2.1 Compare glm routines (Brucella, DU, MTOR/Scrambled)

Nicolas Bennett

2015-05-28

The following investigation marks the starting point to fitting glm models to single cell feature data. The regular glm routine has convergence problems with the investigated data set and as the data can be completely separated, the resulting p-values are unusable.

Several glm routines, in addition the the standard glm function provided by base R, are tested for their ability to imporve the analysis of the problematic dataset.

- The *glm2* package employs a different fitting method which should provide greater stability for models that fail to converge using glm.
- The glm routine in *safeBinaryRegression* extends the default glm function by first testing for the existence of a maximal likelihood estimate.
- For dealing with the complete separation issues, *brglm* does either an adjusted-score approach to bias reduction or maximum penalized likelihood (by Jeffreys invariant prior). The returned estimates are guaranteed to be finite.
- A lasso or elastic-net regularized solution is fitted by *glmnet*
- while *bayesglm* uses a weakly informative prior as a constraint.
- Finally, *bestglm* does a best subset search using AIC, BIC, EBIC, BICq or Cross-Validation.

```
mtor.loc <- findWells(experiments="brucella-du-k[12]", contents="MTOR")
```

```
## there are 8 wells remaining:
##   J101-2C  H6   SIRNA  DHARMACON_L-003008-00_A  2475  MTOR
##   J107-2D  H6   SIRNA  DHARMACON_L-003008-00_C  2475  MTOR
##   J110-2D  H6   SIRNA  DHARMACON_L-003008-00_D  2475  MTOR
##   J104-2C  H6   SIRNA  DHARMACON_L-003008-00_B  2475  MTOR
##   J107-2C  H6   SIRNA  DHARMACON_L-003008-00_C  2475  MTOR
##   J110-2C  H6   SIRNA  DHARMACON_L-003008-00_D  2475  MTOR
##   J101-2D  H6   SIRNA  DHARMACON_L-003008-00_A  2475  MTOR
##   J104-2D  H6   SIRNA  DHARMACON_L-003008-00_B  2475  MTOR
```

```
scr1.loc <- findWells(plates=sapply(mtor.loc, getBarcode), contents="SCRAMBLED",
                      well.names="G23")
```

```
## there are 8 wells remaining:
##   J101-2C  G23  CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J107-2D  G23  CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J110-2D  G23  CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J104-2C  G23  CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J107-2C  G23  CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J110-2C  G23  CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J101-2D  G23  CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J104-2D  G23  CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
```

```
scr2.loc <- findWells(plates=sapply(mtor.loc, getBarcode), contents="SCRAMBLED",
                      well.names="H2")
```

```
## there are 8 wells remaining:
##   J101-2C  H2   CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J107-2D  H2   CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J110-2D  H2   CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J104-2C  H2   CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J107-2C  H2   CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J110-2C  H2   CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J101-2D  H2   CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
##   J104-2D  H2   CONTROL  SCRAMBLED  none  ON-TARGETplus Non-targeting Pool
```

```
data <- suppressMessages(getSingleCellData(c(mtor.loc, scr1.loc, scr2.loc)))

mtor.dat <- lapply(data, function(x) {
  return(list(meta=x$H6$meta, data=meltData(cleanData(x$H6))))
})
scr1.dat <- lapply(data, function(x) {
  return(list(meta=x$G23$meta, data=meltData(cleanData(x$G23))))
})
scr2.dat <- lapply(data, function(x) {
  return(list(meta=x$H2$meta, data=meltData(cleanData(x$H2))))
})
```

```
## well H2  (J101-2C): discarding 2 images because count.cells not in [54, 433]
```

First, wells containing siRNA for the gene *MTOR* are searched for within the kinome-wide Dharmacon unpooled screens (replicates 1 and 2). Then on the plates containing those wells, scrambled control experiments are looked up (one set in a well located close to the *MTOR* well and one set located further away). The data for the resulting 24 wells is loaded, cleaned up and melted into data frames. A quick check in openBIS reveals that the two images discarded in well H2 on `J101-2C` are completely out of focus.

```
dat1 <- suppressMessages(prepareDataforGlm(
  mtor.dat[["J101-2C"]]$data$mat$Cells, scr1.dat[["J101-2C"]]$data$mat$Cells)
)
## Warning in prepareDataforGlm(mtor.dat[["J101-2C"]]$data$mat$Cells,
## scr1.dat[["J101-2C"]]$data$mat$Cells): removed 25 variables containing Na/
## NaN.
system.time(glm11 <- glmRegular(dat1))
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
##    user  system elapsed
##    17.3     0.2    17.5
system.time(glm12 <- glmGlm2(dat1))
## Warning in makeRankFull(data): removed 46 zero variance variables.
## Warning in makeRankFull(data): removed 8 variables due to highly
## correlation (>0.9999)
## Warning: glm.fit2: algorithm did not converge. Try increasing the maximum
## iterations
```

```
## Warning: glm.fit2: fitted probabilities numerically 0 or 1 occurred
##    user  system elapsed
##   15.65    0.27   15.94
system.time(glm13 <- glmSafe(dat1))
## Warning in makeRankFull(data): removed 46 zero variance variables.
## Warning in makeRankFull(data): removed 8 variables due to highly
## correlation (>0.9999)
## Warning in Ops.factor(y, 0.5): '-' not meaningful for factors
## err: complete separation
##    user  system elapsed
##   2.513   0.095   2.622
system.time(glm14 <- glmBrglm(dat1))
## Warning in makeRankFull(data): removed 46 zero variance variables.
## Warning in makeRankFull(data): removed 8 variables due to highly
## correlation (>0.9999)
## Warning in fit.proc(x = X, y = Y, weights = weights, start = start,
## etastart = etastart, : Iteration limit reached
##    user  system elapsed
##    1439      22    1463
system.time(glm15 <- glmGlmnet(dat1))
##    user  system elapsed
##    3.32    0.02    3.35
system.time(glm16 <- glmBayesglm(dat1))
## Warning: fitted probabilities numerically 0 or 1 occurred
##    user  system elapsed
##    50.5     1.2    51.8
system.time(glm17 <- glmBestglm(dat1))
## Warning in makeRankFull(data): removed 46 zero variance variables.
## Warning in makeRankFull(data): removed 8 variables due to highly
## correlation (>0.9999)
## Morgan-Tatar search since family is non-gaussian.
## Warning in bestglm(Xy, family = binomial): NAs introduced by coercion
## err: too many subsets
##    user  system elapsed
##    2.37    0.16    2.53
```

As mentioned earlier, the standard glm routine has both convergence issues and trouble with complete data separation. The modified fitting procedure in *glm2* does not improve the situation however. The extension provided in *safeBinaryRegression* correctly identifies the problem of complete separation but then fails and therefore is of no use. *brglm* also has problems with fitting a solution, exceedes the default numbe rof iterations, runs for a very long time and yields worse prediction accuracy (0.89) than the first two packages (0.98 and 0.98, respectivley). A regularized solution fitted by *glmnet* is found without issues while *bayesglm* again complains about complete separation. Finally, *bestglm* is completely useless, as it tries to choose the best result via an all subsets search (2^n), which is prohibititive if $n \approx 500$.

```
dat2 <- suppressMessages(prepareDataforGlm(
  mtor.dat[["J101-2C"]]$data$mat$Cells, scr2.dat[["J101-2C"]]$data$mat$Cells)
)
## Warning in prepareDataforGlm(mtor.dat[["J101-2C"]]$data$mat$Cells,
## scr2.dat[["J101-2C"]]$data$mat$Cells): removed 25 variables containing Na/
## NaN.
glm21 <- glmRegular(dat2)
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
glm22 <- glmGlm2(dat2)
## Warning in makeRankFull(data): removed 46 zero variance variables.
## Warning in makeRankFull(data): removed 9 variables due to highly
## correlation (>0.9999)
## Warning: glm.fit2: algorithm did not converge. Try increasing the maximum
## iterations
## Warning: glm.fit2: fitted probabilities numerically 0 or 1 occurred
glm25 <- glmGlmnet(dat2)
glm26 <- glmBayesglm(dat2)
## Warning: fitted probabilities numerically 0 or 1 occurred
```

Of the remaining glm routines, *glm* and *glm2* again suffer from convergence issues and complete data separation, while *glmnet* does fine and *bayesglm* only warns about complete separation. The prediction results are very good which of course is not at all surprising, given the separation issues.

method, data	true positive rate	true negative rate	accuracy
glm, data1	0.96	0.99	0.98
glm, data2	0.99	0.94	0.96
glm2, data1	0.97	0.99	0.98
glm2, data2	0.97	0.95	0.95
glment, data1	0.98	1	0.99
glmnet, data2	0.99	1	0.99
bayesglm, data1	1	1	1
bayesglm, data2	0.99	1	0.99

All data is from plate `J101-2C` and data1 refers to the *MTOR* well, combined with the scrambled control well `G23` while data2 corresponds to the same *MTOR* well paired with the much closer scrambled control well `H2`.

```
dat3 <- suppressMessages(prepareDataforGlm(
  mtor.dat[["J107-2C"]]$data$mat$Cells, scr2.dat[["J107-2C"]]$data$mat$Cells)
)
## Warning in prepareDataforGlm(mtor.dat[["J107-2C"]]$data$mat$Cells,
## scr2.dat[["J107-2C"]]$data$mat$Cells): removed 25 variables containing Na/
## NaN.
glm31 <- glmRegular(dat3)
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
glm32 <- glmGlm2(dat3)
## Warning in makeRankFull(data): removed 46 zero variance variables.
## Warning in makeRankFull(data): removed 8 variables due to highly
## correlation (>0.9999)
## Warning: glm.fit2: fitted probabilities numerically 0 or 1 occurred
glm35 <- glmGlmnet(dat3)
glm36 <- glmBayesglm(dat3)
## Warning: fitted probabilities numerically 0 or 1 occurred
```

For this run, data from a differen plate is used (`J107-2C`) and the well pair lying closer on the plate (`H6` for *MTOR* and `H2` for *SCRAMBLED*) is fitted. The issue of complete separation remains and therefore prediction again is very good:

method	true positive rate	true negative rate	accuracy
glm	0.91	0.88	0.89
glm2	0.9	0.88	0.89
glment	0.92	0.89	0.91
bayesglm	0.93	0.89	0.91

```
dat4 <- suppressMessages(prepareDataforGlm(
  rbind(mtor.dat[["J101-2C"]]$data$mat$Cells,
        mtor.dat[["J104-2C"]]$data$mat$Cells),
  rbind(scr2.dat[["J101-2C"]]$data$mat$Cells,
        scr2.dat[["J104-2C"]]$data$mat$Cells))
)
## Warning in prepareDataforGlm(rbind(mtor.dat[["J101-2C"]]$data$mat$Cells, :
## removed 25 variables containing Na/NaN.
glm41 <- glmRegular(dat4)
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
glm42 <- glmGlm2(dat4)
## Warning in makeRankFull(data): removed 46 zero variance variables.
## Warning in makeRankFull(data): removed 9 variables due to highly
## correlation (>0.9999)
## Warning: glm.fit2: algorithm did not converge. Try increasing the maximum
## iterations
## Warning: glm.fit2: fitted probabilities numerically 0 or 1 occurred
glm45 <- glmGlmnet(dat4)
glm46 <- glmBayesglm(dat4)
## Warning: fitted probabilities numerically 0 or 1 occurred
```

Up until now, one single *MTOR* well was compared to a single *SCRAMBLED* well. Now, two wells for each setting are combined: The two `H6` wells on plates `J101-2C` and `J104-2C` are combined and fitted against the two `H2` wells on the same plates. The previous issues remain and prediction accuracy is still high.

method	true positive rate	true negative rate	accuracy
glm	1	0.99	0.99
glm2	1	0.99	0.99
glment	1	1	1
bayesglm	1	1	1

```
dat5 <- suppressMessages(prepareDataforGlm(
  do.call(rbind, lapply(mtor.dat, function(x) return(x$data$mat$Cells))),
  do.call(rbind, lapply(scr2.dat, function(x) return(x$data$mat$Cells))))
)
## Warning in prepareDataforGlm(do.call(rbind, lapply(mtor.dat, function(x)
## return(x$data$mat$Cells))), : removed 25 variables containing Na/NaN.
system.time(glm51 <- glmRegular(dat5))
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
##    user  system elapsed
##    70.0     1.4    71.4
system.time(glm52 <- glmGlm2(dat5))
## Warning in makeRankFull(data): removed 46 zero variance variables.
## Warning in makeRankFull(data): removed 8 variables due to highly
## correlation (>0.9999)
## Warning: glm.fit2: fitted probabilities numerically 0 or 1 occurred
##    user  system elapsed
##    73.9     2.2    76.1
system.time(glm55 <- glmGlmnet(dat5))
##    user  system elapsed
##  188.41    0.69  189.42
system.time(glm56 <- glmBayesglm(dat5))
## Warning: fitted probabilities numerically 0 or 1 occurred
##    user  system elapsed
##   131.3     4.5   136.1
```

Finally, all available data is combined (8 *MTOR* wells against 8 *SCRAMBLED* wells, `H2`). Prediction accuracy is lowerde considerably but is still very high considering the circumstances.

method	true positive rate	true negative rate	accuracy
glm	0.82	0.87	0.85
glm2	0.82	0.87	0.85
glment	0.82	0.87	0.85
bayesglm	0.82	0.88	0.85