Diagnosing Models and Code in R

Andrew Redd, PhD. VINCI Helpdesk R Expert

VINCI CyberSeminar 2/18/2021

Notes

- **R + Statistics + Artistry**
- **tidyverse** is always loaded and used
  - **dplyr** – data manipulations
  - **purrr** – map reduce operations
  - **magrittr** – the pipe to chain operations together.
    - **%>%** pass this (result of left-hand side) into that (right-hand side function)

Poll – What is your experience with R?

- **Expert**: I should be giving this presentation. ;-)
- **Practitioner**: R is my usual go to program for analysis. I am comfortable writing functions, using packages, modeling, graphs, etc.
- **Journeyman**: I'm mostly comfortable with R, but there are things I prefer to do in other languages.
- **Learner**: I use R for one or two things, but it is not my usual software.
- **Noob**: What's R?

An Example

- Modeling death after discharge
- 1,192,486 records with 21,890 deaths
- Generalized linear mixed effects model

Poll 2 –
What is a typical size datasets to you?

- <10, forget statistics I'm into case studies.
- Tens (11-99)
- Hundreds (100-1,000)
- Thousands (1,000-10,000)
- Tens of thousands (10,000-100,000)
- Hundreds of thousands (100,000 – 999,999)
- Millions
- Billions
- More?

How Long?
`system.time()`

First step is usually to know how long things take, enter `system.time()`.

R

```
library(lme4)
run.time <- system.time(
     model.fit <- data %>%
          sample_n(1e5) %>% #< only a ‘reasonable amount’
          glmer( model.formula, data=., family=binomial()
               , control = glmerControl(optimizer=“bobyqa”, optCtrl=list(maxfun=2e5)
               )
)
```

How Long?
Output

First step is usually to know how long things take, enter `system.time()`.

R

```
run.time
```

Output

```
    user   system  elapsed
12451.32  1899.34 14377.97
```

How Long?
Interpretation

Output

```
    user   system  elapsed
12451.32  1899.34 14377.97
```

- How do we read this?
  - **User** (approx. 3.5 hours) is the time of the actual computations.
  - **System** (approx. 30 minutes) is the time of system operations such as disk IO.
  - **Elapsed** (Approx. 4 hours) is the total time elapsed.
  - There are two other values that are present but usually not shown that relate to child processes.

How Long?
Useful trick

First step is usually to know how long things take, enter `system.time()`.

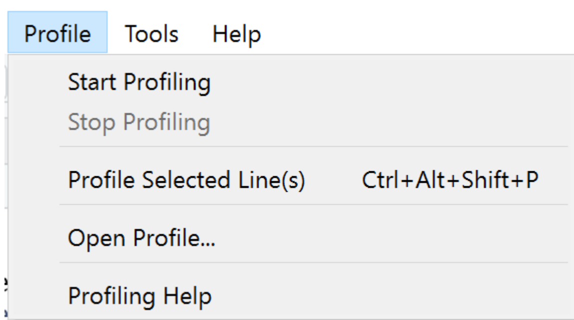
R

```
run.time %>% unclass %>% lubridate::dseconds()
```

Output

```
“12451.32s (~3.46 hours)” “1899.34s (~31.66 minutes)” “14377.97s
(~3.99 hours)”  NA  NA
```

A better way – Profiling



- Profiling gives a detailed view of what is happening in the code.

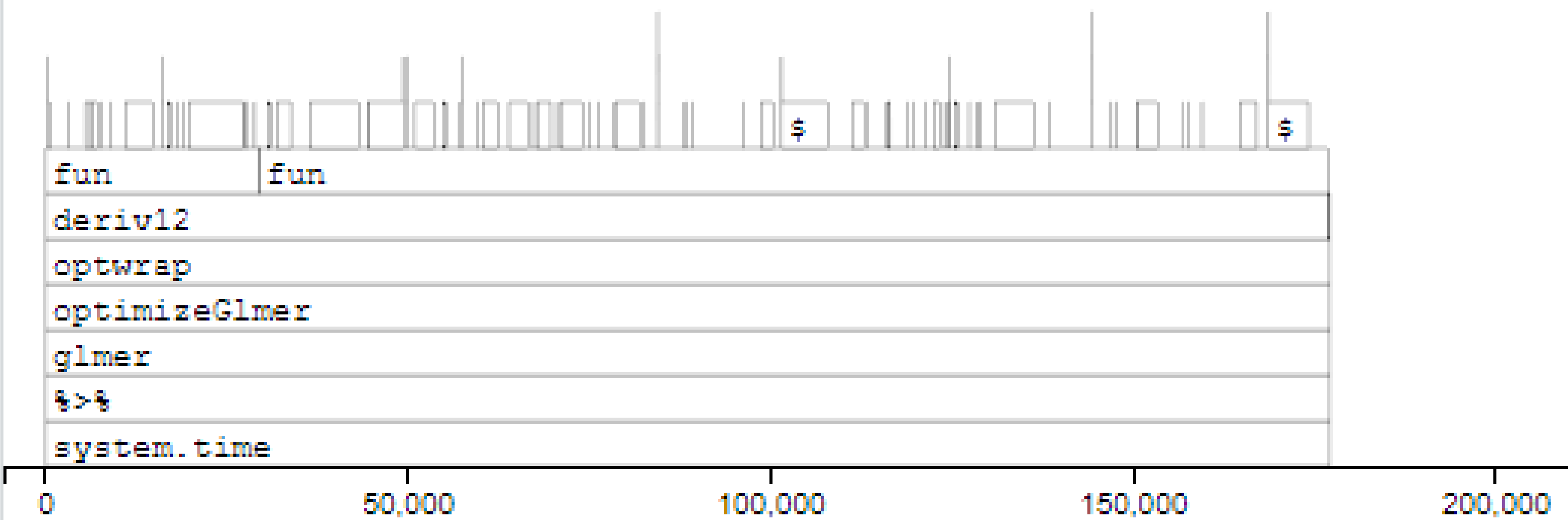
Profiling

- When you run profiling you will know by the profiling icon (clock) and the extra option on the console pane to stop profiling.

- After running a report will be displayed

Profiling – Flame Graph

- Flame graphs show function calls as horizontal bars
- The higher the bar the deeper the call
  - Ex. glmer calls optimizeGlmer which in turn calls optwrap, and so on



Profiling – Data Table

- The data table summarizes the time that is spent in each call at each level.

Flame Graph | Data | Options ▾

Code	File	Memory (MB)		Time (ms)
▼ system.time		-742.3	373.2	177020
▼ %>%		-742.3	373.2	177020
▼ glmer		-742.3	373.2	177020
▼ optimizeGlmer		-742.3	373.2	177020
▼ optwrap		-742.3	373.2	177020
▼ deriv12		-742.3	372.9	176920
▼ fun		-742.3	372.9	176900
resp$updateWts		0	0.0	41920
resp$updateMu		0	0.1	17020
▶ $		0	0.2	12180
resp$setOffset		-693.1	1.1	200
▶ pwrssUpdate		0	0.7	60
pp$setTheta		0	0.0	20
spos		0	0.0	20
fn		0	0.2	100
profvis::profvis		-269.7	0.1	720

Sample Interval: 20ms | 291040ms

Problem with fit

- In this case the hessian is degenerate, i.e. Not full rank, i.e. there are extraneous variables in the formula.
  - Check the variables:

R

```
model.fit %>% model.frame %>% summary
```

Output (abbreviated to relevant)

```
 i_rurality_hr     drivetimesc          i_ED       
 Min.   :0.0000   Min.   :  0.00   Min.   :0.0000  
 1st Qu.:0.0000   1st Qu.: 12.00   1st Qu.:1.0000  
 Median :0.0000   Median : 22.00   Median :1.0000  
 Mean   :0.0014   Mean   : 31.37   Mean   :0.9811  
 3rd Qu.:0.0000   3rd Qu.: 41.00   3rd Qu.:1.0000  
 Max.   :1.0000   Max.   :322.00   Max.   :1.0000  

    BedCount        OccuRate        AcademicAffil   
 Min.   : 10.0   Min.   :0.08767   Min.   :0.0000  
 1st Qu.:109.0   1st Qu.:0.64061   1st Qu.:1.0000  
 Median :137.0   Median :0.72037   Median :1.0000  
 Mean   :145.2   Mean   :0.70026   Mean   :0.9954  
 3rd Qu.:166.0   3rd Qu.:0.77462   3rd Qu.:1.0000  
 Max.   :279.0   Max.   :1.03585   Max.   :1.0000  
```

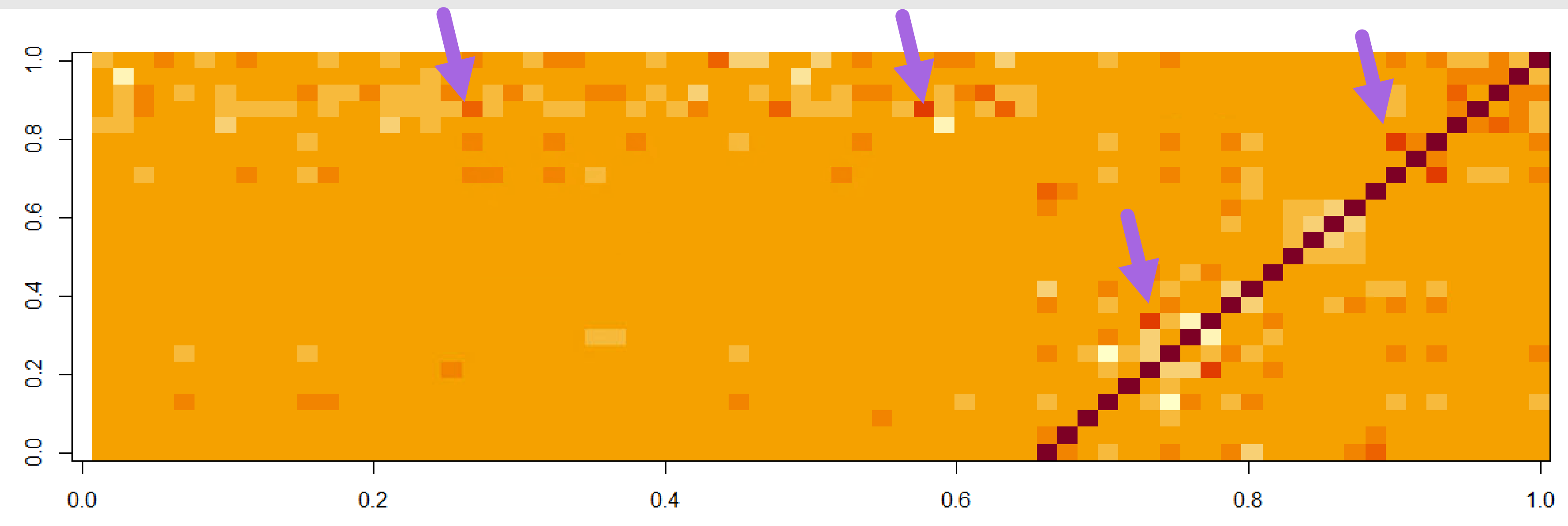
Correlation Matrix

- Examining the correlation matrix can reveal where there are deficiencies

R

```
cmat <- model.fit %>% model.matrix %>% cor
Cmat[,48:72] %>% image
```

Output (Standard graphics plot)



Find the highest correlations

- Many ways to do this, this is my way.

R

```
cmat %>% as.data.frame %>%  #< Matrix to data frame
    dplyr::add_rownames() %>% #< add  the row names as a column
    tidyr::pivot_longer(-1, 'colname') %>% #< make longer format
    filter(match(rowname, row.names(cmat)) < match(colname, row.names(cmat))) %>%
        #^ filter out duplicates
    arrange(desc(value))
```

```
# A tibble: 370 x 3
   rowname                    colname                       value
   <chr>                      <chr>                         <dbl>
 1 i_rurality_r               drivetimesc                   0.476
 2 race_nUnknown/Missing      ethnicity_nUnknown/Missing    0.427
 3 Age                        VW_SCORE                      0.316
 4 i_ED                       OccuRate                      0.302
 5 OccuRate                   AcademicAffil                 0.224
 6 i_ED                       BedCount                      0.191
 7 Age                        gender_nMALE                  0.185
 8 OccuRate                   Likelihood                    0.181
 9 ethnicity_nUnknown/Missing MaritalStatus_nUNKNOWN/MISSING 0.167
10 race_nWhite                i_rurality_r                  0.155
# ... with 360 more rows
```

Variance Inflation Factors

- Another option to identify problematic variables is to use the Variance Inflation Factor (VIF)

R

```
car::vif(model.fit)
```

Output

```
                          GVIF Df GVIF^(1/(2*Df))
DischYr           1.313190e+00  1        1.145945
sta6a             4.471407e+10 45        1.313222
Age               1.123859e+00  1        1.060122
gender_n          1.022122e+00  1        1.011000
race_n            2.022680e+00  5        1.072983
ethnicity_n       1.528088e+00  2        1.111827
MaritalStatus_n   1.362427e+00  3        1.052896
PriorityStatus_n  1.143252e+00  4        1.016875
VW_SCORE          1.073885e+00  1        1.036284
i_rurality_r      1.718008e+00  1        1.310728
i_rurality_hr     1.132329e+00  1        1.064109
drivetimesc       1.697575e+00  1        1.302910
i_ED              4.535656e+06  1     2129.707950
BedCount          8.568441e+01  1        9.256588
OccuRate          4.968718e+00  1        2.229062
AcademicAffil     3.156436e+00  1        1.776636
Likelihood        3.884546e+00  1        1.970925
```

Standardize Variables

- Fitting Algorithms don't like large differences in scale of fitting variables, particularly generalized versions.
- Example, Age_std is the centered(mean=0) and re-scaled(sd=1) version of Age

R

```
system.time(glmer(death~Age, data=mf2, family=binomial()))
```

Output

```
 user system  elapsed
83.36   6.03    90.97
```

R

```
system.time(glmer(death~Age_std, data=mf2, family=binomial()))
```

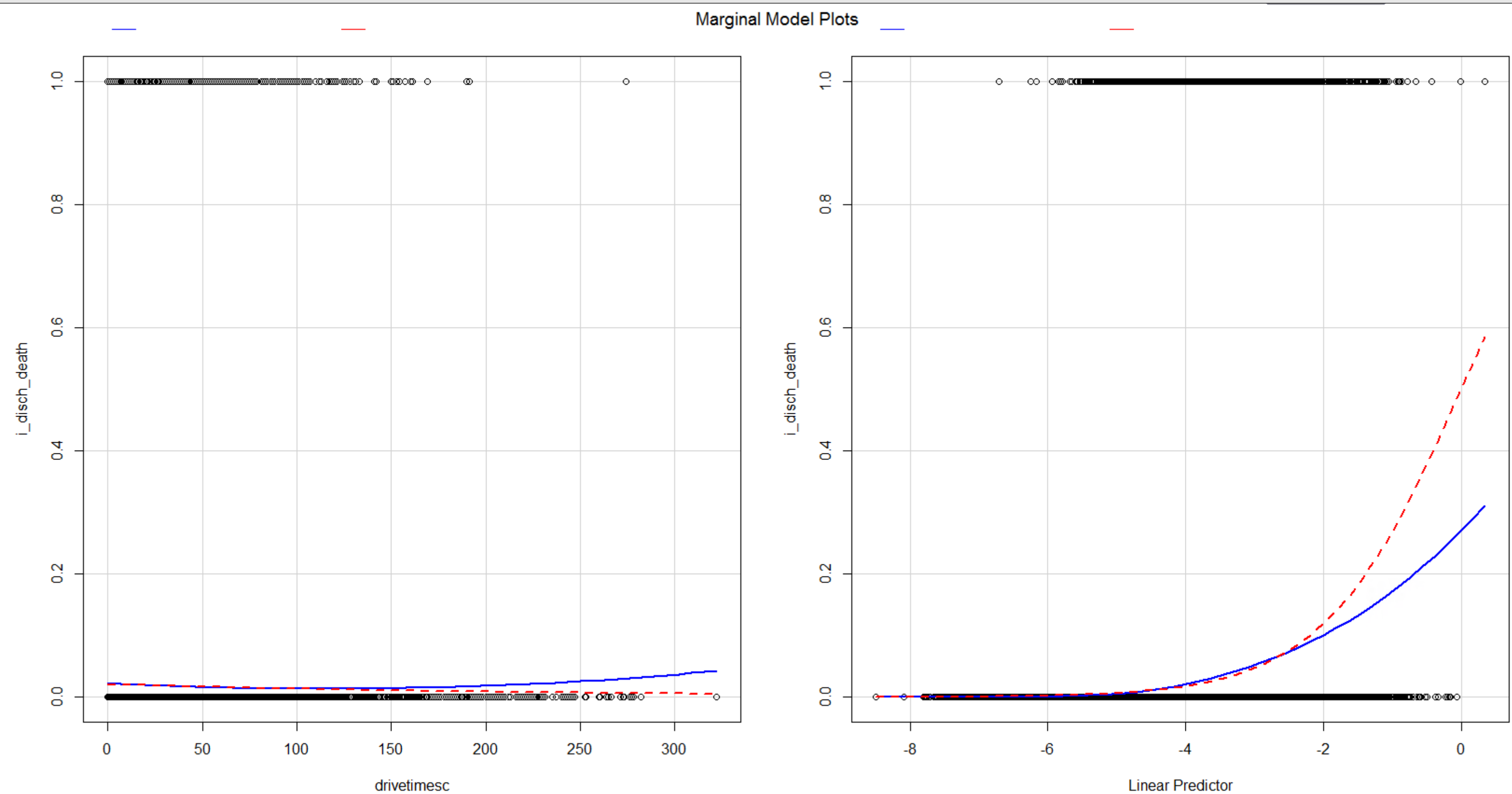
Output

```
 user system  elapsed
15.50   1.23    16.74
```

Misfit – Marginal Model Plots

R

```
Car::mmps(model.fit, ~drivetimesc)
```



Summary for diagnosing problems

- Evaluate if it is worthwhile. Spending a week of work to save a few hours of computation time is rarely worthwhile.
- Figure out why there are problems
  - Profiling
  - Traceback
  - Debug
  - Sequentially adding or deleting variables
- Figure out if the model is correct or if adjustments such as transformations can improve the fit.
- If all else fails throw more $money$ at it

Parallel

- By money I mean computational resources.
- Two major approaches
  - `foreach` explicit parallelism
  - `futures` for implicit parallelism

`foreach` paradigm

- We need a set of workers (cluster)
  - processes that can handle the computations.
- Special functions handle delegating computations to the group.
- The main package is [foreach]

Clusters

- doParallel - Interfaces a cluster communicating over sockets.
- doMC - Interfaces a cluster utilizing multiple cores of a single processor.
- doSNOW - Interfaces a 'snow' cluster, simple network of workstations.
- doMPI - Cluster communicating over message passing interface (MPI). Typical in High performance computing clusters but is overkill for most statistics applications. *Requires extensive out of R configuration of the cluster.*

`future` paradigm

Parallelism is achieved through registering a plan or strategies.

- *sequential* (default)
- *transparent* - essentially the same as sequential
- *multisession* - multiple R sessions on host computer
- *multicore* - forked R Processes, not available on Windows.
- *multiprocess* - shortcut for multicore if available, otherwise multisession.
- *cluster* - Heterogeneous cluster of machines
- *remote* - Execute code on remote session/machine.