Analysis

Paul E. Johnson[1]

[1]Center for Research Methods and Data Analysis, University of Kansas

2018

Outline

Outline

This is a brief overview

- Using R (R Core Team, 2017) for analysis is a huge topic
- In following days of the workshop, we investigate analysis methods in considerably more detail

Outline

How is X related to Y?

- A predictor and an outcome.
- A simple theory: temperature in Oregon depends on the elevation above sea level.

$$tann_i = \beta_0 + \beta_1 * elevation_i + e_i$$

I'll get that Oregon temperature file again

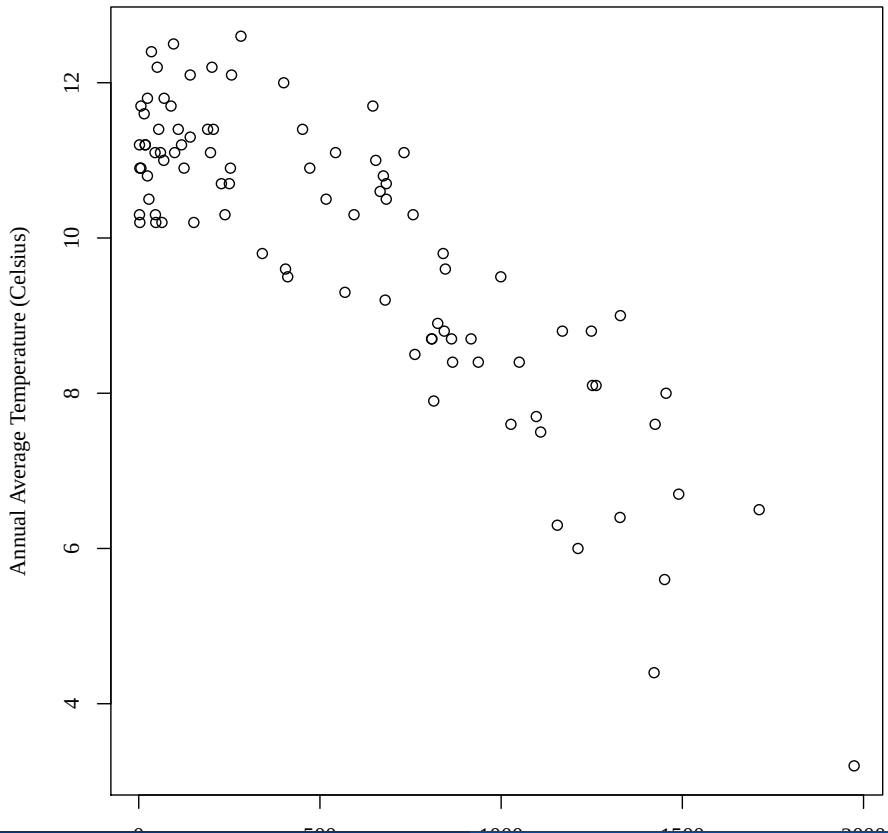
```
dat1 <- read.table("data/oregon.csv", sep = ",",
    header = TRUE, stringsAsFactors = FALSE)
head(dat1)
```

```
  station latitude longitude elevation tann
1     ANT   44.917  -120.717       846  9.6
2     ARL   45.717  -120.200        96 12.5
3     ASH   42.217  -122.717       543 11.1
4     AST   46.150  -123.883         2 10.3
5     BKA   44.833  -117.817      1027  7.6
6     BKK   44.783  -117.833      1050  8.4
```

Check The Scatterplot!

```
plot(tann ~ elevation, data = dat1,
    xlab="Elevation above sea level",
    ylab="Annual Average Temperature (Celsius)",
    main="")
```

- The "formula" interface, "$y \sim x$". Same is used in regression estimation.
- Because both variables are numeric, R makes a standard "scatterplot".



A Simple One-Predictor Regression

- The linear regression function is "lm"

```
mod1 <- lm (tann ~ elevation, data = dat1)
```

- If that command succeeds, it generates no "output". Nothing is written out, the command prompt simply returns.
- Part of the R philosophy is that we should interact with the model, ask it for info, make plots of it, etc

Follow-up 1: summary

```
summary(mod1)
```

```
Call:
lm(formula = tann ~ elevation, data = dat1)

Residuals:
    Min      1Q  Median      3Q     Max
-2.6841 -0.6026 -0.1081  0.7613  2.1034

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 11.6881385  0.1502978   77.77   <2e-16 ***
elevation   -0.0032377  0.0002016  -16.06   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9582 on 90 degrees of freedom
Multiple R-squared:  0.7413,  Adjusted R-squared:  0.7385
F-statistic: 257.9 on 1 and 90 DF,  p-value: < 2.2e-16
```

Digression: Generic Functions

- Note: the `summary()` function does something very different with `mod1` than it did with a data frame.

Generic Function: A function for which "methods" have been written to customize R's response to input objects.

`summary()` is one of the generic functions because there are specialized method implementation named `summary.lm`.

- An invention due to the ATT S development team in the 1980s.

What goes on inside the R Runtime engine?

- When you run `summary(mod1)`, the R runtime system says to itself
  - Hmm. I've got `summary.default()`, and I also I have `summary.lm()` and `summary.data.frame()`. And `summary.anova()` and `summary.glm()`. And ....
  - Which do they expect me to use? Its a puzzler!
  - Let's ask the object itself for a hint

```
class(mod1)
```

```
[1] "lm"
```

  - Aha! It is an "lm" object, that means I use `summary.lm()` to display that.
- If you happen to have `myUnknownObject`, then R will use `summary.default()` when you run `summary(myUnknownObject)`.

What goes on inside the R Runtime engine? ...

- An explanation of how objects are assigned into classes, and where these "method functions" like `summary.lm()` come from, is left for another day.

summary creates an object

```
mod1sum <- summary(mod1)
```

- And running the function `coef()` does something different with the summary object and the object itself.
- Observe `coef(mod1)` simply splats out the regression coefficients

```
coef(mod1)
```

```
 (Intercept)    elevation
11.688138523 -0.003237686
```

- But the output from `coef` on the summary object is a coefficient table

```
coef(mod1sum)
```

```
                Estimate   Std. Error    t value     Pr(>|t|)
(Intercept) 11.688138523 0.1502978171  77.76652 2.556377e-84
elevation   -0.003237686 0.0002015983 -16.06009 3.638765e-28
```

summary creates an object ...

- That last one is actually calling a method `coef.summary.lm(mod1sum)` because the class of the summary object is

```
class(mod1sum)
```

```
[1] "summary.lm"
```

Follow-up 2: confidence intervals

```
confint(mod1)
```

```
                   2.5 %       97.5 %
(Intercept) 11.389545676 11.986731369
elevation   -0.003638196 -0.002837176
```

Follow-up 3: plot diagnostics

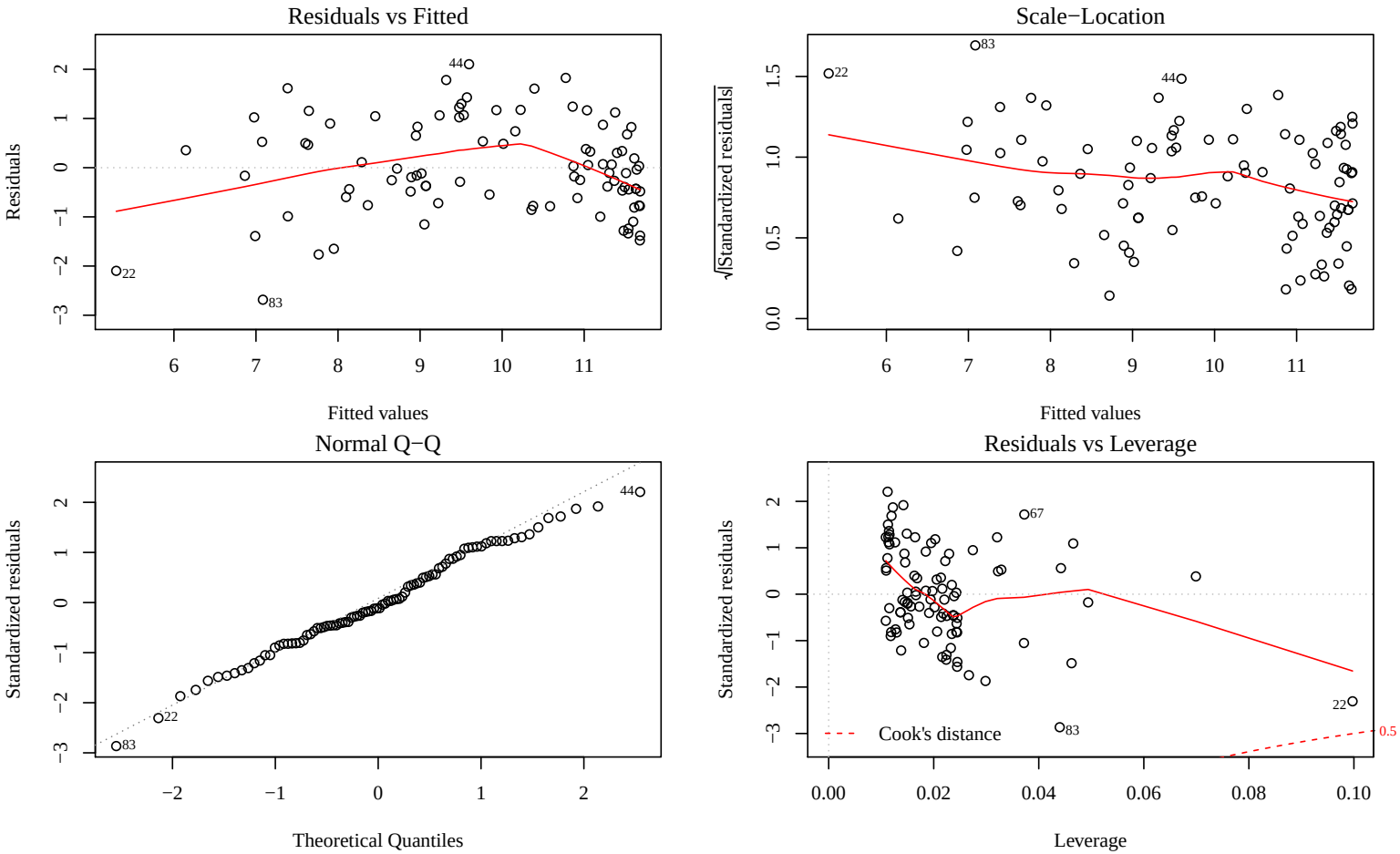
- When run interactively, the command `plot(mod1)` will pause between 4 graphs.
- This code creates a 2 x 2 plot including those 4 graphs

```
par(mfcol = c(2,2))
plot(mod1)
par(mfcol = c(1,1))
```

And then it sets the default display back to one graph per device.

- If you want fancy control of page "layout", the `layout()` function in R is recommended.

Follow-up 3: plot diagnostics ...

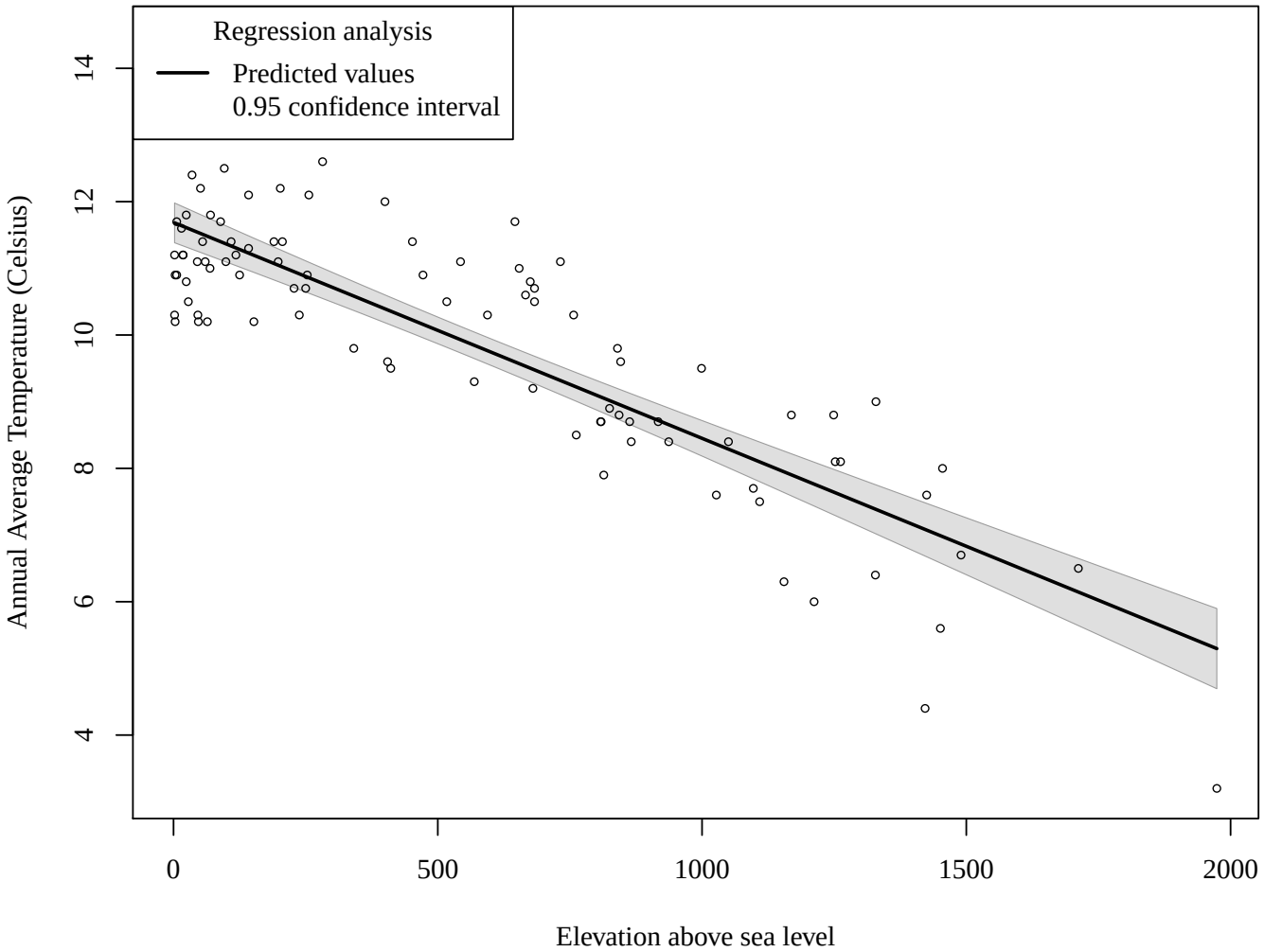


Follow-up 4: plot predictive relationship

- Because I do this all the time with my classes, I wrote a function in rockchalk called `plotSlopes()` for this purpose. I think its output is nicer.

```
library(rockchalk)
plotSlopes(mod1, plotx = "elevation", interval =
    "confidence", xlab = "Elevation above sea
    level", ylab = "Annual Average Temperature
    (Celsius)", main="")
```

Follow-up 4: plot predictive relationship



Follow-ups: many many more exist

```
> methods(class = "lm")
 [1] add1.lm*              alias.lm*             anova.lm*             case.names.lm*
 [5] confint.lm            cooks.distance.lm*    deviance.lm*          dfbeta.lm*
 [9] dfbetas.lm*           drop1.lm*             dummy.coef.lm         effects.lm*
[13] extractAIC.lm*        family.lm*            formula.lm*           hatvalues.lm*
[17] influence.lm*         kappa.lm              labels.lm*            logLik.lm*
[21] model.frame.lm*       model.matrix.lm       nobs.lm*              plot.lm*
[25] predict.lm            print.lm*             proj.lm*              qr.lm*
[29] residuals.lm          rstandard.lm*         rstudent.lm*          simulate.lm*
[33] summary.lm            variable.names.lm*    vcov.lm*
```

- I know what one-half of these are for.
- If I were you, I would understand the R "predict" method, especially its "newdata" argument. That's where most of my work in rockchalk has been concentrated.

Use a table writer to make a table

```
library(rockchalk)
or10 <- outreg(list("Temperature" = mod1))
```

	Temperature
	Estimate
	(S.E.)
(Intercept)	11.688***
	(0.150)
elevation	-0.003***
	(0.000)
N	92
RMSE	0.958
R^2	0.741

$*p < 0.05$** $p < 0.01$***$p < 0.001$

```
cat(or10)
```

Use a table writer to make a table ...

	Temperature Estimate (S.E.)
(Intercept)	11.688*** (0.150)
elevation	-0.003*** (0.000)
N	92
RMSE	0.958
R^2	0.741

$*p \leq 0.05$** $p \leq 0.01$***$p \leq 0.001$

Use a table writer to make a table

```
dat1$elevationP1000 <-
    dat1$elevation / 1000
mod2 <- lm(tann ~
    elevationP1000, data
    = dat1)
or20 <-
    outreg(list("Temperature
    = mod2), varLabels =
    list("elevationP1000"
    = "Elev. per 1000
    ft"))
cat(or20)
```

	Temperature Estimate (S.E.)	
(Intercept)	11.688***	(Inter
	(0.150)	
Elev. per 1000 ft	-3.238***	Elev.
"	(0.202)	
N	92	N
RMSE	0.958	RMSE
R^2	0.741	R^2
$*p \leq 0.05$** $p \leq 0.01$***$p \leq 0.001$		$*p \leq$

Outline

Where is R different than other Programs?

- In some programs (SPSS), the user runs a command, and the program "spits out" everything worth knowing.
- SAS and Stata, to a lesser degree, still have that approach, although they do leave a door open to export fitted models and do some follow-up work with them.
- R is almost completely different. If SPSS is "idiot proof", the R approach is "idiots not welcome".

KU

Fit, then Follow Up

- The S/R approach has us create a fitted model object
- And then run follow-up functions, such as
  - `summary()`
  - `plot()`

What to work on next

- Find code that works, and then test variations on that.
- The example folders
  Example-els
  Example-mpg
  have full worked examples. Look for R/import-1.R, R/analysis-1.R.
- WorkingExamples in
  `http://pj.freefaculty.org/R/WorkingExamples`.
  Exciting new feature!: HTML files are browsable in there

KU

References

R Core Team (2017). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Session

```
sessionInfo()
```

```
R version 3.6.0 (2019-04-26)
Platform: x86_64-pc-linux-gnu (64-bit)
Running under: Ubuntu 19.04

Matrix products: default
BLAS:   /usr/lib/x86_64-linux-gnu/atlas/libblas.so.3.10.3
LAPACK: /usr/lib/x86_64-linux-gnu/atlas/liblapack.so.3.10.3

locale:
 [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
     LC_TIME=en_US.UTF-8
 [4] LC_COLLATE=en_US.UTF-8     LC_MONETARY=en_US.UTF-8
     LC_MESSAGES=en_US.UTF-8
 [7] LC_PAPER=en_US.UTF-8       LC_NAME=C                  LC_ADDRESS=C
[10] LC_TELEPHONE=C             LC_MEASUREMENT=en_US.UTF-8
    LC_IDENTIFICATION=C

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base

other attached packages:
[1] rockchalk_1.8.144
```

Session ...

```
loaded via a namespace (and not attached):
 [1] Rcpp_1.0.1        lattice_0.20-38 MASS_7.3-51.4   grid_3.6.0
     plyr_1.8.4        nlme_3.1-140
 [7] xtable_1.8-4      stats4_3.6.0    zip_2.0.2       carData_3.0-2
     minqa_1.2.4       nloptr_1.2.1
[13] Matrix_1.2-17     pbivnorm_0.6.0  boot_1.3-22     openxlsx_4.1.0
     splines_3.6.0     lme4_1.1-21
[19] tools_3.6.0       foreign_0.8-71  kutils_1.69     compiler_3.6.0
     mnormt_1.5-5      lavaan_0.6-3
```