Reproducible Research Tools for Creating Books

Max Kuhn

Pfizer Global R&D, Groton, CT

Introduction

In 2010, Kjell Johnson and I started working on a book for Springer called *Applied Predictive Modeling*.

The book is a practitioner's guide to building machine learning models for regression and classification.

On thing that frustrated me about other books on this subject is that they were either:

- highly theoretical with no guidance on how do *do* this type of modeling or
- essentially software manuals with no insight into the strengths and weaknesses of of the models

We wanted to be somewhere in–between: intuitive descriptions, strategies for creating good models and some software to build them.

Data and Reproducibility

One of the most frustrating aspect to most books is the lack of public data and software.

- This is especially true outside of biology, chemistry and a few other sciences. Little public data exists for marketing and business applications.
- Additionally, most used the same data sets (i.e. Box's "data grave–robbers") that were dissimilar from real problems in the wild.

We wanted readers to be able to reproduce *almost* every analysis in the text, as well as extending our approach and easily trying other techniques.

In comparison to other texts, readers are also more likely to find more errors. My ego has been suitably prepared.

Other Aspects

Kjell and I are in different locations (Connecticut and Michigan).

Many of the models are computationally expensive and several of the data sets are moderate to large (in predictors and/or data points).

The tools we used were:

- R, `Sweave` and Bioconductor's weaver package for computations
- Springer's LaTeXstyles
- `make` to coordinate everything
- The multicore package for parallel processing with two Mac Pros (32 GB each, 8 cores supporting 16 distinct processes)
- tikz and R's tikzDevice package
- Dropbox
- Papers for OS X (for research and `bibtex` management)

Dropbox

In leu of a proper version control system, Dropbox was used. “Check–ins” are automatically occur every time a file is saved.

This was effective at keeping our local computers synced as well as dealing with conflicting file saves.

However, we did have to coordinate our activities to be sure that we did not edit the same file at the same time.

Dropbox is able to reconstruct previous versions (but has no diff mechanism built in).

Surprisingly, my company’s IT group has not blocked the relevant port(s).

make

A makefile was used to coordinate the system level activities.

The book eventually had 20 chapters and *very few inter–chapter dependencies*.

Targets were created for each chapter's `tex` file, the `bibtex` file, the index files and the final pdf.

Most targets looked like this:

```
RegPerformance.tex: RegPerformance.Rnw
  @date '+ %Y-%m-%d %H:%M:%S: Building Regression Performance chapter'
  @bash weaver.sh RegPerformance > RegPerformance.Rout 2>&1
  @$(RCMD) Stangle RegPerformance.Rnw
```

More on `weaver.sh` in a minute.

An `achive` target as used to create a zip archive of the current version.

LaTeX and Sweave

Sweave was used to combine the LaTeX markup with R code to create the models, figures, tables etc. For example:

```
The MARS prediction equation for the \Sexpr{nrow(trees)} tree samples was
<<MarsEq, echo = FALSE, results = tex>>=
library(earth)
fit <- earth(Volume ~ ., data = trees)
cat(marsEQ(fit))
@
```

results in:

The MARS prediction equation for the 31 tree samples was:

$$
\begin{aligned}
27 &+ 6.2 \times h(Girth - 14) \\
&- 3.3 \times h(14 - Girth) \\
&+ 0.49 \times h(Height - 72)
\end{aligned}
$$

Translations

We can't automate the contextual implications of our analyses, but some of the text can be made cleaner via code.

For example, more informative names can be developed and matched to variable names (e.g. `NumCI` = "number of chief investigators"). In this way, this:

```
ContractValueBand = J AND
CI.Dept2713 <= 0: unsuccessful (20.0/1.0)
```

becomes

> "If Contract Value Band J and no chief investigators from Department 2713 then the results were unsuccessful"

in the text.

LaTeX and Sweave

The `latex` function in Frank Harrell's Hmisc package was enormously helpful as was the LaTeX version of his `describe` function (very cool).

There are a variety of other tools cataloged at the CRAN Task View for Reproducible Research:

http://cran.r-project.org/web/views/ReproducibleResearch.html

```
latex(describe(iris[,3:5]), file = "")
```

3 Variables 150 Observations

Petal.Length

n	missing	unique	Mean	.05	.10	.25	.50	.75	.90	.95
150	0	43	3.758	1.30	1.40	1.60	4.35	5.10	5.80	6.10

```
lowest : 1.0 1.1 1.2 1.3 1.4, highest: 6.3 6.4 6.6 6.7 6.9
```

Petal.Width

n	missing	unique	Mean	.05	.10	.25	.50	.75	.90	.95
150	0	22	1.199	0.2	0.2	0.3	1.3	1.8	2.2	2.3

```
lowest : 0.1 0.2 0.3 0.4 0.5, highest: 2.1 2.2 2.3 2.4 2.5
```

Species

n	missing	unique
150	0	3

```
setosa (50, 33%), versicolor (50, 33%), virginica (50, 33%)
```

Sweave and Caching

One issue was that some models took days to complete (even using multiple cores) and some took a few trial–and–error attempts to optimize.

There are a few packages that can be used to *cache* the results from Sweave's code chunks. Two (pre–knitr) packages are cacheWeave and Bioconductors's weaver. These would:

- Save the R objects generated by a code chunk and the R code
- When the Sweave file is re–run and the code chunk (or the dependencies) have not been modified, it loads the previous version.

weaver seemed to have the most functionality.

With some work, we can recover weaver–cached objects.

Sweave and Caching

There were some issues though:

- Some code chunks would always re–run even without any dependencies. This occurred with to $< 5\%$ of the chunks.
- A cached code chunk could not be printed (a minor issue)

Our strategy was to build and initial version of the book using caching. The final version would be created from non–cached versions of the code chunks.

In R:

```
library(weaver)
Sweave("RegPerformance.Rnw", driver = weaver())
```

This necessitated `weaver.sh`:

```
#!/bin/bash
echo "library(weaver);Sweave(\"$1.Rnw\",driver=weaver())"\ | R --no-save --no-resto
```

Parallel Processing

Many of the computations involved cross–validation, bootstrapping and other "embarrassingly parallel" operations.

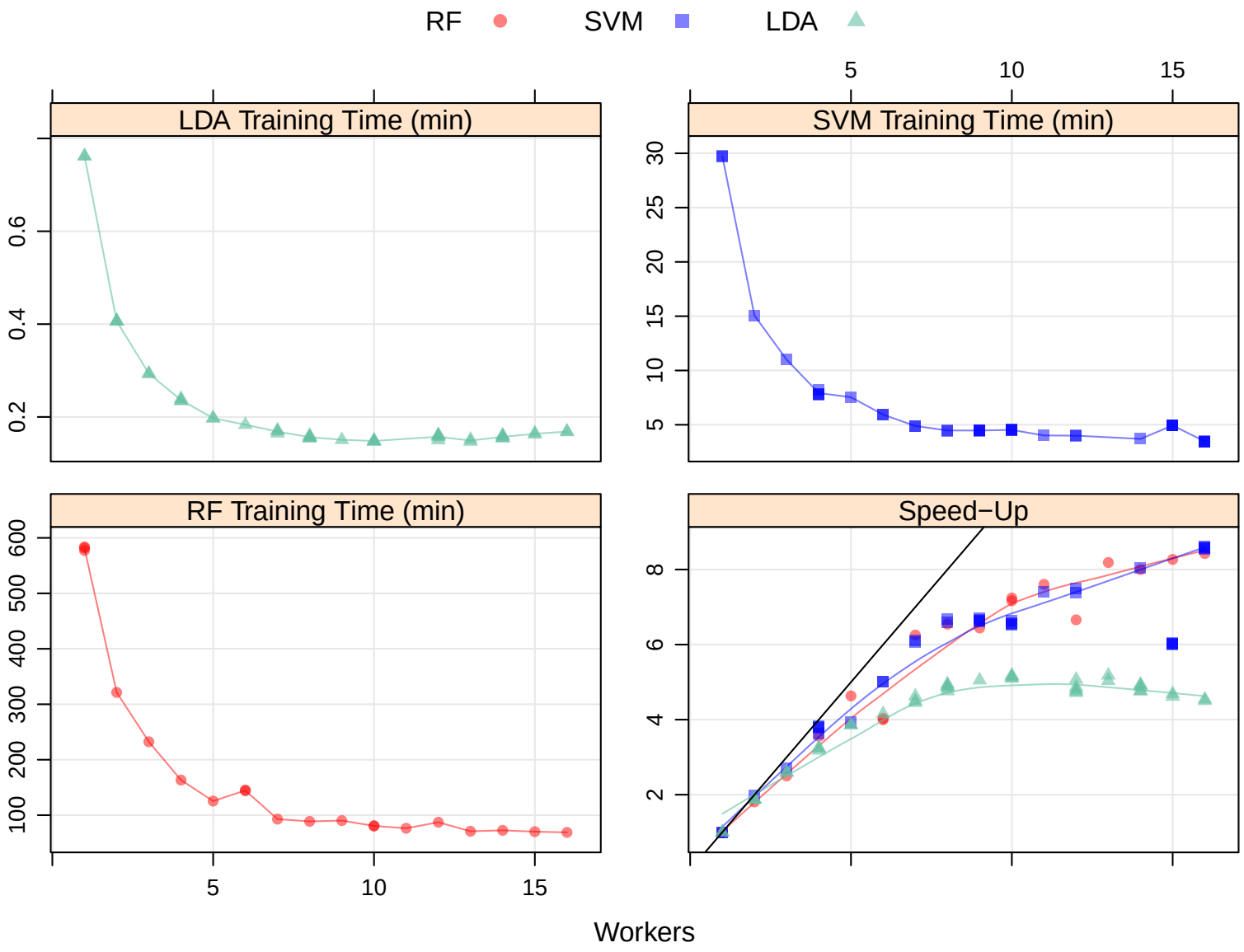
R has many options for *explicit parallelism* and my packages (e.g. caret) utilize these with the foreach package in conjunction with the multicore package.

Using $M = 10$–15 "workers" dramatically reduced the time to build models.

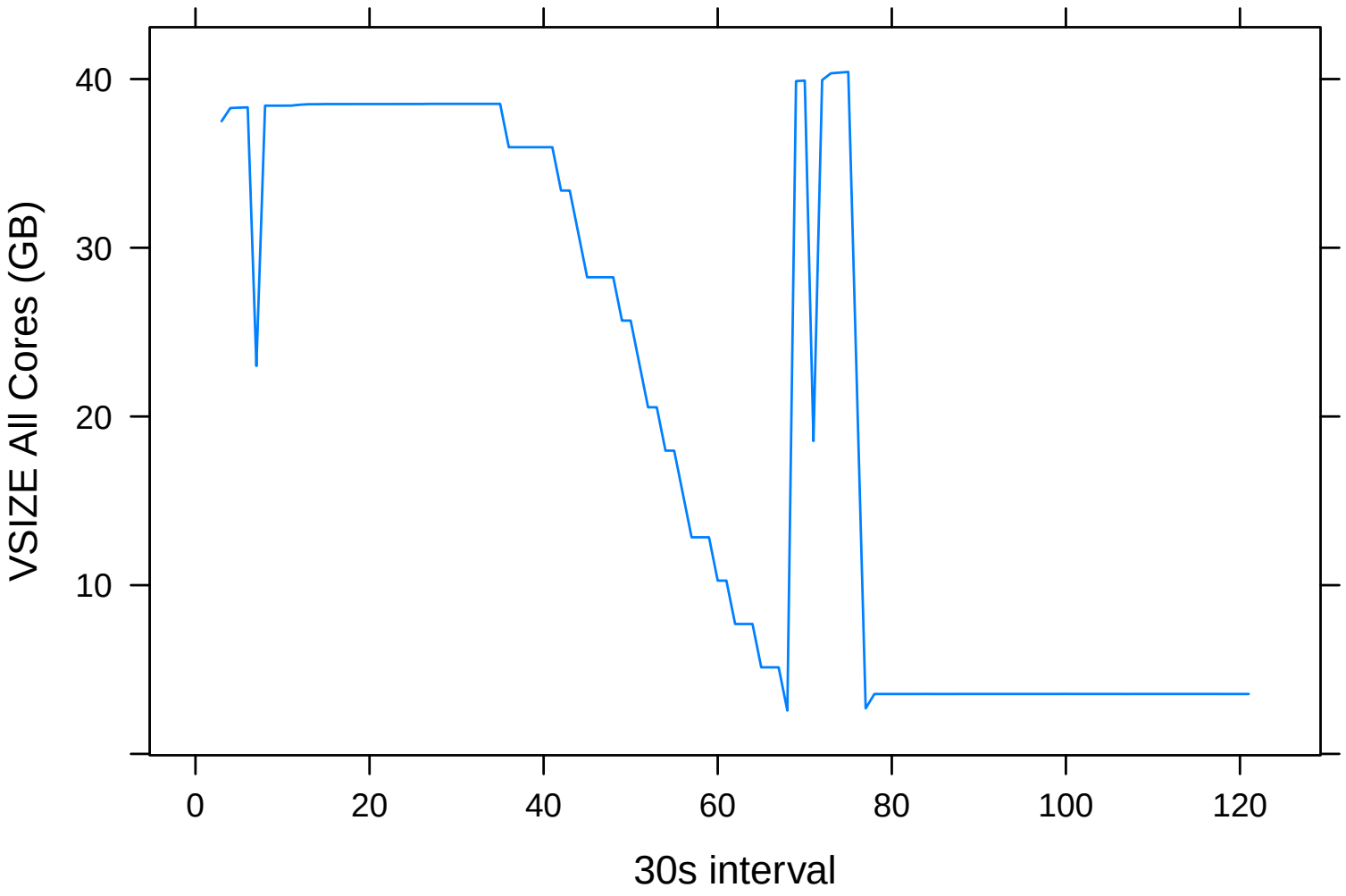
However, with M workers, there are $M + 1$ copies of the data in memory. In some cases (e.g. the party package's `cforest` function) only $M = 3$ workers could be used.

Since we used parallelism was used within each chapter's `Rnw` file, `make` was not invoked in parallel (i.e. `make -J 8`).

Parallel Processing



Parallel Processing Memory Requirements



LaTeX Tools and Tikz

The `latex` function in Frank Harrell's Hmisc package was enormously helpful as was the LaTeXversion of his `describe` function (very cool).

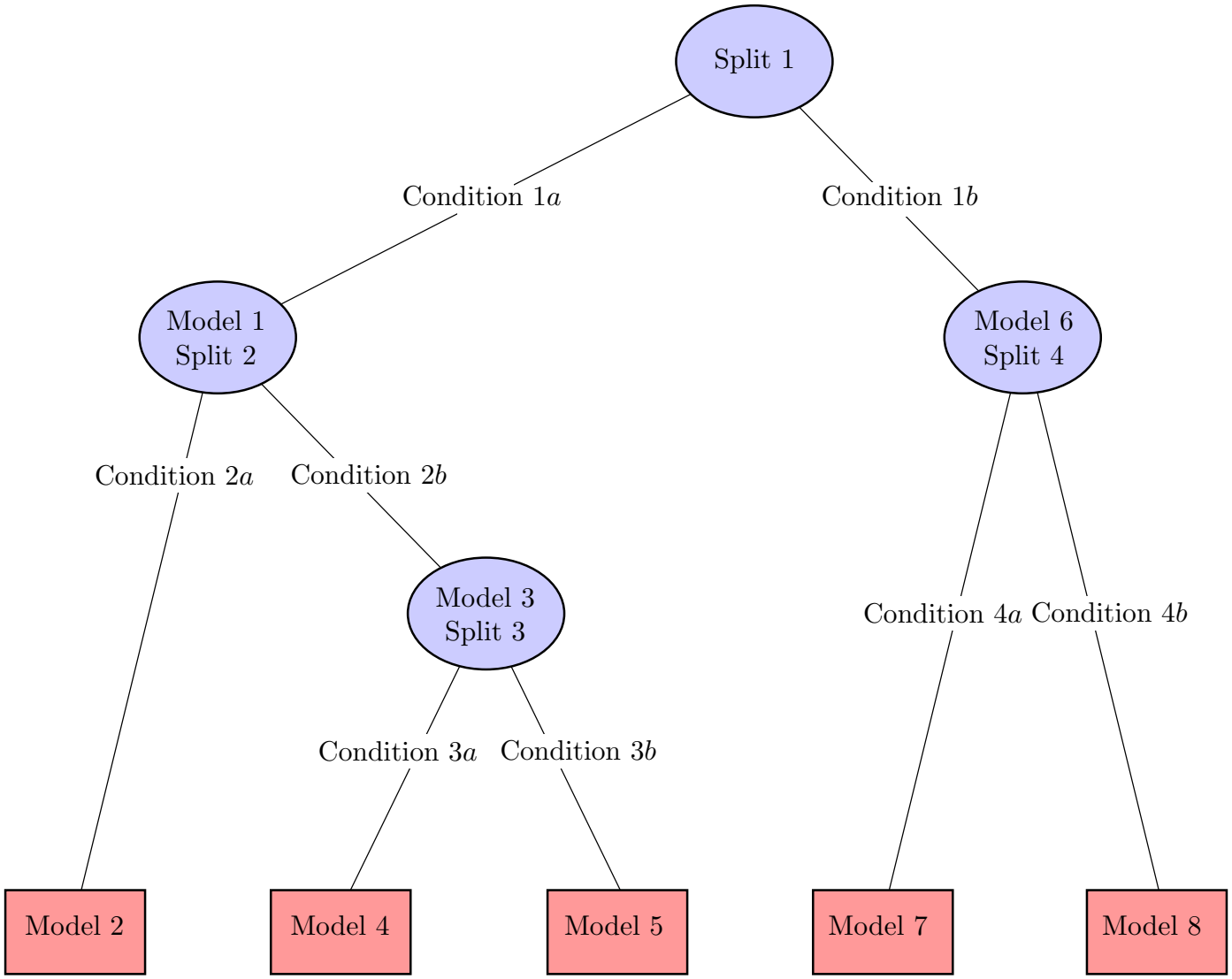
LaTeXhas a module called Tikz that enables high quality vector graphics.

There were a few concepts that we wanted to illustrate and Tikz was a good choice.

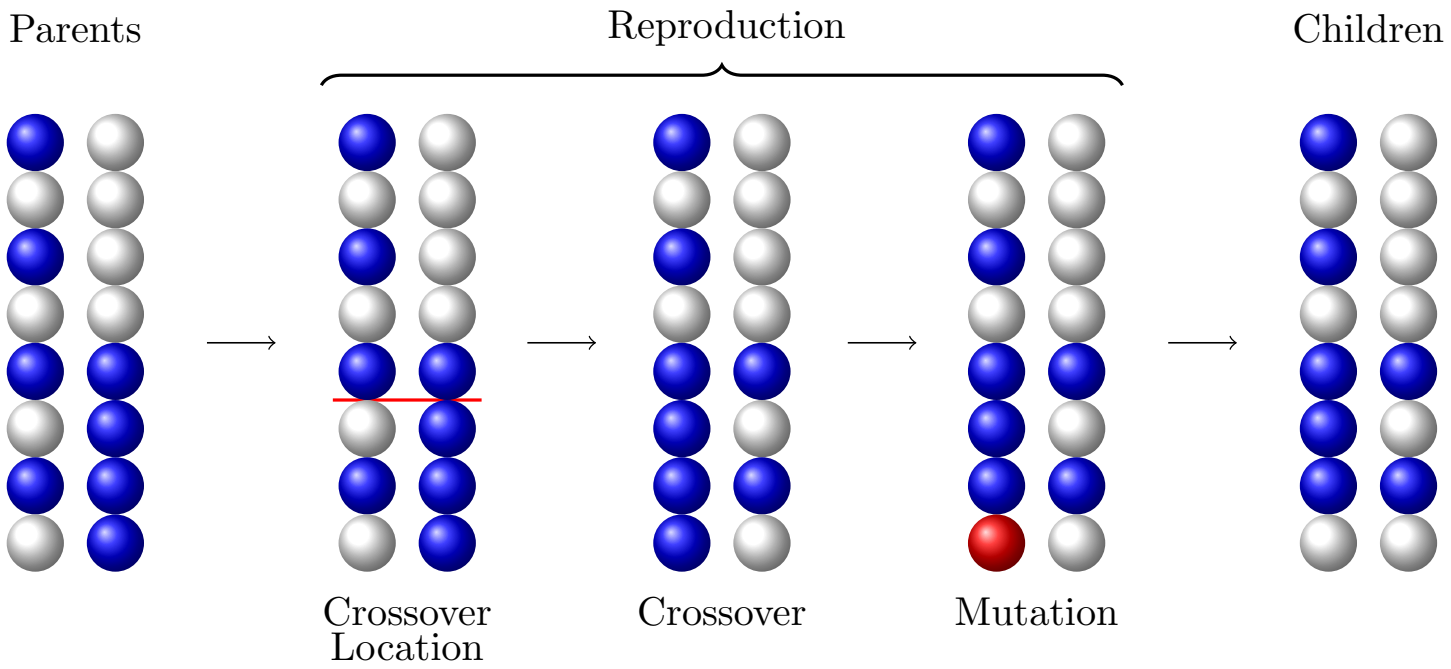
Additionally, R has a package called tikzDevice that allowed us to create a figure close to what we wanted in R, then edit the LaTeXmarkup to make it look exactly how we wanted.

Examples:

Model Tree Diagram Adapted from `plot.party`



Genetic Algorithm Diagram From Scratch



Minor ESS Issues

I used emacs with ESS and sometimes...

```
When you create a paragraph and get too far from the right--
hand side, ESS automatically adds a carriage  return after
the next space is added. This works great until \Sexpr{x
  == 2}.
```

This was frustrating and we tried to eliminate white space in `Sexpr`.

After many attempts, I gave up on modifying the ESS arguments to prevent this from happening.

What Would We Have Done Differently?

After we started, knitr was released and may have a better set of integrated features.

It allows for caching and printing of objects from cached code chunks.

One down–side of add–on packages for `Sweave` is that you can only use one at a time (e.g. `driver = weaver()`).

Dropbox was very effective, but a proper version control system is probably a better idea.

Thanks

- Kjell Johnson
- Springer, specifically Marc Strauss and Hannah Bracken
- Fritz Leisch for `Sweave`
- Frank Harrell for Hmisc