LITERATE PROGRAMMING

Marcus Birkenkrahe

Winter 2020

Contents

1 WHAT WILL YOU LEARN?

- What is the main idea behind the Web?
- What are Markup and Markdown?
- What is "literate programming"?
- What does it have to do with data science notebooks?
- Which data science notebooks are out there?
- How can I use notebooks for data science?
- How can I use notebooks in my own work?

2 IT'S PERSONAL

- DESY Particle Physics PhD
- CERN WWW development
- What is the main idea behind the Web?

Figure 1: Photo: Peggy Bacon in mid-air backflip, Sydney 1937. Source: State Lib. NSW@flickr

See figure 2 for a glimpse of the early days of the Web.

3 WHAT IS MARKUP?

- HTML = HyperText Markup Language
- Hide meta information - unlike "WYSIWIG"
- Example - active text element behind This is a link.

```
<a href="https://www.w3schools.com">This is a link</a>
```

Q: *Who can write HTML (and CSS) documents?*[1]

4 WHAT IS MARKDOWN? (1)

> »The idea for Markdown is to make it easy to read, write, and edit prose. HTML is a publishing format; Markdown is a writing format. Thus, Markdown's formatting syntax only addresses issues that can be conveyed in plain text.« – John Gruber

5 WHAT IS MARKDOWN? (2)

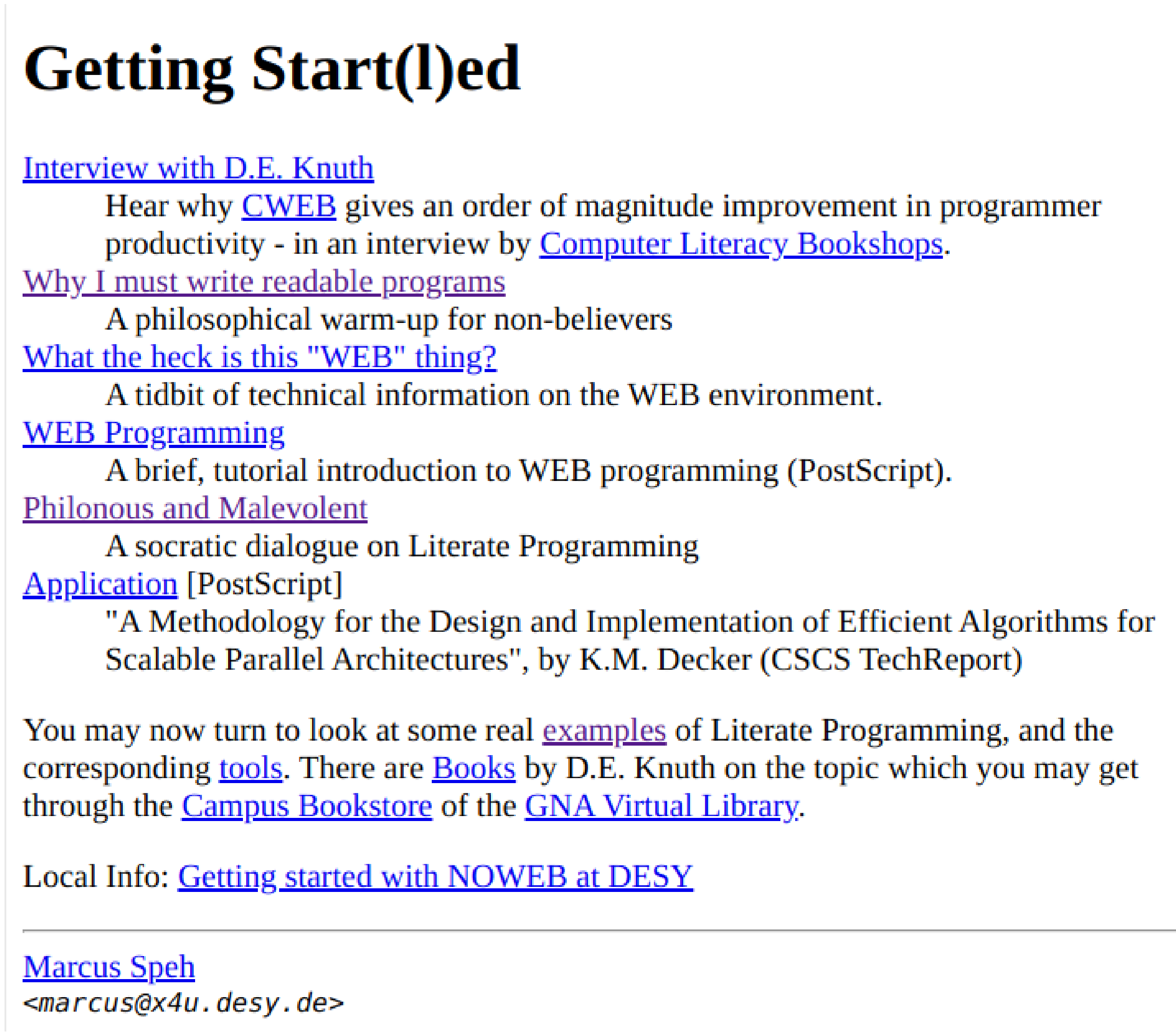
- Easy-to-read and easy-to-write
- Easy to customize (see figure 3)
- Even easier than HTML:

```
[This is a link](https://www.w3schools.com)
```

```
<a href="https://www.w3schools.com">This is a link</a>
```

Q: Have you come across Markup or Markdown?

[1] For a live view, right click & pick "View page source" in your browser.



Getting Start(l)ed

Interview with D.E. Knuth
    Hear why CWEB gives an order of magnitude improvement in programmer productivity - in an interview by Computer Literacy Bookshops.
Why I must write readable programs
    A philosophical warm-up for non-believers
What the heck is this "WEB" thing?
    A tidbit of technical information on the WEB environment.
WEB Programming
    A brief, tutorial introduction to WEB programming (PostScript).
Philonous and Malevolent
    A socratic dialogue on Literate Programming
Application [PostScript]
    "A Methodology for the Design and Implementation of Efficient Algorithms for Scalable Parallel Architectures", by K.M. Decker (CSCS TechReport)

You may now turn to look at some real examples of Literate Programming, and the corresponding tools. There are Books by D.E. Knuth on the topic which you may get through the Campus Bookstore of the GNA Virtual Library.

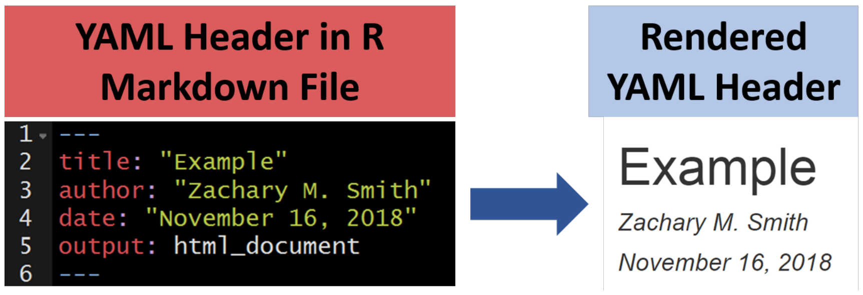
Local Info: Getting started with NOWEB at DESY

Marcus Speh
<marcus@x4u.desy.de>

Figure 2: Virtual Library page for Literate Programming @DESY, 1994 (Source: Wayback Machine)

6 WHAT IS MARKDOWN (3)

- YAML ("YAML Ain't Markup Language")
- Used for configuration (e.g. headers)
- Used in R Markdown Notebooks



```yaml
---
title: "Example"
author: "Zachary M. Smith"
date: "November 16, 2018"
output: html_document
---
```

Figure 3: YAML header in R Markdown file (left) and rendering (right) (Source: Smith 2020)

7 ORG-MODE

- Major GNU Emacs editor mode[2]
- Plain text markup + export + publishing
- Literate Programming environment[3]

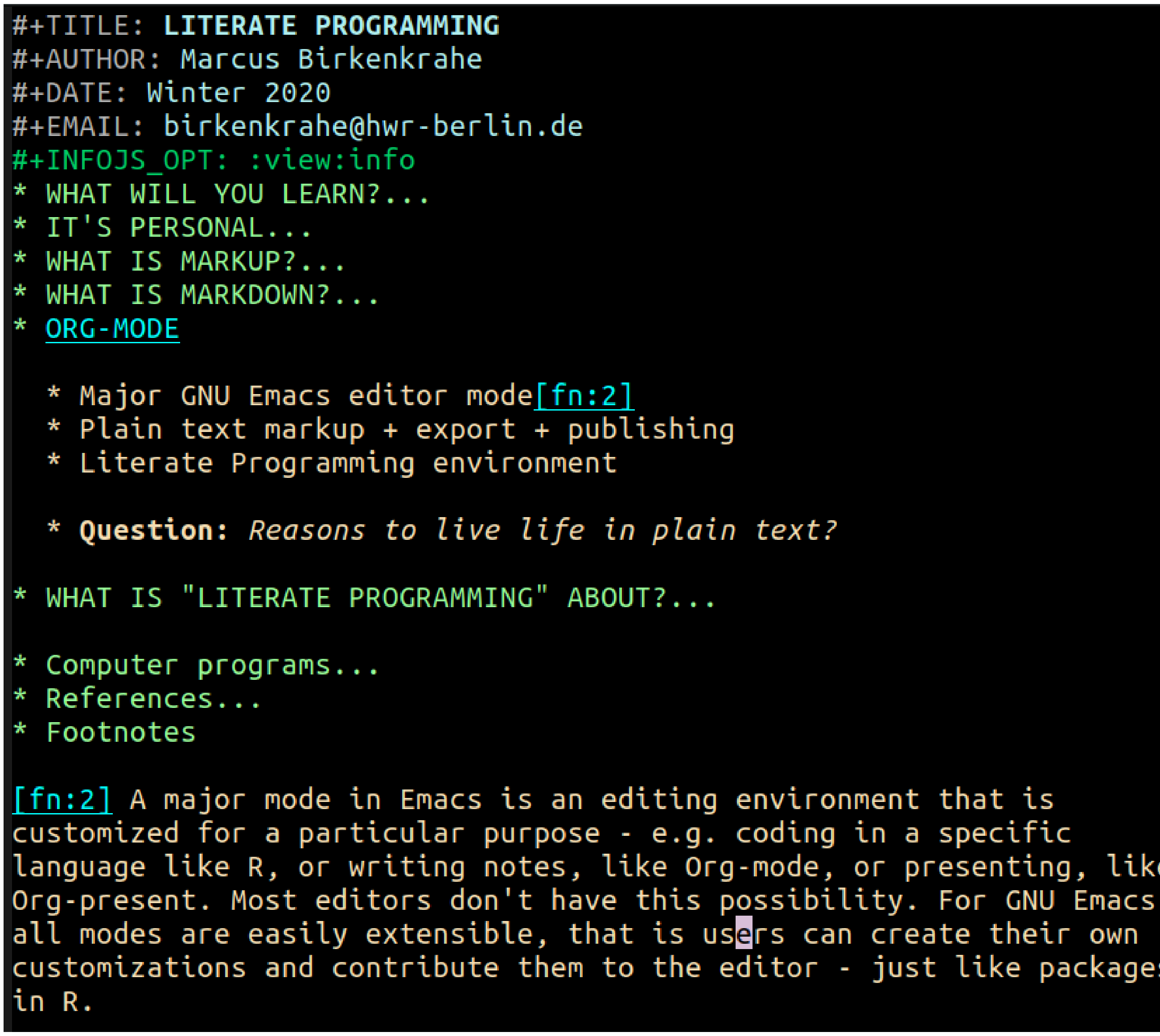
Q: Can you think of any reasons to live life in plain text?

See figure 4 for an example of this very page displayed in Emacs Org-mode.

---

[2] A major mode in Emacs is an editing environment that is customized for a particular purpose - e.g. coding in a specific language like R, or writing notes, like Org-mode, or presenting, like Org-present. Most editors don't have this possibility. For GNU Emacs, all modes are easily extensible, that is users can create their own customizations and contribute them to the editor - just like packages in R.

[3] See also: Org-mode spreadsheets (Gif)



```
#+TITLE: LITERATE PROGRAMMING
#+AUTHOR: Marcus Birkenkrahe
#+DATE: Winter 2020
#+EMAIL: birkenkrahe@hwr-berlin.de
#+INFOJS_OPT: :view:info
* WHAT WILL YOU LEARN?...
* IT'S PERSONAL...
* WHAT IS MARKUP?...
* WHAT IS MARKDOWN?...
* ORG-MODE

  * Major GNU Emacs editor mode[fn:2]
  * Plain text markup + export + publishing
  * Literate Programming environment

  * Question: Reasons to live life in plain text?

* WHAT IS "LITERATE PROGRAMMING" ABOUT?...

* Computer programs...
* References...
* Footnotes

[fn:2] A major mode in Emacs is an editing environment that is
customized for a particular purpose - e.g. coding in a specific
language like R, or writing notes, like Org-mode, or presenting, lik
Org-present. Most editors don't have this possibility. For GNU Emacs
all modes are easily extensible, that is users can create their own
customizations and contribute them to the editor - just like package
in R.
```

Figure 4: GNU Emacs Org-mode Markup example

8 LITERATE PROGRAMMING

»Instead of imagining that our main task is to instruct a *computer* what to do, let us concentrate rather on explaining to *human beings* what we want a computer to do.« – Donald Knuth

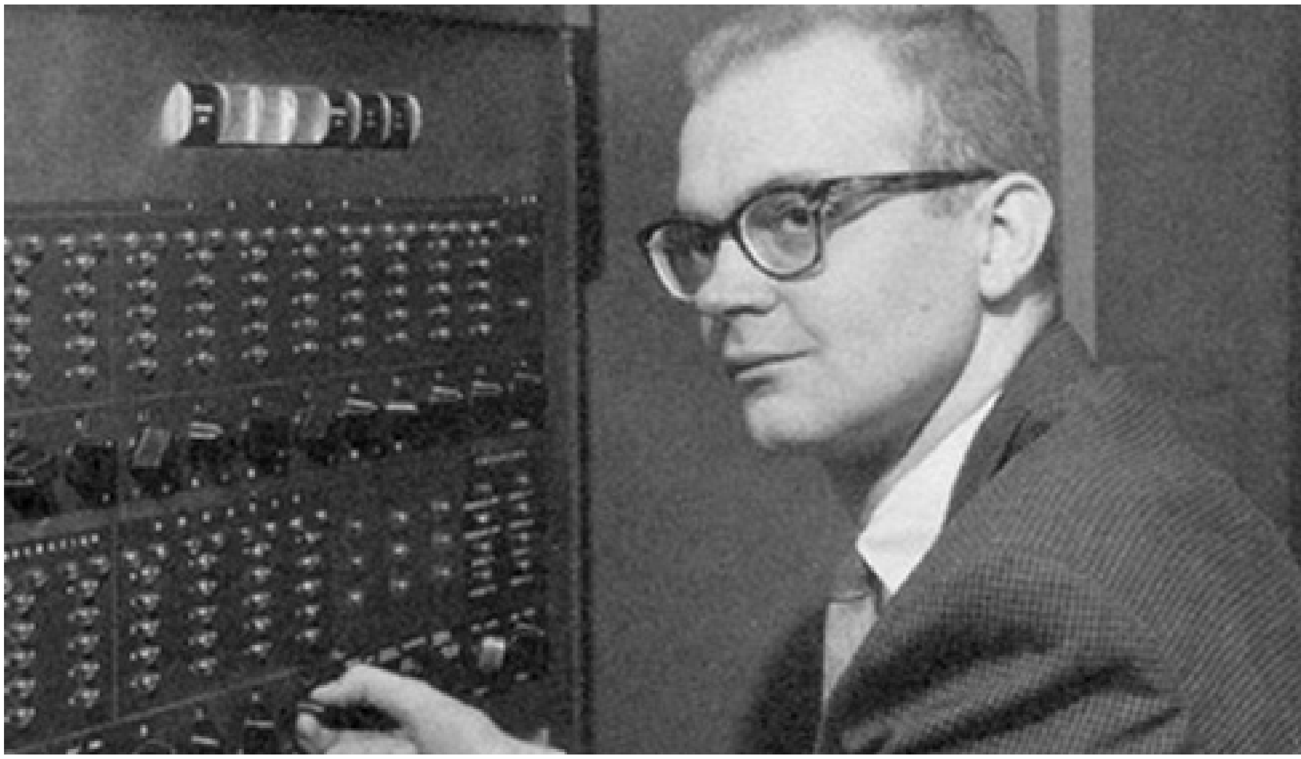
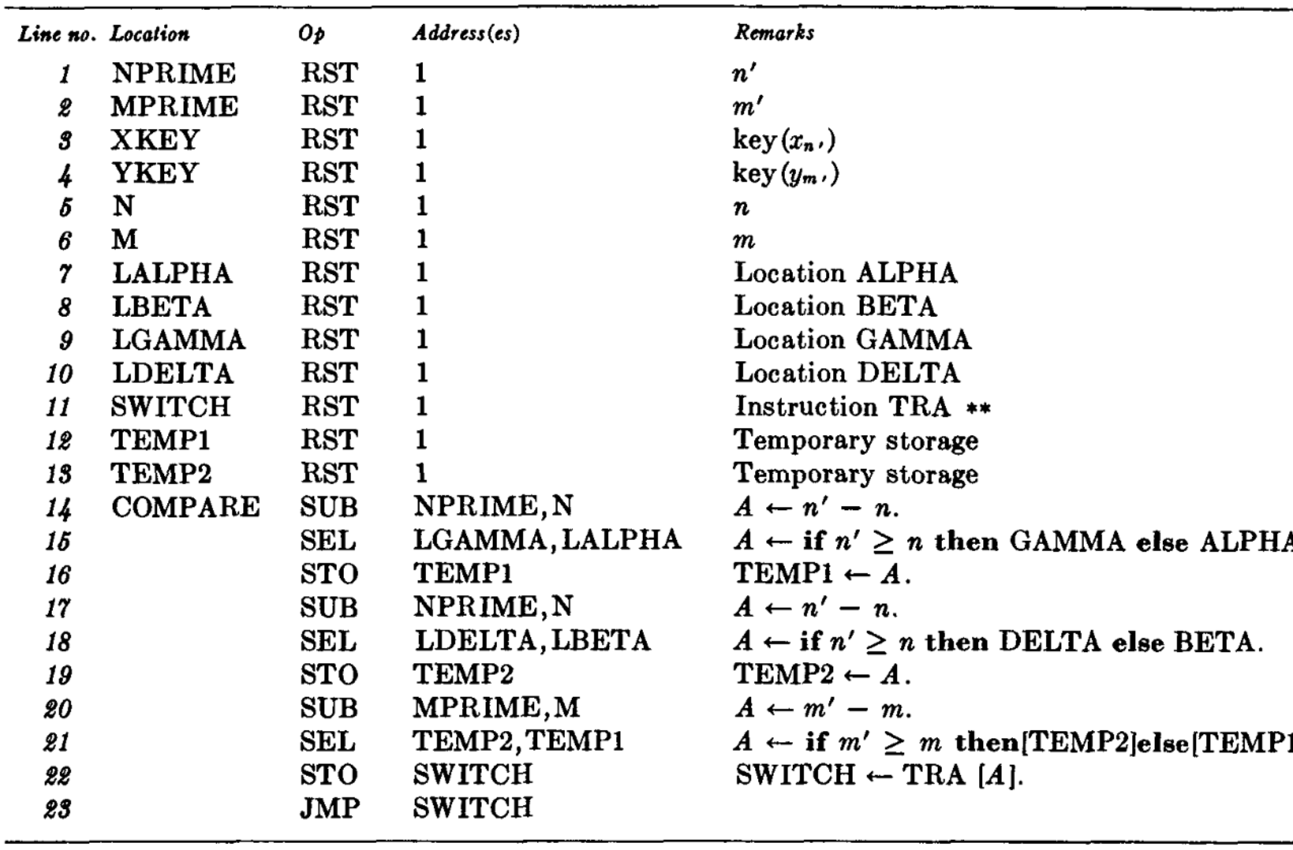


Figure 5: Donald M. Knutz, 1958, working on an IBM 650 computer (Source).

9 LITPROG: WHY?

- Machines cannot handle uncertainty (figure 6)
- Humans like stories



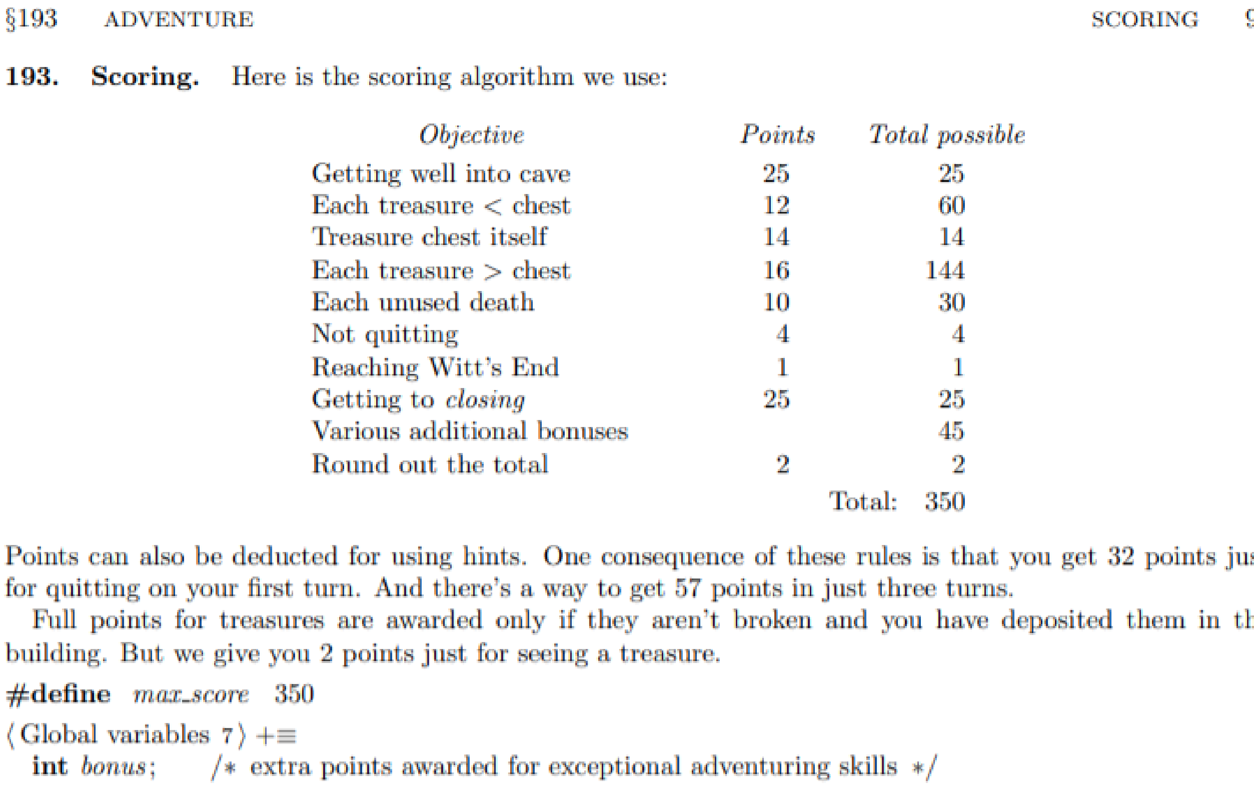
Line no.	*Location*	*Op*	*Address(es)*	*Remarks*
1	NPRIME	RST	1	n'
2	MPRIME	RST	1	m'
3	XKEY	RST	1	key$(x_{n'})$
4	YKEY	RST	1	key$(y_{m'})$
5	N	RST	1	n
6	M	RST	1	m
7	LALPHA	RST	1	Location ALPHA
8	LBETA	RST	1	Location BETA
9	LGAMMA	RST	1	Location GAMMA
10	LDELTA	RST	1	Location DELTA
11	SWITCH	RST	1	Instruction TRA **
12	TEMP1	RST	1	Temporary storage
13	TEMP2	RST	1	Temporary storage
14	COMPARE	SUB	NPRIME,N	$A \leftarrow n' - n$.
15		SEL	LGAMMA,LALPHA	$A \leftarrow$ **if** $n' \geq n$ **then** GAMMA **else** ALPHA
16		STO	TEMP1	TEMP1 $\leftarrow A$.
17		SUB	NPRIME,N	$A \leftarrow n' - n$.
18		SEL	LDELTA,LBETA	$A \leftarrow$ **if** $n' \geq n$ **then** DELTA **else** BETA.
19		STO	TEMP2	TEMP2 $\leftarrow A$.
20		SUB	MPRIME,M	$A \leftarrow m' - m$.
21		SEL	TEMP2,TEMP1	$A \leftarrow$ **if** $m' \geq m$ **then**[TEMP2]**else**[TEMP1
22		STO	SWITCH	SWITCH $\leftarrow$ TRA $[A]$.
23		JMP	SWITCH	

Figure 6: Von Neumann's First Computer Program (Knuth, 1970)

10 LITPROG: HOW?

- Write programs for use by humans *and* by machines
- Write mainly documentation that also contains code

Figure 7 shows part of a literate program. program



§193 ADVENTURE SCORING 9

193. Scoring. Here is the scoring algorithm we use:

Objective	*Points*	*Total possible*
Getting well into cave	25	25
Each treasure < chest	12	60
Treasure chest itself	14	14
Each treasure > chest	16	144
Each unused death	10	30
Not quitting	4	4
Reaching Witt's End	1	1
Getting to *closing*	25	25
Various additional bonuses		45
Round out the total	2	2
	Total:	350

Points can also be deducted for using hints. One consequence of these rules is that you get 32 points jus
for quitting on your first turn. And there's a way to get 57 points in just three turns.
Full points for treasures are awarded only if they aren't broken and you have deposited them in th
building. But we give you 2 points just for seeing a treasure.

#define *max_score* 350

⟨ Global variables 7 ⟩ +≡
  int *bonus*; /* extra points awarded for exceptional adventuring skills */

Figure 7: Another screenshot of Knuth's `cweb` version of `advent`.

11 LITPROG: WORKFLOW

- Documentation + code is contained in one file (`file.w`)
- Tangling leads to a file that the computer can run

- Weaving leads to a file that can be printed

See figure 8 for the complete workflow.

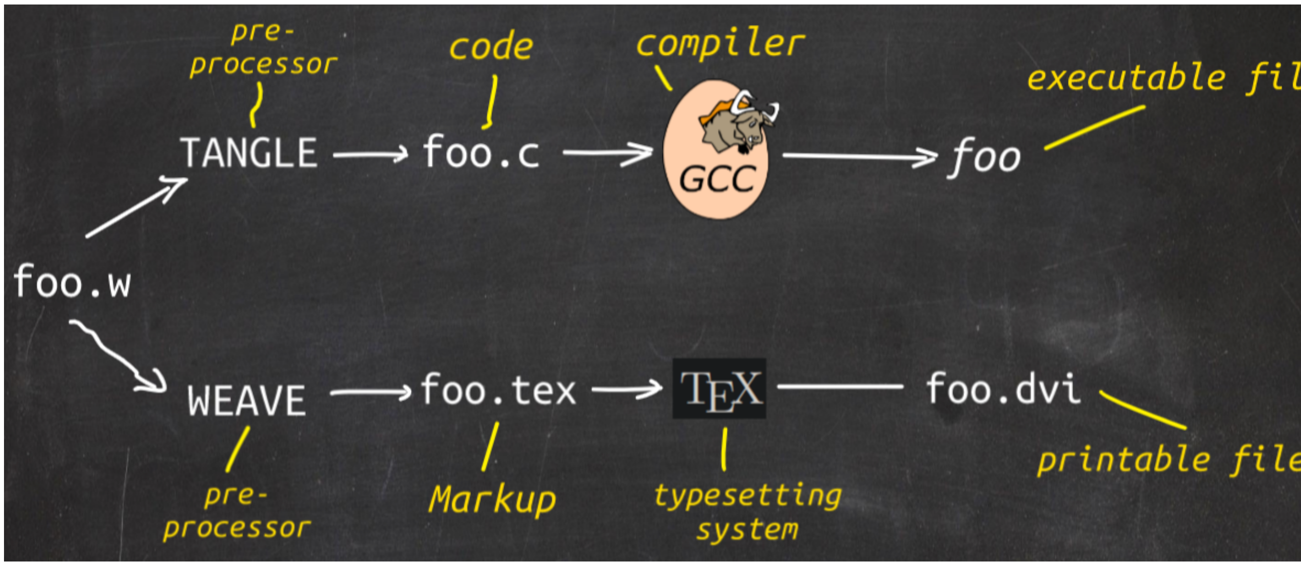
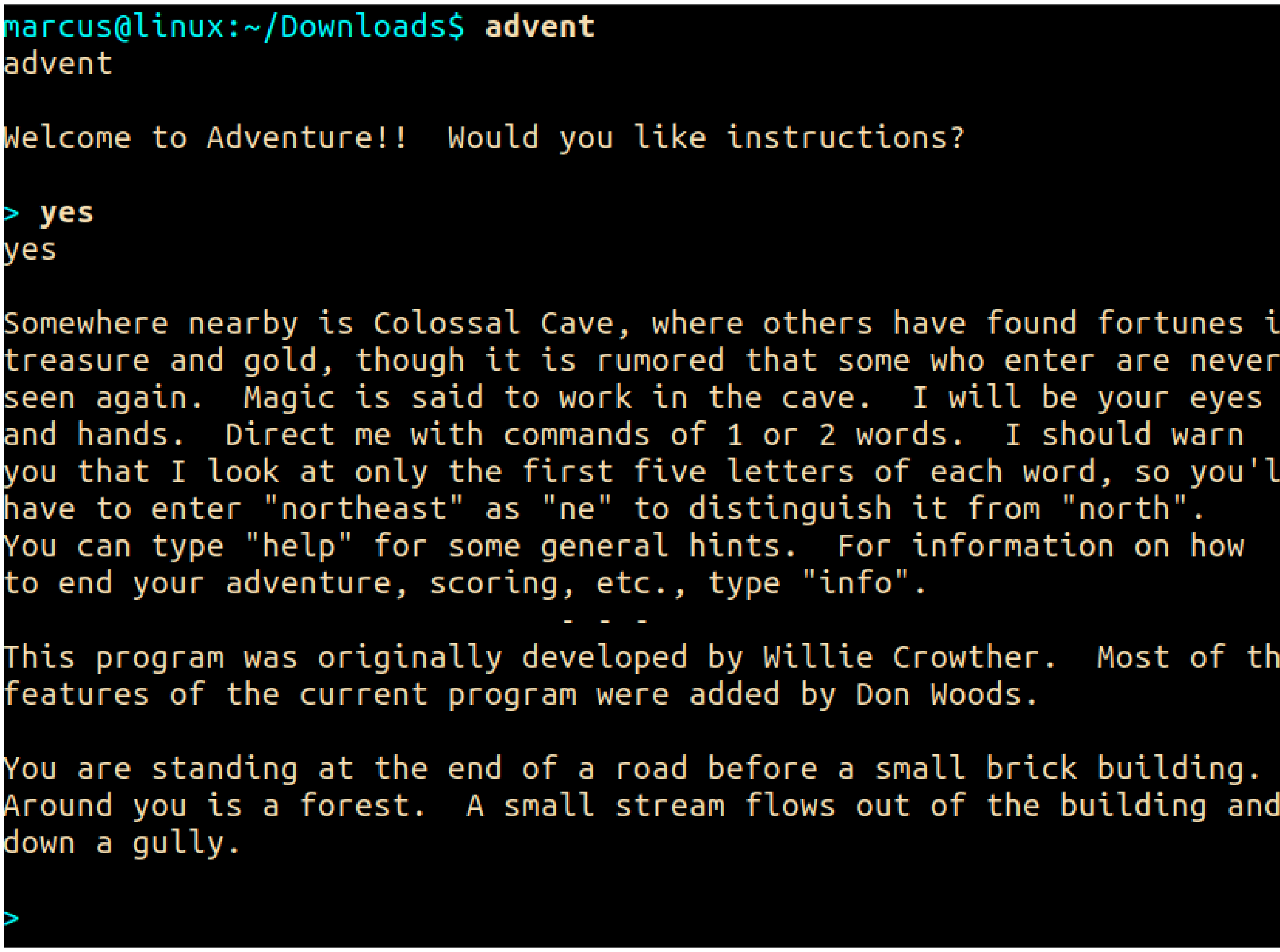


Figure 8: Literate programming process chain (Knuth/Levy, 2002).

12 LITPROG: EXAMPLE

- `advent` is the first digital Role Playing Game (RPG)
- It was rewritten in `cweb` by Don Knuth (see figure 11)
- *Try typing `advent` in your terminal!*

Figure 9 shows the first few moments of the game (source).



```
marcus@linux:~/Downloads$ advent
advent

Welcome to Adventure!!  Would you like instructions?

> yes
yes

Somewhere nearby is Colossal Cave, where others have found fortunes i
treasure and gold, though it is rumored that some who enter are never
seen again.  Magic is said to work in the cave.  I will be your eyes
and hands.  Direct me with commands of 1 or 2 words.  I should warn
you that I look at only the first five letters of each word, so you'l
have to enter "northeast" as "ne" to distinguish it from "north".
You can type "help" for some general hints.  For information on how
to end your adventure, scoring, etc., type "info".
                              - - -
This program was originally developed by Willie Crowther.  Most of th
features of the current program were added by Don Woods.

You are standing at the end of a road before a small brick building.
Around you is a forest.  A small stream flows out of the building and
down a gully.


>
```

Figure 9: Adventure game in Linux - see `advent(6)`

13 LITPROG: PRINTOUT

Figure 10 shows the printout that corresponds to figure 9 (source).

14 LITPROG: STORY

- Automatic index of commands, variables, objects
- Index of subroutines, table of contents
- Support for digital, code-based storytelling

Figures 11 and 12 show part of the index and the table of contents of the `cweb` printout.

---

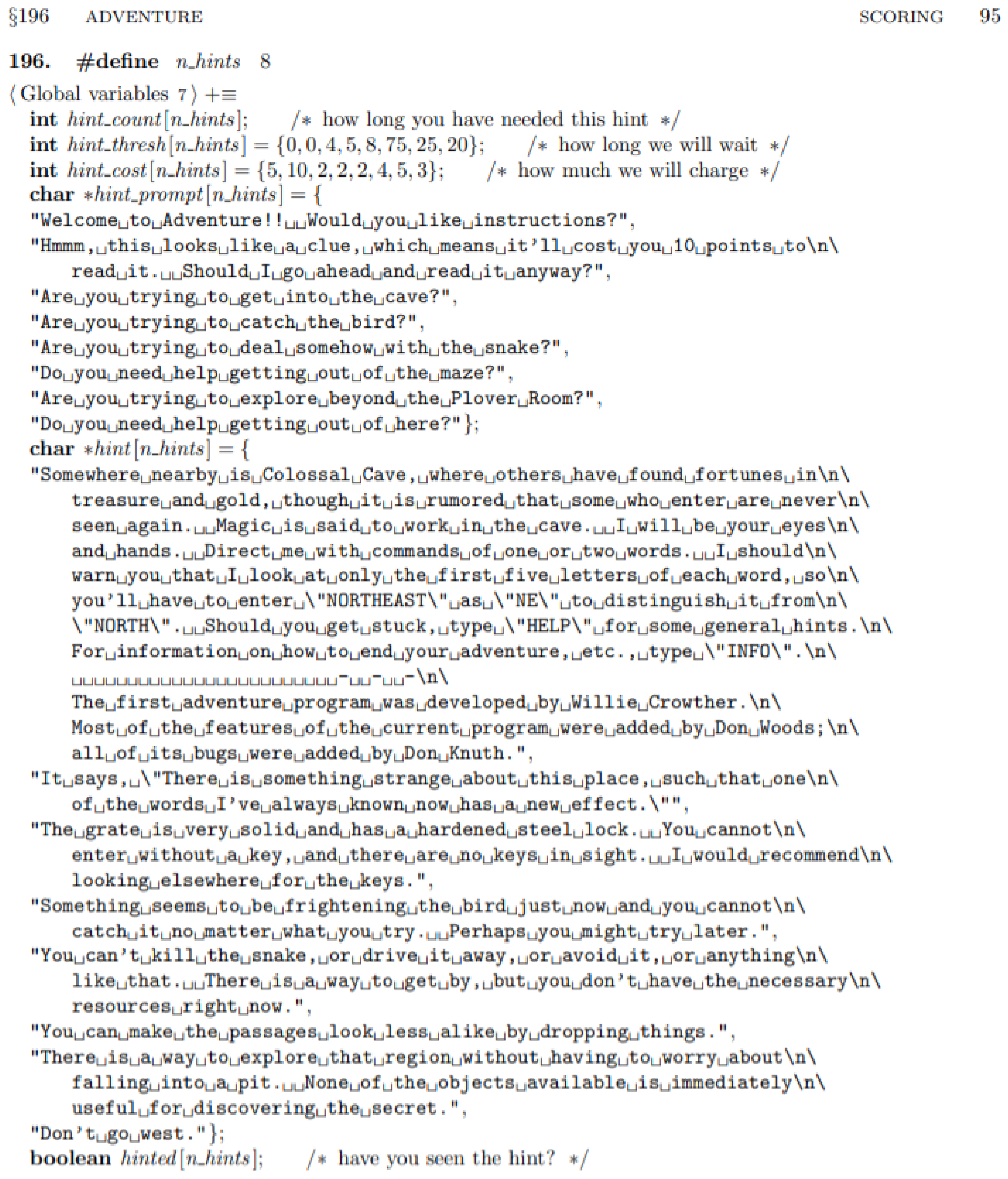
15 LITPROG PROS AND CONS

LITPROG PROS	**LITPROG *CONS**
Storytelling supported	Requires thought
Prettyprinting w/TEX	TEX difficult to learn
Automatic index/TOC	Requires (different) training
Free Software	Standardisation difficult

16 THE CASE FOR LITPROG

- Code and documentation in separate files and rarely synchronized,
- Variable names that are mnemonics and acronyms, not words,
- Documentation that is seldom created by the programmer, and
- Documentation that has a lower priority than the program.

See also: Childs, 2010:



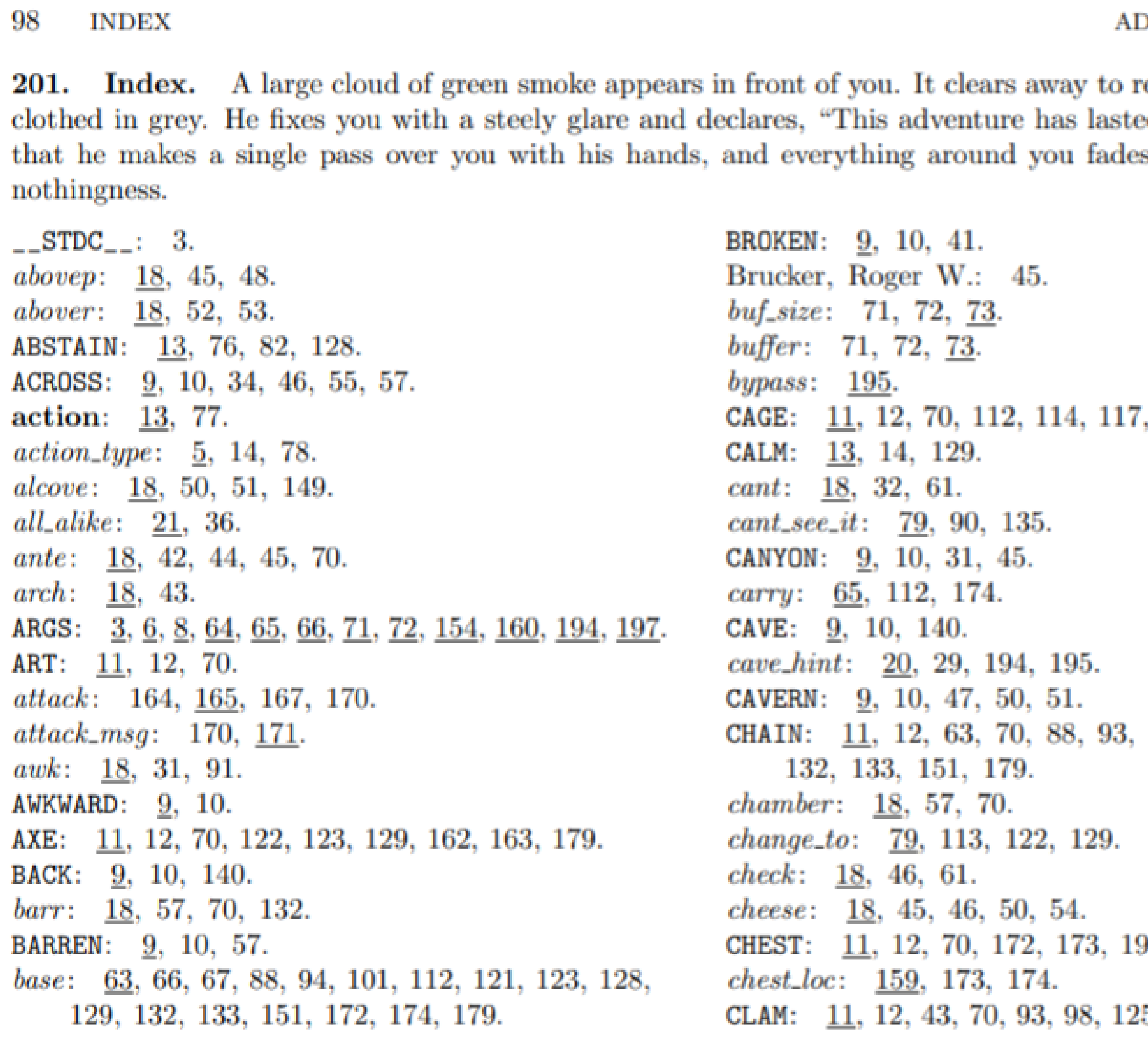
§196 ADVENTURE SCORING 95

196. **#define** *n_hints* 8

⟨ Global variables 7 ⟩ +≡

```
int hint_count[n_hints];   /* how long you have needed this hint */
int hint_thresh[n_hints] = {0, 0, 4, 5, 8, 75, 25, 20};   /* how long we will wait */
int hint_cost[n_hints] = {5, 10, 2, 2, 2, 4, 5, 3};   /* how much we will charge */
char *hint_prompt[n_hints] = {
"Welcome␣to␣Adventure!!␣␣Would␣you␣like␣instructions?",
"Hmmm,␣this␣looks␣like␣a␣clue,␣which␣means␣it'll␣cost␣you␣10␣points␣to\n\
    read␣it.␣␣Should␣I␣go␣ahead␣and␣read␣it␣anyway?",
"Are␣you␣trying␣to␣get␣into␣the␣cave?",
"Are␣you␣trying␣to␣catch␣the␣bird?",
"Are␣you␣trying␣to␣deal␣somehow␣with␣the␣snake?",
"Do␣you␣need␣help␣getting␣out␣of␣the␣maze?",
"Are␣you␣trying␣to␣explore␣beyond␣the␣Plover␣Room?",
"Do␣you␣need␣help␣getting␣out␣of␣here?"};
char *hint[n_hints] = {
"Somewhere␣nearby␣is␣Colossal␣Cave,␣where␣others␣have␣found␣fortunes␣in\n\
    treasure␣and␣gold,␣though␣it␣is␣rumored␣that␣some␣who␣enter␣are␣never\n\
    seen␣again.␣␣Magic␣is␣said␣to␣work␣in␣the␣cave.␣␣I␣will␣be␣your␣eyes\n\
    and␣hands.␣␣Direct␣me␣with␣commands␣of␣one␣or␣two␣words.␣␣I␣should\n\
    warn␣you␣that␣I␣look␣at␣only␣the␣first␣five␣letters␣of␣each␣word,␣so\n\
    you'll␣have␣to␣enter␣\"NORTHEAST\"␣as␣\"NE\"␣to␣distinguish␣it␣from\n\
    \"NORTH\".␣␣Should␣you␣get␣stuck,␣type␣\"HELP\"␣for␣some␣general␣hints.\n\
    For␣information␣on␣how␣to␣end␣your␣adventure,␣etc.,␣type␣\"INFO\".\n\
    ␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣-␣␣-␣␣-\n\
    The␣first␣adventure␣program␣was␣developed␣by␣Willie␣Crowther.\n\
    Most␣of␣the␣features␣of␣the␣current␣program␣were␣added␣by␣Don␣Woods;\n\
    all␣of␣its␣bugs␣were␣added␣by␣Don␣Knuth.",
"It␣says,␣\"There␣is␣something␣strange␣about␣this␣place,␣such␣that␣one\n\
    of␣the␣words␣I've␣always␣known␣now␣has␣a␣new␣effect.\"",
"The␣grate␣is␣very␣solid␣and␣has␣a␣hardened␣steel␣lock.␣␣You␣cannot\n\
    enter␣without␣a␣key,␣and␣there␣are␣no␣keys␣in␣sight.␣␣I␣would␣recommend\n\
    looking␣elsewhere␣for␣the␣keys.",
"Something␣seems␣to␣be␣frightening␣the␣bird␣just␣now␣and␣you␣cannot\n\
    catch␣it␣no␣matter␣what␣you␣try.␣␣Perhaps␣you␣might␣try␣later.",
"You␣can't␣kill␣the␣snake,␣or␣drive␣it␣away,␣or␣avoid␣it,␣or␣anything\n\
    like␣that.␣␣There␣is␣a␣way␣to␣get␣by,␣but␣you␣don't␣have␣the␣necessary\n\
    resources␣right␣now.",
"You␣can␣make␣the␣passages␣look␣less␣alike␣by␣dropping␣things.",
"There␣is␣a␣way␣to␣explore␣that␣region␣without␣having␣to␣worry␣about\n\
    falling␣into␣a␣pit.␣␣None␣of␣the␣objects␣available␣is␣immediately\n\
    useful␣for␣discovering␣the␣secret.",
"Don't␣go␣west."};
boolean hinted[n_hints];   /* have you seen the hint? */
```

Figure 10: Screenshot of Knuth's `cweb` version of `advent`.



98 INDEX AD

201. Index. A large cloud of green smoke appears in front of you. It clears away to r
clothed in grey. He fixes you with a steely glare and declares, "This adventure has laste
that he makes a single pass over you with his hands, and everything around you fades
nothingness.

`__STDC__`: 3.
abovep: 18, 45, 48.
abover: 18, 52, 53.
`ABSTAIN`: 13, 76, 82, 128.
`ACROSS`: 9, 10, 34, 46, 55, 57.
action: 13, 77.
action_type: 5, 14, 78.
alcove: 18, 50, 51, 149.
all_alike: 21, 36.
ante: 18, 42, 44, 45, 70.
arch: 18, 43.
`ARGS`: 3, 6, 8, 64, 65, 66, 71, 72, 154, 160, 194, 197.
`ART`: 11, 12, 70.
attack: 164, 165, 167, 170.
attack_msg: 170, 171.
awk: 18, 31, 91.
`AWKWARD`: 9, 10.
`AXE`: 11, 12, 70, 122, 123, 129, 162, 163, 179.
`BACK`: 9, 10, 140.
barr: 18, 57, 70, 132.
`BARREN`: 9, 10, 57.
base: 63, 66, 67, 88, 94, 101, 112, 121, 123, 128,
129, 132, 133, 151, 172, 174, 179.
`BROKEN`: 9, 10, 41.
Brucker, Roger W.: 45.
buf_size: 71, 72, 73.
buffer: 71, 72, 73.
bypass: 195.
`CAGE`: 11, 12, 70, 112, 114, 117,
`CALM`: 13, 14, 129.
cant: 18, 32, 61.
cant_see_it: 79, 90, 135.
`CANYON`: 9, 10, 31, 45.
carry: 65, 112, 174.
`CAVE`: 9, 10, 140.
cave_hint: 20, 29, 194, 195.
`CAVERN`: 9, 10, 47, 50, 51.
`CHAIN`: 11, 12, 63, 70, 88, 93,
132, 133, 151, 179.
chamber: 18, 57, 70.
change_to: 79, 113, 122, 129.
check: 18, 46, 61.
cheese: 18, 45, 46, 50, 54.
`CHEST`: 11, 12, 70, 172, 173, 19
chest_loc: 159, 173, 174.
`CLAM`: 11, 12, 43, 70, 93, 98, 12

Figure 11: Index for the "Adventure" game by (Crowther, 1975), Knuth (1998).

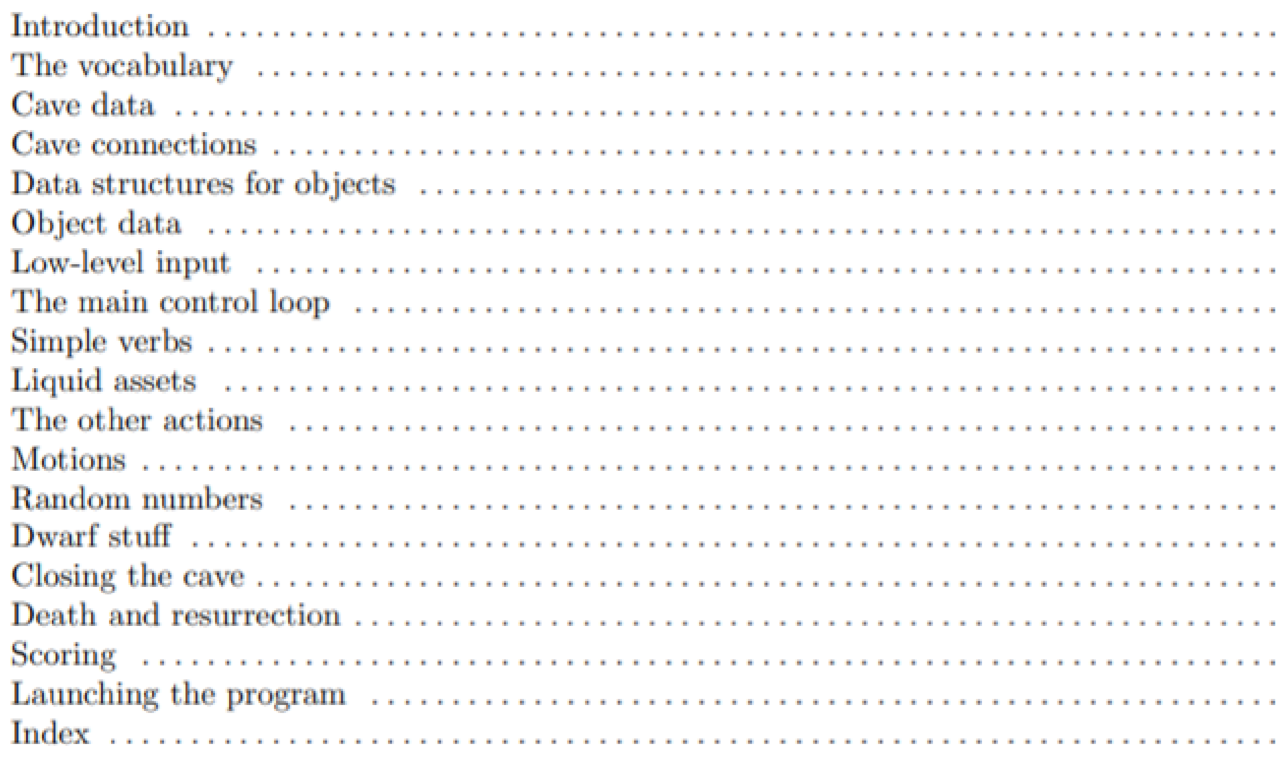


Figure 12: Table of Contents for the "Adventure" game (Crowther, 1975), Knuth (1998).

»It is commonly accepted in software engineering circles that one of the greatest needs in computing is the reduction of the cost of maintenance of codes. Maintenance programmers spend at least half of their time trying to understand what code does and maintenance is accepted to be 60% to 80% of a code's cost.«

17 MODERN APPLICATION EXAMPLES

- Extreme Programming (XP)
- Agile Modeling (AM)
- Interactive programming (see figure 13)

Figure 13 shows a computational IPython notebook from 2005. IPython is the precursor of Jupyter notebooks.

---

18 AGILE METHODOLOGIES

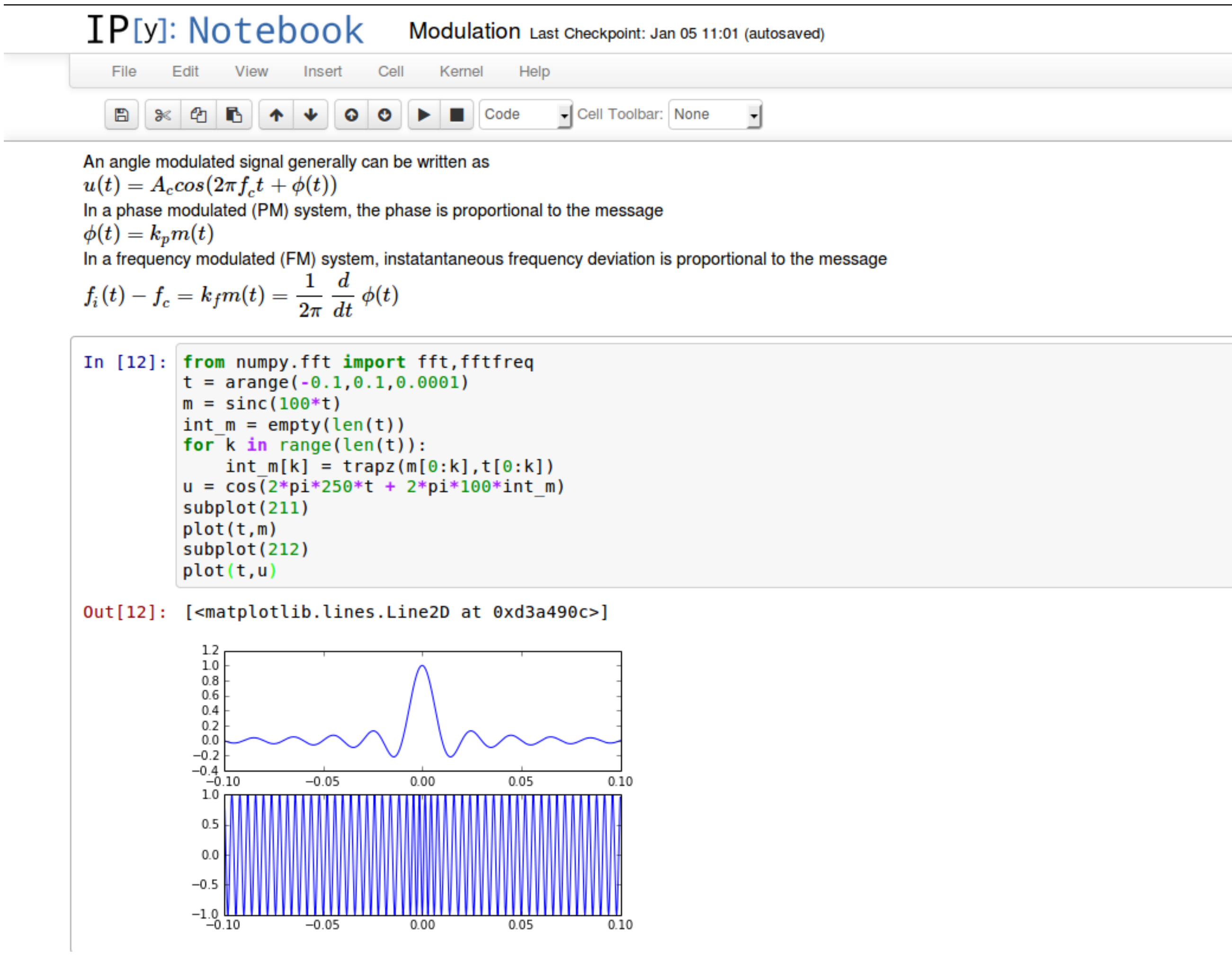
- Ways to develop and document anything
- Better suited for *complex* projects and *culturally* diverse teams
- Core value: optimize customer *communication*

Figure 14: Scrum is the best known agile methodology.

---

19 NOTEBOOK EXAMPLES

- Subsetting quiz as shiny app with `learnr` package
- GNA Internet Course on Literate Programming (1994)
- SQL cells in Deepnote (demo, 1 min)
- Kaggle notebook from Pima Indians database


IP[y]: Notebook
Modulation Last Checkpoint: Jan 05 11:01 (autosaved)
File Edit View Insert Cell Kernel Help
Code
Cell Toolbar: None

An angle modulated signal generally can be written as

$u(t) = A_c cos(2\pi f_c t + \phi(t))$

In a phase modulated (PM) system, the phase is proportional to the message

$\phi(t) = k_p m(t)$

In a frequency modulated (FM) system, instatantaneous frequency deviation is proportional to the message

$$f_i(t) - f_c = k_f m(t) = \frac{1}{2\pi}\frac{d}{dt}\phi(t)$$

```
In [12]: from numpy.fft import fft,fftfreq
         t = arange(-0.1,0.1,0.0001)
         m = sinc(100*t)
         int_m = empty(len(t))
         for k in range(len(t)):
             int_m[k] = trapz(m[0:k],t[0:k])
         u = cos(2*pi*250*t + 2*pi*100*int_m)
         subplot(211)
         plot(t,m)
         subplot(212)
         plot(t,u)

Out[12]: [<matplotlib.lines.Line2D at 0xd3a490c>]
```

Figure 13: IPython notebook. By Shishirdasika, CC BY-SA 3.0, via Wikimedia Commons

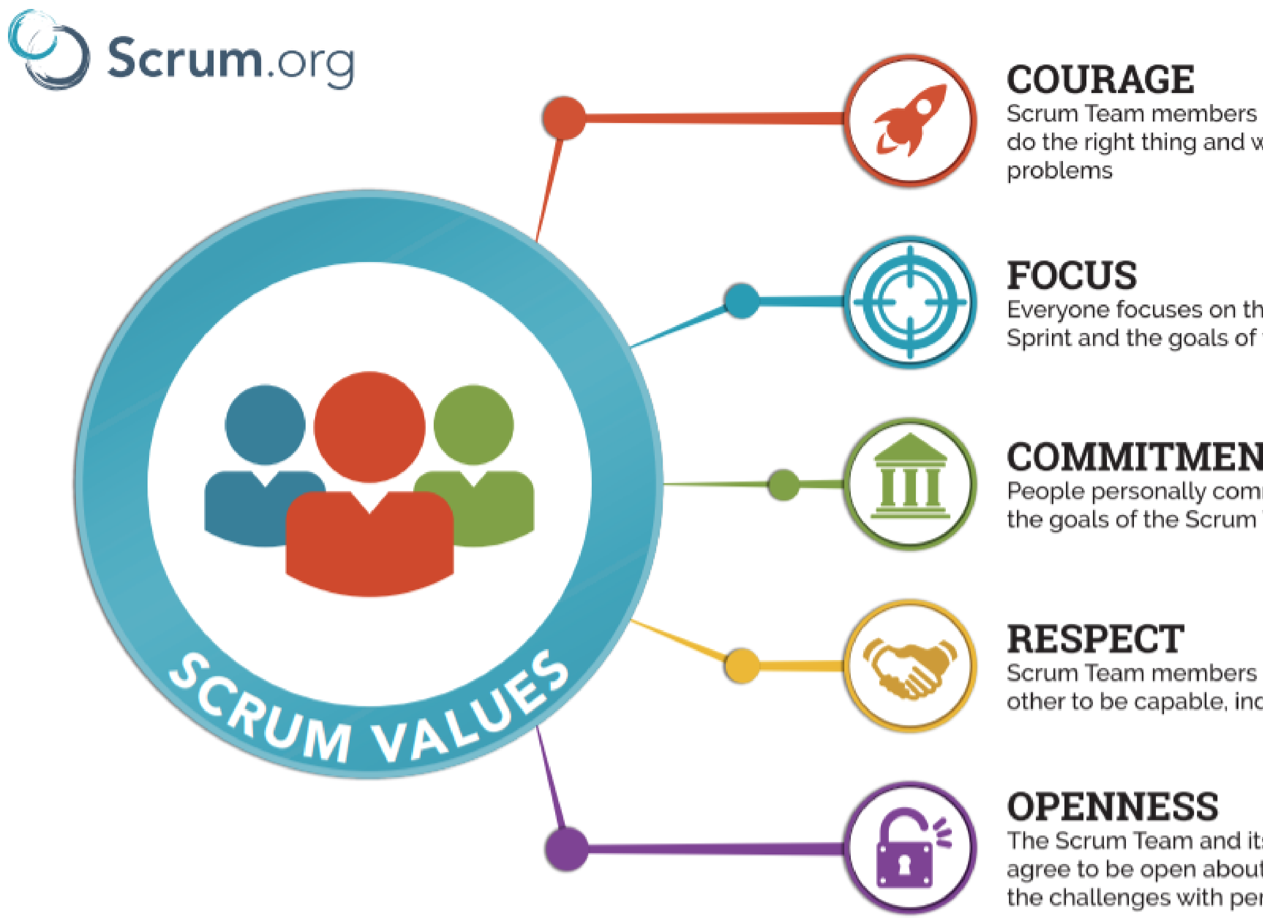


Figure 14: Scrum values (Source: scrum.org)

- Count cloud notebook
- Introduction to DataCamp projects (with R)
- R Markdown Outputs Gallery

20 NOTEBOOK TUTORIALS

- Tutorial: Jupyter and R Markdown: Notebooks with R (2016)
- Book: R notebook (bookdown)
- Article: R notebooks for dummies (2020)
- Course: Reporting with R Markdown (2020)
- Course: R Markdown from RStudio

21 ORG-MODE AGAIN

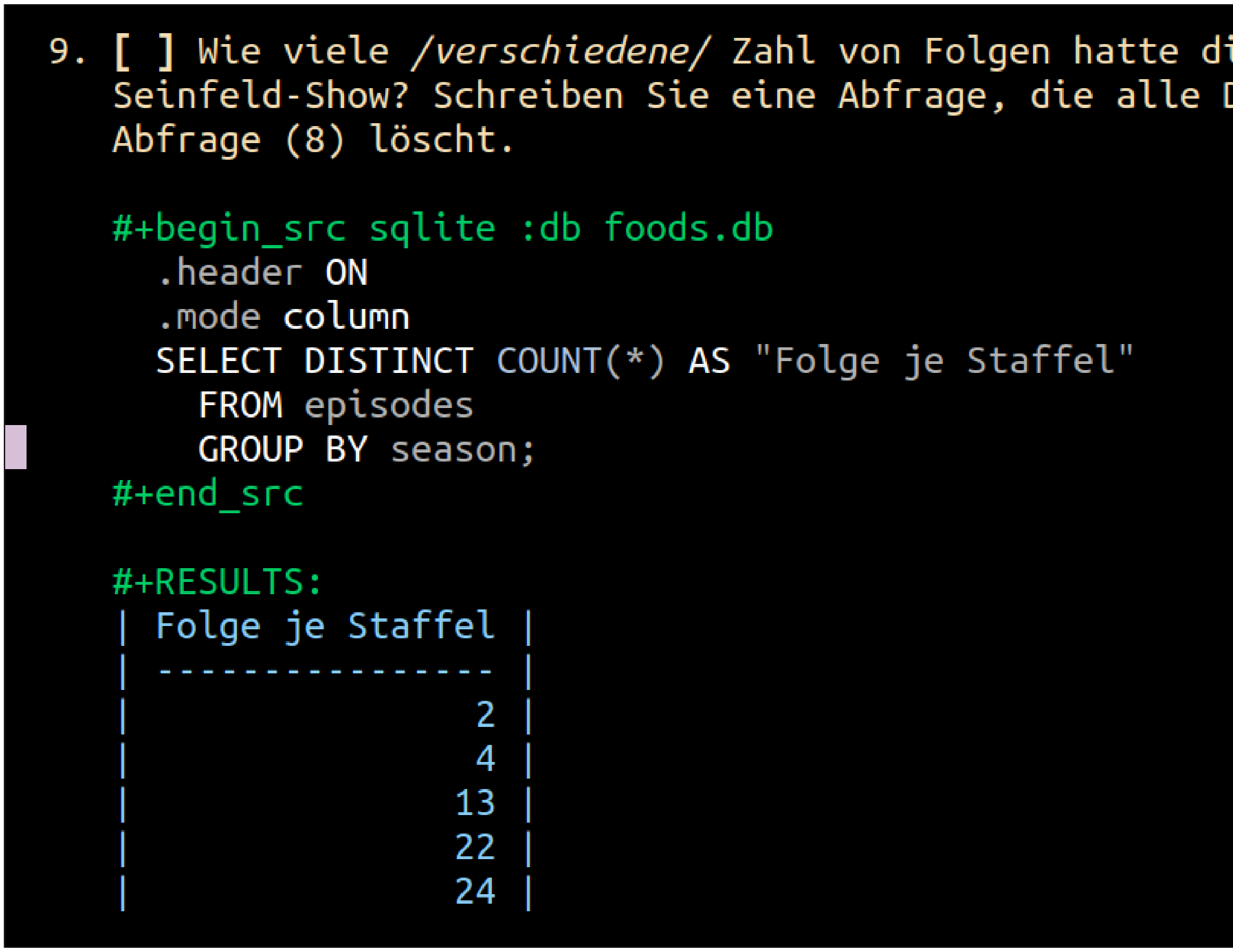
- Notebooks work with R, SQL, Python,... anything
- SQLite example (SQLite = SQL for IoT)
- Present, too, if you like
- R notebook example (print+plot)

Figure 15 shows an SQLite notebook example

---

22 NOTEBOOK DEMO (RSTUDIO CLOUD)

- EDA using the `Pima` Indian data set (via Matloff)
- Head over to this RStudio cloud notebook to start
- Compare your results with this solution (PDF)



```
9. [ ] Wie viele /verschiedene/ Zahl von Folgen hatte di
   Seinfeld-Show? Schreiben Sie eine Abfrage, die alle D
   Abfrage (8) löscht.

   #+begin_src sqlite :db foods.db
     .header ON
     .mode column
     SELECT DISTINCT COUNT(*) AS "Folge je Staffel"
       FROM episodes
       GROUP BY season;
   #+end_src

   #+RESULTS:
   | Folge je Staffel |
   | ---------------- |
   |                2 |
   |                4 |
   |               13 |
   |               22 |
   |               24 |
```

Figure 15: SQLite notebook example (Emacs/Org-mode)

Figure 16 shows a screenshot from the RStudio cloud workspace where we will practice R notebook creation and execution.

---

23 NOTEBOOK APPLICATIONS FOR YOU

- Emacs + ESS + Org-mode (Tutorial)
- RStudio notebooks
- Write your next paper or thesis as a "literate program"[4]

24 REFERENCES

(1) Donald E. Knuth, "Von Neumann's First Computer Program". Computing Surveys, 2(4), 1970.

(2) John Gruber, "Markdown: Syntax". Blog. daringfireball.net

(3) Donald E. Knuth and Silvio Levy, "The `CWEB` System of Structured Documentation", 2002. Manual. literateprogramming.com

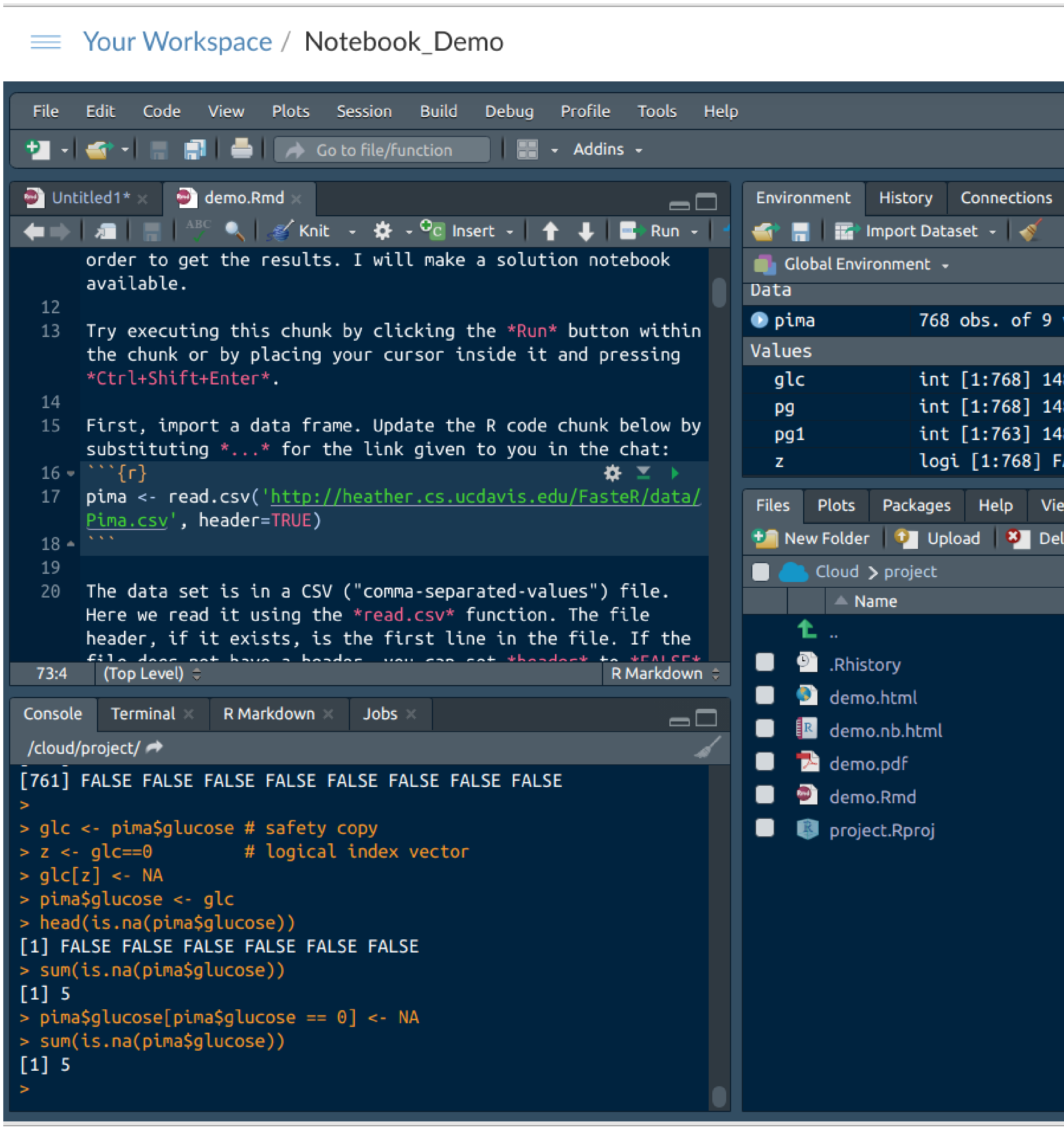
(4) Don Woods and Don Knuth, 1998.

(5) Bart Childs, "Literate Programming, Why?" (n.d.). literateprogramming.com

(6) Bart Childs, "Thirty years of literate programming and more?". TUGboat, Volume 31(2), 2010:183-188.

(7) Zachary M. Smith, "R Markdown Crash Course", 2020-03-02. github.com

[4]Remember: litprog means "documentation first" - this is data-driven storytelling from the story rather than from the data end - much easier and much more likely to result in a good story!


Your Workspace / Notebook_Demo
File Edit Code View Plots Session Build Debug Profile Tools Help
Go to file/function
Addins
Untitled1*
demo.Rmd
Knit
Insert
Run
Environment History Connections
Import Dataset
Global Environment
Files Plots Packages Help
New Folder Upload
Cloud > project
Name
73:4 (Top Level) R Markdown
Console Terminal R Markdown Jobs
/cloud/project/

~~~
order to get the results. I will make a solution notebook
available.

Try executing this chunk by clicking the *Run* button within
the chunk or by placing your cursor inside it and pressing
*Ctrl+Shift+Enter*.

First, import a data frame. Update the R code chunk below by
substituting *...* for the link given to you in the chat:
```{r}
pima <- read.csv('http://heather.cs.ucdavis.edu/FasteR/data/
Pima.csv', header=TRUE)
```

The data set is in a CSV ("comma-separated-values") file.
Here we read it using the *read.csv* function. The file
header, if it exists, is the first line in the file. If the
~~~

```
Data
pima        768 obs. of 9
Values
glc         int [1:768] 14
pg          int [1:768] 14
pg1         int [1:763] 14
z           logi [1:768] F
```

```
..
.Rhistory
demo.html
demo.nb.html
demo.pdf
demo.Rmd
project.Rproj
```

```
[761] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
>
> glc <- pima$glucose # safety copy
> z <- glc==0         # logical index vector
> glc[z] <- NA
> pima$glucose <- glc
> head(is.na(pima$glucose))
[1] FALSE FALSE FALSE FALSE FALSE FALSE
> sum(is.na(pima$glucose))
[1] 5
> pima$glucose[pima$glucose == 0] <- NA
> sum(is.na(pima$glucose))
[1] 5
>
```

Figure 16: RStudio cloud workspace with R notebook demo.