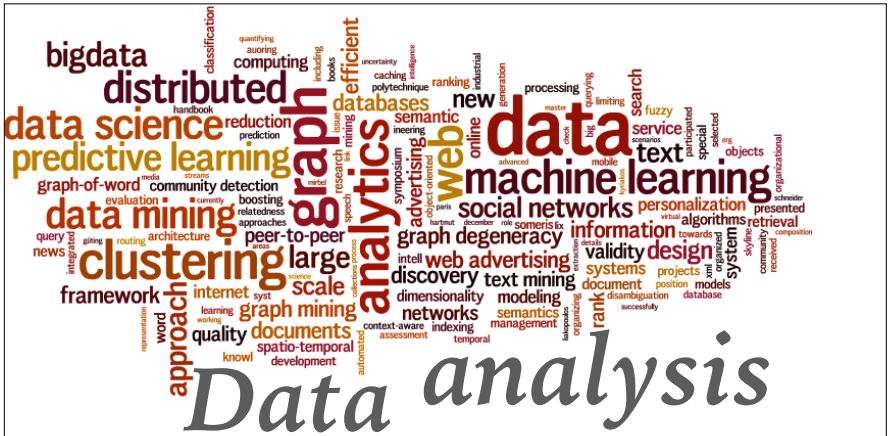
DATA ANALYSIS FOR THE MASSES

Jan Vitek

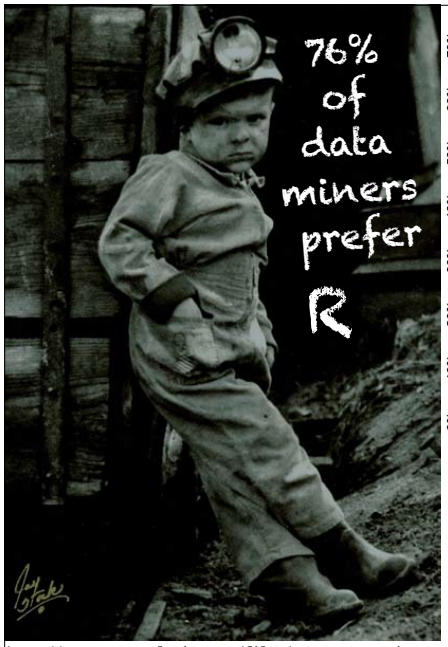
FACULTY OF INFORMATION TECHNOLOGY CTU IN PRAGUE

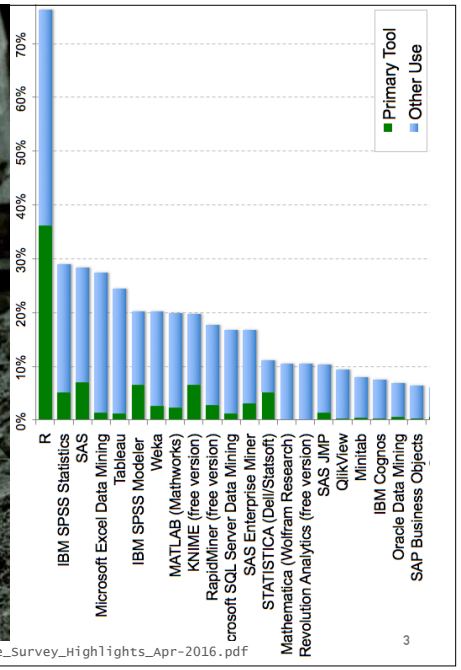
PRL@PRG



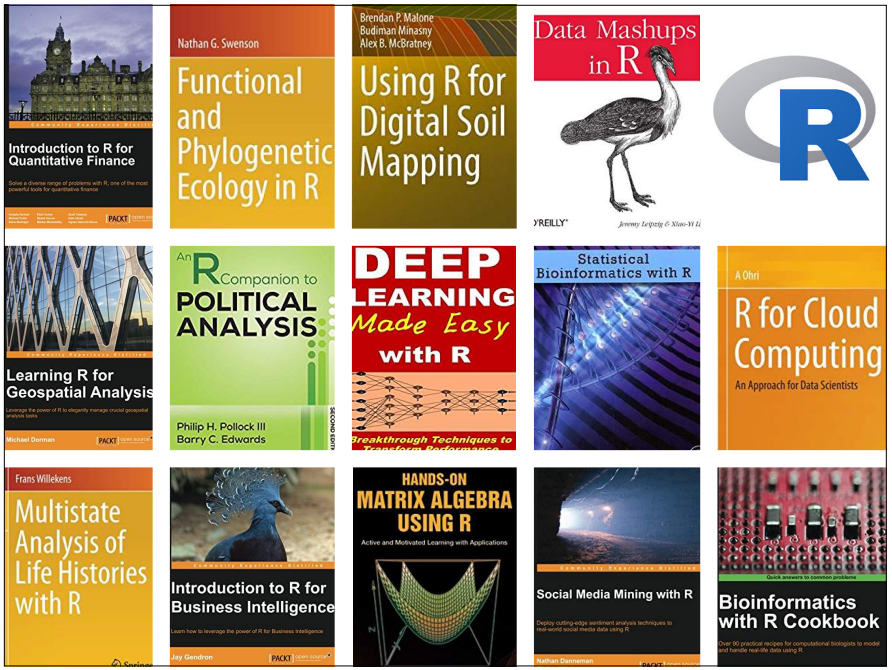
*…process of **inspecting**, **cleansing**, **transforming**, and **modeling** data in order to discover useful information…*

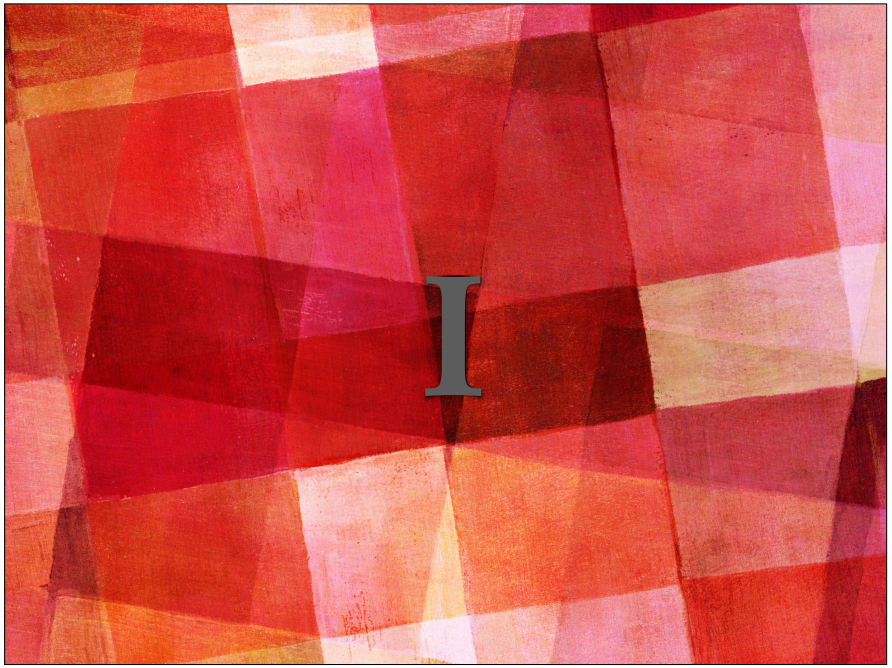
The state of DA today? *The face of DA for the next decade?*





http://www.rexeranalytics.com/files/Rexer_Data_Science_Survey_Highlights_Apr-2016.pdf





THE R PROJECT FOR STATISTICAL COMPUTING

R is a free software environment for statistical computing and graphics.

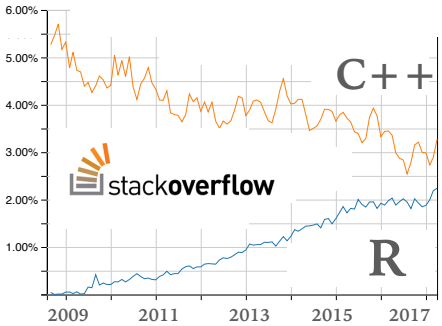
2,000,000+ users in academe and industry

11,172 open source packages on CRAN

195,609 questions on stack overflow

900+ attendees useR! conference

55+ user groups worldwide



FORTY YEARS OF S

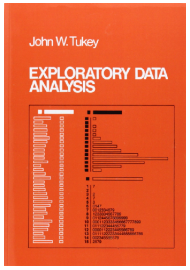
SCS Library (AT&T) convenient statistical routines written in FORTRAN for portability

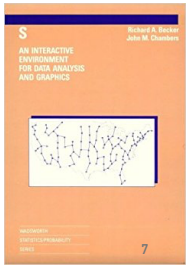
1976: First release of S; influenced by Tukey's work, allows interactive use of SCS routines for EDA

Key idea was a uniform interface to native routines, same calling sequence and a standard data representation

1978: S integrated to the newly developed UNIX operating system, takes advantage of it portability and tooling

1985: John Chambers & Richard Becker publish S book





TWENTY FIVE YEARS OF R

Need a free teaching language for statistical computing

Started as a 1000 line Scheme interpreter

- *1993 R announced on an S mailing list*
- *1995 R open-sourced (GNU)*
- *1997 R core group*
- *2000 R 1.0*
- *2003 R Foundation*
- *2004 UseR! conference in Vienna*
- *2015 R Consortium*

Latest version has over 800,000 lines of C and R code

Ross Ihaka
U. Auckland

Robert Gentleman
23andMe

TWENTY YEARS OF CRAN

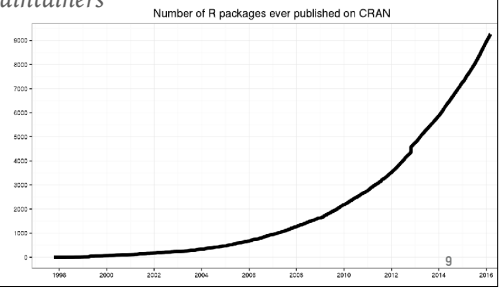
Provide a decentralized and modularized way to develop software

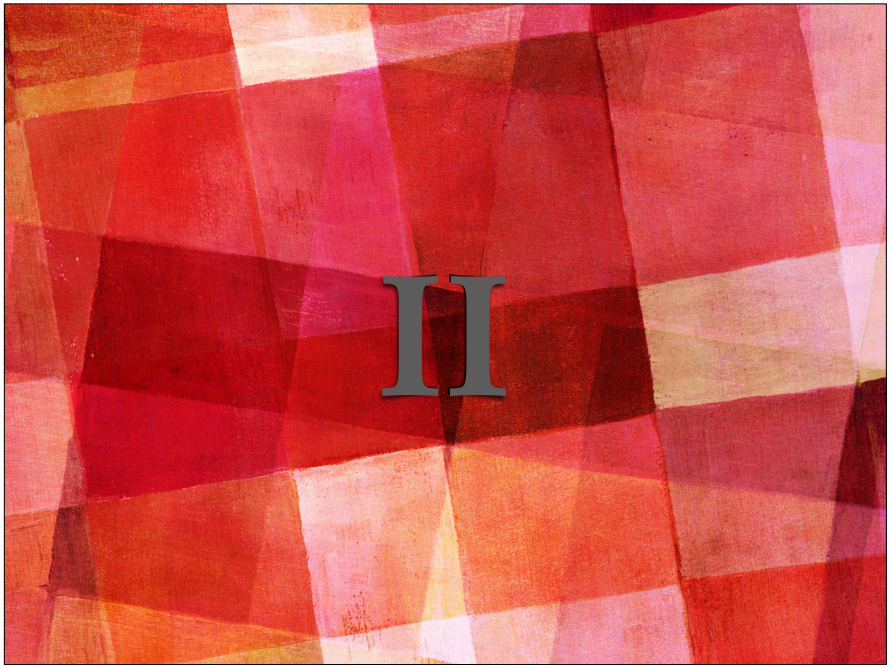
Inspired by Perl (CPAN) and LaTex (CTAN).

- *standardized format for packages, vignettes and tests*
- *automated and manual checking of incoming software*
- *testing on 3 releases of R, for 12 OSes and 7 architectures*
- *manual communication with maintainers*

Growth

- *6 accepted packages/day*
- *230 updates/day*





NUTSHELL

What R is…	*What R isn't…*
• vectorized	• fast
• dynamic	• low-footprint
• lazy	• concurrent
• functional	• distributed

LAZY

Lazy evaluation delays evaluation of arguments until their value is needed while avoiding repeated evaluations.

```
assert(x==2, print("Error"))
```

```
assert<-function(C,P) if(C){P}
```

```
with(fd, carb*den)
```

DYNAMIC

Dynamic languages support run-time extension of programs, by adding new code, by extending definitions or by modifying types.

```
with  <-  function(df, e,...)
               eval( substitute(e),
                     df,
                     parent.frame)))
```


COPY-ON-WRITE

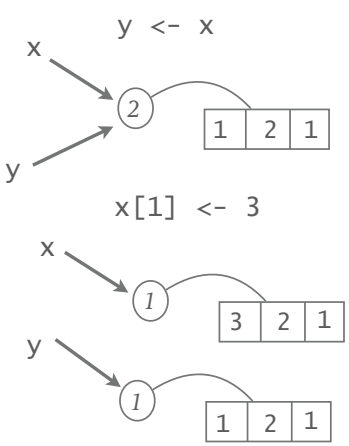
Modification of a value creates a copy, unless it is unshared

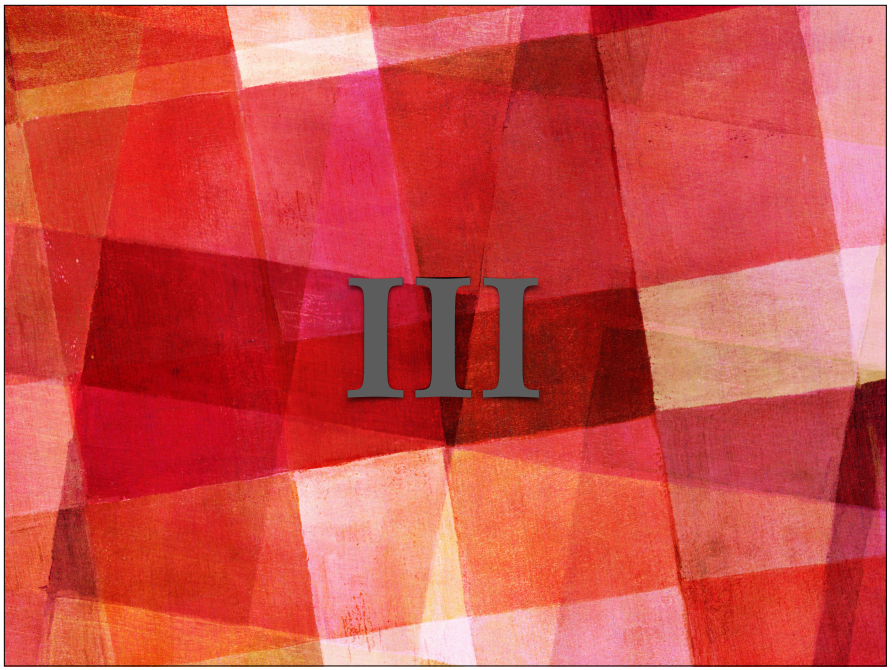
This allows the following pleasing property to hold

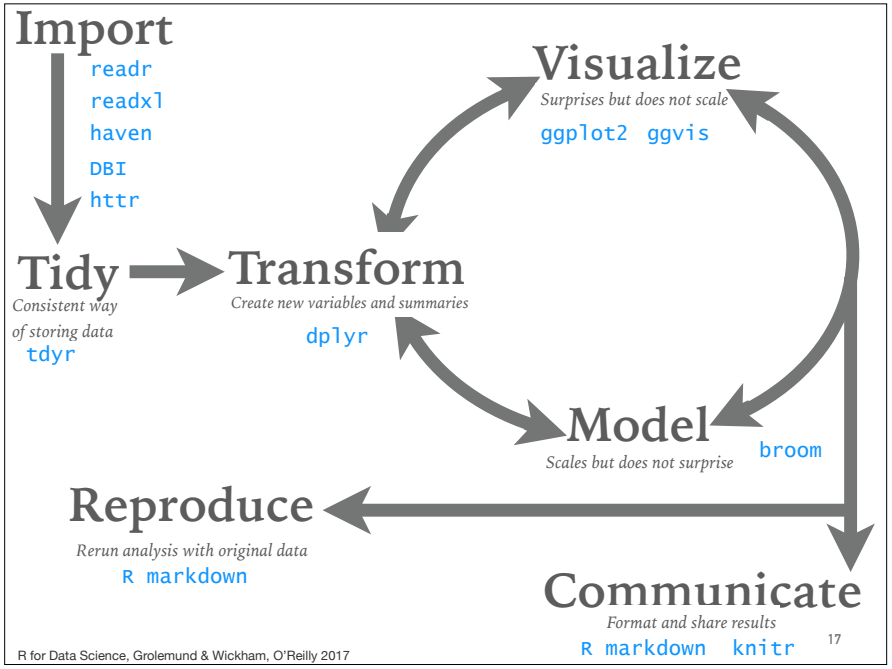
```
assert(x[1]==5)
f(x)                    f <- function(a) { a[1]<-0}
assert(x[1]==5)
```


COPY-ON-WRITE

Modification of a value creates a copy, if the value is shared







R for Data Science, Grolemund & Wickham, O'Reilly 2017

TIDY

Imported data is often 'messy', obtaining tidy data entails changing representation, validating encodings, and reshaping so that

Each variable forms a column.

Each observation forms a row.

Each type of observational unit is a data frame

"80% of data analysis is … cleaning & preparing"
— Dasu, Johnson 2003

Hadley Wickham (RStudio)

Tidy Data

J. of Statistical Software, 2014

TIDY

```
grades <- tbl_df(read.table("tbl.csv"))
```

ID	Test	Fall	Spring	Winter
1	Final	90	94	NA
2	Midtm	16	??	20
2	Final	93	90	96

```
grades[grades=="??"] <- NA
```

ID	Test	Fall	Spring	Winter
1	Final	90	94	NA
2	Midtm	16	NA	20
2	Final	93	90	96

TIDY

Column headers as values

Missing observations (explicit)

Missing observations (implicit)

Different units in a column

Wrong data type

ID	Test	Fall	Spring	Winter
<int>	<fctr>	<int>	<fctr>	<int>
1	Final	90	94	NA
2	Midtm	16	NA	20
2	Final	93	90	96

TIDY

ID	Test	Fall	Spring	Winter
<int>	<fctr>	<int>	<fctr>	<int>
1	Final	90	94	NA
2	Midtm	16	NA	21
2	Final	93	90	96

```
grades %>%
  mutate_at('Spring', as.character) %>%
  mutate_at('Spring', as.integer)   %>%
  complete(ID, Test) %>%
  gather(Semester, Score, Fall:Winter) %>%
  spread(Test, Score) -> grades2
```


TIDY

ID	Test	Fall	Spring	Winter
<int>	<fctr>	<int>	<int>	<int>
1	Final	90	94	NA
2	Midtm	16	NA	21
2	Final	93	90	96

```
grades %>%
  mutate_at('Spring', as.character) %>%
  mutate_at('Spring', as.integer)   %>%
```


TIDY

ID	Test	Fall	Spring	Winter
<int>	<fctr>	<int>	<int>	<int>
1	Final	90	94	NA
2	Midtm	16	NA	21
2	Final	93	90	96

```
grades %>%
  mutate_at('Spring', as.character) %>%
  mutate_at('Spring', as.integer)   %>%
  complete(ID, Test)
```


ID	Test	Fall	Spring	Winter
1	Final	90	94	NA
1	Midtm	NA	NA	NA
2	Final	93	90	96
2	Midtm	16	NA	21

ID	Test	Semester	Score
1	Final	Fall	90
1	Midterm	Fall	NA
2	Final	Fall	93
2	Midterm	Fall	16
1	Final	Spring	94
1	Midterm	Spring	NA
2	Final	Spring	90
2	Midterm	Spring	NA
1	Final	Winter	NA
1	Midterm	Winter	NA
2	Final	Winter	96
2	Midterm	Winter	20

```
grades %>%
  mutate_at('Spring', as.character)
  mutate_at('Spring', as.integer)
  complete(ID, Test) %>%
  gather(Semester, Score, Fall:Winter)
```

ID	Test	Semester	Score
1	Final	Fall	90
1	Midtm	Fall	NA
2	Final	Fall	93
2	Midtm	Fall	16
1	Final	Spring	94
1	Midtm	Spring	NA
2	Final	Spring	90
2	Midtm	Spring	NA
1	Final	Winter	NA
1	Midtm	Winter	NA
2	Final	Winter	96
2	Midtm	Winter	20

ID	Semester	Final	Midtm
1	Fall	90	NA
1	Spring	94	NA
1	Winter	NA	NA
2	Fall	93	16
2	Spring	90	NA
2	Winter	96	20

```
spread(Test, Score) -> grades2
```


TRANSFORM

A grammar of data manipulation

One data frame verbs

- `filter()` *to select observations based on their values.*
- `arrange()` *to reorder the observations*
- `select()`, `rename()` *to select variables based on their names*
- `mutate()` *to add new variables as functions of existing ones*
- `summarise()` *to condense multiple values to a single value*
- `sample_n()` *to take random samples*

Grouping verb

- `group_by()` *perform any operation "by group"*

Multiple table verbs

- `inner_join(x,y)`, `left_join`, `right_join`, `full_join`
  only includes observations in both x and y, in x, in y, in x or y
- `semi_join(x,y)`, `anti_join` *keep, drop, all in x that have a match in y*
- `intersect(x,y)`, `union`, `setdiff` *set operations*

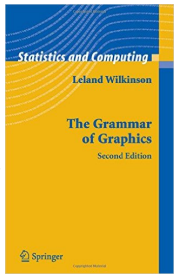
ID	Semester	Final	Midtm
1	Fall	90	NA
1	Spring	94	NA
1	Winter	NA	NA
2	Fall	93	16
2	Spring	90	NA
2	Winter	96	20

ID	cnt	avgF	avgM	grade
1	3	92	0	73.6
2	3	93	18	92.4

```
grades2 %>%
   group_by(ID) %>%
   summarise(cnt=n(),
        avF=mean(Final,na_rm=TRUE),
        avM=mean(Midtm,na_rm=TRUE)) %>%
   replace_na(list(avgM=0)) %>%
   mutate(grade=avgF*.8+avgM)
```


VISUALIZE

The grammar of graphics



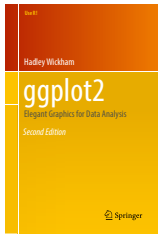
Statistical graphic specifications are expressed in:

1) ***DATA**: a set of operations that create variables from datasets*
2) ***TRANS**: variable transformations (e.g., rank)*
3) ***SCALE**: scale transformations (log)*
4) ***COORD**: a coordinate system (polar)*
5) ***ELEMENT**: graphs (points) and aesthetic attributes (color)*
6) ***GUIDE**: one or more guides (axes, legends, etc.)*

Leland Wilkinson
Chief Scientist, H2O.ai

VISUALIZE

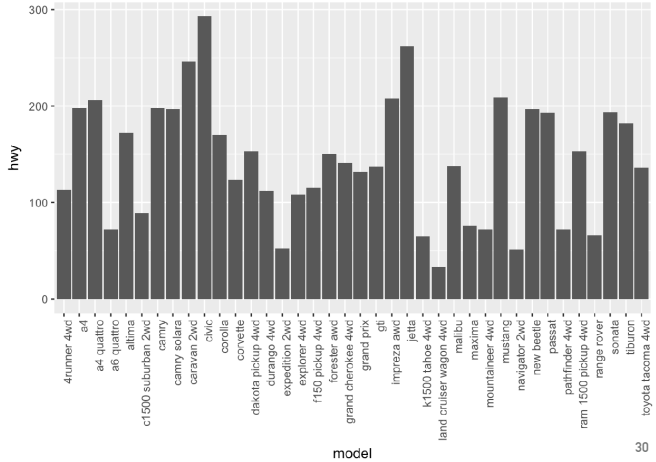
Implementing the grammar of graphics in R



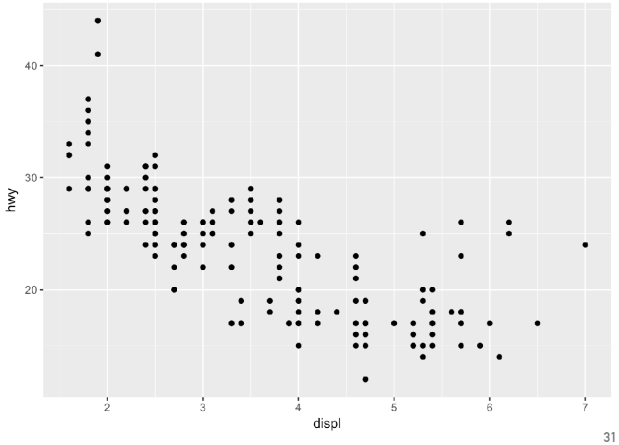

```
> mpg
  manufacturer      model displ  year   cyl      trans   drv   cty   hwy    fl   class
         <chr>      <chr> <dbl> <int> <int>      <chr> <chr> <int> <int> <chr>   <chr>
1         audi         a4   1.8  1999     4   auto(l5)     f    18    29     p compact
2         audi         a4   1.8  1999     4 manual(m5)     f    21    29     p compact
3         audi         a4   2.0  2008     4 manual(m6)     f    20    31     p compact
4         audi         a4   2.0  2008     4   auto(av)     f    21    30     p compact
5         audi         a4   2.8  1999     6   auto(l5)     f    16    26     p compact
6         audi         a4   2.8  1999     6 manual(m5)     f    18    26     p compact
7         audi         a4   3.1  2008     6   auto(av)     f    18    27     p compact
8         audi a4 quattro   1.8  1999     4 manual(m5)     4    18    26     p compact
9         audi a4 quattro   1.8  1999     4   auto(l5)     4    16    25     p compact
10        audi a4 quattro   2.0  2008     4 manual(m6)     4    20    28     p compact
# ... with 224 more rows
```



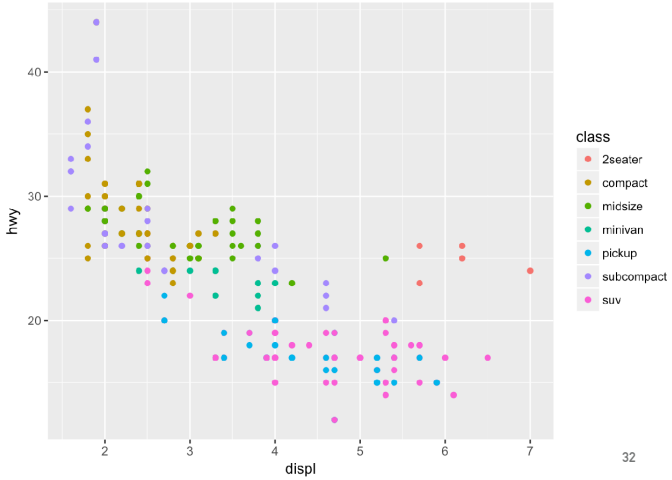
```
ggplot(mpg, aes(x=model,y=hwy)) +
  geom_bar(stat="identity")
```



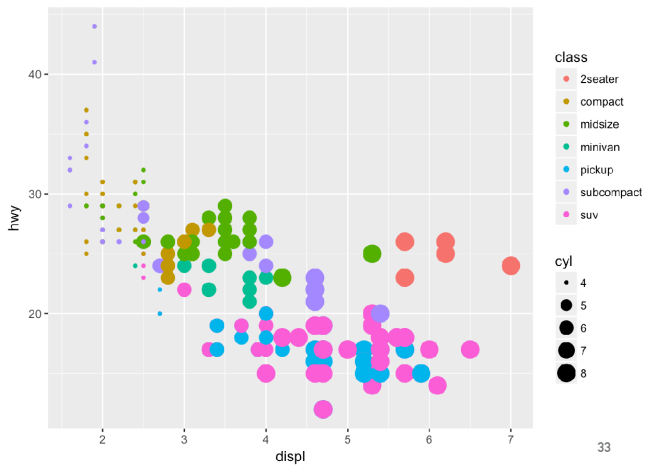

```
ggplot(mpg, aes(x=displ, y=hwy)) +
   geom_point()
```



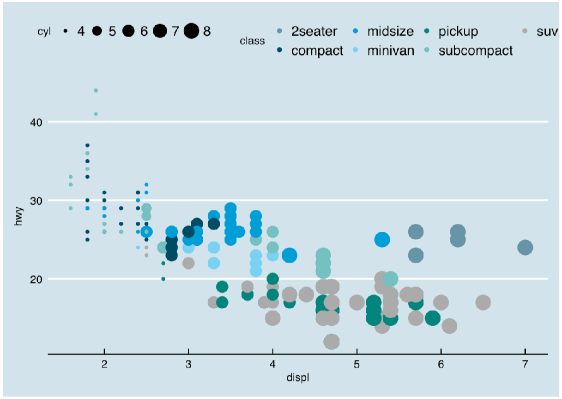

```
ggplot(mpg, aes(x=displ, y=hwy)) +
   geom_point(color=class)
```



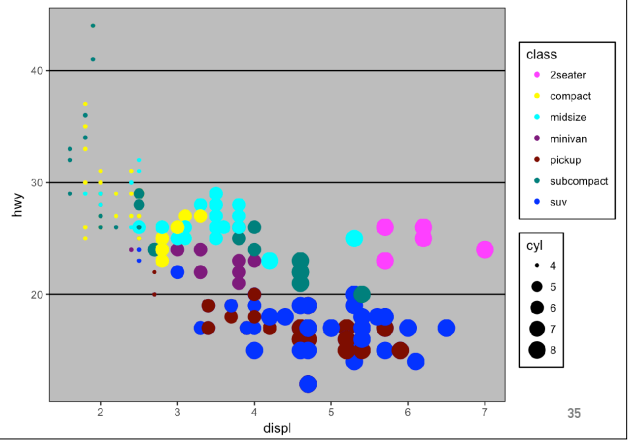
```
ggplot(mpg, aes(x=displ, y=hwy)) +
   geom_point(color=class,size=cyl)
```




```
ggplot(mpg, aes(x=displ, y=hwy)) +
   geom_point(color=class,size=cyl)+
    theme_economist() +
    scale_colour_economist()
```



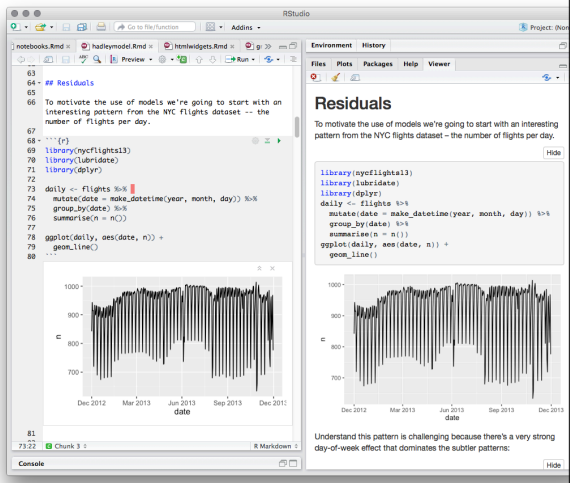

```
ggplot(mpg, aes(x=displ, y=hwy)) +
   geom_point(color=class,size=cyl)+
    theme_excel() +
    scale_colour_excel()
```



REPRODUCE

2001 ***Sweave*** *— integration of R code into LaTeX to create dynamic reports, updated automatically if data or analysis change*

2011 ***Knitr*** *— inspired by Sweave. Expands it to support R Markdown, caching, TikZ graphics, other languages, outputs PDF, Html, LaTeX…*



EVOLVING LANGUAGE ECOSYSTEMS (ELE)

https://prl-prg-erc.github.io

- *Modern languages are characterized by rich ecosystems that include compilers, interpreters, IDEs, libraries, help pages, manuals, stackoverflow…*
- *To remain relevant, languages need to evolve, they must add new features, their libraries must adapt to new requirements, implementations must change to meet new performance goals.*
- *The ELE project explores the fundamental techniques and algorithms for evolving entire language ecosystems. We will reduce the cost of wide-ranging changes to programming languages and obviate the need for devising new languages.*

BIG DATA

matter: *an R package for direct interactions with larger-than-memory datasets, stored in an arbitrary number of files of any size.*

matter enables a wide variety of data exploration and manipulation steps and is extensible to many bioinformatics applications

Kylie Bemis
Northeastern U.

```
sums ← mapply(ctx, x, sum,
        function(...){sum(...)}
           @:commutative   @:pure
           @:visit(−1:0, 0:−1,
                   0:0,
                   0:1, 1:0))
```

K. Bemis O. Vitek. ***matter: an R package for rapid prototyping with larger-than-memory datasets on disk.*** *Bioinformatics 2017*

TYPES

R is untyped, statically, and strongly typed, at runtime

What if we could add types, gradually, partially?

```
function (x, narm=FALSE, dims=1L) {
    if (is.matrix(x))
        x <- as.matrix(x)
      if (dims < 1L ||
        dims > length(dim(x))-1L)
        stop("invalid 'dims'")
```

A gradual type system can gradually enrich "scripts" with explicit and sound types without changing code"

Matthias Felleisen
Northeastern U.

```
function(x :~ Matrix(N,…),
         narm :: Logical = FALSE,
         dims :: Range(1,dim(x)-1L)= 1L){
```

PARALLELISM

R's implementation is not thread safe, how do we parallelize?

foreach *allows iteration over elements in a collection without explicit loop counter in parallel via doMC (parallel/multicore), doMPI (Rmpi)…*

partools *Miscellaneous utilities for parallelizing large computations. Alternative to MapReduce. File splitting and distributed operations such as sort and aggregate.*

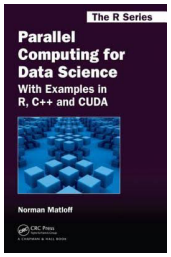
The ELE project investigates safe thread parallelism

Following our collaboration with Intel

Synthesize OpenMP code for a subset of R

"Painless parallel computation is not possible"

Norm Matloff



PERFORMANCE

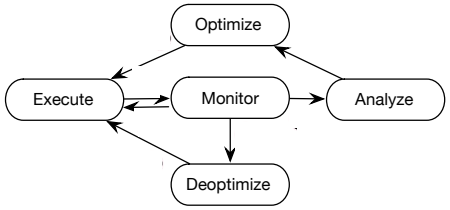
R performance is hampered by dynamic features of the language

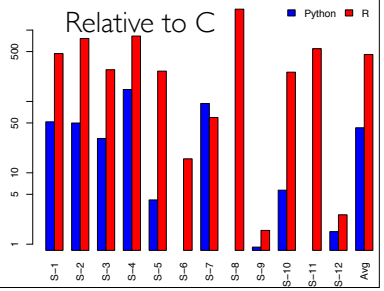
Tierney and Kalibera introduced a byte-code compiler

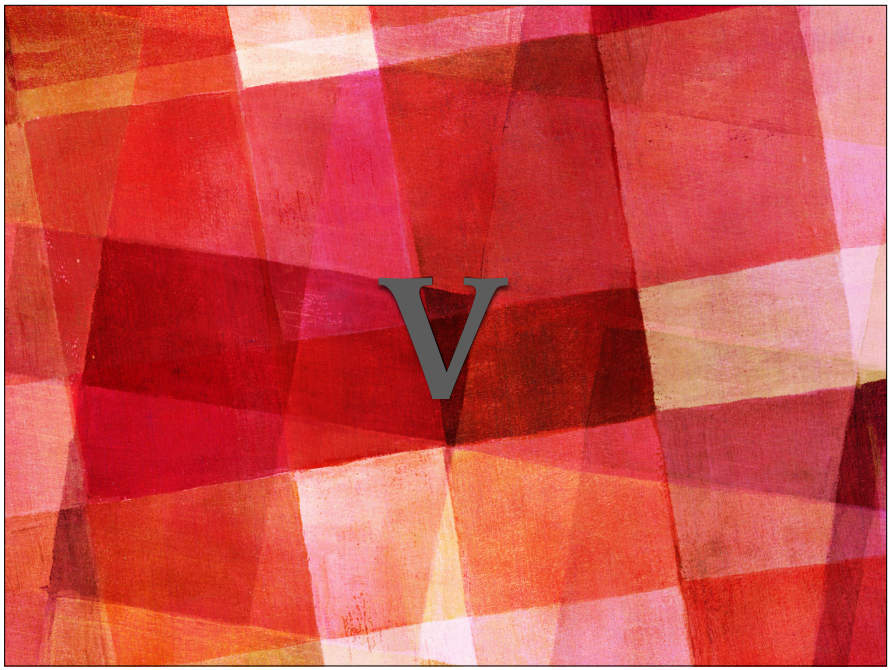
Oracle research is building a Java compiler

The ELE project builds LLVM JIT

Tomas Kalibera
CTU

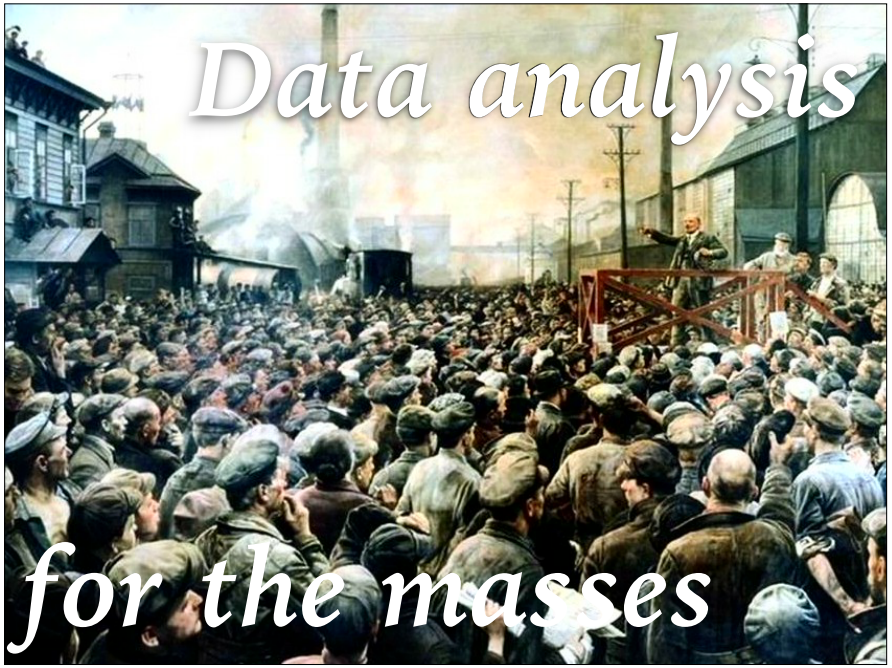






TAKE-AWAYS

- *INTERACTIVITY: the ability to quickly test out ideas is key to exploratory data analysis*
- *DYNAMISM: the dynamic features of the language are key for learning and development*
- *ECOSYSTEM: R is not a language it is a rich ecosystem*
- *HUMANS IN THE LOOP: without a strong open source community R wouldn't be here*


Data analysis
for the masses