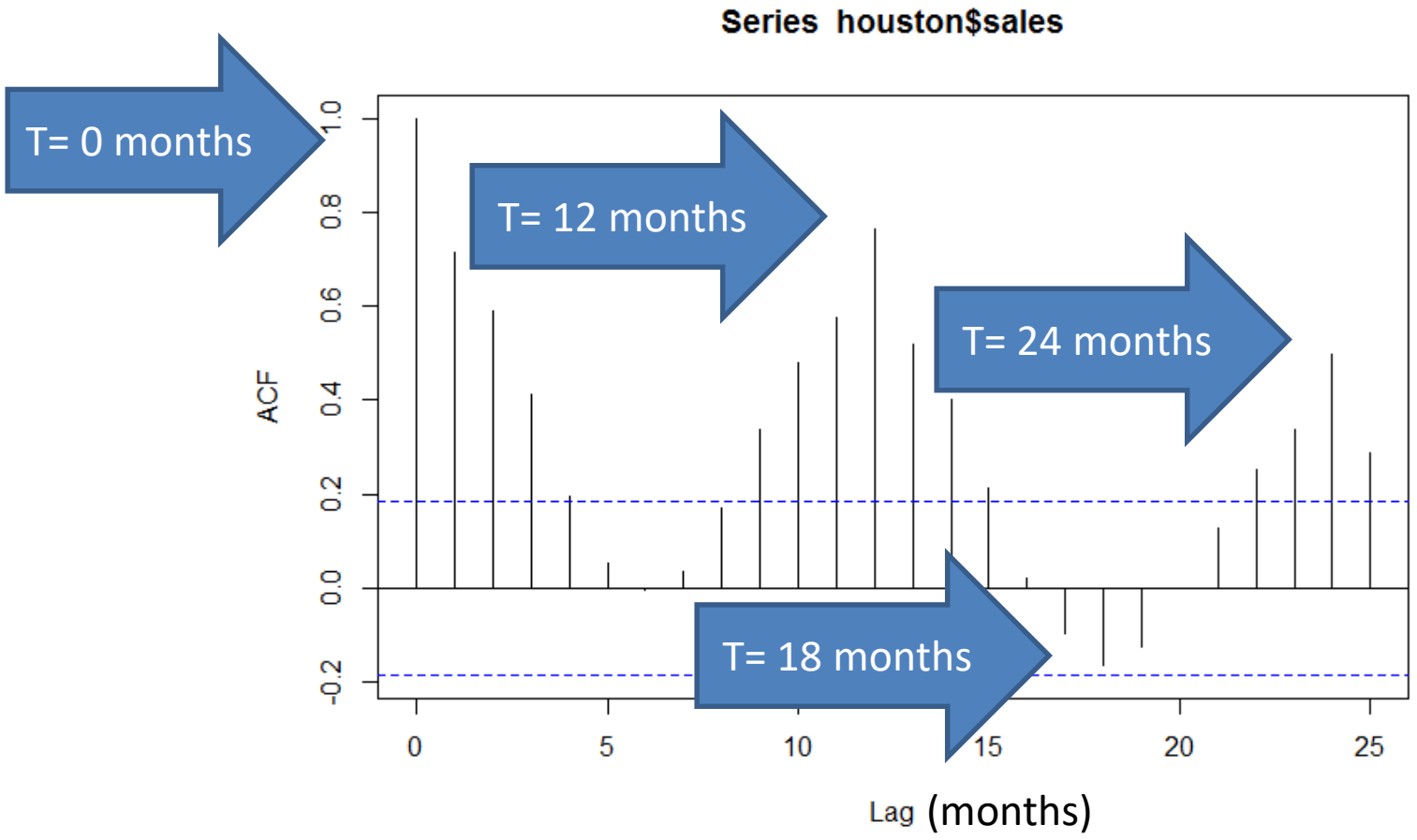
Anscombe Dataset: All these datasets have the same correlation (r = 0.816)



Detecting seasonal cycles with autocorrelation

- Autocorrelation
  - Correlation of a time series signal with time-shift of itself
- Why would that possibly be useful?
  - Similarity of observations as a function of time lag between them
  - A mathematical tool for finding repeating patterns
    - e.g. the presence of a periodic signal obscured by noise

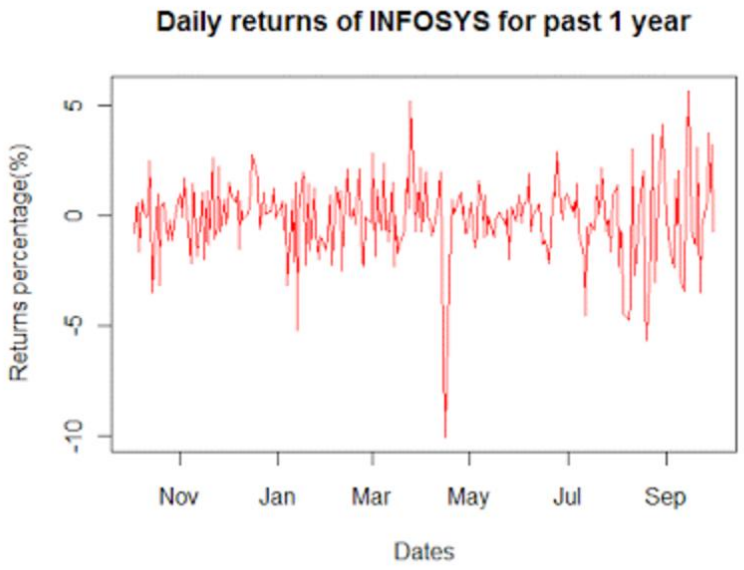
Autocorrelation in the Houston sales data using acf()



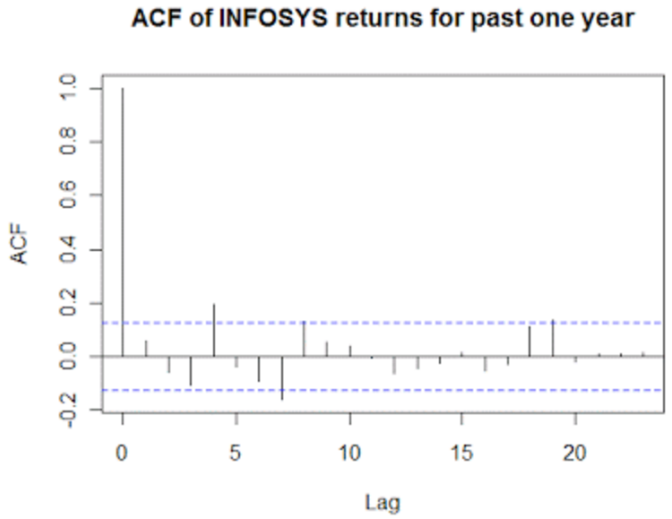
Using the R function acf(houston$sales)

But be careful…

- Example: Stock prices (one particular stock called INFOSYS)



Plotting the ACF of INFY returns for the past one years
acf(infy_ret, main = "ACF of INFOSYS returns for past one year")



4th and 7th lags seem to matter

But be careful…

```
## Regressing the returns till the 7th lag
summary(lm(infy_ret[8:length(infy_ret)] ~ infy_ret[8:length(infy_ret) – 1] + infy_ret[8:length(infy_ret) – 2]+
infy_ret[8:length(infy_ret) – 3] + infy_ret[8:length(infy_ret) – 4] + infy_ret[8:length(infy_ret) – 5] +
infy_ret[8:length(infy_ret) – 6] +infy_ret[8:length(infy_ret) – 7] ))
## This is a simple OLS regression of the "inty_ret" starting from the 8th observation. I have started from the 8th
observation to ensure that the number of obs. are same in the dependents and independent variables.
```

Output:

Coefficients:

	Estimate	Std. Error	t value	Pr(>|t|)	
(Intercept)	-0.09316	0.11321	-0.823	0.41140	
infy_ret[8:length(infy_ret) - 1]	0.08158	0.06479	1.259	0.20920	
infy_ret[8:length(infy_ret) - 2]	-0.04017	0.06537	-0.614	0.53950	
infy_ret[8:length(infy_ret) - 3]	-0.10049	0.06528	-1.539	0.12504	
infy_ret[8:length(infy_ret) - 4]	0.20153	0.06457	3.121	0.00203	**
infy_ret[8:length(infy_ret) - 5]	-0.08566	0.06568	-1.304	0.19344	
infy_ret[8:length(infy_ret) - 6]	-0.06849	0.06584	-1.040	0.29928	
infy_ret[8:length(infy_ret) - 7]	-0.12395	0.06621	-1.872	0.06241	.

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Multiple R-squared: 0.08717, Adjusted R-squared: 0.05998

- Only the coefficient of the 4th lag is statistically significant, and the Adjusted R-squared is small 0.05998 (i.e ~ 6% of the explanation is provided by the above regression).
- The 4 days ago stock price provides a statistically significant explanation of today's stock prices. But we are missing in the above model is the transaction costs – so this is not really capturing how the market moves

Correlation does not imply causation

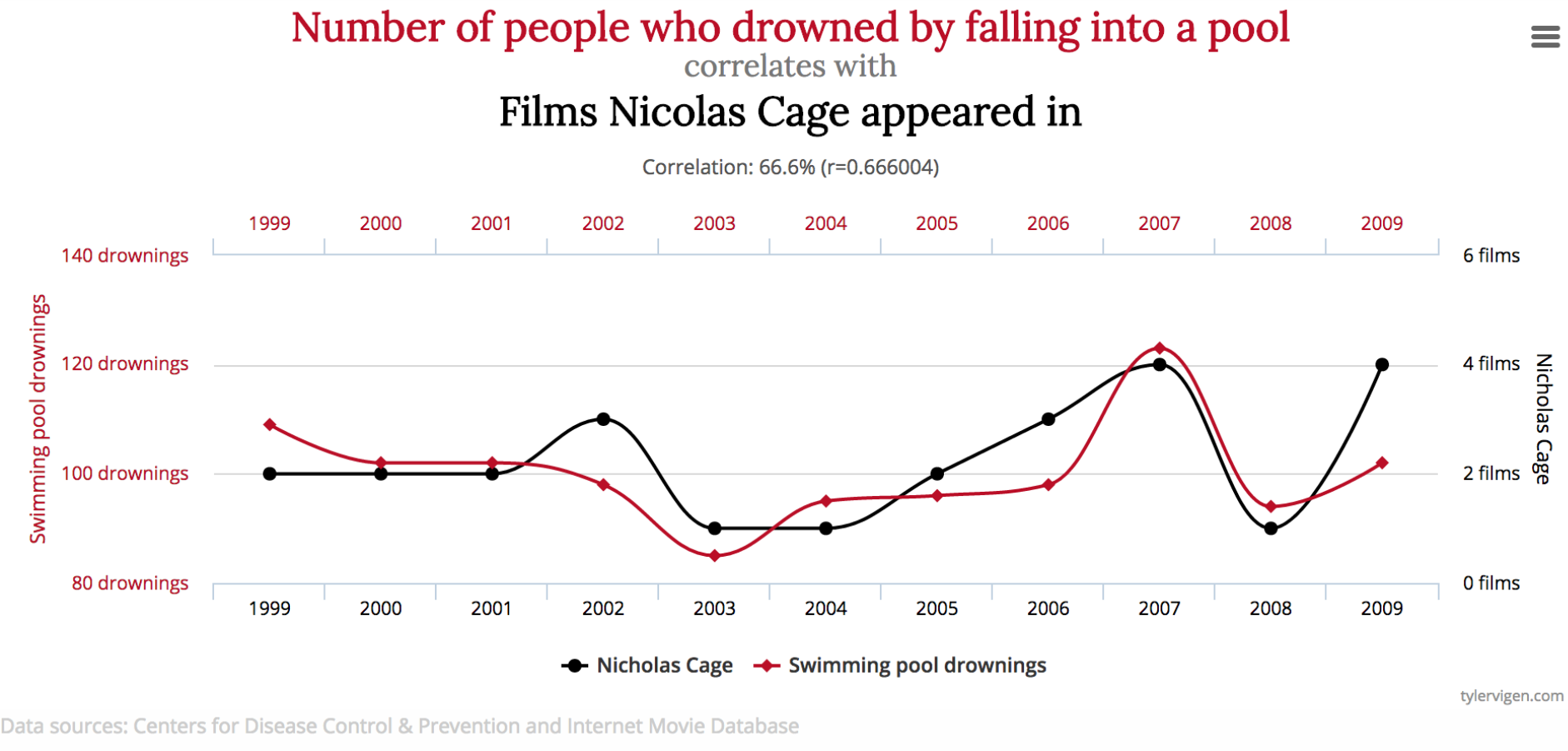
Whenever there's a fire…

…you see the fire department.

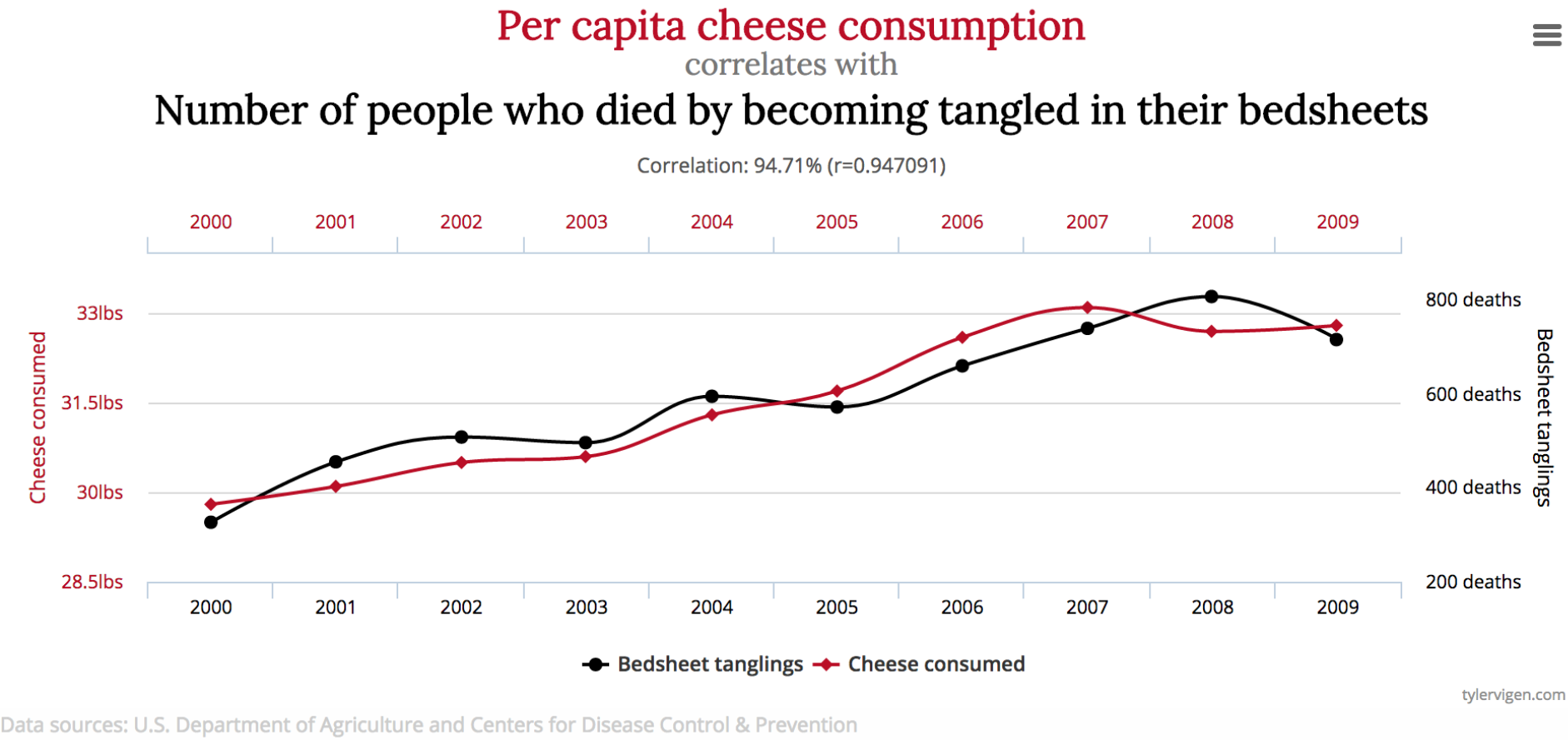
The presence of fires and firefighters is highly correlated.

So… firefighters cause fires?

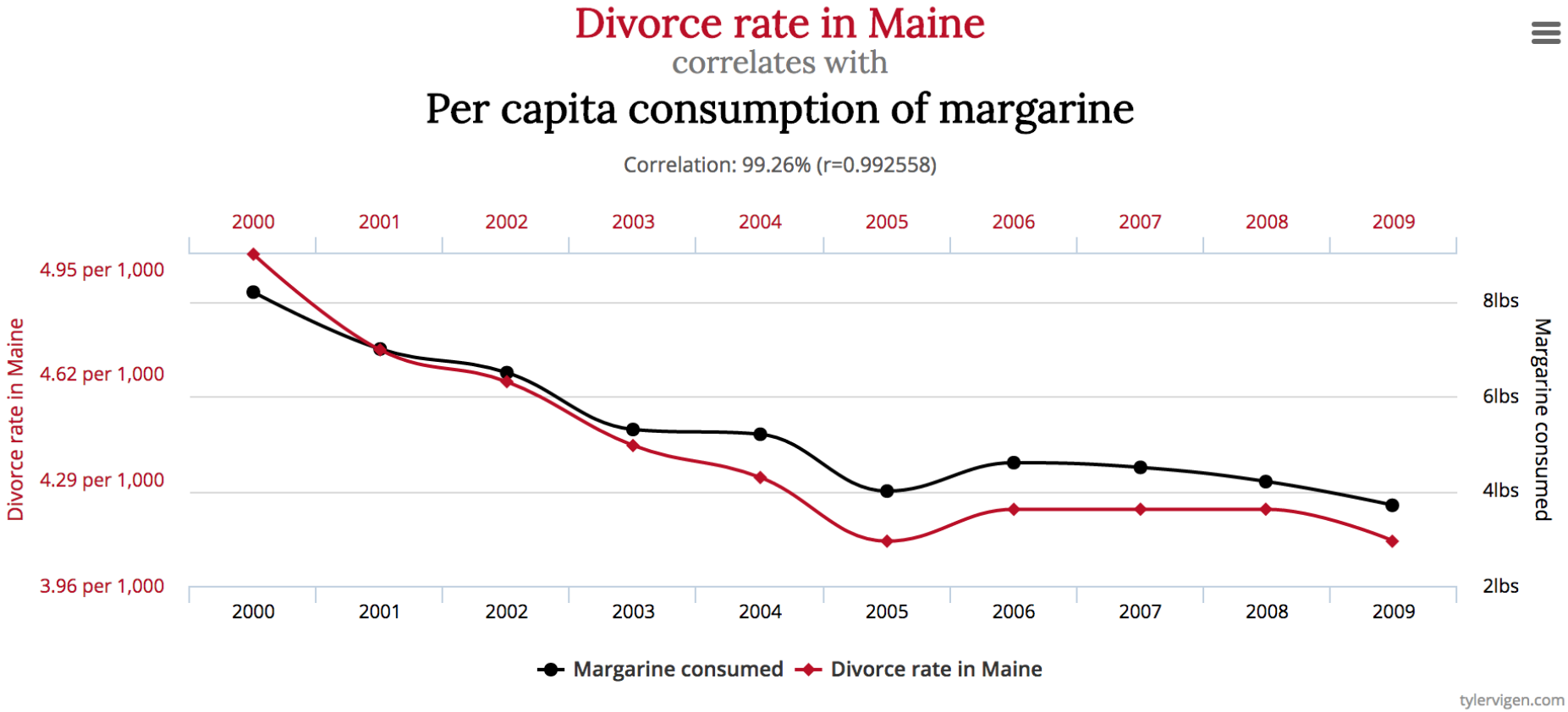
Spurious Correlations



Spurious Correlations



Spurious Correlations



A short trip into geology

- Nature has fascinating geological formations called geysers, a type of hot spring.
- Yellowstone National Park in the U.S. has many of these.
- The most famous geyser is called Old Faithful.
- Spews piping-hot water 120 feet into the air about every hour.

The 'faithful' dataset: eruption observations from Old Faithful

```
> str(faithful)
'data.frame':  272 obs. of  2 variables:
 $ eruptions: num  3.6 1.8 3.33 2.28 4.53 ...
 $ waiting  : num  79 54 74 62 85 55 88 85 51 85 ...
```

Variables:

Eruptions: Length of eruption (min.)

Waiting: Time to next eruption (min.)

Things to notice:

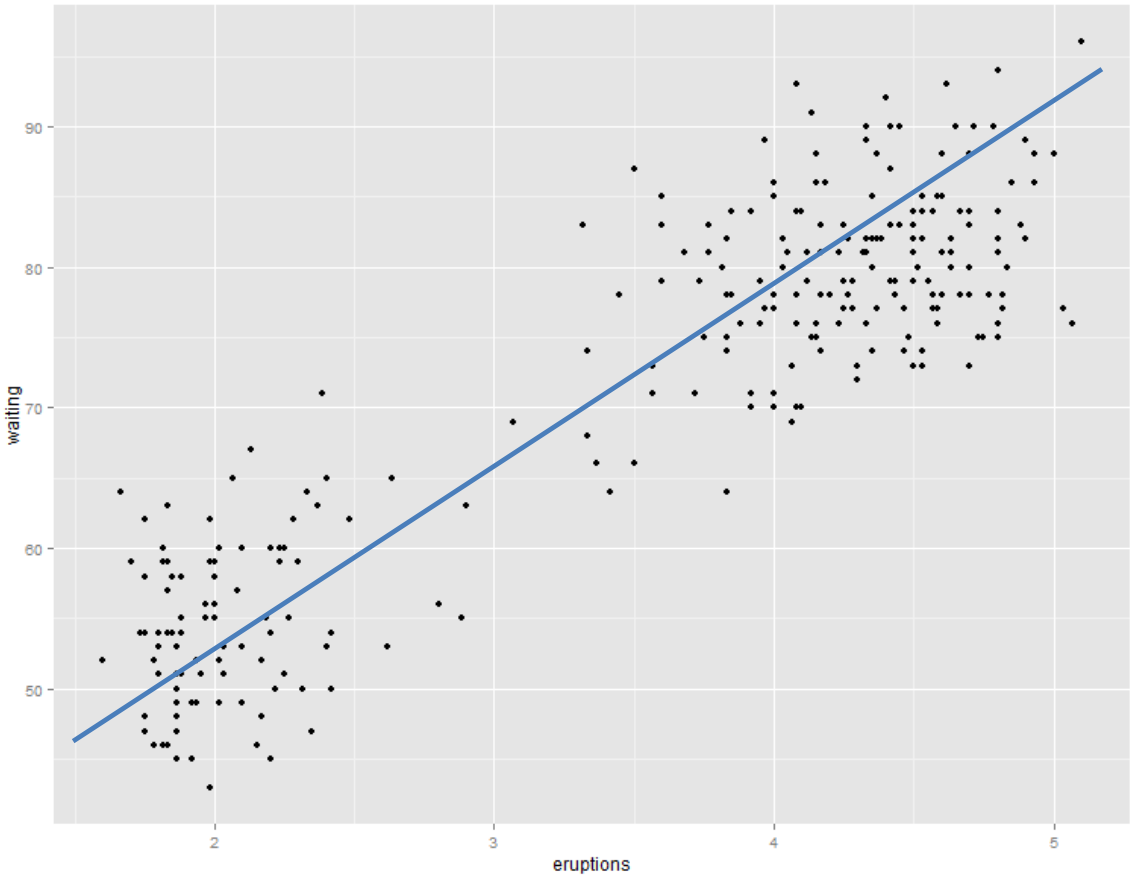
- Bi-modal (short, long) waiting times
- Waiting time is roughly a linear function of eruption length

What is a model of data?

- A model family describes what kind of connection might exist between variables
  - e.g. A linear model predicts a single continuous response to a linear combination of predictor vars + Gaussian noise
  - Specified by the data miner/statistician
- A fitted model is an instance of a model family that has had its parameters estimated from data
  - Fitted model can make predictions on future data
  - Fitted to minimize model prediction error on the data
  - e.g. Linear model minimizes total squared deviation between responses and predictions

Example of a linear model

<u>Response variable</u>: waiting time until next eruption



<u>Predictor variable</u>: length of eruption (minutes)

Linear models in R with lm()

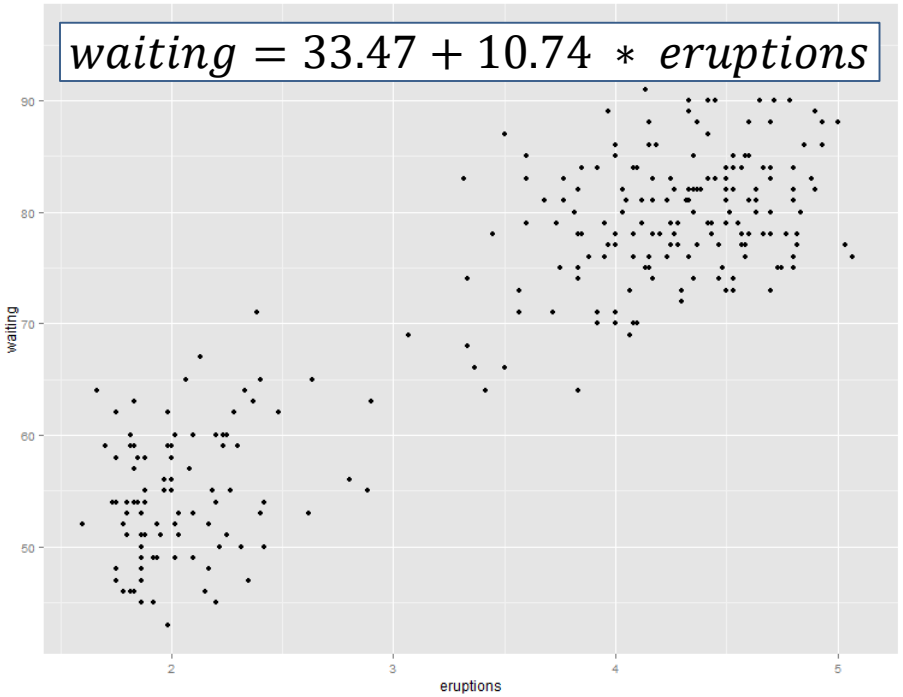
- `lm(y ~ model)` means the response `y` is modelled by a linear predictor described by the expression `model`
- Computes the model coefficients: stores results of the linear fit in *lmfit* object

lm(.) syntax	Model equation	Description
`y ~ x`	$y = \beta_0 + \beta_1 x$	Straight line with y-intercept
`y ~ -1 + x`	$y = \beta_1 x$	Straight line forced thru the origin (0,0)
`y ~ x1 + x2`	$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2$	First-order model with no interaction term between x1, x2.
`y ~ x1:x2`	$y = \beta_0 + \beta_1 x_1 x_2$	Only first-order interaction term between x1, x2.

Predicting duration of Old Faithful eruptions based on time since last eruption

```
> eruption.lm = lm(waiting ~ eruptions, data = faithful)
> eruption.lm
Call:
lm(formula = waiting ~ eruptions, data = faithful)
Coefficients:
(Intercept)    eruptions
      33.47         10.73
```

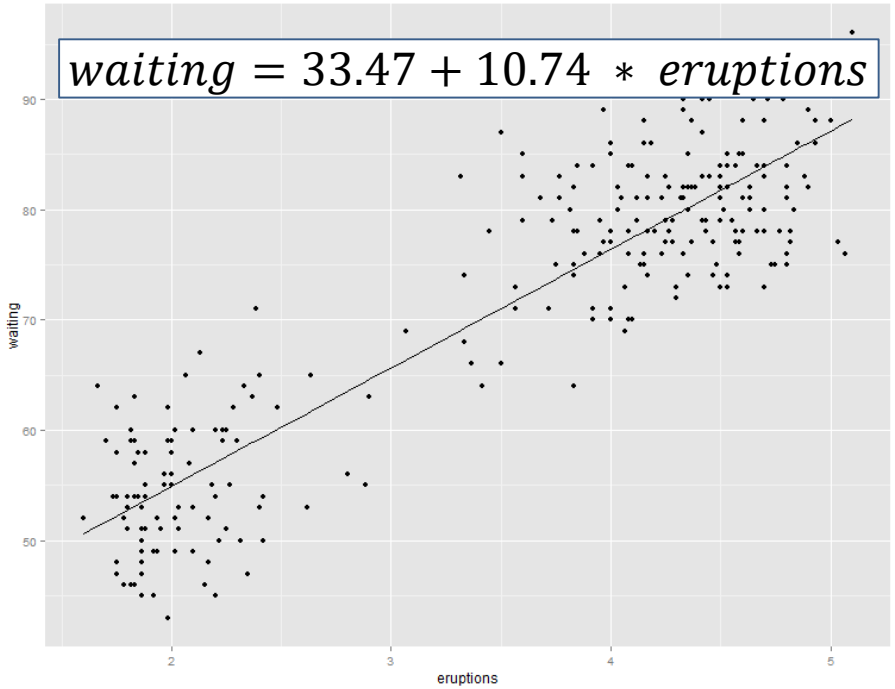
$$waiting = 33.47 + 10.74 * eruptions$$



Every extra minute of eruption means ~10 extra minutes of waiting.

Use `predict`(*lmfit*) to get the fitted line

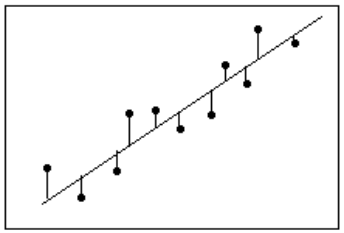
```
> eruption.predict = predict(eruption.lm)
> ggplot() + geom_point(data=faithful, aes(eruptions, waiting))
+ geom_line(data=faithful, aes(eruptions, eruption.predict))
```



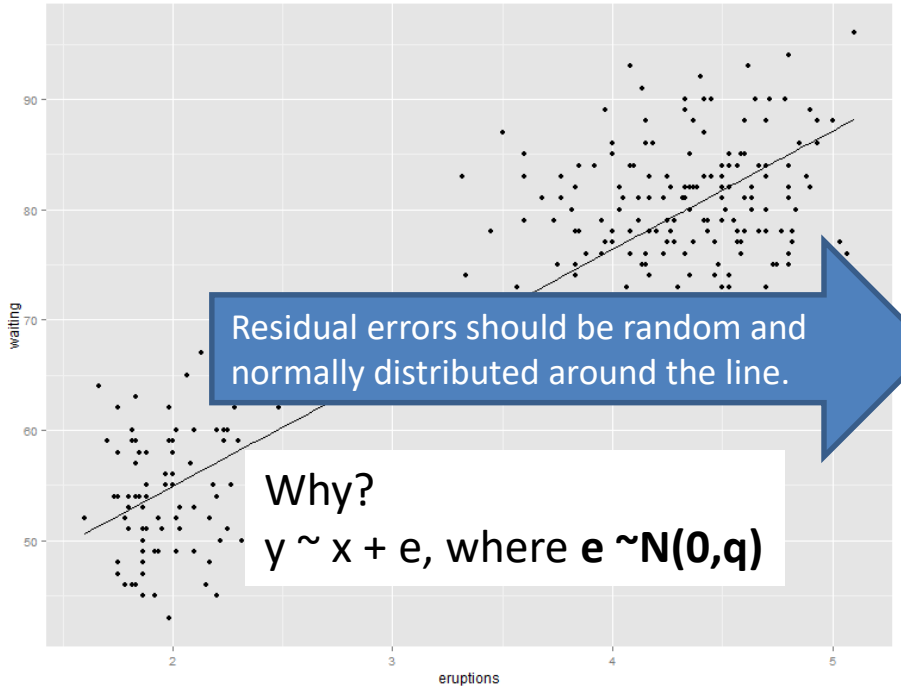
Analyzing prediction errors via residuals: resid()

Duration of previous eruption (min) vs subsequent waiting time (min)

```
> eruption.res = resid(eruption.lm)
> qplot(faithful$waiting, eruption.res)
```



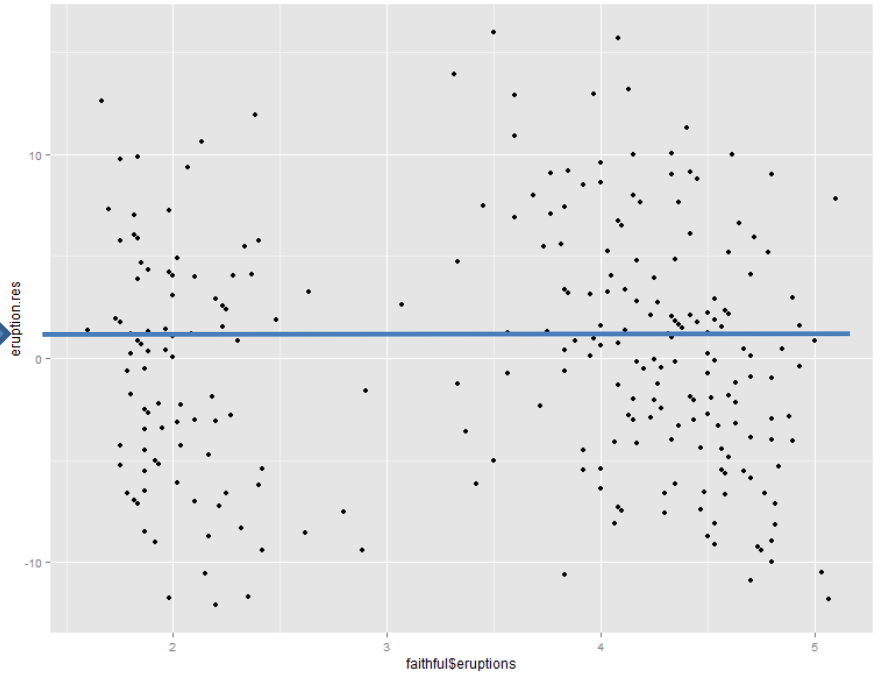
Linear fit



Residual errors should be random and normally distributed around the line.

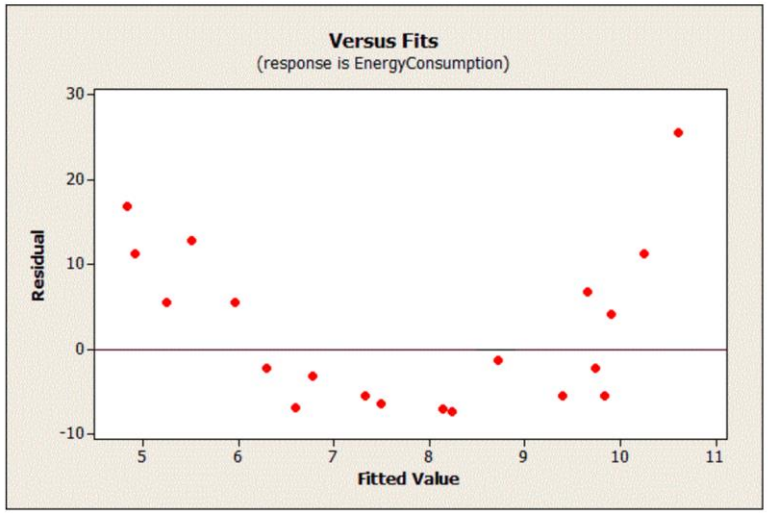
Why?
y ~ x + e, where **e ~N(0,q)**

Residuals (errors) of linear fit



Residuals can show us more subtle patterns after the visually striking pattern is subtracted out

- Residuals
  - Like a magnifying glass on a local area
  - Errors are supposed to be random: any patterns are interesting to find



- The non-random pattern in the residuals indicates that predictor variables are not capturing some explanatory information that is "leaking" into the residuals. Possibilities include:
  - A missing variable
  - A missing higher-order term of a variable in the model to explain the curvature
  - A missing interaction between terms already in the model

We've been analyzing continuous variables

- What if we're interested in relationships between categorical variables?

A 2-d contingency table: diamonds dataset

Attributes: Clarity, PriceLevel

PriceLevel = {“Low” if price <= 3000, “High” if price > 3000}

```
> hiprice <- ifelse(diamonds$price > 3000, "High", "Low")
> table(diamonds$clarity, hiprice, dnn=c('Clarity', 'Price'))

       Price
Clarity High  Low
   I1    425  316
   SI2  6063 3131
   SI1  6285 6780
   VS2  5043 7215
   VS1  3239 4932
   VVS2 1445 3621
   VVS1  734 2921
   IF    370 1420
```

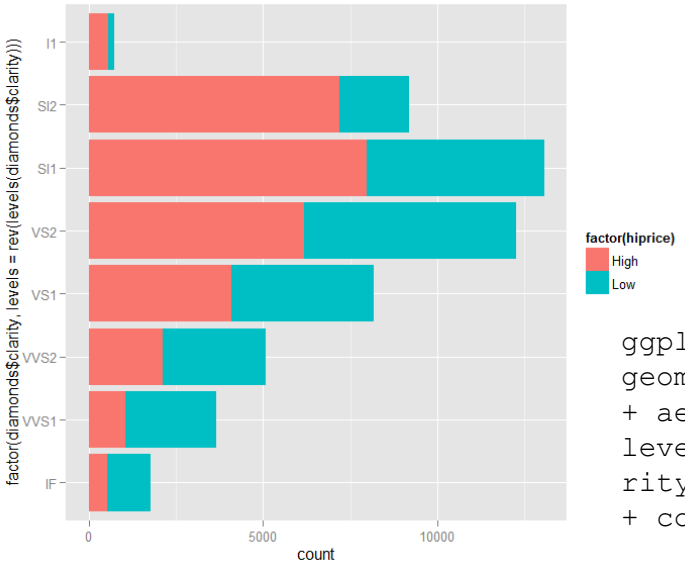
A 2-d contingency table: diamonds dataset

Attributes: Clarity, PriceLevel

PriceLevel = {“Low” if price <= 3000, “High” if price > 3000}

```
> hiprice <- ifelse(diamonds$price > 3000, "High", "Low")
> table(diamonds$clarity, hiprice, dnn=c('Clarity', 'Price'))
```

Clarity	Price High	Low
I1	425	316
SI2	6063	3131
SI1	6285	6780
VS2	5043	7215
VS1	3239	4932
VVS2	1445	3621
VVS1	734	2921
IF	370	1420




```
ggplot(diamonds) +
geom_bar(position="stack")
+ aes(factor(diamonds$clarity,
levels=rev(levels(diamonds$cla
rity))), fill=factor(hiprice))
+ coord_flip()
```

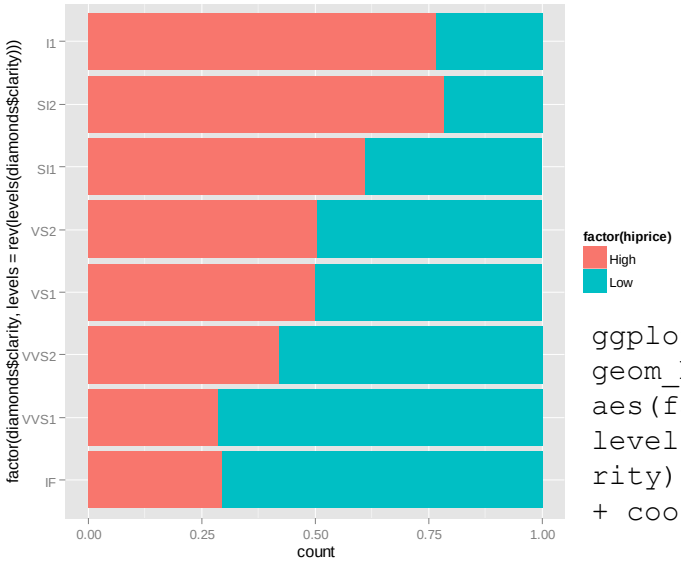
Easier to see interesting patterns with ‘filling’ histogram bars

Attributes: Clarity, PriceLevel

PriceLevel = {“Low” if price <= 3000, “High” if price > 3000}

```
> hiprice <- ifelse(diamonds$price > 3000, "High", "Low")
> table(diamonds$clarity, hiprice, dnn=c('Clarity', 'Price'))
```

Clarity	Price High	Low
I1	425	316
SI2	6063	3131
SI1	6285	6780
VS2	5043	7215
VS1	3239	4932
VVS2	1445	3621
VVS1	734	2921
IF	370	1420




```
ggplot(diamonds) +
geom_bar(position="fill") +
aes(factor(diamonds$clarity,
levels=rev(levels(diamonds$cla
rity))), fill=factor(hiprice))
+ coord_flip()
```

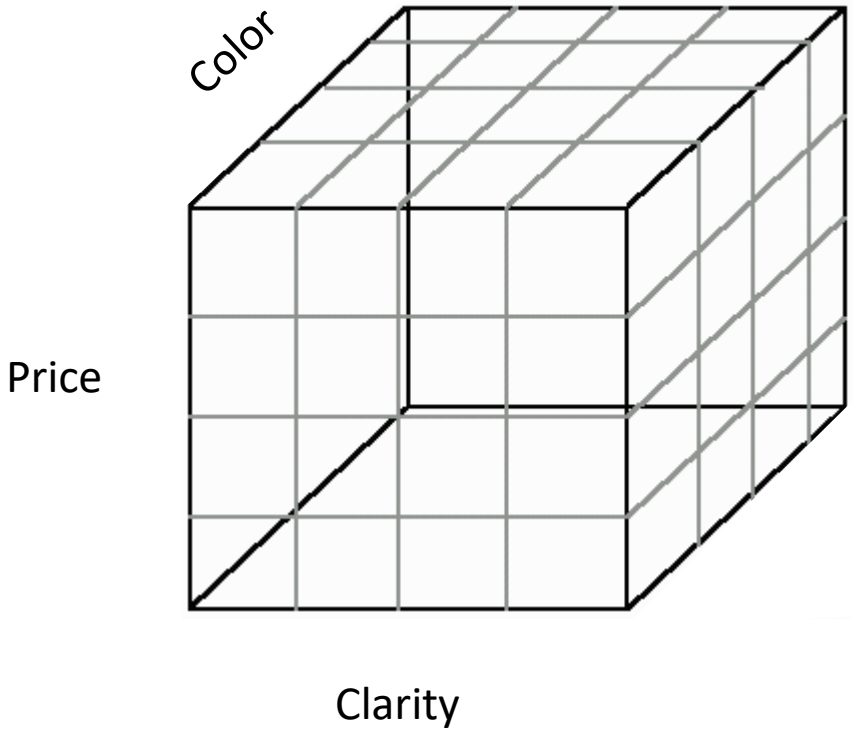
Contingency tables

- Recipe for making a contingency table
  - Pick *k* attributes from your dataset
  - Call them $a_1, a_2, \ldots, a_k$
  - For every possible combination of values

    $a_1 = x_1, a_2 = x_2 \ldots, a_1 = x_k$

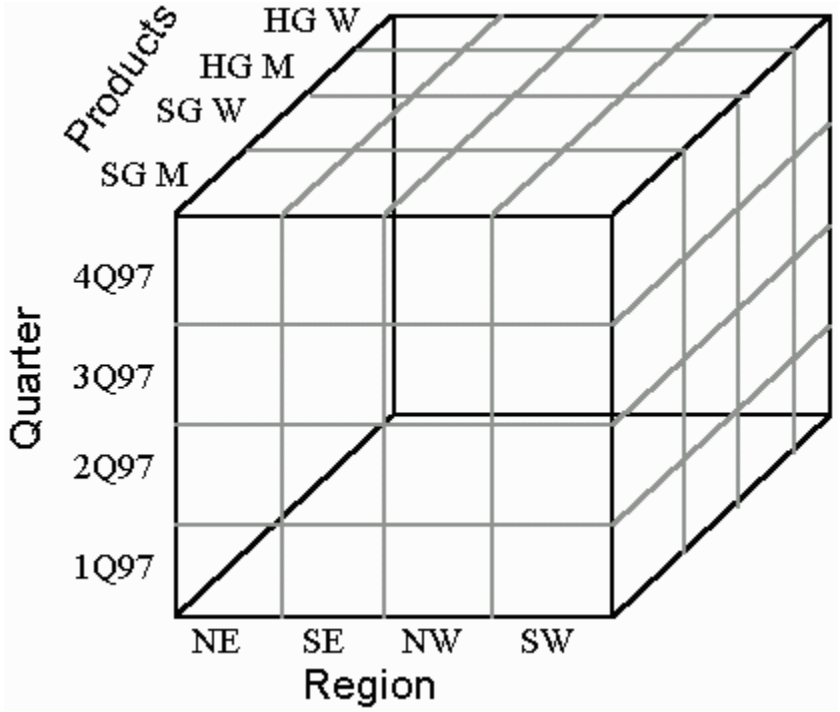
    record how frequently that combination occurs
- Another name for a histogram:
  - a 1-dimensional contingency table

3-dimensional contingency tables



Getting harder to look at ….

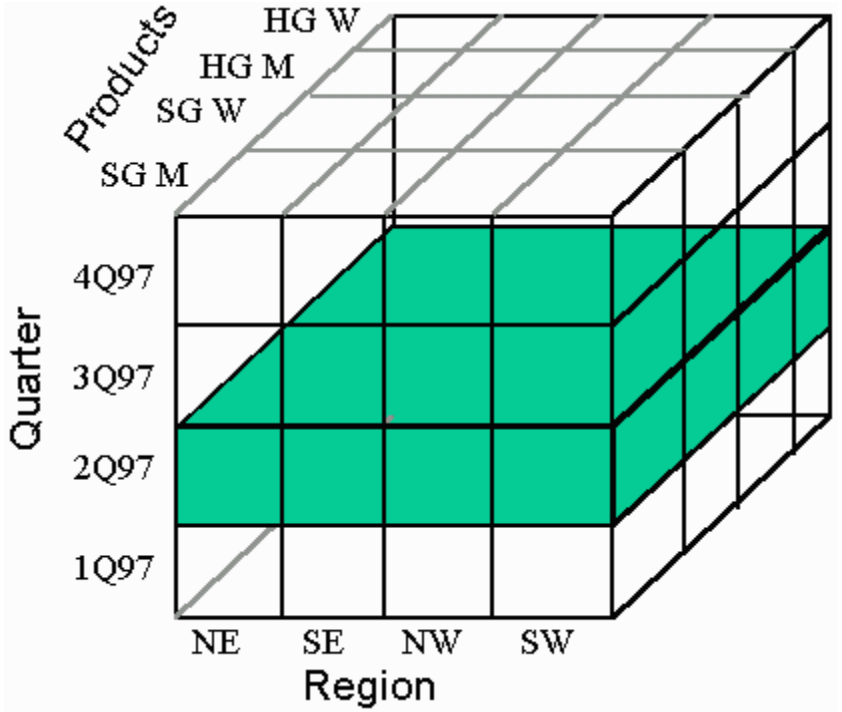
3-dimensional contingency tables



Fun fact: These are called “datacubes” in the database and OLAP (On-Line Analytical Processing) worlds

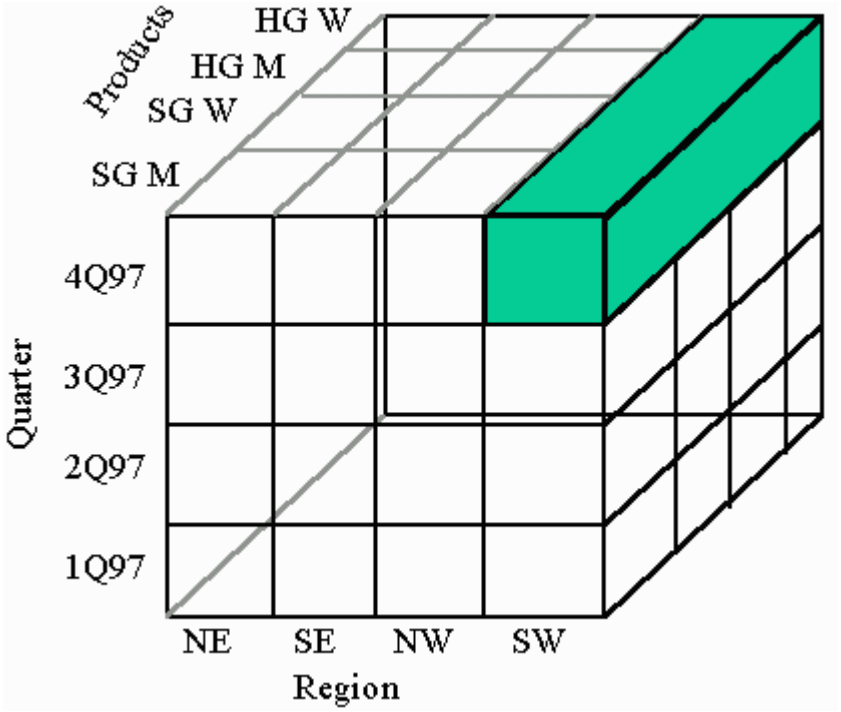
Source: http://cisnet.baruch.cuny.edu/holowczak/classes/9440/olap/

3-dimensional contingency tables



Sales of *all* Products in 2Q 1997 in all regions

3-dimensional contingency tables



Sales of *all* Products in 4Q 1997 in SW region

Time to stop and think

- Why would people want to look at contingency tables, anyway?
- 16 attributes
  - # of 1-d contingency tables: 16-choose-1 = 16
  - # of 2-d contingency tables: 16-choose-2 = 120
  - # of 3-d contingency tables: 16-choose-3 = 560
- 100 attributes
  - # of 3-d contingency tables: 161,700

Which patterns in data are 'interesting'?

- Up until now we've relied mostly on human intelligence and subjective interpretation
- That's fine.. for small numbers of contingency tables, variables.
- What if we have 10,000 variables?
  Not fun anymore!
- Use automated statistical tests that are objective measures of 'interesting'

Contingency tables applied to texts: which words are most characteristic of a category X document?

- 2x2 contingency table for two binary variables
  - Variable 1: Word occurs or doesn't occur in document
  - Variable 2: Document in category X not in category X

	Word occurs	**Word doesn't occur**
Document is category X	a	c
Document not category X	c	d

- Question:

  Which words best characterize category X docs?

Measures of association: how likely did this relationship occur by chance?

- Chi-squared: How far are the expected counts from the observed counts?
  - Widely used, suitable for large samples
  - Null hypothesis: the two variables (word occurs, doc category) are independent.

$$\chi^2 = \sum_{\{i=1\}}^{r} \sum_{\{j=1\}}^{c} (O_{ij} - E_{ij})^2 / E_{ij}$$

	Word occurs	Word doesn't occur	
Document is category X	a = O_{11}	b = O_{12}	r1
Document is category Y	c = = O_{21}	d = O_{22}	r2
	c1	c2	t

Source: http://www.uvm.edu/~dhowell/methods7/Supplements/ChiSquareTests.pdf

The chi-squared measure of association

```
> table(diamonds$clarity, hiprice, dnn = c("Clarity", "Price"))
       Price
Clarity High  Low
  I1    425  316
  SI2  6063 3131
  SI1  6285 6780
  VS2  5043 7215
  VS1  3239 4932
  VVS2 1445 3621
  VVS1  734 2921
  IF    370 1420
> tab = table(diamonds$clarity, hiprice, dnn = c("Clarity", "Price"))
> chisq.test(tab)

        Pearson's Chi-squared test

data:  tab
X-squared = 3783.385, df = 7, p-value < 2.2e-16
```

Chi-squared example

- In 2000 the Vermont State legislature approved a bill authorizing civil unions. The vote can be broken down by gender:

	Vote		
	Yes	No	Total
Women	35 (28.83)	9 (15.17)	44
Men	60 (66.17)	41 (34.83)	101
Total	95	50	145

- E_ij = Ri × Cj/N, where Ri and Cj represent row and column marginal totals and N is the grand total.

$$\chi^2 = \sum\sum \frac{(O_{ij} - E_{ij})^2}{E_{ij}} = \frac{(35 - 28.83)^2}{28.83} + \cdots + \frac{(41 - 34.83)^2}{34.83} = 5.50$$

- Degrees of freedom: (r-1) (c-1)= 1
- χ 2 > 5.50 on 1 df = .019, so we can reject the null hypothesis that voting behavior is independent of gender

A general strategy for analyzing large datasets

1. Start with a single unit

   e.g. a single city in a state database
2. Find interesting patterns
3. Summarize patterns with a model
4. Apply the model to all units
5. Find units that don't fit the pattern
6. Summarize with a single model

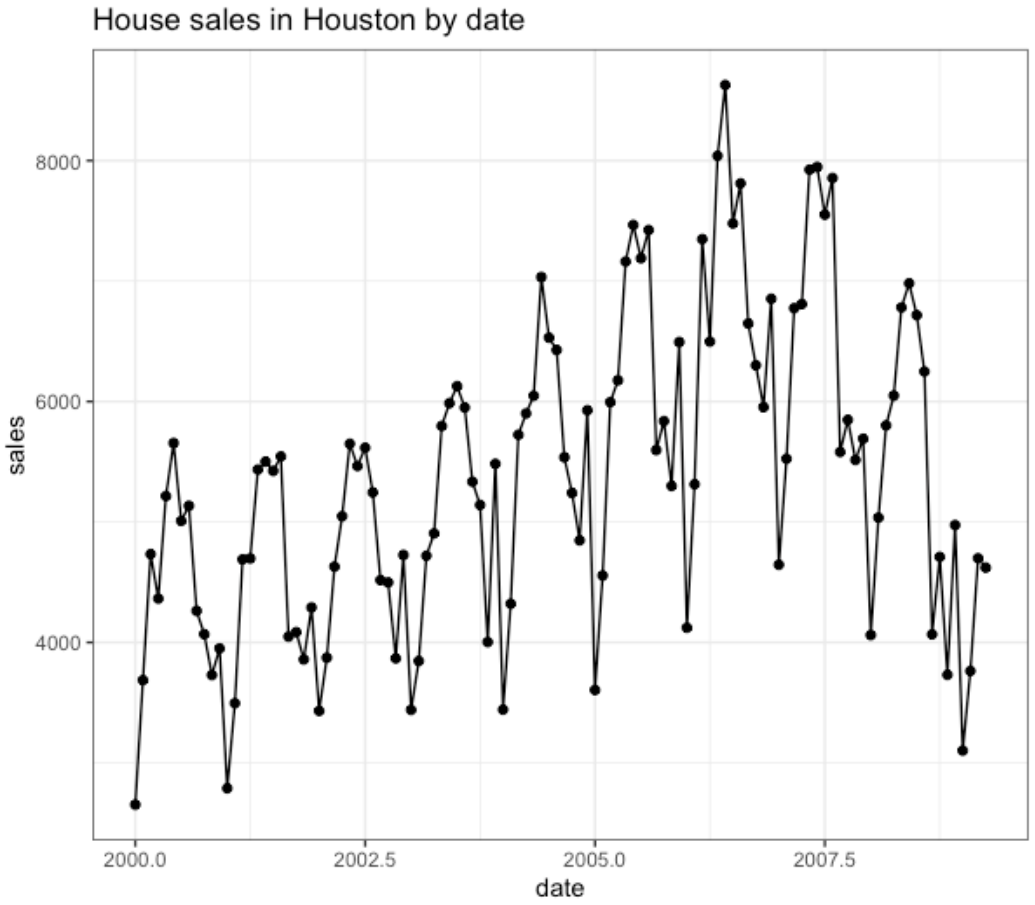
Source for the next section: http://www.slideshare.net/hadley/03-modelling

Case study: Texas housing sales data

- For each metropolitan area in Texas (45 cities)
- For each month 2000-2009 (112 months)
  - Number of houses listed and sold
  - Total value of houses and average sale price
  - Average time on the market

- Strategy:
  - Start with a single city (Houston)
  - Explore patterns and fit models
  - See how the models apply to all cities

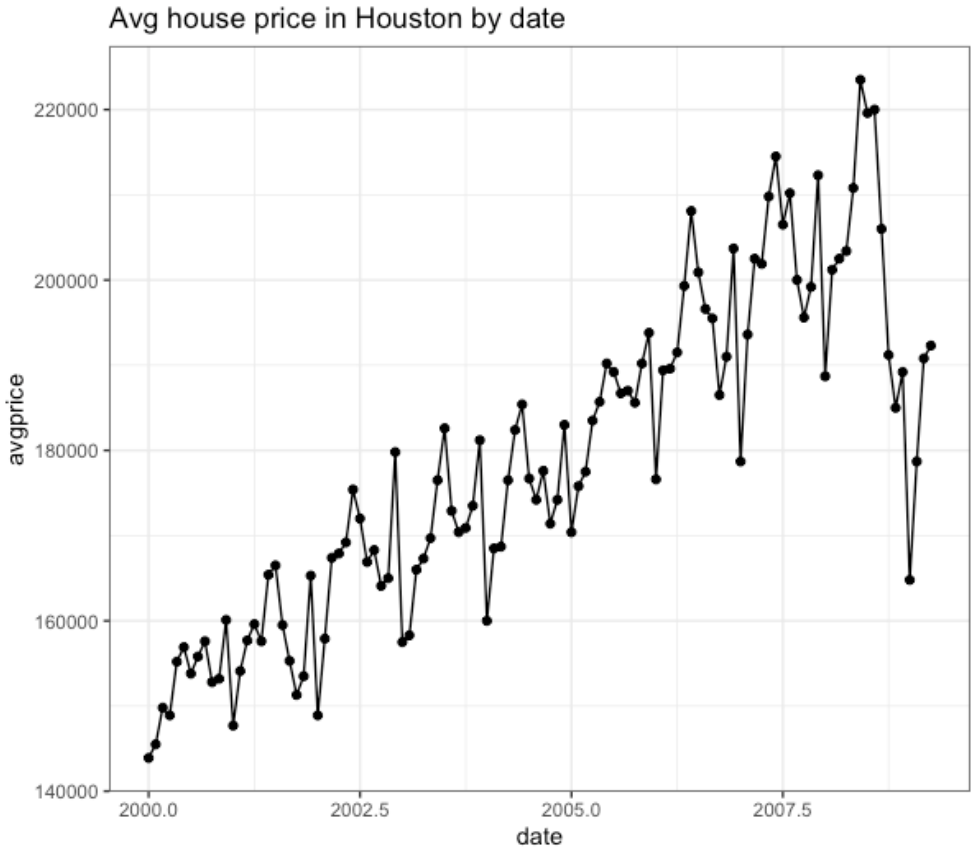
Houston housing sales [2000-2009]




```
ggplot(houston, aes(date, sales)) + geom_point() + geom_line()+
    ggtitle("House sales in Houston by date")
```

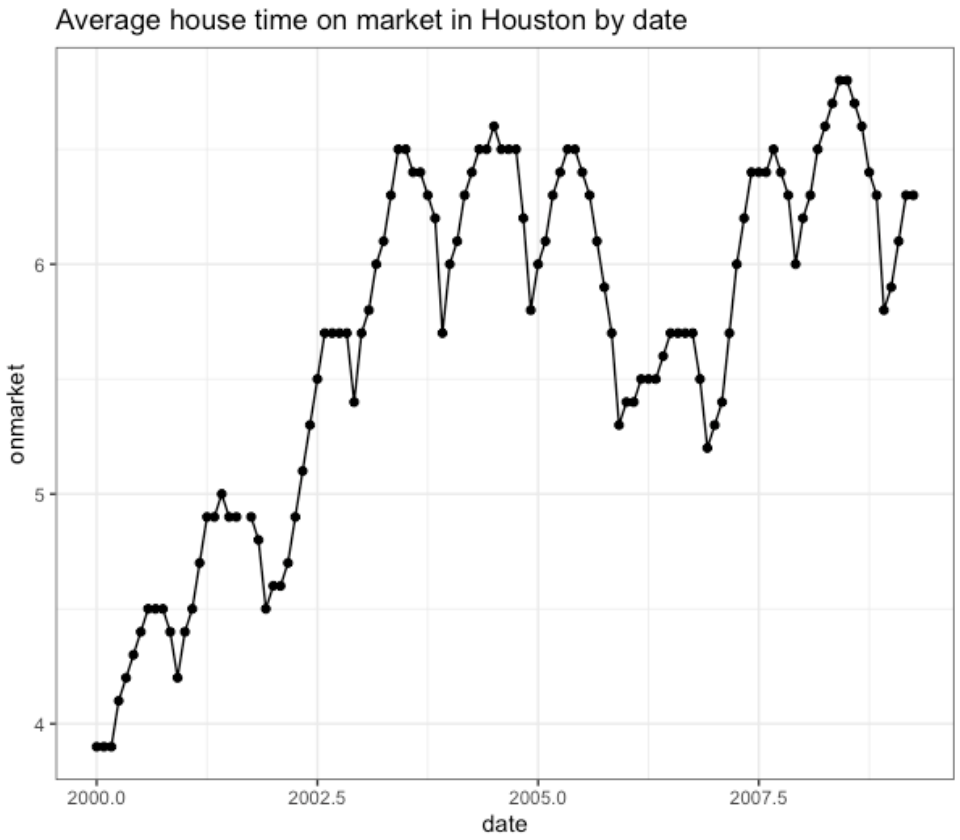
[Source: Wickham] / See Examples for this week on Canvas

Average house price (Houston)




```
ggplot(houston, aes(date, avgprice)) + geom_point() + geom_line()
        + ggtitle("Avg house price in Houston by date")
```

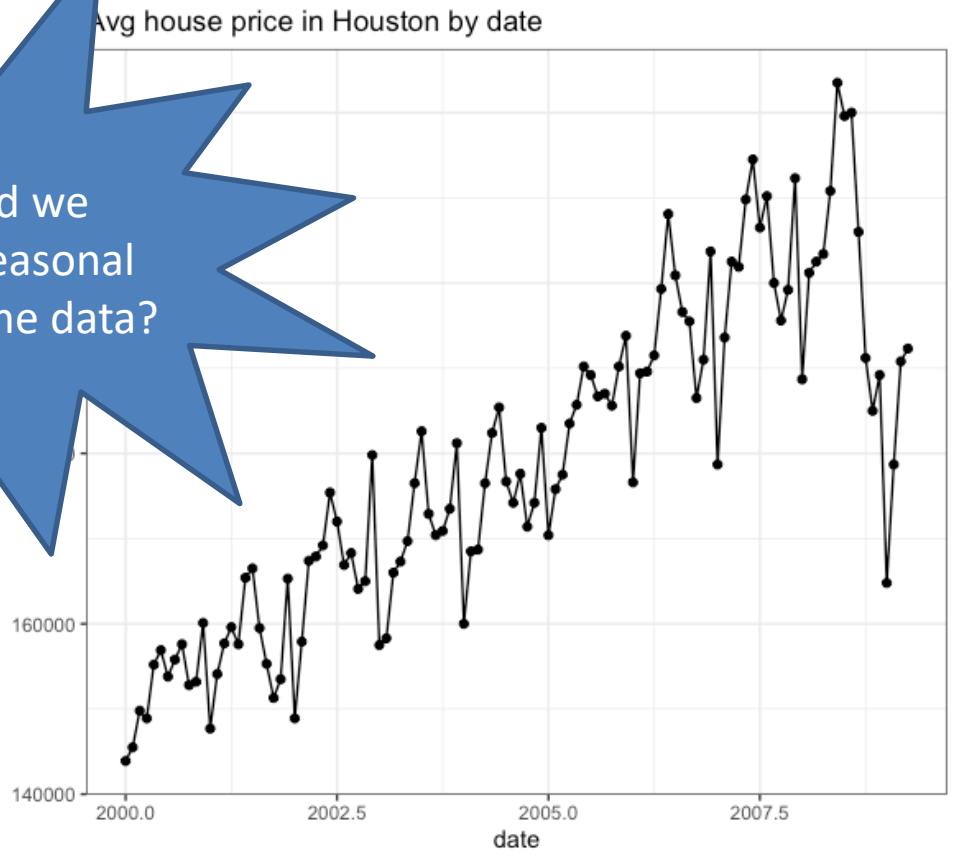
Days on market (Houston)




```
ggplot(houston, aes(date, onmarket)) + geom_point() + geom_line()
+ ggtitle("Average house time on market in Houston by date")
```

Seasonal variations make it harder to see long-term trends and interesting short-term events

How could we "remove" seasonal variation in the data?

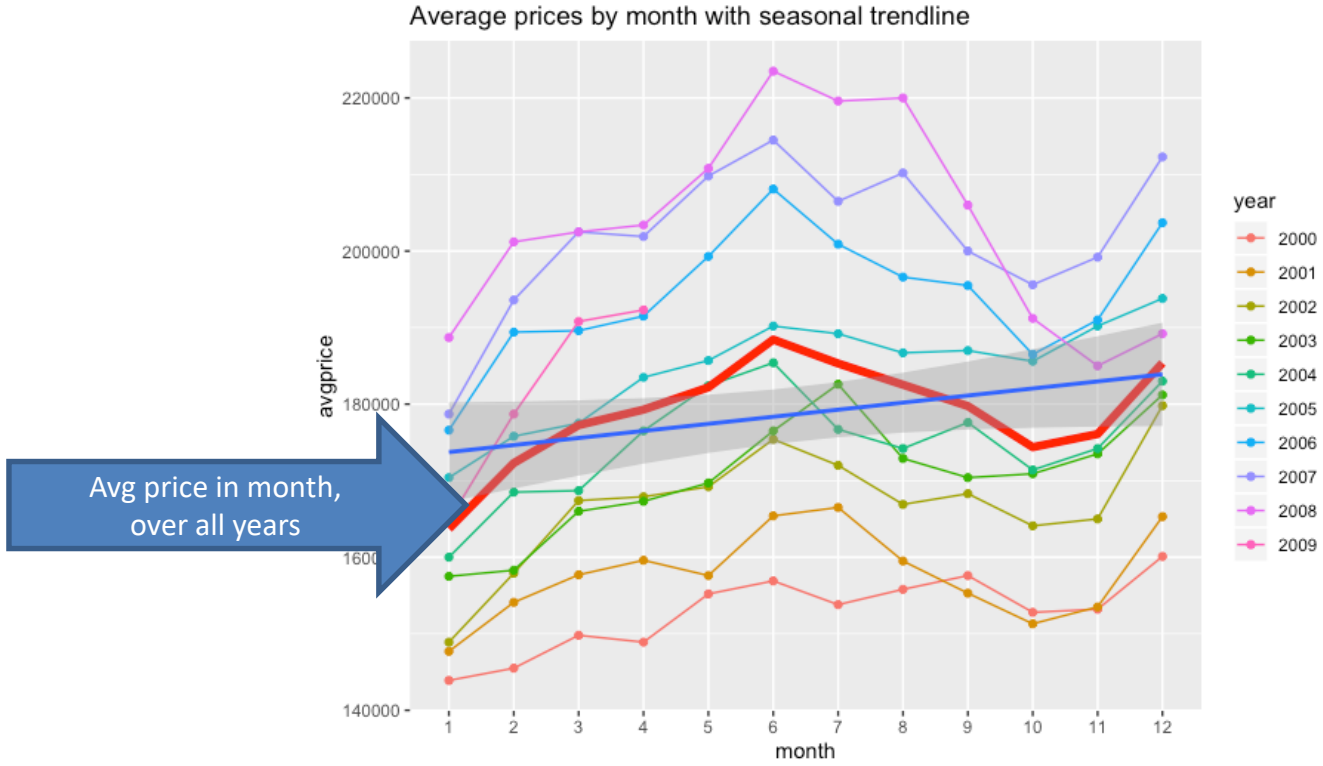



```
ggplot(houston, aes(date, avgprice)) + geom_point() + geom_line()
        + ggtitle("Avg house price in Houston by date")
```

Basic time series analysis

- Understanding the relationship of present values of a time-varying variable with past or future values of the same variable.
- Correlated observations in time series:
  - Global trends
  - Seasonal or periodic patterns

Simple way of looking for seasonal variation: Group by year, plot with x-axis = month




```
avg <- stat_summary(aes(group = 1), fun.y = "mean", geom = "line",
colour = "red", size = 2, na.rm = TRUE)

line <- geom_smooth(aes(group = 1),method=lm) #Looking for linear trend
ggplot(houston, aes(month, avgprice, group = year)) +
  geom_line(aes(colour = year)) +geom_point(aes(colour = year))+
  avg+ line + ggtitle("Average prices by month with seasonal trendline")
```

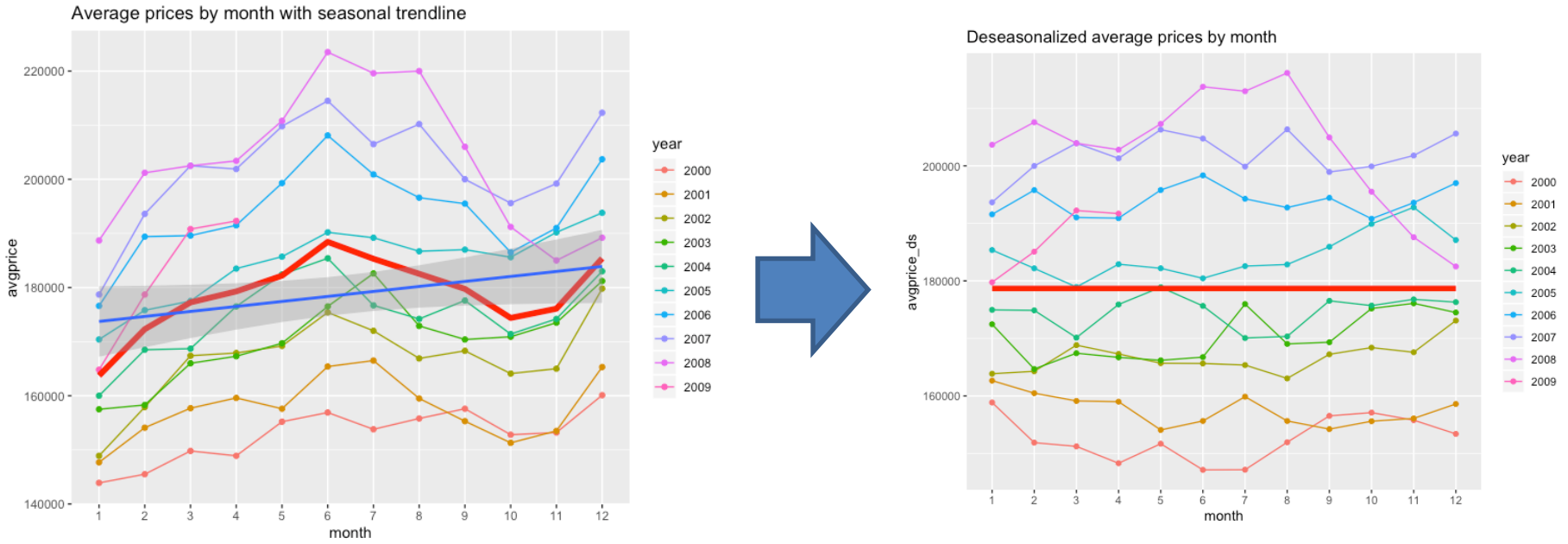
Using linear models to get rid of obvious pattern

- We don't care about the coefficients learned by the linear model
- We're just using it to subtract out the pattern

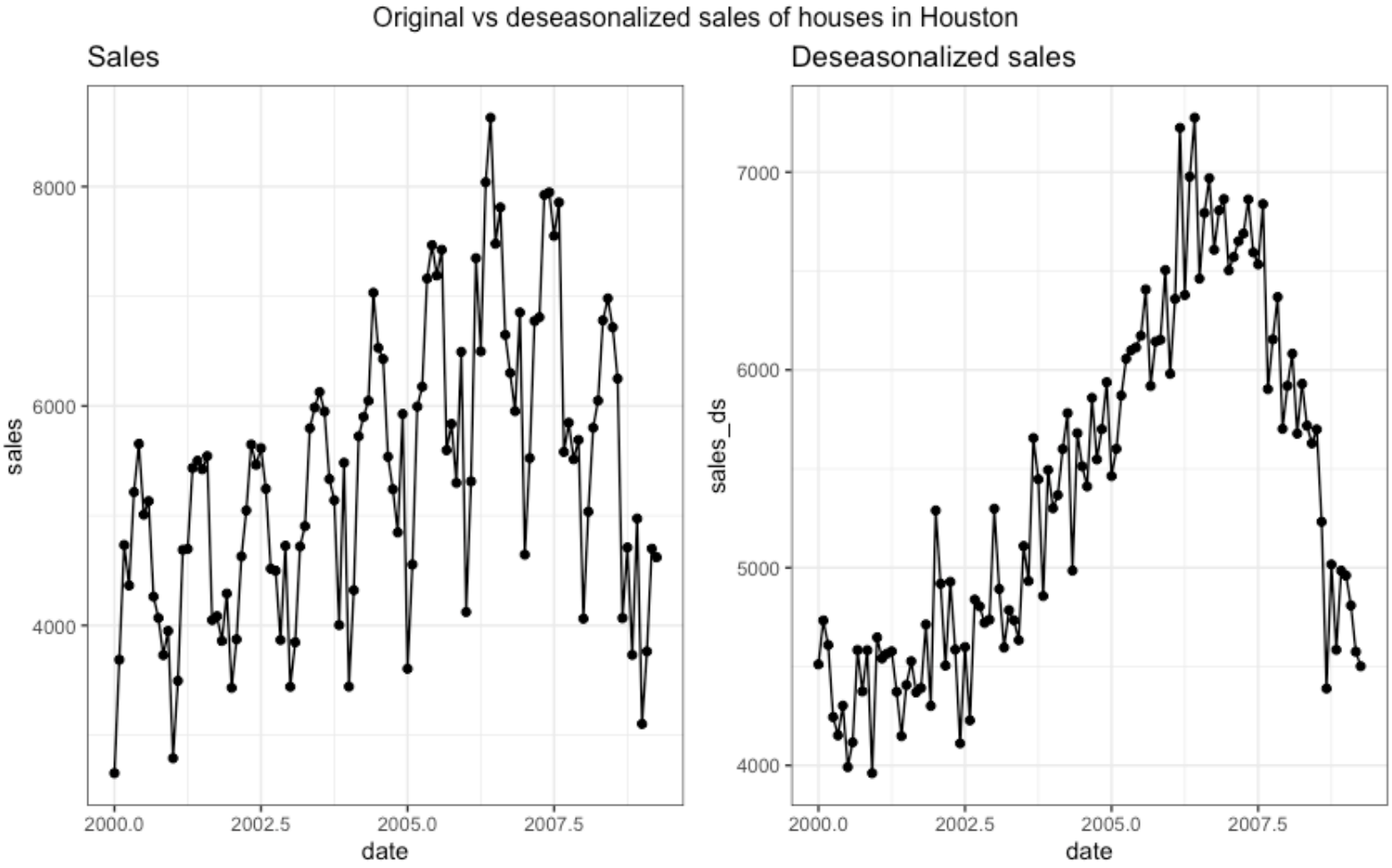
```
deseas <- function(var, month) {
    resid(lm(var ~ factor(month), na.action =
    "na.exclude")) + mean(var, na.rm = TRUE)
}

houston[ ,c("avgprice_ds", "listings_ds",
    "sales_ds", "onmarket_ds") := list(deseas(avgprice, month),
                                       deseas(listings, month),
                                       deseas(sales, month),
                                       deseas(onmarket, month))]
```

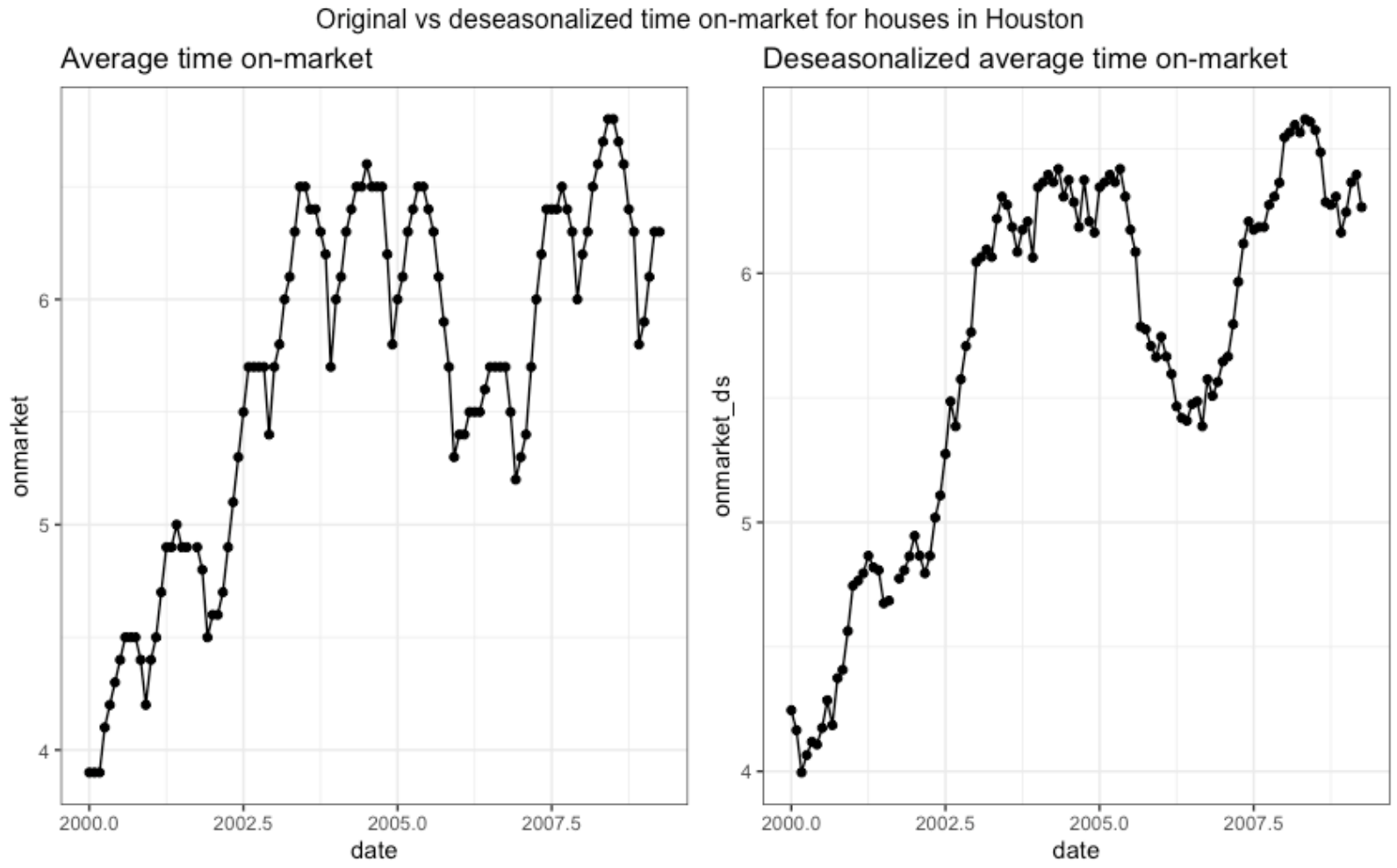
Mean line becomes flat.. After subtracting out mean variation



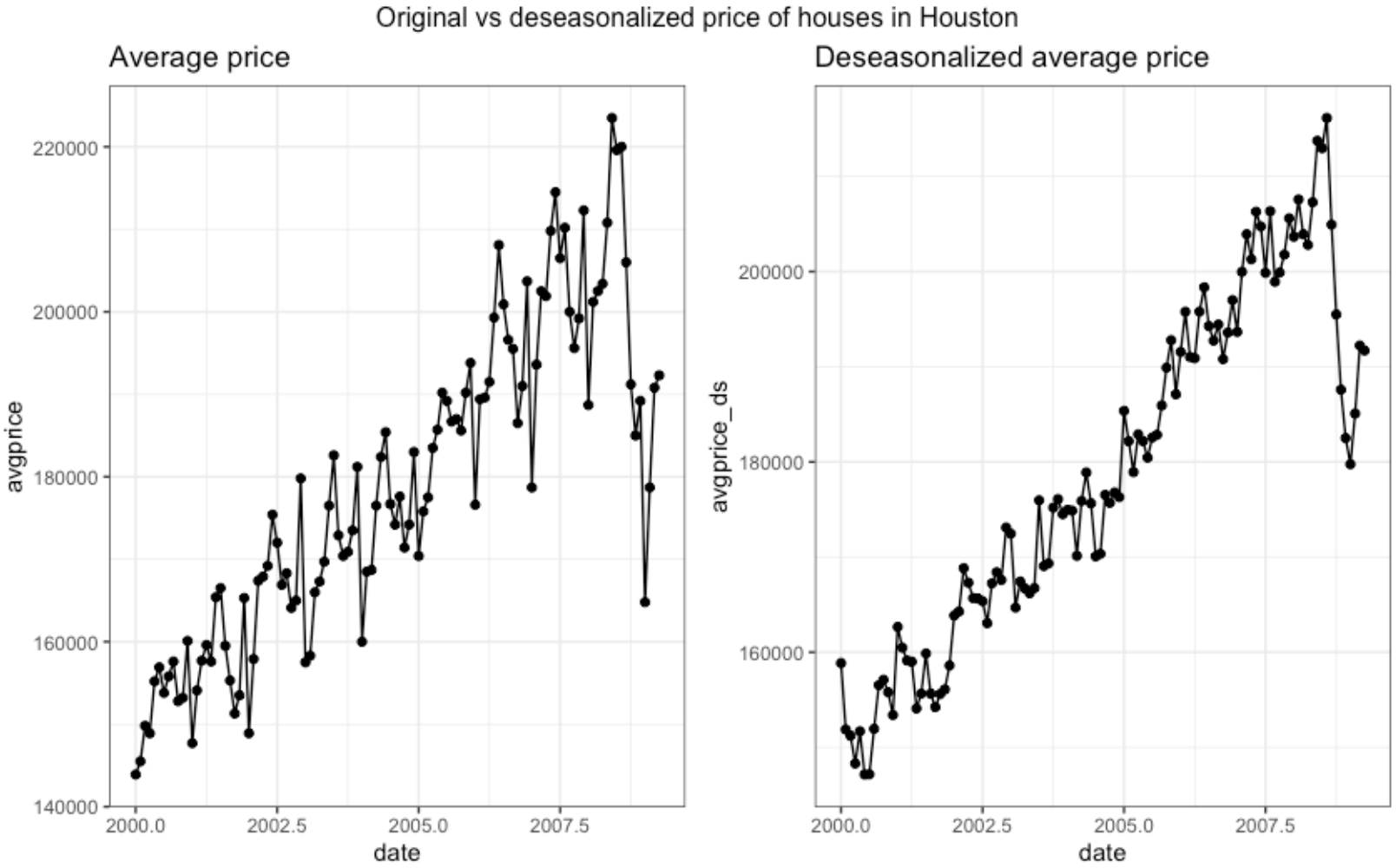
Sales started slowing mid-2006, big drop mid-2008



House listings begin slowing at the start of 2006

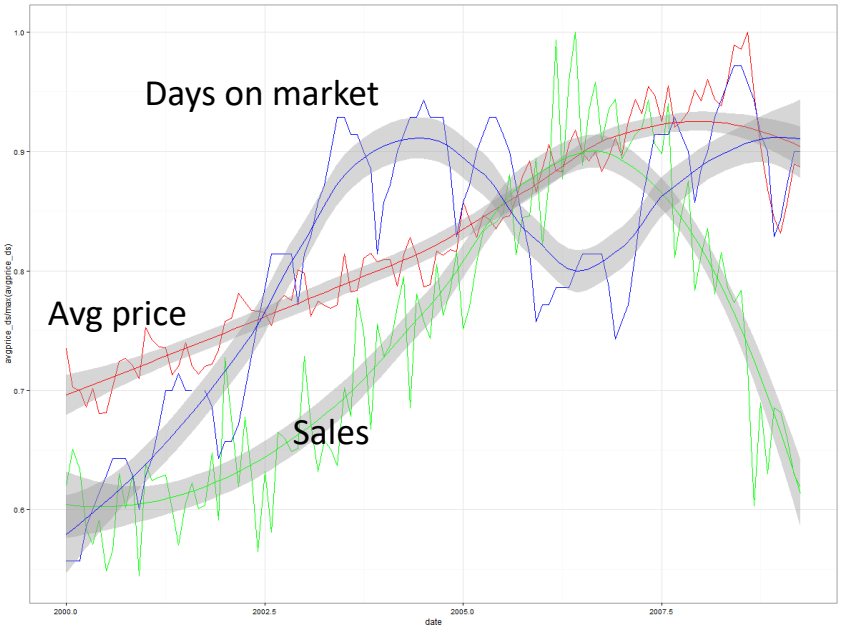


Average price took big hit in mid-late 2008



Examples of hypotheses from exploratory analysis

- Number of sales falling and days on market increasing because people uncertain about buying a new house.
- Avg price not falling as rapidly because people now overvalue their current home.
- Eventually as house stays on market for longer (~6 mo), they drop the price in an effort to sell.

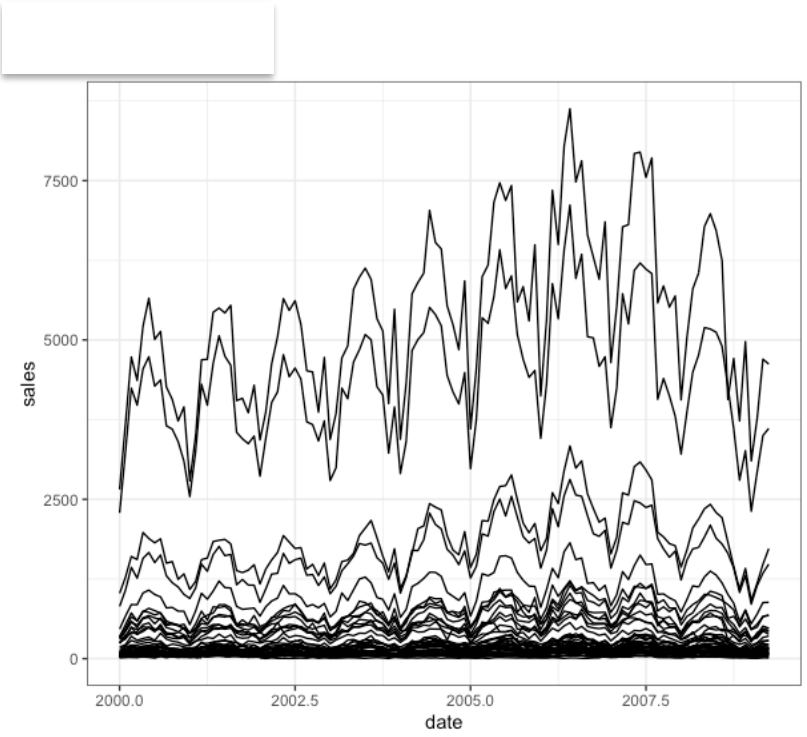


We know from our exploration of Houston data that many of the series have strong seasonal components.

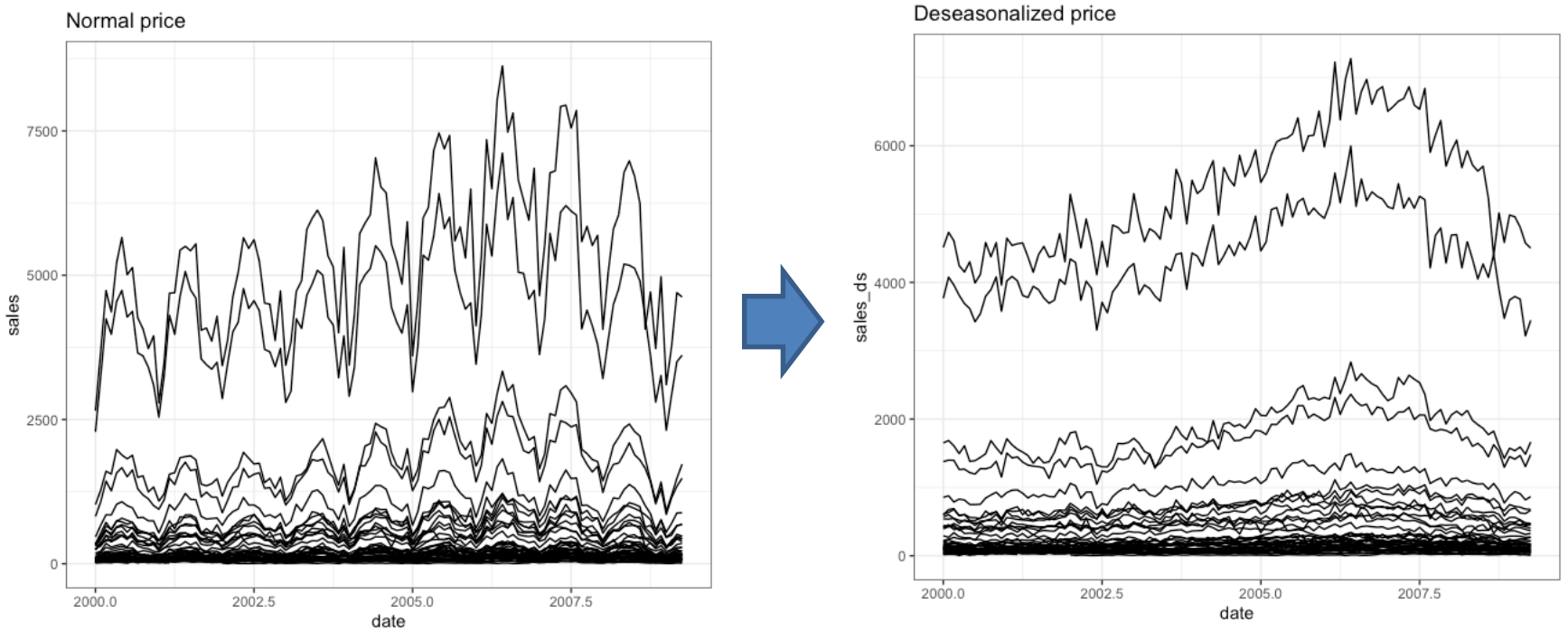
- It's a good idea to check that's true for all cities. Start with sales:

```
tx <- read.csv("tx-house-sales.csv")
qplot(date, sales, data = tx, geom = "line", group = city)
```

Looks promising, but big variation in the scale of absolute sales numbers



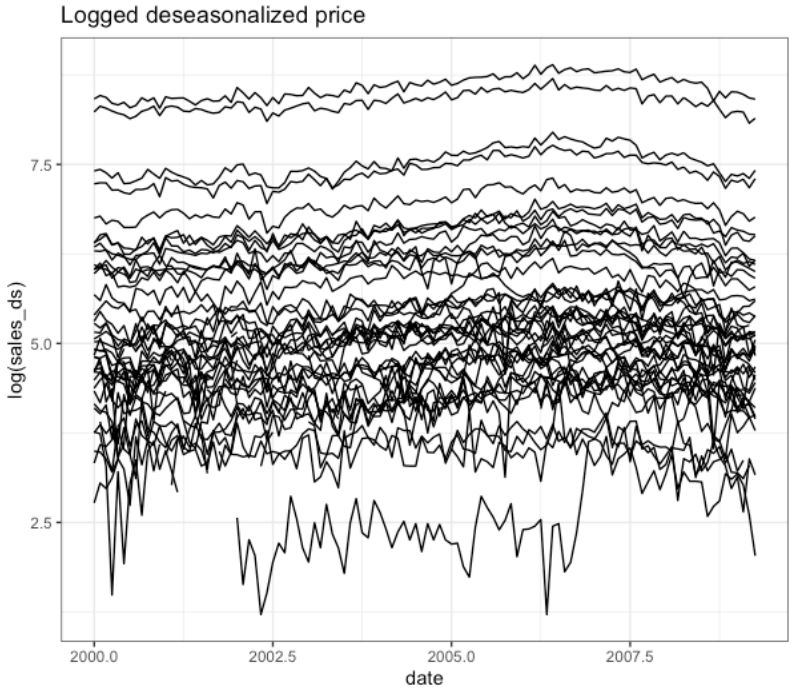
De-seasonify all cities




```
tx = houses[ ,c("avgprice_ds", "listings_ds",
                "sales_ds", "onmarket_ds") := list(deseas(avgprice, month),
                                                   deseas(listings, month),
                                                      deseas(sales, month),
                                                      deseas(onmarket, month)), by=city]

qplot(date, sales_ds, data = tx, geom = "line", group = city, main = "Deseasonalized price")
```

Then zoom in using log(y) scale transform




```
qplot(date, log(sales_ds), data = tx, geom = "line", group = city,
main = "Logged deseasonalized price")
```

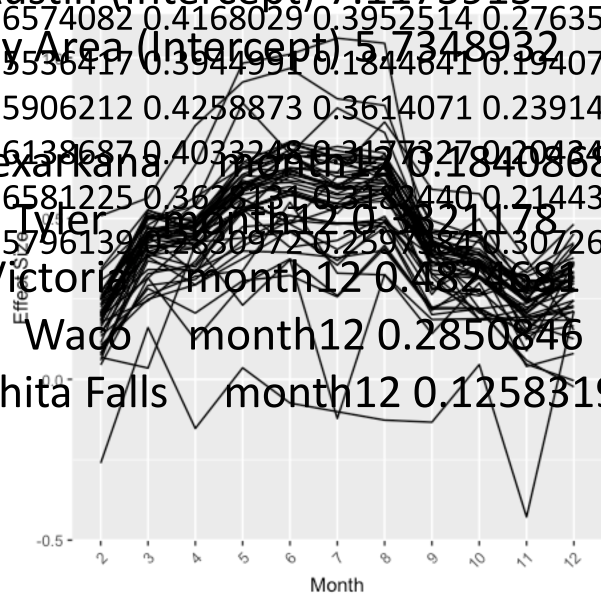
Do the same model fitting we did for Houston, but for all cities

```
coefs = tx[,as.list(coef(lm(log(sales) ~ month))),by=city]
coef.melt = melt(coefs)

p = ggplot(coef.melt[variable!="(Intercept)"],aes(x=variable,y=value,
group =city)) + geom_line()
p = p + scale_x_discrete("Month",
labels=c("2","3","4","5","6","7","8","9","10","11","12"))
p = p + ylab("Effect Size")
p = p + theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
> head(coefs)
        city (Intercept)    month2    month3    month4    month5    month6    month7
1:   Abilene    4.494573 0.2721315 0.4668759 0.4896370 0.6103430 0.6577483 0.6417541
2:  Amarillo    5.068099 0.2506584 0.3849041 0.4084202 0.5279375 0.5719820 0.5443326
3: Arlington    5.696738 0.1938864 0.4555178 0.4688507 0.6062277 0.6119491 0.5728290
4:    Austin    7.117592 0.1628227 0.3969885 0.4390155 0.5834722 0.6361164 0.6082152
5:  Bay Area    5.734893 0.2587622 0.4538445 0.4892976 0.6375921 0.6946926 0.6285423
6:  Beaumont    4.753063 0.5761817 0.4492423 0.4536311 0.5628283 0.4837094 0.5336239
      month8    month9   month10   month11   month12
1: 0.6574082 0.4168029 0.3952514 0.2763562 0.2835502
2: 0.5536417 0.3944991 0.1844641 0.1940762 0.2092959
3: 0.5906212 0.4258873 0.3614071 0.2391424 0.2970186
4: 0.6138687 0.4033248 0.3177327 0.2043451 0.3209600
5: 0.6581225 0.3626131 0.3182440 0.2144378 0.3609763
6: 0.5796139 0.2830972 0.2597184 0.3072626 0.3051792
```

```
> coef.melt
              city    variable     value
  1:       Abilene (Intercept) 4.4945734
  2:      Amarillo (Intercept) 5.0680987
  3:     Arlington (Intercept) 5.6967383
  4:        Austin (Intercept) 7.1175915
  5:      Bay Area (Intercept) 5.7348932
 ---
536:     Texarkana     month12 0.1840868
537:         Tyler     month12 0.3821178
538:      Victoria     month12 0.4824681
539:          Waco     month12 0.2850846
540: Wichita Falls     month12 0.1258319
```



What are the outliers? Use facets


```
p = ggplot(coef.melt[variable != "(Intercept)"],aes(x=variable, y=value, group=city)) + geom_line()
p = p + facet_wrap(~city, nrow=5)
p = p + ggtitle("Mean effect of month on house sales per texas city")
p = p + scale_x_discrete("Month", labels=c("2","3","4","5","6","7","8","9","10","11","12"))
p = p + ylab("Effect Size\n")
p = p + theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

What you should know

- Finding residuals from linear models
- Autocorrelation to find seasonal patterns
- How to construct and use contingency tables