An introduction to R

Thomas Lumley

Dept of Biostatistics,
University of Washington

R Core Development Team

AAPOR — Florida — 2009–5–15

What is R: marketing

- R is a free implementation of a dialect of the S language, the interactive statistics and graphics environment developed at Bell Labs.

- R/S are probably the most widely used software for research in statistical methodology and in genomics, and is popular in financial modelling and medical statistics.

- John Chambers won the 1999 ACM Software Systems award for S, which *will forever alter the way people analyze, visualize, and manipulate data*.

- Ross Ihaka won the Royal Society of New Zealand's 2008 Pickering Medal, recognizing *excellence and innovation in the practical application of technology* for the creation of R.

What is R good at?

Apart from the price:

- Graphics: publication-quality 2-d graphics, designed based on visual perception research at Bell Labs and elsewhere

- Range of methods: In addition to many built-in features, over 2000 add-on packages are available for more specialized analyses

- Flexibility: Data analysis uses the same programming language that R is written in. There is a smooth transition from simple data analyses to customization of analyses to programming.

Why not R: Speed/memory

R (and S) are accused of being slow, memory-hungry, and able to handle only small data sets.

This is completely true.

Fortunately, computers are fast and have lots of memory. Standard laptop computers can handle tens or hundreds of thousands of observations.

Computers with 32Gb memory or more to handle tens of millions of observations are still expensive, but the price is coming down fast. Tools for interfacing R with databases allow very large data sets, but this isn't transparent to the user.

Why not R: commercial support

There are companies supplying support and/or consulting services, but they are mostly new and small.

The mailing lists provide better support on average than most software vendors, but there are no guarantees (and they don't have to be polite to you if you ask lazy questions).

Why not R: Too Hard

The problem with a system that "will forever alter the way people analyze, visualize, and manipulate data" is that you have to alter the way you analyze, visualize, and manipulate data.

- No built-in pointy-clicky analyses, although there are tools to program them

- A real programming language works differently from spreadsheet macros or SAS/Stata macros.

- The system is large, and parts of it may use terminology from different areas of statistics

Outline

- Getting data in and out, some simple data analysis and graphics

- A brief look at the survey package, for reweighting and design-based inference.

- odfWeave/Sweave for reports

Reading data

- Text files
- Stata datasets
- Web pages
- (Databases)

Much more information is in the Data Import/Export manual.

Reading text data

The easiest format has variable names in the first row

```
 case    id gender    deg yrdeg field startyr year    rank admin
    1     1      F Other    92 Other      95   95 Assist     0
    2     2      M Other    91 Other      94   94 Assist     0
    3     2      M Other    91 Other      94   95 Assist     0
    4     4      M   PhD    96 Other      95   95 Assist     0
```

and fields separated by spaces. In R, use

```
salary <- read.table("salary.txt", header=TRUE)
```

to read the data from the file `salary.txt` into the data frame `salary`.

Syntax notes

- Spaces in commands don't matter (except for readability), but Capitalisation Does Matter.

- `TRUE` (and `FALSE`) are logical constants

- Unlike many systems, R does not distinguish between commands that do something and commands that compute a value. Everything is a function: ie returns a value.

- Arguments to functions can be named (`header=TRUE`) or unnamed (`"salary.txt"`)

- A whole data set (called a `data frame` is stored in a variable (`salary`), so more than one dataset can be available at the same time.

Reading text data

Sometimes columns are separated by commas (or tabs)

```
Ozone,Solar.R,Wind,Temp,Month,Day
41,190,7.4,67,5,1
36,118,8,72,5,2
12,149,12.6,74,5,3
18,313,11.5,62,5,4
NA,NA,14.3,56,5,5
```

Use

```
ozone <- read.table("ozone.csv", header=TRUE, sep=",")
```

or

```
ozone <- read.csv("ozone.csv")
```

Syntax notes

- Functions can have optional arguments (`sep` wasn't used the first time). Use `help(read.table)` for a complete description of the function and all the arguments.

- There's more than one way to do it.

- `NA` is the code for missing data. Think of it as "Don't Know". R handles it sensibly in computations: eg 1+NA, NA & FALSE, NA & TRUE. You cannot test `temp==NA` (Is temperature equal to some number I don't know?), so there is a function `is.na()`.

Data from other packages

Data from the American National Election Studies, 2006 pilot study, in SPSS portable format

```
>  library(foreign)
> nespanel <- read.spss("~/Downloads/NESPIL06.por")
```

- Lots of functionality in R comes in **packages**, loaded with the `library()` function.

- The **foreign** package in the standard R distribution and reads data from SPSS, Stata, SAS PROC XPORT, and some others.

The web

Files for `read.table` can live on the web

```
fl2000<-read.table("http://faculty.washington.edu/tlumley/
data/FLvote.dat", header=TRUE)
```

It's also possible to read from more complex web databases (such as the genome databases, or financial 'ticker' services).

Simple manipulation and graphs

```
> str(fl2000)
'data.frame': 67 obs. of  8 variables:
 $ GORE    : int  47365 2392 18850 3075 97318 386565 2155 29645 25525
 $ BUSH    : int  34124 5610 38637 5414 115185 177323 2873 35426 2976
 $ BUCHANAN: int  263 73 248 65 570 788 90 182 270 186 ...
 $ NADER   : int  3226 53 828 84 4470 7101 39 1461 1379 562 ...
 $ NELSON  : int  49091 3104 22914 4118 112255 377081 2809 28947 2758
 $ MCCOLLUM: int  31060 4578 33901 4699 98813 174980 2055 37026 27056
 $ LOGAN   : int  1735 50 358 92 2304 6166 31 746 948 561 ...
 $ county  : chr  "ALACHUA" "BAKER" "BAY" "BRADFORD" ...
```

Simple manipulation and graphs

```
> summary(fl2000)
      GORE              BUSH            BUCHANAN            NADER
 Min.   :   789   Min.   :  1317   Min.   :   9.0   Min.   :   19.0
 1st Qu.:  3058   1st Qu.:  4757   1st Qu.:  46.5   1st Qu.:   95.5
 Median : 14167   Median : 20206   Median : 120.0   Median :  562.0
 Mean   : 43435   Mean   : 43439   Mean   : 260.8   Mean   : 1453.9
 3rd Qu.: 46015   3rd Qu.: 56546   3rd Qu.: 285.5   3rd Qu.: 1870.5
 Max.   :386565   Max.   :289492   Max.   :3407.0   Max.   :10022.0
    MCCOLLUM          LOGAN            county
 Min.   :   948   Min.   :   27   Length:67
 1st Qu.:  3757   1st Qu.:  110   Class :character
 Median : 18934   Median :  392   Mode  :character
 Mean   : 40352   Mean   : 1203
 3rd Qu.: 52503   3rd Qu.: 1242
 Max.   :264801   Max.   :11796
```

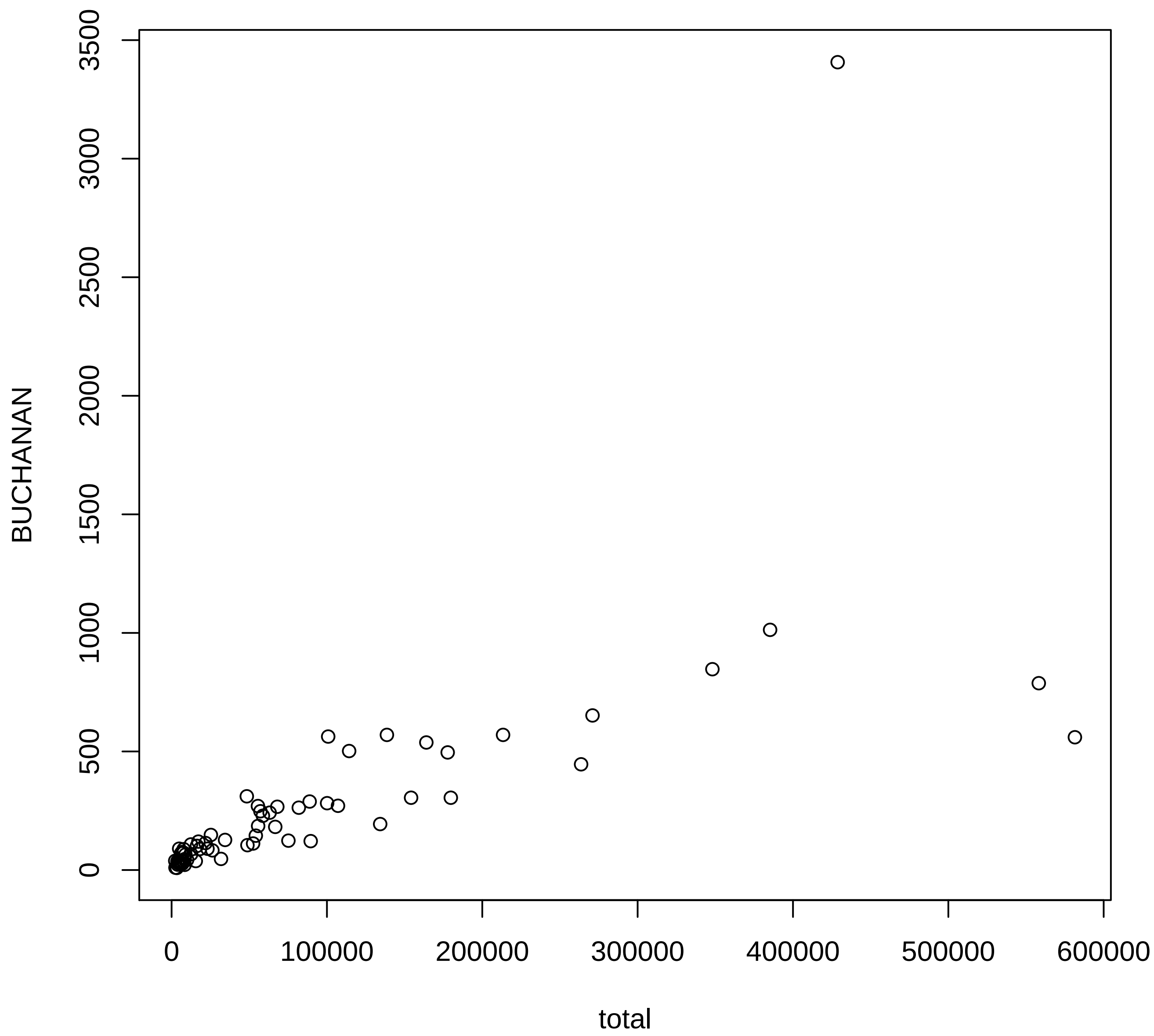
Simple manipulation and graphs

```
> fl2000$total <- with(fl2000, NELSON+MCCOLLUM+LOGAN)
> summary(fl2000$total)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   2356    7759   34430   86150  100500  581500
> plot(BUCHANAN~total, data=fl2000)
```

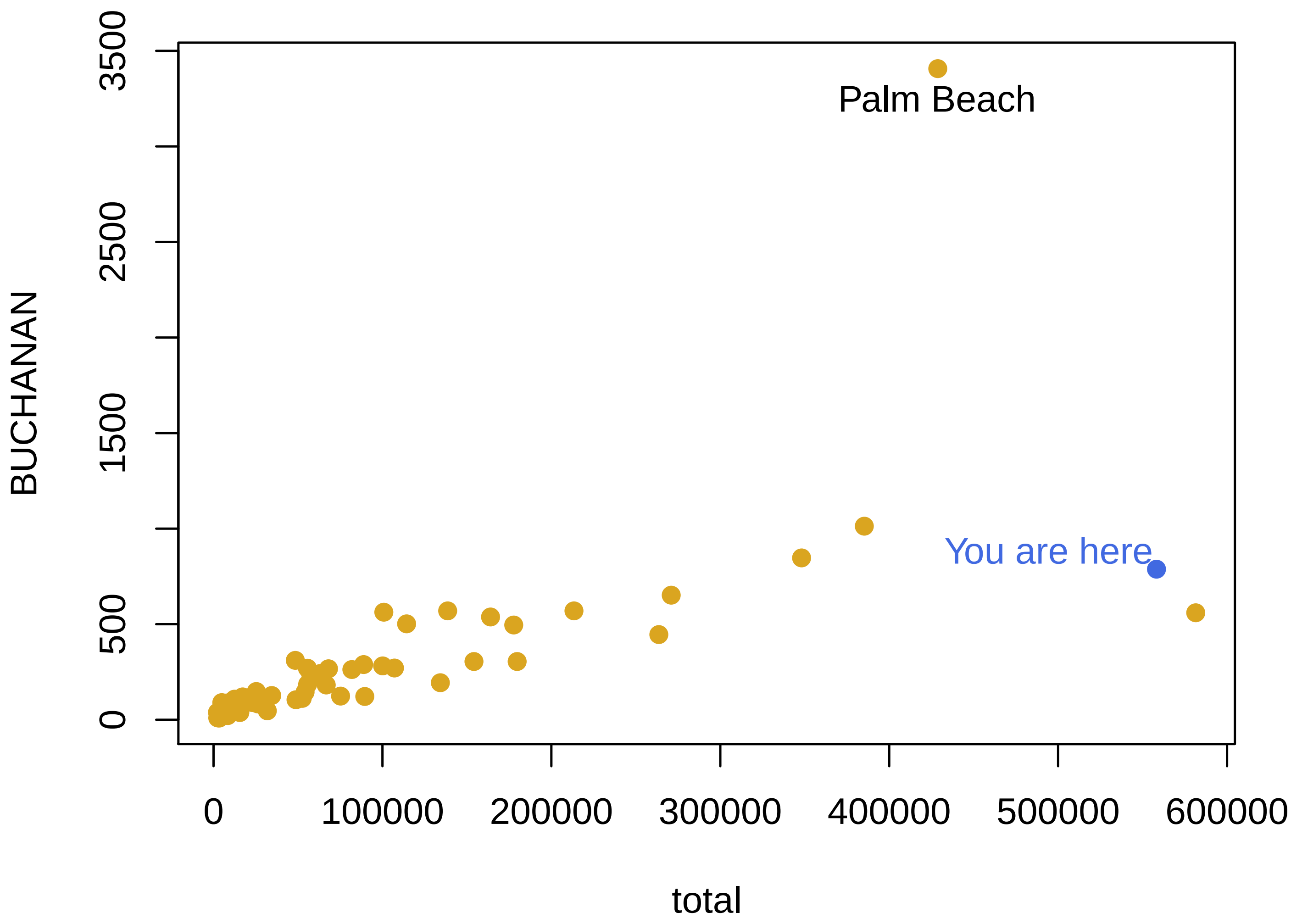
Because R can have more than one data set loaded, we need to specify which `BUCHANAN` and which `total` we mean. The `$` is like the possessive 's. `with()` explicitly specifies which data set we mean.

Many regression and graphics functions, like `plot`, take the the data set as an argument and use a **model formula** to specify variables.

Simple manipulation and graphs



Simple manipulation and graphs



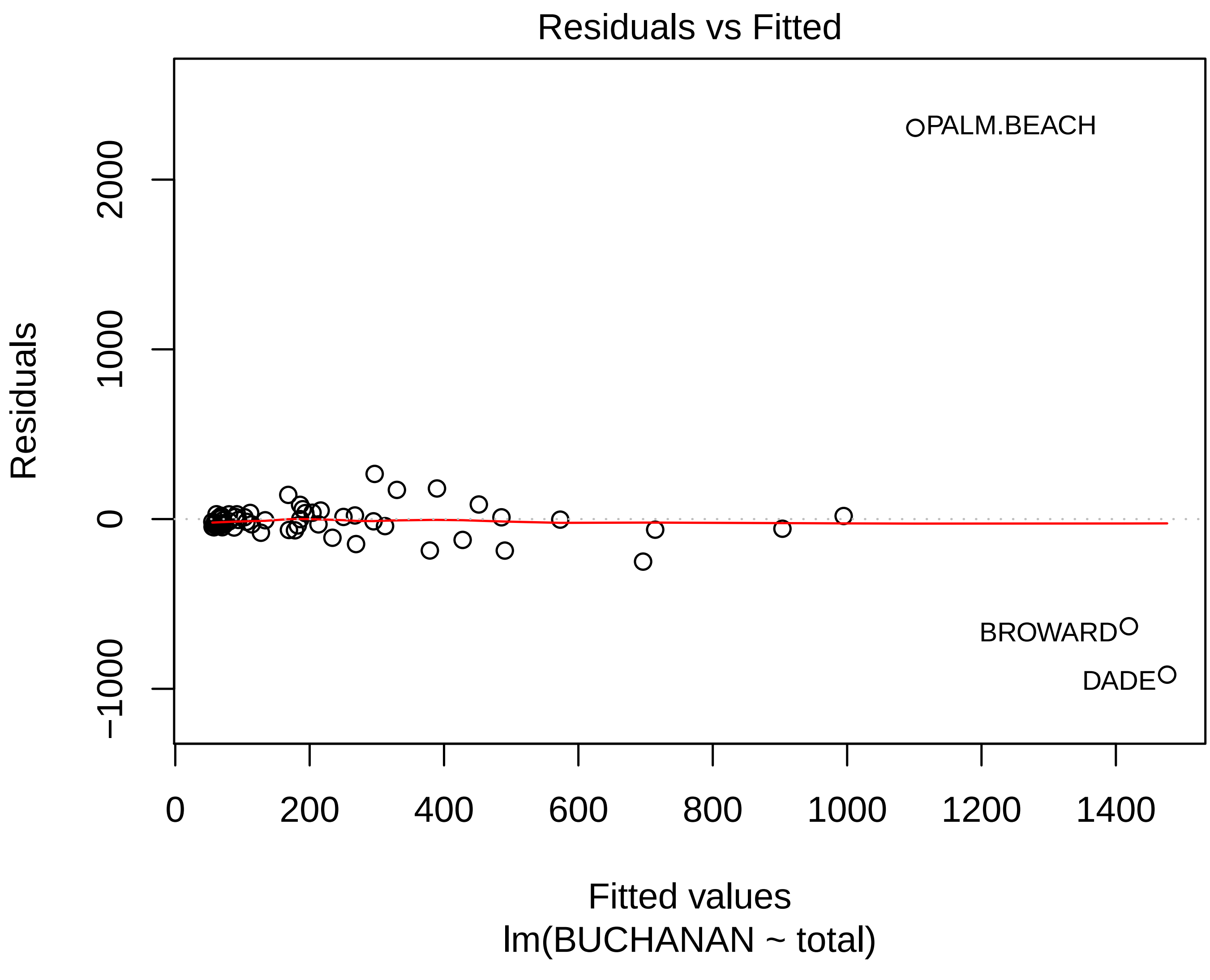
Simple manipulation and graphs

```
> plot(BUCHANAN~total,data=fl2000,
    col=ifelse(county=="BROWARD", "royalblue","goldenrod"),pch=19)
> with(fl2000, identify(total, BUCHANAN,labels=county)
> text(556155,885,"You are here",col="royalblue",adj=1)
> model <- lm(BUCHANAN~total, data=fl2000)
> plot(model)
> model2 <- update(model, .~.+I(county=="PALM.BEACH"))
> plot(model2)
```

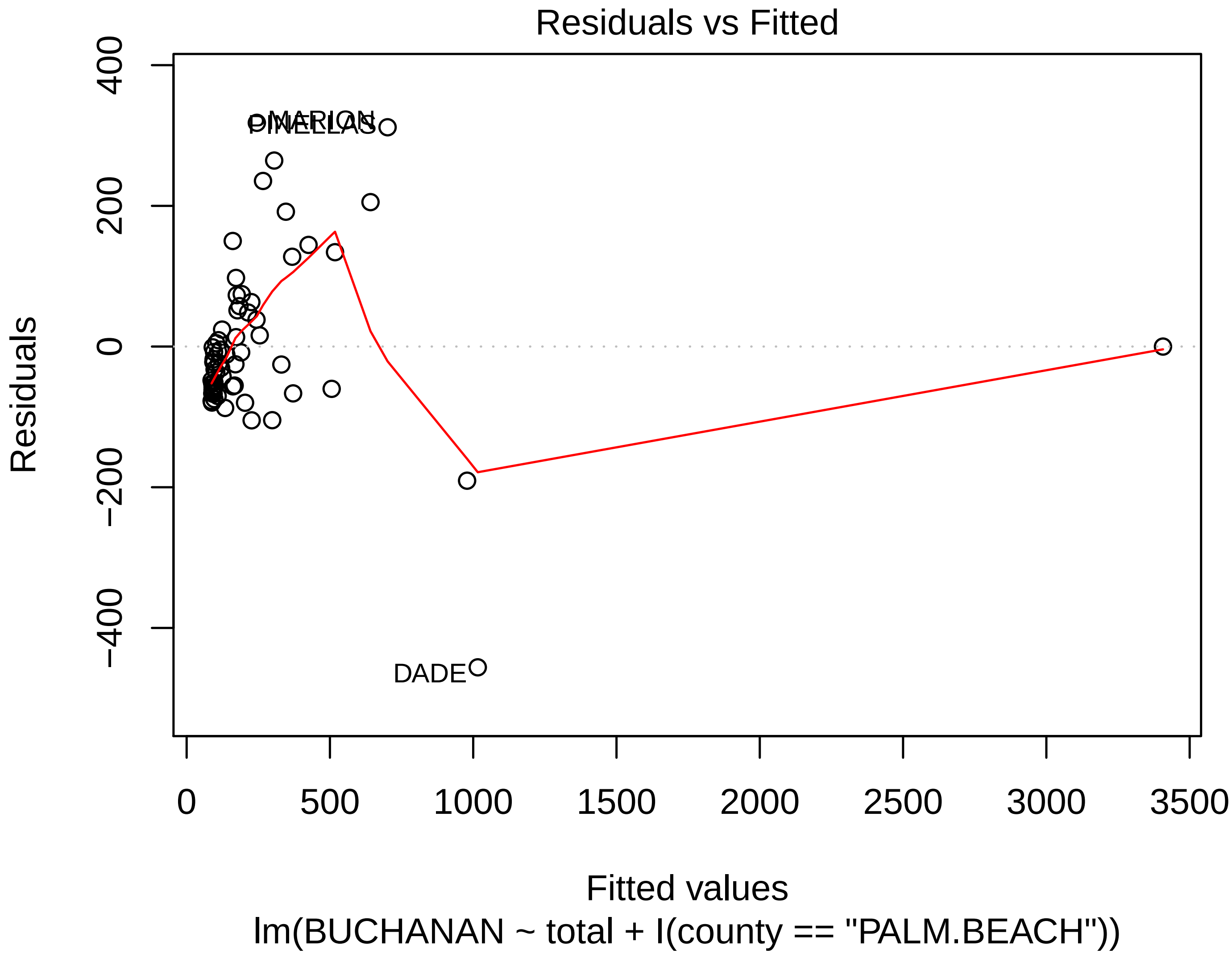
Simple manipulation and graphs

```
> summary(model2)
Call:
lm(formula = BUCHANAN ~ total + I(county == "PALM.BEACH"), data = fl2000)
Residuals:
    Min      1Q  Median      3Q     Max
-455.95  -61.98  -23.47   43.30  318.07

Coefficients:
                                  Estimate   Std. Error t value  Pr(>|t|)
(Intercept)                     83.2331534   17.5099884   4.753 0.0000117 ***
total                            0.0016041    0.0001213  13.227   < 2e-16 ***
I(county == "PALM.BEACH")TRUE 2636.0312127  125.9480666  20.930   < 2e-16 ***
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 117.8 on 64 degrees of freedom
Multiple R-squared: 0.9336,Adjusted R-squared: 0.9315
F-statistic: 449.6 on 2 and 64 DF,  p-value: < 2.2e-16
```

Simple manipulation and graphs



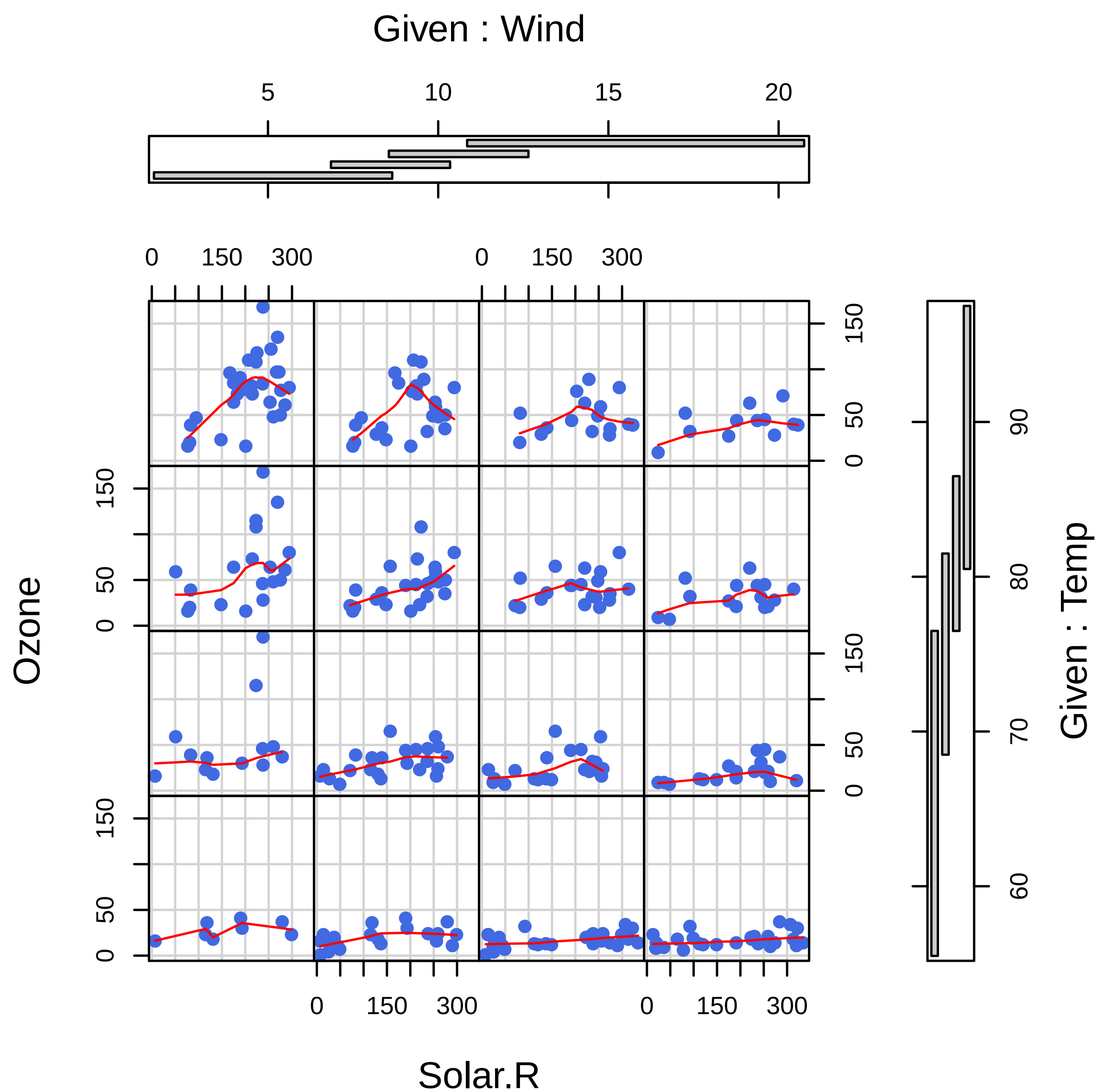
Simple manipulation and graphs



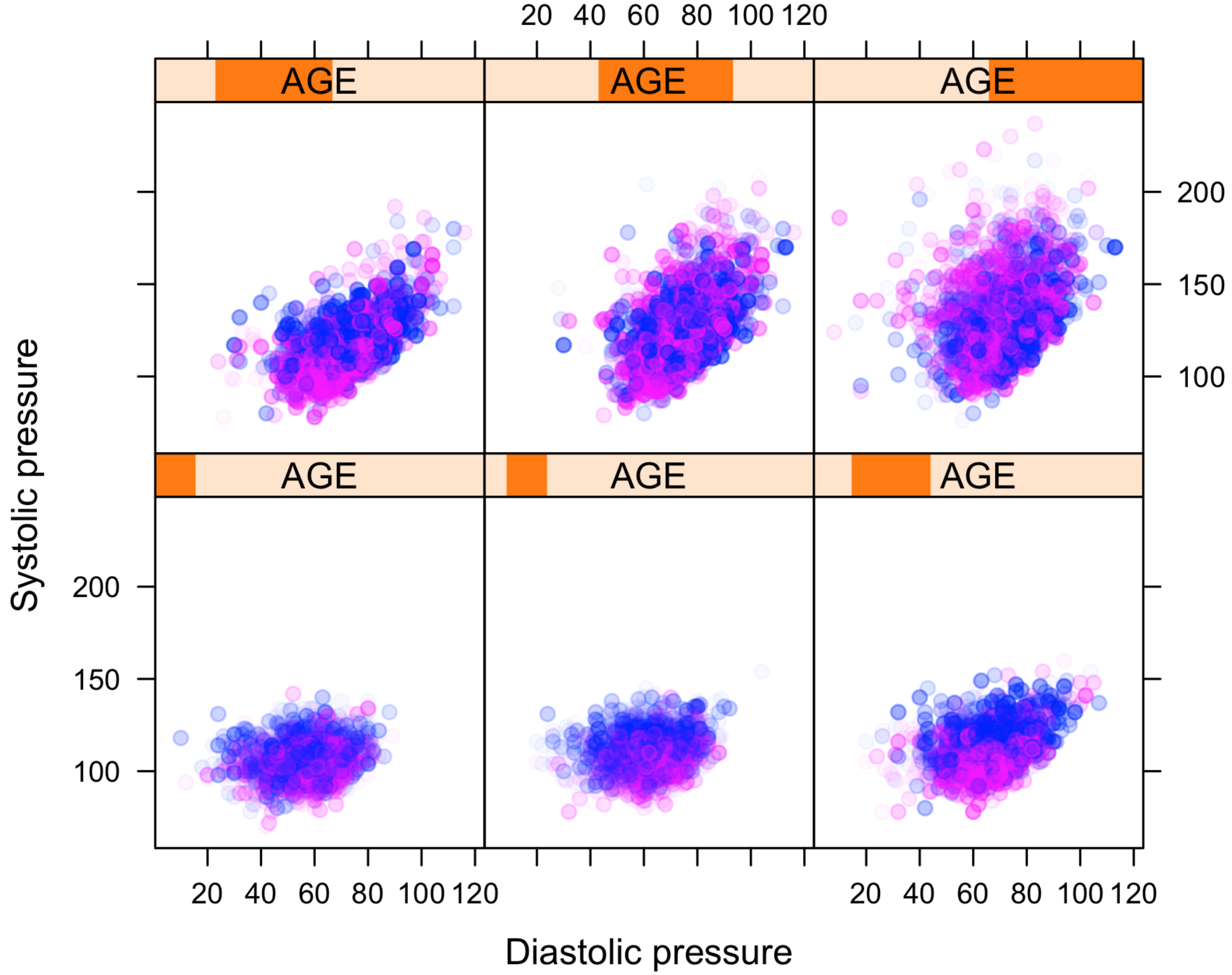
More complex graphs

```
coplot(Ozone~Solar.R|Wind*Temp,data=airquality,
   panel=panel.smooth,pch=19,col="royalblue",number=4)

svycoplot(sysbp~diabp|agegp, style="transparent",
   basecol=function(d) c("magenta","royalblue")[d$sex]
   data=nhanes_design)

spplot(states,c("age1","age2","age3","age4"),
    names.attr=c("<35","35-50","50-65","65+"))
```

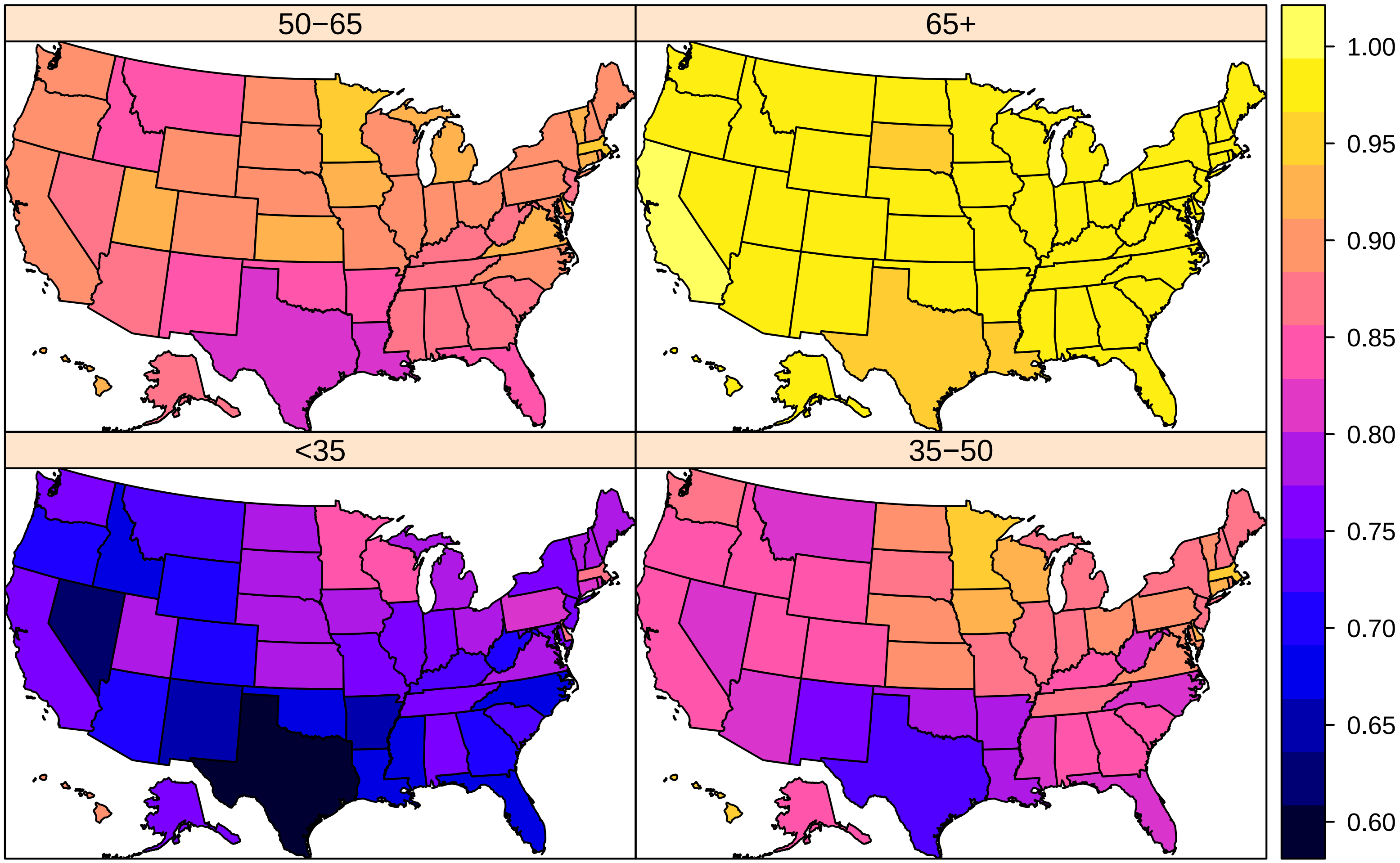
Smog and sunlight



Blood pressure by age and gender



Health insurance by age and state



Probability samples

The **survey** package analyses data from complex probability samples

- stratification, clustering, unequal probability sampling
- post-stratification, raking, calibration for non-response
- summary statistics, regression models, graphics.
- multiply imputed data
- large data via relational databases

`http://faculty.washington.edu/tlumley/survey`

Describing a sampling design

Specify clusters, strata, sampling weights, and data set. All the information is packaged into a single object.

```
shs<-svydesign(id=~psu, strata=~stratum, weight=~grosswt,
    data=shs_data)
brfss <- svydesign(id=~X_PSU, strata=~X_STATE, weight=~X_FINALWT,
   data="brfss", dbtype="SQLite", dbname="brfss07.db", nest=TRUE)
```

Multistage sampling, finite population corrections, PPS sampling, replicate weights, are also supported.

SHS is a subset of the Scottish Household Survey. It fits in memory easily.

BRFSS is the 2007 Behavioral Risk Factor Surveillance System data, the world's largest telephone survey. It lives in a database.

Simple summaries

```
> brfss <- update(brfss, insured=(HLTHPLAN==1))
> svymean(~insured, brfss)
                mean     SE
insuredFALSE 0.15622 0.0015
insuredTRUE  0.84378 0.0015
> svymean(~insured, subset(brfss, X_STATE==48))
                mean     SE
insuredFALSE 0.26062 0.0057
insuredTRUE  0.73938 0.0057
> for_map <- svyby(~insured, ~agegp+X_STATE, svymean, design=brfss)
```

Similarly, `svytotal()`, `svyquantile()`, `svyvar()`, `svykappa()`

Testing

```
> svychisq(~X_AGE_G+insured,brfss)

Pearson's X^2: Rao & Scott adjustment

data:  NextMethod("svychisq", design)
F = 817.7281, ndf = 3.987, ddf = 1717617.068, p-value < 2.2e-16
```

Also `svyttest()` for one-sample and two-sample t-test.

Regression models

Modelling internet use in Scotland (2001) by logistic regression on age, sex, and income

```
> model <- svyglm(intuse~I(age-18)*sex+groupinc, design=shs,  family=binomial)
> summary(model)
Call:
svyglm(intuse ~ I(age - 18) * sex + groupinc, design = shs, family = binomial)
Survey design:
svydesign(id = ~psu, strata = ~stratum, weight = ~grosswt, data = ex2)

Coefficients:
                     Estimate Std. Error t value        Pr(>|t|)
(Intercept)          0.258307   0.120749   2.139          0.0324 *
I(age - 18)         -0.039431   0.001549 -25.448         < 2e-16 ***
sexfemale           -0.066039   0.066869  -0.988          0.3234
groupincunder 10K   -0.612557   0.117055  -5.233 0.000000169627 ***
groupinc10-20K      -0.040161   0.112927  -0.356          0.7221
groupinc20-30k       0.708368   0.114609   6.181 0.000000000659 ***
groupinc30-50k       1.665127   0.119688  13.912         < 2e-16 ***
groupinc50K+         2.265943   0.167362  13.539         < 2e-16 ***
I(age - 18):sexfemale -0.011199   0.002131  -5.255 0.000000150794 ***
```

Regression models

```
> regTermTest(model,~groupinc)
Wald test for groupinc
 in svyglm(intuse ~ I(age - 18) * sex + groupinc, design = shs, family = binomial)
Chisq =  1886.269  on  5  df: p= < 2.22e-16
```

`svyglm()` fits linear and generalized linear models, `svyolr()` fits ordinal models, `svyloglin()` fits loglinear models, `svycoxph()` fits the Cox proportional hazards model.

Reweighting

Toy example: standardized testing data on California schools

```
> svytotal(~enroll, dclus1)
         total     SE
enroll 3404940 932235
> pop.types
stype
   E    H    M
4421  755 1018
> dclus1p <-postStratify(dclus1, ~stype, pop.types)
> svytotal(~enroll, dclus1p)
         total     SE
enroll 3680893 406293
```

`rake()` will rake on multiple categorical variables, `calibrate()` does calibration aka generalized raking.

Reweighting

```
>  svymean(~api00, dclus1)
        mean     SE
api00 644.17 23.542
> sum(apipop$api99)
[1] 3914069
> dclus1_cal <- calibrate(dclus1, ~api99,
     pop=c(`(Intercept)`=6194, api99=3914069))
> svymean(~api00, dclus1_cal)
        mean     SE
api00 666.72 3.2959
```

In practice the goal is bias reduction, not variance reduction, but the computations are the same.

Generating reports with Sweave

Sweave is a reporting system that takes a file of text+code, runs the code, and puts the resulting output back into the file.

The original version works with LaTeX code, the odfWeave package supplies a version that works with OpenOffice (and so with new enough versions of MS Office).

Designed for ’reproducible data analysis’ and for documentation: since the output comes from processing the code in the input file, the output has to match the code.

Do example if time permits.