Vignette for Modeling Interactions with Treatment in High Dimensions: Binary Outcome

Abraham Apfel

Wednesday, May 1, 2024

Contents

1 Purpose

The purpose of this vignette is to offer a guidance for colleagues looking to discover biomarkers whose association with an outcome may differ by treatment arm. We particularly focus on the high dimensional setting, however many of these ideas are easily applicape to a low dimensional setting as well.

Much of this vignette is similarly covered in another vignette made for when there is a survival outcome (https://github.com/apfela2/Vignettes/Vignette_grpLasso_Survival) however there are some differences. In particular, there is a different function used to run the group lasso regression (and consequently different wrapper functions). Here we also demonstrate an example where we use standard dummy coding for treatment as opposed to +1/-1 coding, as is discussed in the vignettes.

In this vignette we will cover parameterizing your model in a way to focus on identifying biomarkers which interact with treatment, on the importance of using splines when investigating interactions, and our recommended approach to model certain biomarkers where there seems to be a point of discontinuity. We also focus on several challenges unique to the penalized regression setting such as including splines and interactions, scaling continuous variables appropriately for proper inference when categorical variables are also present, and appropriate visualizations.

2 Dataset

We used data from 9 clinical trials which were made up of 1,137 patients who were evaluable for Outcome and received one of 2 possible treatments (initial cohort of 1,803). Out of the 1,137 patients in our analysis, 337 were positive for the binary outcome of interest ("Outcome"). Thus leaving us with an effective sample size of 337. See Vignette on Regression Modeling Strategies (https://github.com/apfela2/Vignettes/Vignette_Regression_Strategies) for more information on calculating and how to best make use of effective sample size.

The goal of our analysis was to discover if any biomarkers, from a pre-selected list of 18, differ in their association with "Outcome" by treatment.

3 Setting up Your Data

3.1 Scaling

The first step we do after loading our data is to scale all continuous variables by 2 times the standard deviation (instead of the more typically applied 1 times the sd). This is in order to allow the continuous variables to be on the same scale as the categorical ones. See (Gelman, 2008).

Note due to the complexity of the analysis beyond the scope of this vignette, we had to develop several models and thus I needed to run a Config file to help reduce redundant coding. Thus the code I present below makes use of one of the Config files for which I share the code below.

```r
library(bmsPURR)
# library(GGally)
library(corrplot)
# library(mice)
# library(lme4)
library(rms)
library(rmsb)
library(parallel)
# library(doParallel)
# library(glinternet)
# library(TunePareto)
# library(survivalROC)
# library(survcomp)
# library(timeROC)
library(grpreg)
# library(CalibrationCurves)
# library(splines)
# library(ellipse)
```

```r
library(tidyverse)

# Config File:

yimp <- "No"
lab_yimp <- "NoY_"

simon <- "No"
lab_simon <- "NoSimon"

ver <- "v01"

priorI <- "No"
lab_priorI <- "NoPriorI_"
```

```r
describe(imp_dat)
```

```
## imp_dat
##
##  20  Variables      1140  Observations
## --------------------------------------------------------------------------------
## Cont01
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1138        2      766        1    4.841    2.433    2.266    2.650
##      .25      .50      .75      .90      .95
##    3.307    4.226    5.598    7.300    9.307
##
## lowest : 1.178  1.22   1.596  1.652  1.7    , highest: 24.9   26.877 28.336 30.031 36.774
## --------------------------------------------------------------------------------
## Cont02
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1140        0       67    0.999    58.44    15.31       33       39
##      .25      .50      .75      .90      .95
##       49       60       68       74       79
##
## lowest : 18 23 24 25 26, highest: 85 86 87 89 90
## --------------------------------------------------------------------------------
## Cont03
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1140        0      236        1    70.16    58.39     12.0     17.0
##      .25      .50      .75      .90      .95
##     29.0     53.0     93.0    141.0    185.1
##
## lowest : 9     10    10.07 10.1  10.76, highest: 332   339   343   372   374
## --------------------------------------------------------------------------------
## Cont04
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1113       27      289        1    27.78    5.876    20.40    21.90
##      .25      .50      .75      .90      .95
##    24.00    27.10    30.70    34.88    37.31
##
## lowest : 12.6 17   17.1 17.2 17.3, highest: 49.5 51.1 53.4 54   56.5
## --------------------------------------------------------------------------------
## Cont05
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##      943      197      255        1    119.1    140.5     19.0     24.2 
##      .25      .50      .75      .90      .95 
##     36.0     57.0     99.0    201.6    350.1 
## 
## lowest :    1    2    4    5    6, highest: 2417 2480 2513 2681 2702
## --------------------------------------------------------------------------------
## Cont06 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     1138        2       89        1    135.2    18.73      103      113 
##      .25      .50      .75      .90      .95 
##      125      137      147      155      160 
## 
## lowest :  77  78  79  81  87, highest: 169 170 171 175 180
## --------------------------------------------------------------------------------
## Cont07 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     1138        2      579        1     1.56   0.7054   0.7000   0.8527 
##      .25      .50      .75      .90      .95 
##   1.1100   1.4595   1.8992   2.3600   2.7500 
## 
## lowest : 0.19  0.272 0.3   0.31  0.32 , highest: 4.439 4.77  4.896 5.98  7.3  
## --------------------------------------------------------------------------------
## Cont08 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     1137        3      338        1    266.6     99.2    152.8    170.0 
##      .25      .50      .75      .90      .95 
##    203.0    248.0    305.0    378.4    439.6 
## 
## lowest :   77  100  107  110  111, highest:  692  708  776  831 1056
## --------------------------------------------------------------------------------
## Cont09 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##      958      182       31    0.941    9.953    15.81        0        0 
##      .25      .50      .75      .90      .95 
##        0        1        6       30       60 
## 
## lowest :   0   1   2   3   4, highest:  75  80  90  95 100
## --------------------------------------------------------------------------------
## Cont10 
##        n  missing distinct     Info     Mean      Gmd      .05      .10 
##     1125       15      441        1    2.415   0.2706    2.137    2.176 
##      .25      .50      .75      .90      .95 
##    2.233    2.338    2.538    2.755    2.907 
## 
## lowest : 1.88081 1.90309 1.99564 2.00432 2.01703
## highest: 3.43489 3.44871 3.50365 3.71332 3.76856
## --------------------------------------------------------------------------------
## Cat01 
##        n  missing distinct 
##     1137        3        2 
## 
## Value          0   >=1
## Frequency    847   290
## Proportion 0.745 0.255
## ---------------------------------------------------------------------------
## Cat02
##        n  missing distinct     Info      Sum     Mean      Gmd
##     1140        0        2    0.655      367   0.3219    0.437
##
## ---------------------------------------------------------------------------
## Cat03
##        n  missing distinct
##     1140        0        2
##
## Value         NO   YES
## Frequency    393   747
## Proportion 0.345 0.655
## ---------------------------------------------------------------------------
## Cat04
##        n  missing distinct
##     1066       74        4
##
## Value      TypeA TypeB TypeC TypeD
## Frequency     55   157   215   639
## Proportion 0.052 0.147 0.202 0.599
## ---------------------------------------------------------------------------
## Cat05
##        n  missing distinct
##     1140        0        3
##
## Value      TypeA TypeB TypeC
## Frequency    610   382   148
## Proportion 0.535 0.335 0.130
## ---------------------------------------------------------------------------
## Cat06
##        n  missing distinct     Info      Sum     Mean      Gmd
##     1140        0        2    0.182       74  0.06491   0.1215
##
## ---------------------------------------------------------------------------
## Cat07
##        n  missing distinct     Info      Sum     Mean      Gmd
##     1140        0        2    0.469      221   0.1939   0.3128
##
## ---------------------------------------------------------------------------
## Cat08
##        n  missing distinct
##     1058       82        2
##
## Value         NO   YES
## Frequency    690   368
## Proportion 0.652 0.348
## ---------------------------------------------------------------------------
## ARM2
##        n  missing distinct
##     1140        0        2
##
## Value        PLACEBO TREATMENT
## Frequency        521       619
## Proportion     0.457     0.543
## ---------------------------------------------------------------------------
## Outcome
##        n  missing distinct     Info      Sum     Mean      Gmd
##     1140        0        2    0.627      339   0.2974   0.4182
##
## ---------------------------------------------------------------------------
```

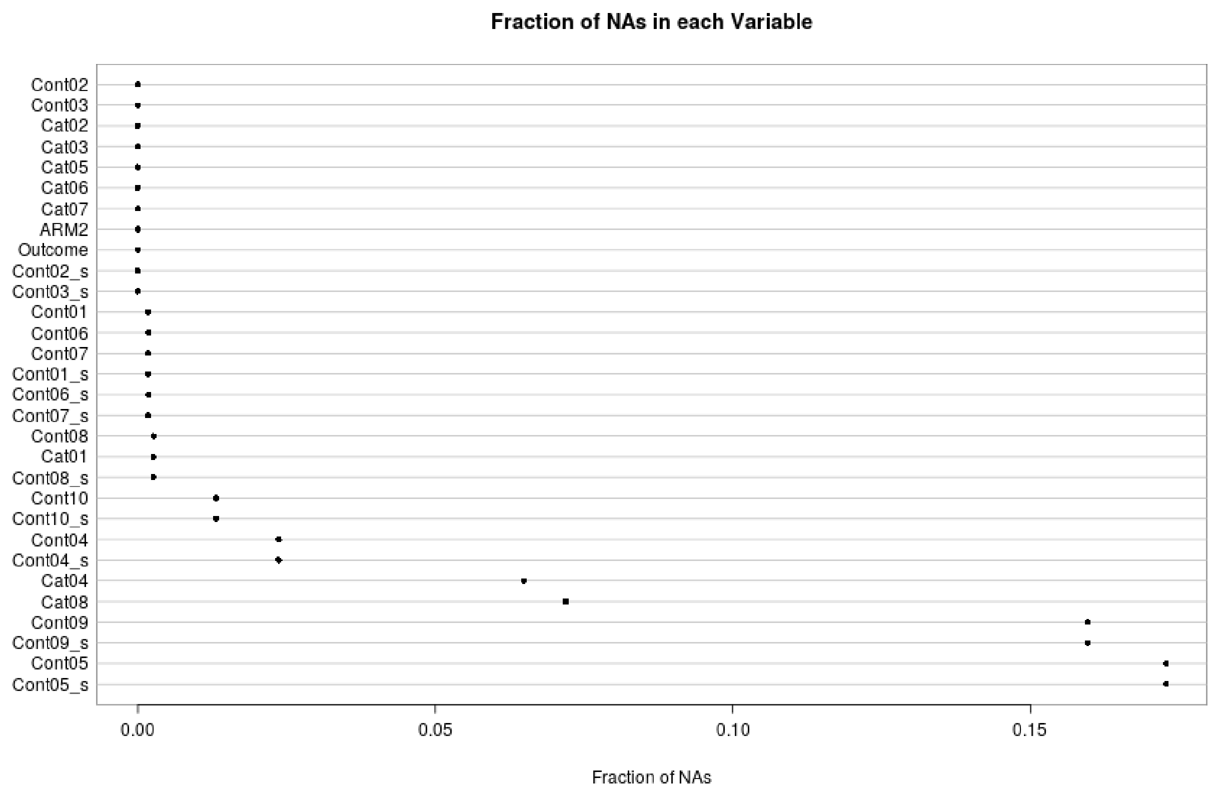
```
cont_bm <- c("Cont01", "Cont02", "Cont03", "Cont04", "Cont05", "Cont06",
                         "Cont07", "Cont08", "Cont09", "Cont10")
cat_bm <- c("Cat01", "Cat02", "Cat03", "Cat04", "Cat05", "Cat06", "Cat07", "Cat08")
trt <- "ARM2"

# Capture mean and sd for purpose of external validation
means <- apply(imp_dat[, cont_bm], 2, mean, na.rm = T)
sds <- apply(imp_dat[, cont_bm], 2, sd, na.rm = T)

# Create new data frame with means and standard deviations
mean_sd <- data.frame(variable = cont_bm, mean = means, sd = sds)

scale2 <- function(x) {
  (x - mean(x, na.rm = T))/(2*sd(x, na.rm = T))
}

# Fix up data

imp_dat <- mutate_at(imp_dat, cont_bm, list(s = ~scale2(.)))
cont_bm <- paste0(cont_bm, "_s")
```

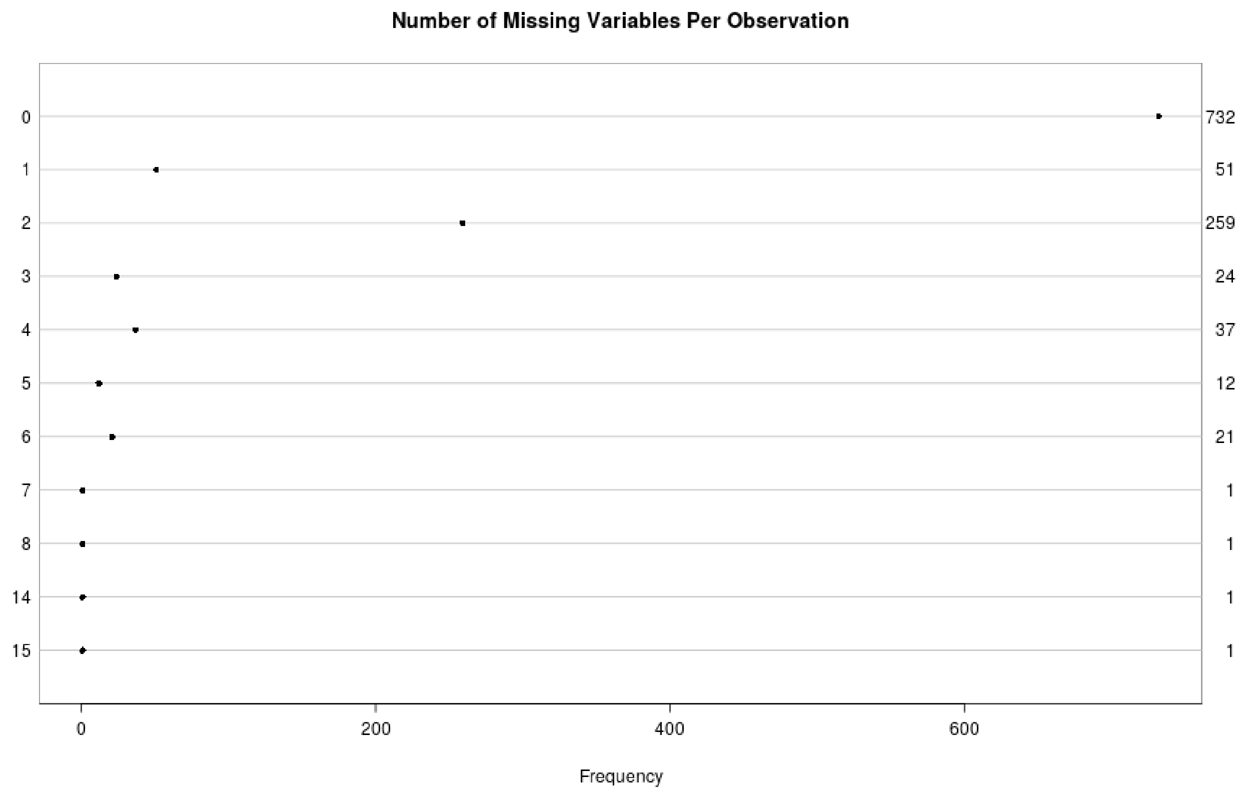
3.2 Imputation

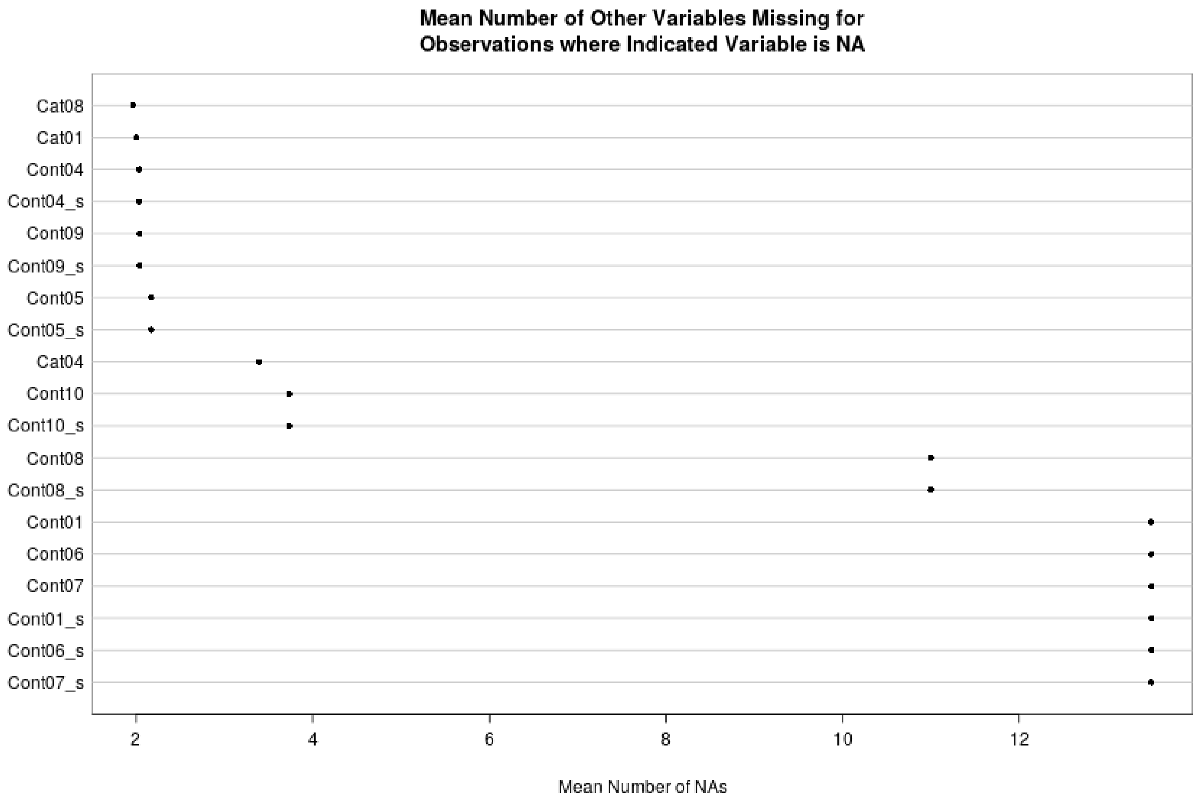
We then use single imputation WITHOUT using outcome for missing data. This is a conservative approach. If we would use multiple imputation we would be able to use the outcome as part of the imputation model but due to complexities with combining multiple imputation with penalized regression we opt for the simpler, conservative approach of single imputation without the outcome in the imputation model.

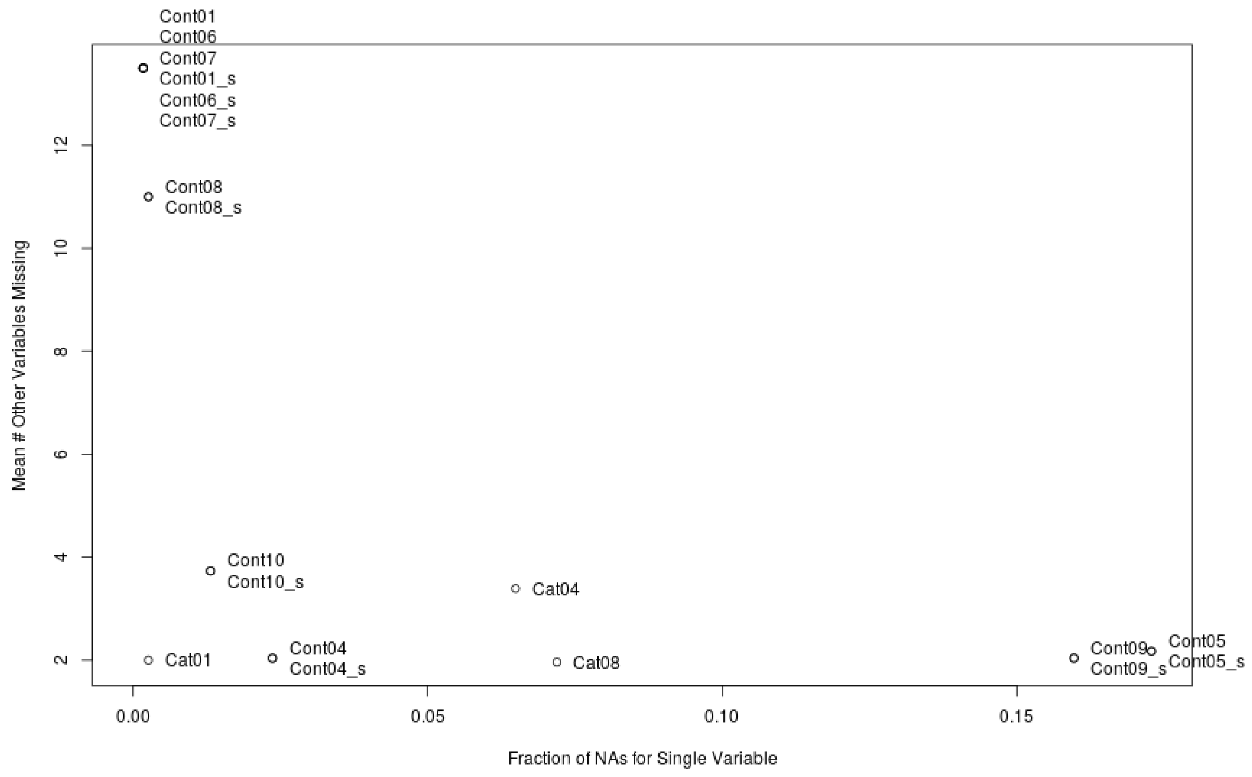
See Regression Modeling Strategies vignette (https://github.com/apfela2/Vignettes/Vignette_Regression_S trategies) for more details about best imputation practices.

```
# Perform single imputation WITHOUT using outcome: conservative approach

# Explore missing patterns
na.patterns <- naclus(imp_dat)
naplot(na.patterns)
```

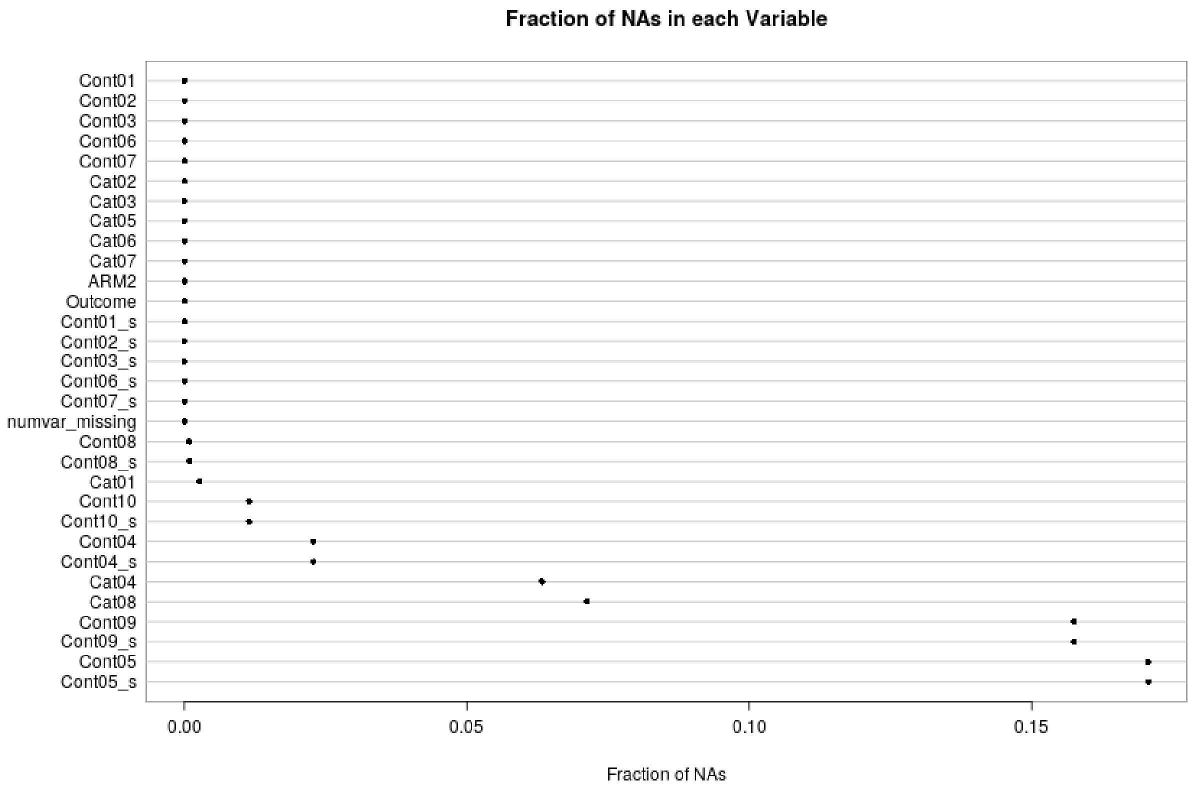

Fraction of NAs in each Variable
Cont02
Cont03
Cat02
Cat03
Cat05
Cat06
Cat07
ARM2
Outcome
Cont02_s
Cont03_s
Cont01
Cont06
Cont07
Cont01_s
Cont06_s
Cont07_s
Cont08
Cat01
Cont08_s
Cont10
Cont10_s
Cont04
Cont04_s
Cat04
Cat08
Cont09
Cont09_s
Cont05
Cont05_s
0.00
0.05
0.10
0.15
Fraction of NAs

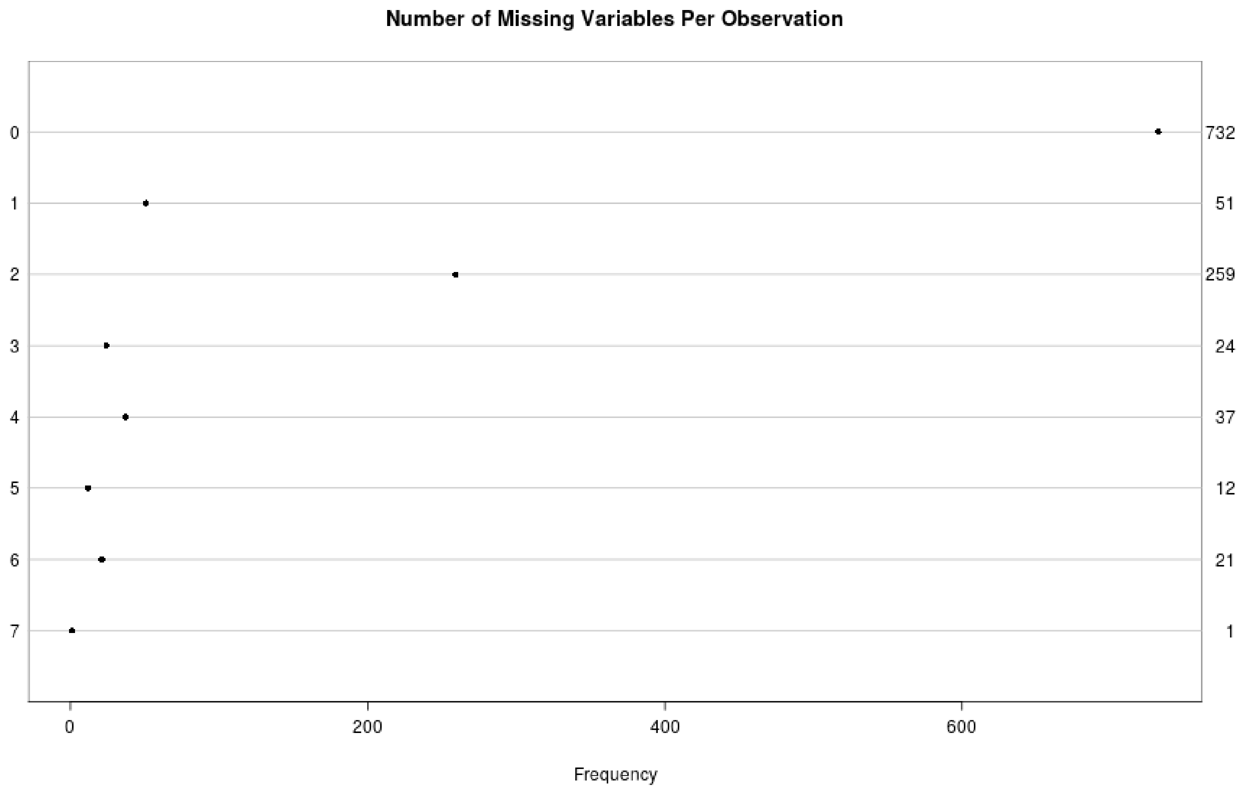

Number of Missing Variables Per Observation
0
1
2
3
4
5
6
7
8
14
15
732
51
259
24
37
12
21
1
1
1
1
0
200
400
600
Frequency

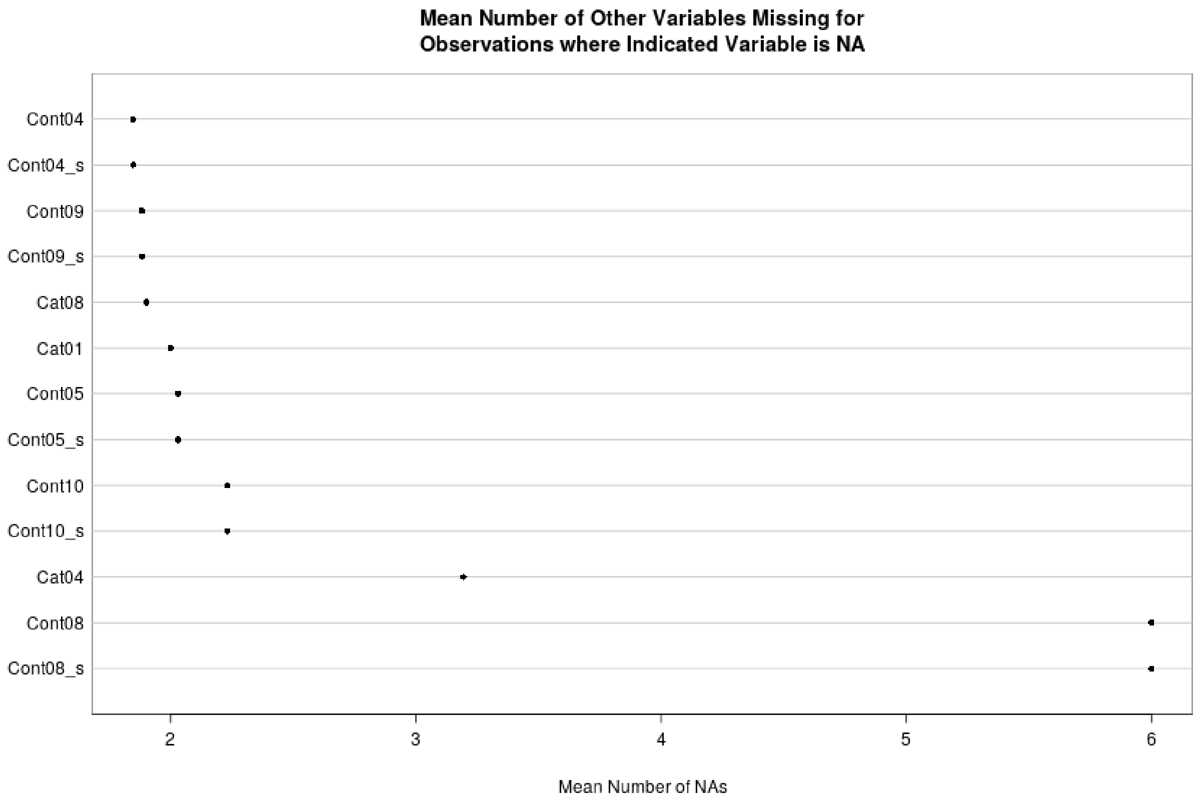

Mean Number of Other Variables Missing for
Observations where Indicated Variable is NA
Cat08
Cat01
Cont04
Cont04_s
Cont09
Cont09_s
Cont05
Cont05_s
Cat04
Cont10
Cont10_s
Cont08
Cont08_s
Cont01
Cont06
Cont07
Cont01_s
Cont06_s
Cont07_s
2
4
6
8
10
12
Mean Number of NAs

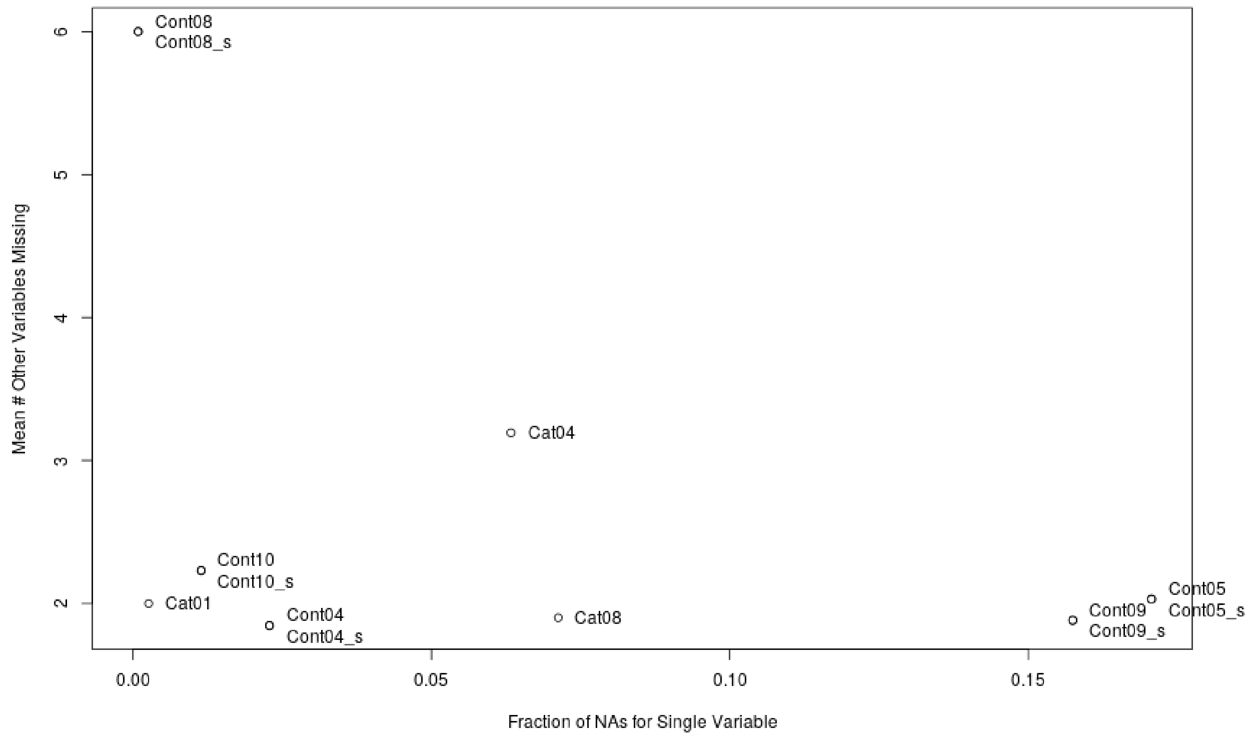


```r
imp_dat$numvar_missing <- na.patterns$na.per.obs

# # Remove subjects missing on >=5 variables (3 subjects)
red_dat <- filter(imp_dat, numvar_missing <= 7)

na.patterns <- naclus(red_dat)
naplot(na.patterns)
```

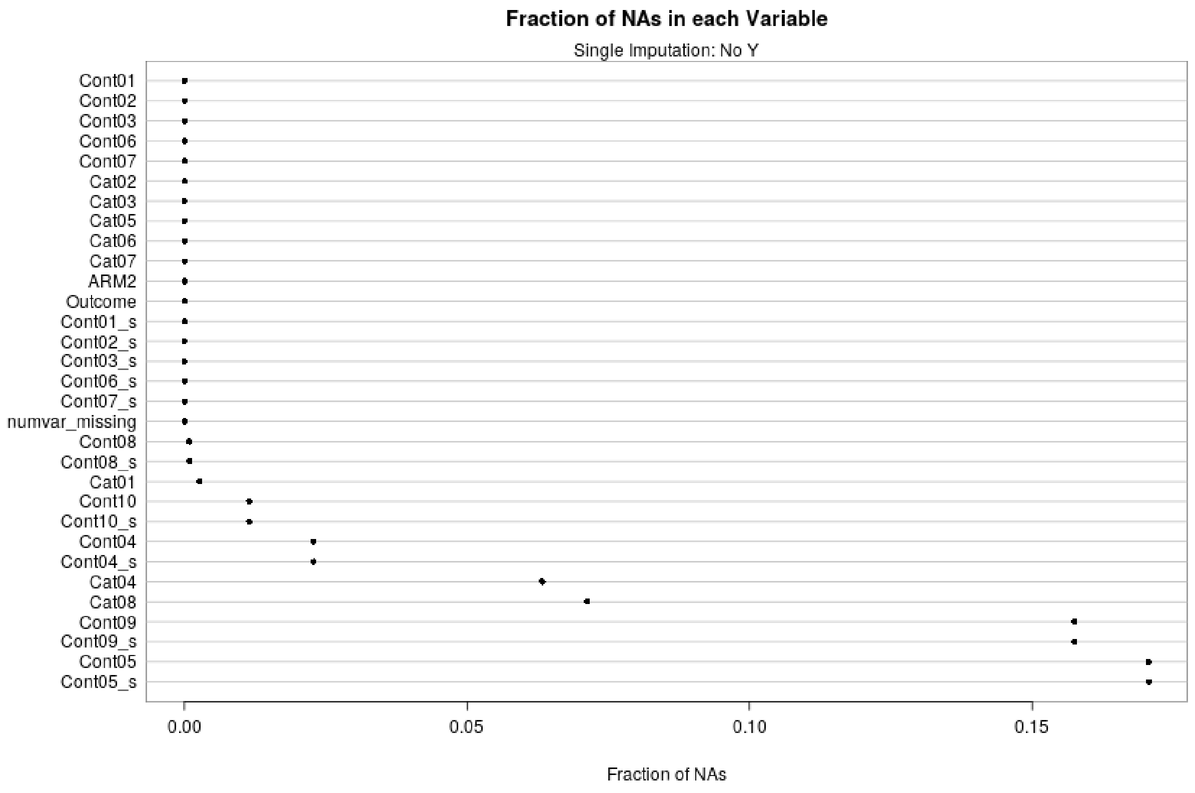

Fraction of NAs in each Variable
Cont01
Cont02
Cont03
Cont06
Cont07
Cat02
Cat03
Cat05
Cat06
Cat07
ARM2
Outcome
Cont01_s
Cont02_s
Cont03_s
Cont06_s
Cont07_s
numvar_missing
Cont08
Cont08_s
Cat01
Cont10
Cont10_s
Cont04
Cont04_s
Cat04
Cat08
Cont09
Cont09_s
Cont05
Cont05_s
0.00
0.05
0.10
0.15
Fraction of NAs


Number of Missing Variables Per Observation
0
1
2
3
4
5
6
7
732
51
259
24
37
12
21
1
0
200
400
600
Frequency


Mean Number of Other Variables Missing for
Observations where Indicated Variable is NA
Cont04
Cont04_s
Cont09
Cont09_s
Cat08
Cat01
Cont05
Cont05_s
Cont10
Cont10_s
Cat04
Cont08
Cont08_s
2
3
4
5
6
Mean Number of NAs



```r
if(yimp == "No"){
naplot(na.patterns, "na per var")
mtext("Single Imputation: No Y", side = 3)
} else if(yimp == "Yes"){
  naplot(na.patterns, "na per var")
  mtext("Single Imputation: With Y", side = 3)
}
```



```r
# Explore correlation of continuous biomarkers
describe(red_dat)
```

```
## red_dat
##
##  31  Variables      1137  Observations
## ---------------------------------------------------------------------------------
## Cont01
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1137        0      766        1     4.84    2.434    2.265    2.650
##      .25      .50      .75      .90      .95
##    3.306    4.224    5.592    7.300    9.310
##
## lowest : 1.178  1.22   1.596  1.652  1.7    , highest: 24.9   26.877 28.336 30.031 36.774
## ---------------------------------------------------------------------------------
## Cont02
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1137        0       67    0.999    58.44    15.29       33       39
##      .25      .50      .75      .90      .95
##       49       60       68       74       79
##
## lowest : 18 23 24 25 26, highest: 85 86 87 89 90
## ---------------------------------------------------------------------------------
## Cont03
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1137        0      236        1     70.2    58.42     12.0     17.0
##      .25      .50      .75      .90      .95
##     29.0     53.0     93.0    141.0    185.6
##
## lowest : 9     10    10.07 10.1  10.76, highest: 332   339   343   372   374
## --------------------------------------------------------------------------------
## Cont04
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1111       26      288        1     27.8     5.87    20.40    21.90
##      .25      .50      .75      .90      .95
##    24.05    27.10    30.70    34.90    37.31
##
## lowest : 12.6 17   17.1 17.2 17.3, highest: 49.5 51.1 53.4 54   56.5
## --------------------------------------------------------------------------------
## Cont05
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      943      194      255        1    119.1    140.5     19.0     24.2
##      .25      .50      .75      .90      .95
##     36.0     57.0     99.0    201.6    350.1
##
## lowest :    1    2    4    5    6, highest: 2417 2480 2513 2681 2702
## --------------------------------------------------------------------------------
## Cont06
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1137        0       89        1    135.2    18.73      103      113
##      .25      .50      .75      .90      .95
##      125      137      147      155      160
##
## lowest :  77  78  79  81  87, highest: 169 170 171 175 180
## --------------------------------------------------------------------------------
## Cont07
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1137        0      579        1    1.561   0.7052   0.7000   0.8554
##      .25      .50      .75      .90      .95
##   1.1100   1.4600   1.9000   2.3600   2.7500
##
## lowest : 0.19  0.272 0.3   0.31  0.32 , highest: 4.439 4.77  4.896 5.98  7.3
## --------------------------------------------------------------------------------
## Cont08
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1136        1      338        1    266.6    99.26    152.8    170.0
##      .25      .50      .75      .90      .95
##    203.0    248.0    305.0    378.5    439.8
##
## lowest :   77  100  107  110  111, highest:  692  708  776  831 1056
## --------------------------------------------------------------------------------
## Cont09
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      958      179       31    0.941    9.953    15.81        0        0
##      .25      .50      .75      .90      .95
##        0        1        6       30       60
##
## lowest :   0   1   2   3   4, highest:  75  80  90  95 100
## --------------------------------------------------------------------------------
## Cont10
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1124       13      440        1    2.415   0.2707    2.137    2.176
##      .25      .50      .75      .90      .95
##    2.233    2.338    2.538    2.755    2.907
##
## lowest : 1.88081 1.90309 1.99564 2.00432 2.01703
## highest: 3.43489 3.44871 3.50365 3.71332 3.76856
## ---------------------------------------------------------------------------
## Cat01
##        n  missing distinct
##     1134        3        2
##
## Value          0   >=1
## Frequency    845   289
## Proportion 0.745 0.255
## ---------------------------------------------------------------------------
## Cat02
##        n  missing distinct     Info      Sum     Mean      Gmd
##     1137        0        2    0.656      367   0.3228   0.4376
##
## ---------------------------------------------------------------------------
## Cat03
##        n  missing distinct
##     1137        0        2
##
## Value         NO   YES
## Frequency    391   746
## Proportion 0.344 0.656
## ---------------------------------------------------------------------------
## Cat04
##        n  missing distinct
##     1065       72        4
##
## Value      TypeA TypeB TypeC TypeD
## Frequency     55   156   215   639
## Proportion 0.052 0.146 0.202 0.600
## ---------------------------------------------------------------------------
## Cat05
##        n  missing distinct
##     1137        0        3
##
## Value      TypeA TypeB TypeC
## Frequency    609   380   148
## Proportion 0.536 0.334 0.130
## ---------------------------------------------------------------------------
## Cat06
##        n  missing distinct     Info      Sum     Mean      Gmd
##     1137        0        2     0.18       73   0.0642   0.1203
##
## ---------------------------------------------------------------------------
## Cat07
##        n  missing distinct     Info      Sum     Mean      Gmd
##     1137        0        2     0.47      221   0.1944   0.3135
##
## ---------------------------------------------------------------------------
## Cat08
##        n  missing distinct
##     1056       81        2
##
## Value         NO   YES
## Frequency    689   367
## Proportion 0.652 0.348
## ---------------------------------------------------------------------------
## ARM2
##        n  missing distinct
##     1137        0        2
##
## Value        PLACEBO TREATMENT
## Frequency        518       619
## Proportion     0.456     0.544
## ---------------------------------------------------------------------------
## Outcome
##        n  missing distinct     Info      Sum     Mean      Gmd
##     1137        0        2    0.626      337   0.2964   0.4175
##
## ---------------------------------------------------------------------------
## Cont01_s
##          n    missing   distinct       Info       Mean        Gmd        .05
##       1137          0        766          1 -0.0001899     0.4276    -0.4528
##        .10        .25        .50        .75        .90        .95
##    -0.3851    -0.2698    -0.1085     0.1319     0.4321     0.7852
##
## lowest : -0.643694 -0.636314 -0.570244 -0.560404 -0.551969
## highest: 3.5247    3.87209   4.12846   4.42631   5.61117
## ---------------------------------------------------------------------------
## Cont02_s
##          n    missing   distinct       Info       Mean        Gmd        .05
##       1137          0         67      0.999 -0.0001845     0.5654   -0.94088
##        .10        .25        .50        .75        .90        .95
##   -0.71899   -0.34918    0.05761    0.35346    0.57535    0.76026
##
## lowest : -1.4956  -1.31069 -1.27371 -1.23673 -1.19975
## highest: 0.982145 1.01913  1.05611  1.13007  1.16705
## ---------------------------------------------------------------------------
## Cont03_s
##         n   missing  distinct      Info      Mean       Gmd       .05       .10
##      1137         0       236         1 0.0003173     0.508   -0.5058   -0.4623
##       .25       .50       .75       .90       .95
##   -0.3579   -0.1492    0.1986    0.6160    1.0038
##
## lowest : -0.531853 -0.523157 -0.522548 -0.522287 -0.516548
## highest: 2.27691   2.33778   2.37256   2.62474   2.64213
## ---------------------------------------------------------------------------
## Cont04_s
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1111       26      288        1 0.001219   0.5419 -0.68158 -0.54311
##      .25      .50      .75      .90      .95
## -0.34463 -0.06307  0.26927  0.65699  0.87974
##
```

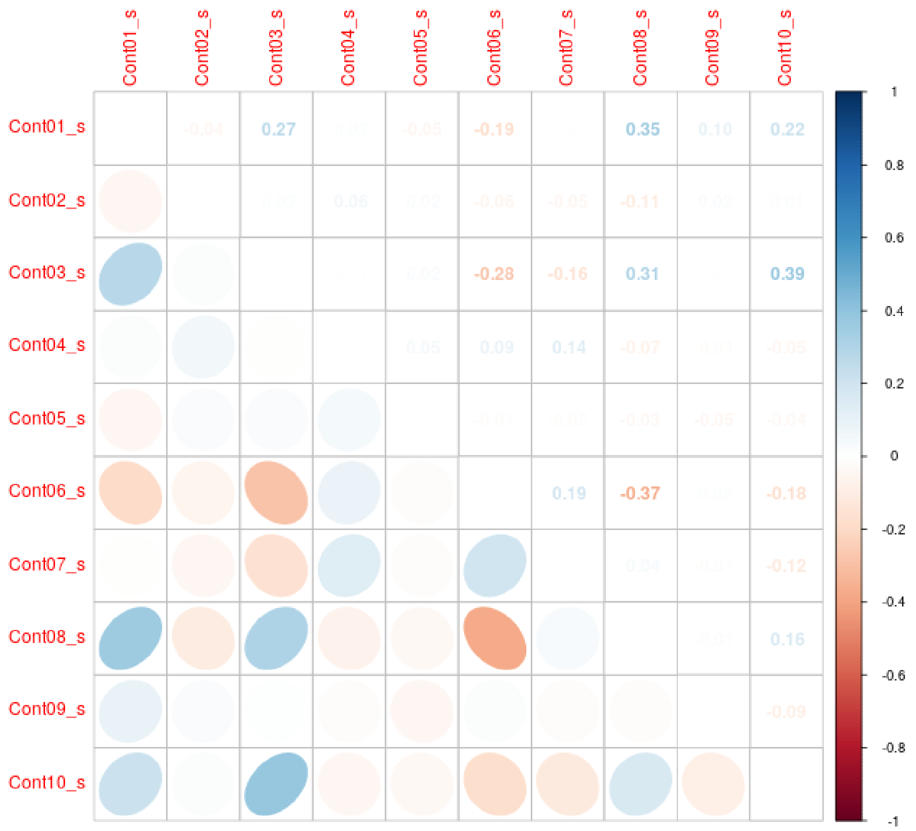
```
## lowest : -1.40165  -0.995458 -0.986226 -0.976995 -0.967763
## highest: 2.0048    2.15251   2.36483   2.42022   2.65101
## --------------------------------------------------------------------------------
## Cont05_s
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##      943      194      255        1 5.741e-18   0.2692 -0.19182 -0.18185
##      .25      .50      .75      .90      .95
## -0.15924 -0.11900 -0.03853  0.15806  0.44260
##
## lowest : -0.226305 -0.224389 -0.220557 -0.218641 -0.216725
## highest: 4.40296   4.52367   4.5869    4.90881   4.94904
## --------------------------------------------------------------------------------
## Cont06_s
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1137        0       89        1 0.0003209   0.5584 -0.96101 -0.66292
##      .25      .50      .75      .90      .95
## -0.30521  0.05249  0.35058  0.58905  0.73810
##
## lowest : -1.73604 -1.70623 -1.67642 -1.6168  -1.43795
## highest: 1.00638  1.03618  1.06599  1.18523  1.33427
## --------------------------------------------------------------------------------
## Cont07_s
##       n  missing distinct     Info     Mean      Gmd      .05      .10
##    1137        0      579        1 0.000541   0.5224 -0.63738 -0.52226
##     .25      .50      .75      .90      .95
## -0.33367 -0.07441  0.25152  0.59227  0.88116
##
## lowest : -1.01516  -0.954415 -0.933675 -0.926267 -0.91886
## highest: 2.13228   2.37747   2.4708    3.27377   4.25156
## --------------------------------------------------------------------------------
## Cont08_s
##        n  missing distinct     Info     Mean      Gmd      .05      .10
##     1136        1      338        1 8.856e-05   0.5104 -0.58526 -0.49655
##      .25      .50      .75      .90      .95
## -0.32686 -0.09546  0.19765  0.57560  0.89056
##
## lowest : -0.974778 -0.856507 -0.820512 -0.805085 -0.799943
## highest: 2.18769   2.26996   2.61963   2.90245   4.05945
## --------------------------------------------------------------------------------
## Cont09_s
##         n   missing  distinct      Info      Mean       Gmd       .05
##       958       179        31     0.941 -2.519e-17    0.3907  -0.24597
##       .10       .25       .50       .75       .90       .95
##  -0.24597  -0.24597  -0.22125  -0.09769   0.49541   1.23679
##
## lowest : -0.245966 -0.221253 -0.196541 -0.171828 -0.147115
## highest: 1.60749   1.73105   1.97818   2.10174   2.2253
## --------------------------------------------------------------------------------
## Cont10_s
##         n   missing  distinct      Info      Mean       Gmd       .05
##      1124        13       440         1 -6.324e-05    0.5206   -0.5349
##       .10       .25       .50       .75       .90       .95
##   -0.4592   -0.3497   -0.1469    0.2372    0.6545    0.9463
##
```

```
## lowest : -1.02711  -0.984269 -0.806269 -0.789562 -0.765112
## highest: 1.96196   1.98854   2.09422   2.49749   2.60374
## --------------------------------------------------------------------------------
## numvar_missing
##        n  missing distinct     Info     Mean      Gmd
##     1137        0        8    0.721   0.8637    1.257
##
## Value          0     1     2     3     4     5     6     7
## Frequency    732    51   259    24    37    12    21     1
## Proportion 0.644 0.045 0.228 0.021 0.033 0.011 0.018 0.001
##
## For the frequency table, variable is rounded to the nearest 0
## --------------------------------------------------------------------------------
```

```
num_dat <- as.matrix(dplyr::select(red_dat, cont_bm))
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(cont_bm)
##
##   # Now:
##   data %>% select(all_of(cont_bm))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
cor_mat <- rcorr(num_dat)

corrplot.mixed(cor_mat$r, lower = "ellipse", upper = "number", tl.pos = "lt")
```



```
red_dat <- dplyr::select(red_dat, -numvar_missing)


set.seed(122)
if (yimp == "No"){
MI <- aregImpute( ~ Cont09_s + Cat08  + Cont04_s + Cat04 + Cont10_s + Cont02_s + Cat05 + Cat06 +
                    Cont03_s + Cat01 + Cont06_s + Cont08_s + Cont07_s + Cont01_s + Cat02 + ARM2 + Cat03
                    Cat07,
                  data = red_dat, n.impute = 1, tlinear = F)
} else if (yimp == "Yes"){
  MI <- aregImpute( ~ Outcome + Cont09_s + Cat08  + Cont04_s + Cat04 + Cont10_s + Cont02_s + Cat05 + Ca
                      Cont03_s + Cat01 + Cont06_s + Cont08_s + Cont07_s + Cont01_s + Cat02 + ARM2 + Cat
                      Cat07,
                    data = red_dat, n.impute = 1, tlinear = F)
}
```

```
## Iteration 1 Iteration 2 Iteration 3 Iteration 4
```

```
# Create dataset with imputed values
imp <- as.data.frame(impute.transcan(MI, imputation=1, data=red_dat, list.out=TRUE, pr=FALSE, check=FAL

# Add USUBJID and PrimRes_na
imp_dat <- red_dat
imp_dat[, names(imp)] <- bind_rows(imp)
imp_dat <- imp_dat[, c(cont_bm, cat_bm, trt, "Outcome")]

# Redundancy Analysis
red <- redun(formula = ~ Cont09_s + Cat08  + Cont04_s + Cat04 + Cont10_s + Cont02_s + Cat05 + Cat06 +
               Cont03_s + Cat01 + Cont06_s + Cont08_s + Cont07_s + Cont01_s + Cat02 + ARM2 + Cat03 + Cor
               Cat07,
             r2 = 0.6, type = "adjusted", data = imp_dat)

print(red)
```

```
##
## Redundancy Analysis
##
## ~Cont09_s + Cat08 + Cont04_s + Cat04 + Cont10_s + Cont02_s +
##     Cat05 + Cat06 + Cont03_s + Cat01 + Cont06_s + Cont08_s +
##     Cont07_s + Cont01_s + Cat02 + ARM2 + Cat03 + Cont05_s + Cat07
##
## n: 1137  p: 19   nk: 3
##
## Number of NAs:    0
##
## Transformation of target variables forced to be linear
##
## R-squared cutoff: 0.6    Type: adjusted
##
## R^2 with which each variable can be predicted from all other variables:
##
## Cont09_s    Cat08 Cont04_s    Cat04 Cont10_s Cont02_s    Cat05    Cat06
##    0.006    0.135    0.080    0.251    0.220    0.179    0.126    0.025
## Cont03_s    Cat01 Cont06_s Cont08_s Cont07_s Cont01_s    Cat02     ARM2
##    0.327    0.174    0.358    0.336    0.125    0.197    0.207    0.092
##    Cat03 Cont05_s    Cat07
##    0.195    0.024    0.070
##
## No redundant variables
```

3.3 Indicator Functions for Cont09

Now that we have a complete dataset (no more missing), we must make several transformations to run the most appropriate model.

While in general when modeling continuous variables we recommend keeping them in continuous form instead of dichotomizing them to avoid throwing away information, for Cont09 specifically, experience tells us that that there appears to be a point of discontinuity at 0. Therefore we recommend best practice to create an indicator variable for when Cont09 = 0 and model all values greater than 0 as continuous.

Note that this recommendation is independent of the complications of penalized regression and cubic splines we are applying in this analysis.

```
# All biomarkers with Indicator functions
Ind_bm <- c("Cont09_s")

# Create Indicator functions for Cont09
imp_dat <- mutate(imp_dat, Ind_Cont09_s_Ind = ifelse(Cont09_s == min(Cont09_s), 1, 0))

# All continuous BM excluding those with Indicator functions
cont_bm <- cont_bm[!(cont_bm %in% Ind_bm)]
```

3.4 Basis Functions for Cubic Splines

In order to perform penalized regression with splines, we must transform the continuous variables into basis functions. Here, we chose to apply splines with 3 knots, thus each continuous variable will be transformed to 2 columns of data each representing a different basis function for the spline.

In order to aid in the interpretability of the results, we will group the basis functions together so that they will undergo the same amount of shrinkage, thus we do not have to worry about one basis function for a given variable being removed from the model while the other basis function remains. We will implement this via the group lasso penalty.

NOTE: If you plan on performing External Validation on a different cohort, it is imperative to use the same knot locations to make up the basis functions in the new data as was used in the training data. Thus, we demonstrate in this example by saving our data object after creating the basis functions.

```r
# Create dataframe of basis functions for splines of continuous variables
cont_dat <- list()
for(i in seq_len(length(cont_bm))){
  cont_dat[[paste0("mod_", cont_bm[i], "1")]] <- rcs(imp_dat[[cont_bm[[i]]]], 3)[,1]
  cont_dat[[paste0("mod_", cont_bm[i], "2")]] <- rcs(imp_dat[[cont_bm[[i]]]], 3)[,2]
}

for(i in seq_len(length(Ind_bm))){
  cont_dat[[paste0("Ind_", Ind_bm[i], "1")]] <- rcs(imp_dat[[Ind_bm[[i]]]], 3)[,1]
  cont_dat[[paste0("Ind_", Ind_bm[i], "2")]] <- rcs(imp_dat[[Ind_bm[[i]]]], 3)[,2]
}

cont_dat <- as.data.frame(cont_dat)

# Save knot locations for External Validation
if(!file.exists("/stash/results/dev/apfela/P02809_Melanoma_CompositeBM_Meta-analysis/NoY_NoPriorI_NoSim
saveRDS(cont_dat, file.path(results, paste0(lab_yimp, lab_priorI, lab_simon, "cont_dat.rds")))
}
```

3.5 Reparameterizing Treatment

Because the goal of this analysis is to identify biomarkers which we think are likely to have a different association with Primary Resistance depending on treatment, it is sometimes beneficial to parametrize the treatment variable as +1/-1 instead of the standard dummy coding. See (Tian et al., 2014). In this particular analysis we tried both parameterizations and present the model with standard dummy coding. See the corresponding vignette with Survival outcomes for an example with the +1/-1 coding.

```r
# Reparameterize treatment to +1/-1 to apply Tibshirani/Simon method
if (simon == "Yes") {
imp_dat$TRT2 <- with(imp_dat, ifelse(ARM2 == "TREATMENT", 1,
                                     ifelse(ARM2 == "PLACEBO", -1, NA)))
} else if (simon == "No"){
  imp_dat$TRT2 <- with(imp_dat, ifelse(ARM2 == "TREATMENT", 1,
                                       ifelse(ARM2 == "PLACEBO", 0, NA)))
}
```

3.6 Transforming categorical variables

The software for Group Lasso (from the `grpreg` package) requires you to provide all predictors in matrix form with all categorical variables to be coded as 0/1. Thus we make the necessary transformations below.

Note that for categorical variables with >2 levels (as is the case for Cat04 and Cat05), we must create 2 distinct dummy variables and then we will group them together as we do with the basis functions for continuous variables and the indicator variables for Cont09.

```r
# Set up clinical covariates for model
imp_dat <- mutate(imp_dat, cat_Cat01 = as.numeric(as.factor(Cat01)) - 1)
imp_dat <- mutate(imp_dat, cat_Cat03 = as.numeric(as.factor(Cat03)) - 1)
imp_dat <- mutate(imp_dat, Cat04_TypeB = ifelse(Cat04 == "TypeB", 1, 0))
imp_dat <- mutate(imp_dat, Cat04_TypeC = ifelse(Cat04 == "TypeC", 1, 0))
imp_dat <- mutate(imp_dat, Cat04_TypeD = ifelse(Cat04 == "TypeD", 1, 0))
imp_dat <- mutate(imp_dat, Cat05_TypeB = ifelse(Cat05 == "TypeB", 1, 0))
imp_dat <- mutate(imp_dat, Cat05_TypeC = ifelse(Cat05 == "TypeC", 1, 0))
imp_dat <- mutate(imp_dat, cat_Cat08 = as.numeric(as.factor(Cat08)) - 1)
imp_dat <- mutate(imp_dat, cat_Cat07 = Cat07)
imp_dat <- mutate(imp_dat, cat_Cat06 = Cat06)
imp_dat <- mutate(imp_dat, cat_Cat02 = Cat02)

mod_Cat04 <- c("Cat04_TypeB", "Cat04_TypeC", "Cat04_TypeD")
mod_Cat05 <- c("Cat05_TypeB", "Cat05_TypeC")
mod_clin <- c("cat_Cat01", "cat_Cat03", "cat_Cat08", "cat_Cat07", "cat_Cat06", "cat_Cat02")
mod_trt <- "TRT2"

# Dataframe to be used for model: Keep only model-ready variables
mod_dat <- dplyr::select(imp_dat, mod_clin, mod_Cat04, mod_Cat05,
                         mod_trt, "Outcome", "Ind_Cont09_s_Ind")
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(mod_clin)
##
##   # Now:
##   data %>% select(all_of(mod_clin))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(mod_Cat04)
##
##   # Now:
##   data %>% select(all_of(mod_Cat04))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(mod_Cat05)
##
##   # Now:
##   data %>% select(all_of(mod_Cat05))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(mod_trt)
##
##   # Now:
##   data %>% select(all_of(mod_trt))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
mod_dat <- cbind(mod_dat, cont_dat)


# Remove extra levels from factors in dataset
mod_dat <- droplevels(mod_dat)
```

3.7 Interactions

Next we have to manually create the interactions between each of our predictors and treatment. Note that all predictors with the exception of 2-level categorical predictors have multiple columns associated with them. We need to multiply every column by the treatment variable to make the interaction. We will later group these interaction terms which each respective predictor for the group lasso penalty.

```r
# Set up data to apply Noah Simon's group lasso method: Manually make interactions

# Continuous variables
bm <- str_subset(colnames(mod_dat), "mod_")
mod_dat <- mutate_at(mod_dat, bm, list(modINT = ~. * TRT2))
INT_bm <- str_subset(colnames(mod_dat), "_modINT")

# MultiLevel categorical variables
Cat04 <- str_subset(colnames(mod_dat), "Cat04_")
mod_dat <- mutate_at(mod_dat, Cat04, list(Cat04INT = ~. * TRT2))
INT_Cat04 <- str_subset(colnames(mod_dat), "_Cat04INT")

Cat05 <- str_subset(colnames(mod_dat), "Cat05_")
mod_dat <- mutate_at(mod_dat, Cat05, list(Cat05INT = ~. * TRT2))
INT_Cat05 <- str_subset(colnames(mod_dat), "_Cat05INT")
```

```r
# Binary Categorical variables
clin <- str_subset(colnames(mod_dat), "cat_")
mod_dat <- mutate_at(mod_dat, clin, list(catINT = ~. * TRT2))
INT_cat <- str_subset(colnames(mod_dat), "_catINT")

# Indicator function variables
Ind <-  c("Ind_Cont09_s_Ind", "Ind_Cont09_s1", "Ind_Cont09_s2")
mod_dat <- mutate_at(mod_dat, Ind, list(IndINT = ~. * TRT2))
INT_Ind <- c("Ind_Cont09_s_Ind_IndINT", "Ind_Cont09_s1_IndINT", "Ind_Cont09_s2_IndINT")

# List of all variables to be included in model
mod_var <- c(mod_trt, clin, "Cat04", "Cat05", cont_bm, Ind_bm)
```

4 Group Lasso Modeling

4.1 Repeated Group Lasso

Now that the data is finally in shape, we are almost ready to run our model. When working with penalized regression, which typically requires some kind of tuning parameter selection via Cross-Validation (CV), we recommend running the CV numerous times in order to not have to rely on one arbitrary split of the data. Thus we will select the tuning parameter which has the minimum median error across the numerous runs.

To implement this recommendation, we make use of a wrapper function, `rep_grpreg`, which can be found at https://github.com/apfela2/Utility_Functions/blob/main/code/R/grpLasso/rep_grpreg.R

The `rep_grpreg` function contains the following arguments:

Parameter	Details
data	dataframe
x	Vector of predictor names to be placed in the model - IMPORTANT!!! Pay attention to order of predictors so that groups are assigned correctly
y	Name of outcome variable
group	Vector of group number assigned to each predictor
family	Distribution of outcome. Currently only set up for "binomial"
penalty	See grpreg documentation
lambda_seq	Vector of proposed lambda values, recommend to keep NULL
nfold	Number of folds used in inner loop of CV used for tuning parameter selection
nreps	Number of times to repeat the CV procedure
alpha	Tuning parameter for elastic net type penalty (how much weight to put on L1 as opposed to L2)
ncores_rep	Number of cores to use if parallelizing

```r
# Sanity check to make sure groups are properly assigned
tmp <- dplyr::select(mod_dat, c("TRT2", clin, INT_cat, Cat04, INT_Cat04, Cat05, INT_Cat05, bm, INT_bm, I
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(clin)
##
##   # Now:
```

```
##   data %>% select(all_of(clin))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(INT_cat)
##
##   # Now:
##   data %>% select(all_of(INT_cat))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(Cat04)
##
##   # Now:
##   data %>% select(all_of(Cat04))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(INT_Cat04)
##
##   # Now:
##   data %>% select(all_of(INT_Cat04))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(Cat05)
##
##   # Now:
##   data %>% select(all_of(Cat05))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(INT_Cat05)
##
##   # Now:
##   data %>% select(all_of(INT_Cat05))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(bm)
##
##   # Now:
##   data %>% select(all_of(bm))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(INT_bm)
##
##   # Now:
##   data %>% select(all_of(INT_bm))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(Ind)
##
##   # Now:
##   data %>% select(all_of(Ind))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.

## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(INT_Ind)
##
##   # Now:
##   data %>% select(all_of(INT_Ind))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
grp <- c(0, rep(1:length(clin), times = 2), rep(length(clin) + 1, times = length(Cat04)*2),
         rep(length(clin) + 2, times = length(Cat05)*2),
         rep(seq(length(clin) + 3, length(clin) + 2 + length(cont_bm)), each = 2, times = 2),
         rep(length(clin) + 2 + length(cont_bm) + 1, each = 3, times = 2))
cbind(colnames(tmp), grp)
```

```
##                                 grp
##  [1,] "TRT2"                    "0"
##  [2,] "cat_Cat01"               "1"
##  [3,] "cat_Cat03"               "2"
##  [4,] "cat_Cat08"               "3"
##  [5,] "cat_Cat07"               "4"
##  [6,] "cat_Cat06"               "5"
##  [7,] "cat_Cat02"               "6"
##  [8,] "cat_Cat01_catINT"        "1"
##  [9,] "cat_Cat03_catINT"        "2"
## [10,] "cat_Cat08_catINT"        "3"
## [11,] "cat_Cat07_catINT"        "4"
## [12,] "cat_Cat06_catINT"        "5"
## [13,] "cat_Cat02_catINT"        "6"
## [14,] "Cat04_TypeB"             "7"
## [15,] "Cat04_TypeC"             "7"
## [16,] "Cat04_TypeD"             "7"
## [17,] "Cat04_TypeB_Cat04INT"    "7"
## [18,] "Cat04_TypeC_Cat04INT"    "7"
## [19,] "Cat04_TypeD_Cat04INT"    "7"
## [20,] "Cat05_TypeB"             "8"
## [21,] "Cat05_TypeC"             "8"
## [22,] "Cat05_TypeB_Cat05INT"    "8"
## [23,] "Cat05_TypeC_Cat05INT"    "8"
## [24,] "mod_Cont01_s1"           "9"
## [25,] "mod_Cont01_s2"           "9"
## [26,] "mod_Cont02_s1"           "10"
## [27,] "mod_Cont02_s2"           "10"
## [28,] "mod_Cont03_s1"           "11"
## [29,] "mod_Cont03_s2"           "11"
## [30,] "mod_Cont04_s1"           "12"
## [31,] "mod_Cont04_s2"           "12"
## [32,] "mod_Cont05_s1"           "13"
## [33,] "mod_Cont05_s2"           "13"
## [34,] "mod_Cont06_s1"           "14"
## [35,] "mod_Cont06_s2"           "14"
## [36,] "mod_Cont07_s1"           "15"
## [37,] "mod_Cont07_s2"           "15"
## [38,] "mod_Cont08_s1"           "16"
## [39,] "mod_Cont08_s2"           "16"
## [40,] "mod_Cont10_s1"           "17"
## [41,] "mod_Cont10_s2"           "17"
## [42,] "mod_Cont01_s1_modINT"    "9"
## [43,] "mod_Cont01_s2_modINT"    "9"
## [44,] "mod_Cont02_s1_modINT"    "10"
## [45,] "mod_Cont02_s2_modINT"    "10"
## [46,] "mod_Cont03_s1_modINT"    "11"
## [47,] "mod_Cont03_s2_modINT"    "11"
## [48,] "mod_Cont04_s1_modINT"    "12"
## [49,] "mod_Cont04_s2_modINT"    "12"
## [50,] "mod_Cont05_s1_modINT"    "13"
## [51,] "mod_Cont05_s2_modINT"    "13"
## [52,] "mod_Cont06_s1_modINT"    "14"
## [53,] "mod_Cont06_s2_modINT"    "14"
## [54,] "mod_Cont07_s1_modINT"    "15"
## [55,] "mod_Cont07_s2_modINT"    "15"
## [56,] "mod_Cont08_s1_modINT"    "16"
## [57,] "mod_Cont08_s2_modINT"    "16"
## [58,] "mod_Cont10_s1_modINT"    "17"
## [59,] "mod_Cont10_s2_modINT"    "17"
## [60,] "Ind_Cont09_s_Ind"        "18"
## [61,] "Ind_Cont09_s1"           "18"
## [62,] "Ind_Cont09_s2"           "18"
## [63,] "Ind_Cont09_s_Ind_IndINT" "18"
## [64,] "Ind_Cont09_s1_IndINT"    "18"
## [65,] "Ind_Cont09_s2_IndINT"    "18"
```

```
PARAMS <- list(dat = mod_dat,
               x = colnames(tmp),
               y = "Outcome",
               group = grp,
               penalty = "grLasso",
               lambda_seq = NULL,
               nreps    = 50, #inner loop of cv to fit lambda
               nfold         = 5,  #inner loop of cv to fit lambda
               # nlambda        = 100, #number of lambdas to evaluate
               # lambda.type    = "min", #which lambda to select from based on inner loop cv to apply t
               # num_outer_rep  = 100, #number of repeats for outer loop for performance estimation
               # outer_cv_nfolds = 5,  #number of folds for outer loop of cross-validation
               font_size   = 18,
               # run.parallel    = T,
               ncores_rep = max(1, (detectCores() - 1)),
               # response = surv.resp,
               # cont_X = str_subset(colnames(dat), "_Q"),
               alpha = 0.95,
               verbose = T)



set.seed(46363)
if(!file.exists(file.path("/stash/results/dev/apfela/P02809_Melanoma_CompositeBM_Meta-analysis",
                          paste0(lab_yimp, lab_priorI, lab_simon, "full_grpreg_anonymized.rds")))) {

  full_grpreg <- rep_grpreg(data = mod_dat, x = PARAMS$x, y = PARAMS$y, group = PARAMS$group,
                            penalty = PARAMS$penalty, lambda_seq = PARAMS$lambda_seq,
                            nreps = PARAMS$nreps, nfold = PARAMS$nfold, alpha = PARAMS$alpha,
                            ncores_rep = PARAMS$ncores_rep, family = "binomial")

  saveRDS(full_grpreg, file = file.path(results, paste0(lab_yimp, lab_priorI, lab_simon, "full_grpreg_a
} else {
  full_grpreg <- readRDS(file = file.path("/stash/results/dev/apfela/P02809_Melanoma_CompositeBM_Meta-a
                                          paste0(lab_yimp, lab_priorI, lab_simon, "full_grpreg_anonymiz
}
```

```
## Finished
```

5 Visualizing the Results

5.1 CV Error by Lambda

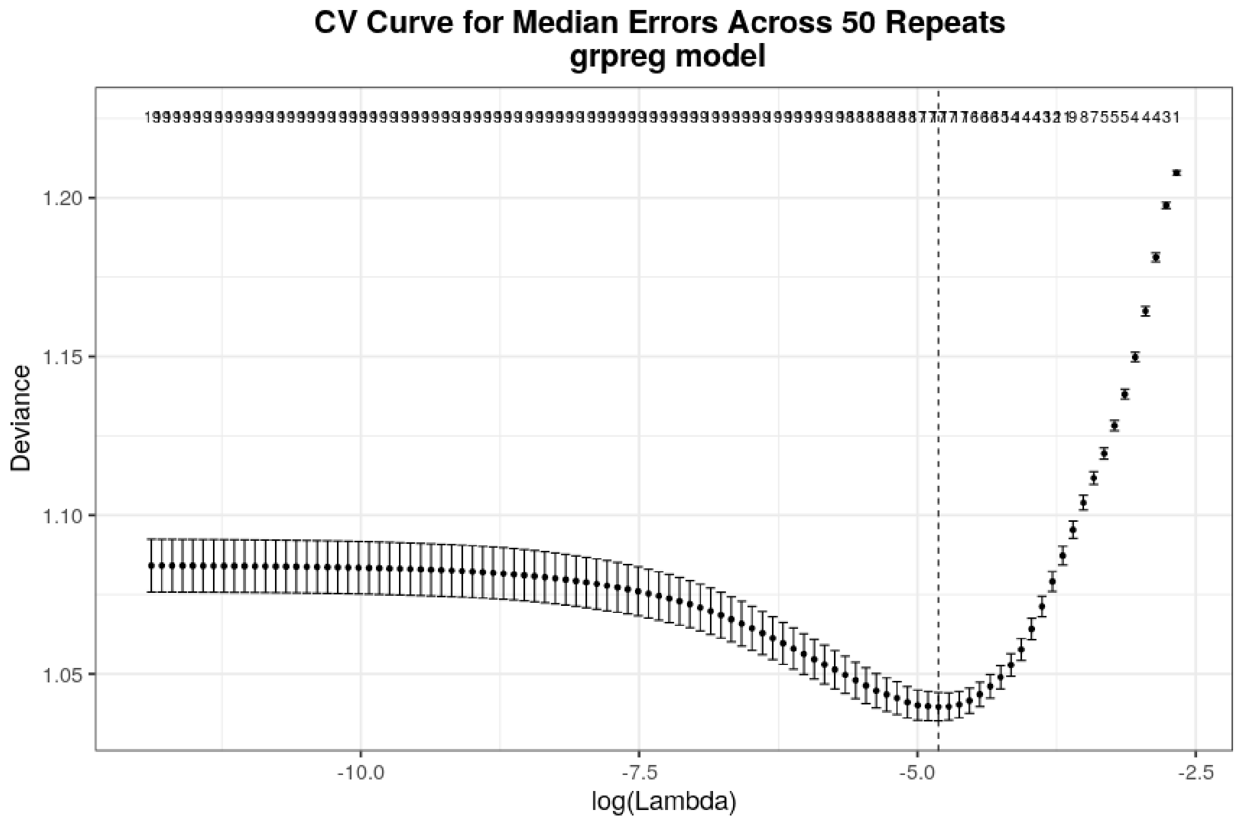
The first useful visualization is to look at the distribution of CV Error across the range of lambda values.

You would like to see a figure which shows a decrease in error as lambda increases and at a certain influx point see the error increase as lambda increases. This point of influx would be the "best lambda".

What you do NOT want to see is the CV Error curve plateau as lambda increases. This would tell us that the model with the highest value of lambda (i.e. the Null Model) is the best performing model. This implies the chosen model does not provide any meaningful information.

```
# Summarize results from full grpreg model

full_PR <- full_grpreg$full_model

# Set up data for CV Error vs lambda plot
err_PR <- as.data.frame(full_grpreg$cv_errors)
sum_err <- data.frame(err.median = apply(err_PR, 2, median), err.sd = apply(err_PR, 2, sd))

sum_err$index <- seq_len(nrow(sum_err))
sum_err$lambda <- full_grpreg$lambda_seq
sum_err$log_lambda <- log(full_grpreg$lambda_seq)


# Calculate number of groups with non-zero coefficients for each lambda
X  <-  PARAMS$dat[, PARAMS$x]

ngroups <- predict(full_PR, X = X, type = "ngroups")

# Prepare for plot
numcoef <- NULL
for(s in 1:ncol(full_PR$beta)) {
  numcoef[s] <- ngroups[s]
}

sum_err <- cbind(sum_err, numcoef)
```

```r
min_log_lambda <- log(full_grpreg$best_lambda)
min_lambda <- full_grpreg$best_lambda
index <- which.min(sum_err$err.median)

plot.title <- paste0("CV Curve for Median Errors Across ", PARAMS$nreps," Repeats \n grpreg model")


ggplot(sum_err, aes(x = log_lambda, y = err.median)) + geom_point() +
  geom_errorbar(aes(ymin = err.median - err.sd, ymax = err.median + err.sd)) +
  geom_text(mapping = aes(x = log_lambda, y = max(err.median + err.sd + ((max(err.median) - min(err.medi
  geom_vline(aes(xintercept = min_log_lambda), linetype = "dashed") +
  labs(title = plot.title) +
  theme_bw(PARAMS$font_size) +
  ylab("Deviance") +
  xlab("log(Lambda)") +
  theme(plot.title = element_text(hjust = 0.5, face = "bold"))
```



In our example we see a nice dip followed by a steep rise in CV Error, a sign that there is helpful information provided by our model.

5.2 Lambda Trajectory Plots

Lambda Trajectory plots help give an overview of the strength of the association between each of the variables in the model with the outcome and how that association changes across the range of lambda. In this application, we are interested in the overall association between each variable with outcome as well as the strength of the interaction between each variable and treatment with the outcome. Therefore we make 2 types of Lambda Trajectory plots. This is also useful forgetting a sense of how important each variable is relative to the others.

1. Overall Effect - We take the Group Norm (sqrt of the sum of the squares) of the coefficient from all terms in each "group" divided by the sqrt of the number of terms in that group.
2. Interaction Terms - We take the Group Norm of only the interaction terms from each group divided by the sqrt of the number of terms in that group.

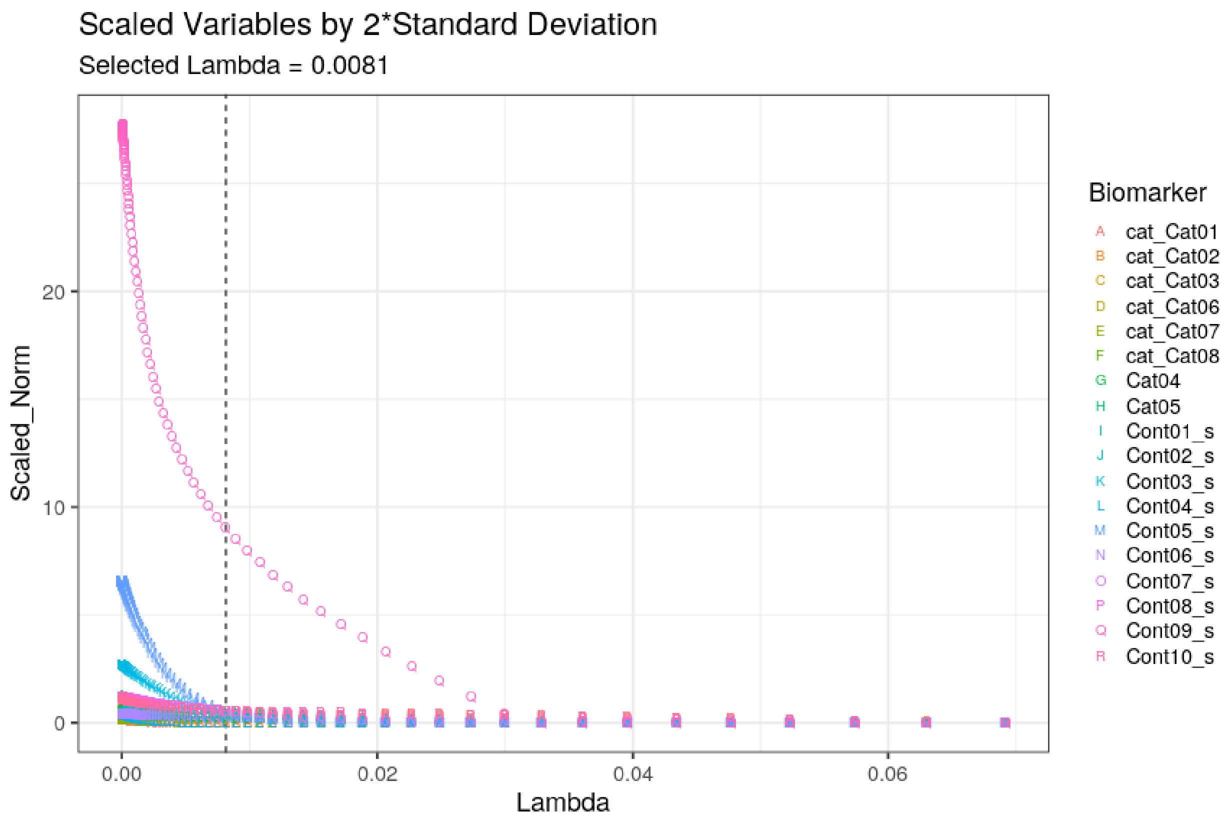
We demonstrate 2 options to visualize these plots: the standard trajectory plot and a faceted version. The faceted version may be particularly useful when there is strong overlap in the trajectories of many variables.

At times it might be useful to zoom in in the event one variable distorts the scale of the y-axis. We illustrate this here as well. to be useful.

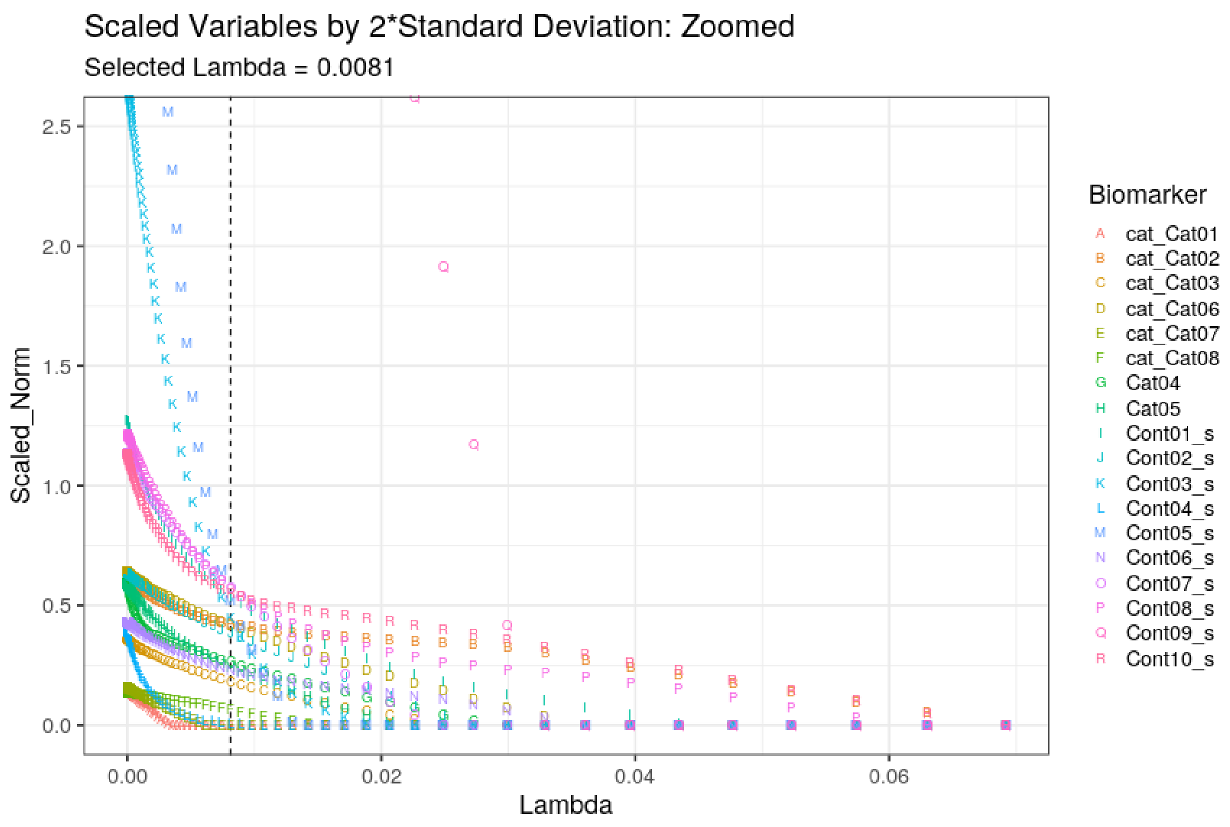
5.2.1 Overall Effect

```
# Make manual lambda trajectory plot in order to customize

# Create dataframe of lambda and group norms
lambda_group <- predict(full_PR, X = X, type = "norm")
lambda_group <- as.data.frame(lambda_group)

# Fix column names so that lambda values aren't in scientific notation
colnames(lambda_group) <- round(full_PR$lambda, 7)



# Set up data for plotting

# Create dataframe mapping coefficients for variables to different lambda values
lambda_group <- rownames_to_column(lambda_group, var = "Group")
lambda_group <- mutate(lambda_group, Biomarker = mod_var)
lambda_group <- filter(lambda_group, Group != 0)

# Calculate number of terms in each group to appropriately calculate Scaled Norms
main_rank <- data.frame(Biomarker = lambda_group$Biomarker,
                        k = c(rep(2, length(clin)),
                              length(Cat04)*2, length(Cat05)*2,
                              rep(4, length(cont_bm)), rep(6, length(Ind_bm))))

# Rank biomarkers by Scaled Norm at selected lambda
main_rank <- mutate(main_rank,
                    Selected_Norm = lambda_group[,as.character(round(min_lambda, 7))])
main_rank <- mutate(main_rank,
                    Selected_Norm_s = Selected_Norm/sqrt(k))

# Reversing order of rank so that largest is first
main_rank <- mutate(main_rank,
                    Rank = nrow(main_rank) - rank(Selected_Norm_s,
                                                  ties.method = "random"))

main_rank <- dplyr::select(main_rank, Biomarker, Rank, Selected_Norm_s)
```

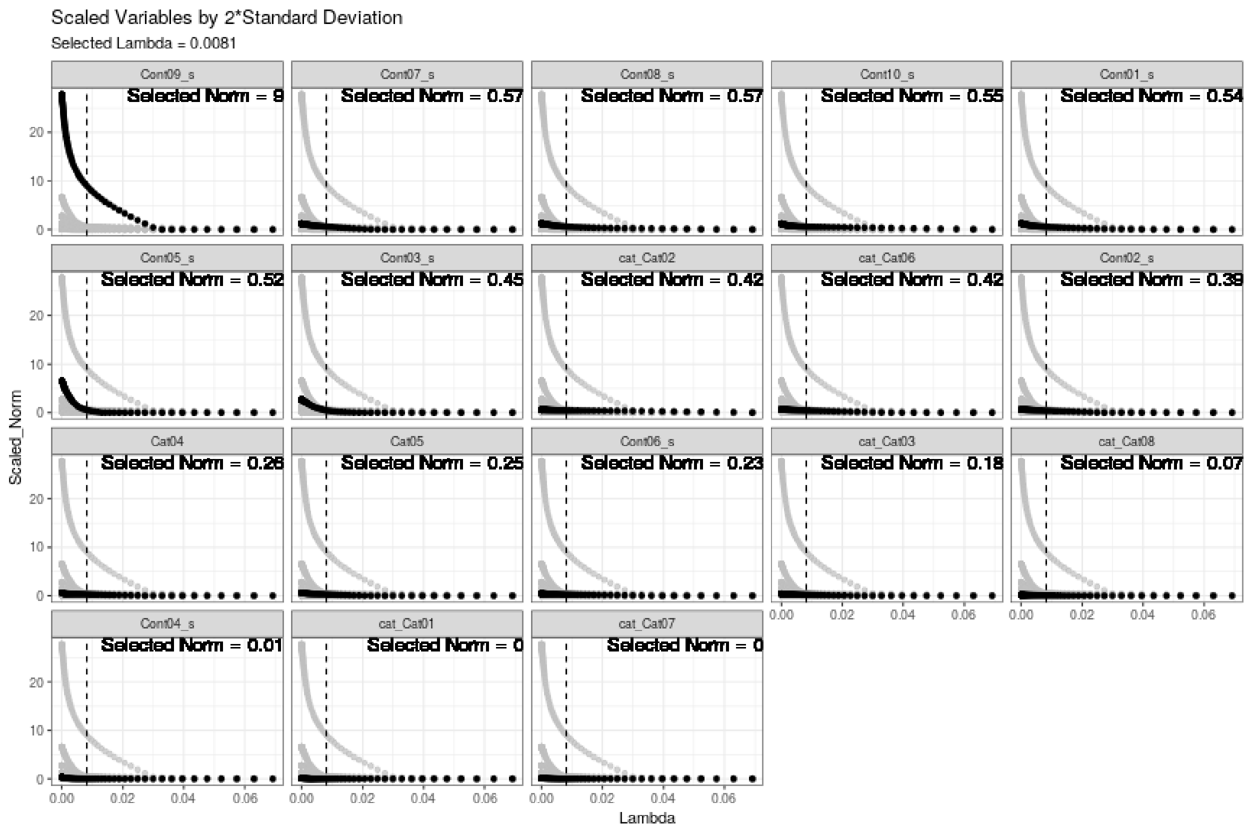
```r
# Transpose such that each row has coefficient for different variable/lambda combination
lambda_group_l <- pivot_longer(lambda_group, -c(Group, Biomarker),
                               names_to = "Lambda", values_to = "Norm")
lambda_group_l$Lambda <- as.numeric(lambda_group_l$Lambda)
lambda_group_l <- mutate(lambda_group_l, log_lambda = log(Lambda))

# Calculate number of terms in each group to appropriately calculate Scaled Norms
tmp <- data.frame(Biomarker = lambda_group$Biomarker,
                  k = c(rep(2, length(clin)),
                        length(Cat04)*2, length(Cat05)*2,
                        rep(4, length(cont_bm)), rep(6, length(Ind_bm))))
lambda_group_l <- inner_join(lambda_group_l, tmp)
lambda_group_l <- mutate(lambda_group_l, Scaled_Norm = Norm/sqrt(k))
lambda_group_l <- inner_join(lambda_group_l, main_rank)

# Make plot with Lambda on x-axis
ggplot(lambda_group_l, aes(x = Lambda, y = Scaled_Norm, color = Biomarker)) +
  scale_shape_manual(values = seq(65, length.out = (length(unique(lambda_group_l$Biomarker))))) +
  # guides(shape = F) +
  # scale_x_reverse() +
  geom_vline(xintercept = min_lambda, linetype = "dashed") +
  geom_point(aes(shape = Biomarker), size = 3) +
  ggtitle("Scaled Variables by 2*Standard Deviation",
          subtitle = paste0("Selected Lambda = ", round(min_lambda, 4))) +
  theme_bw(18)
```



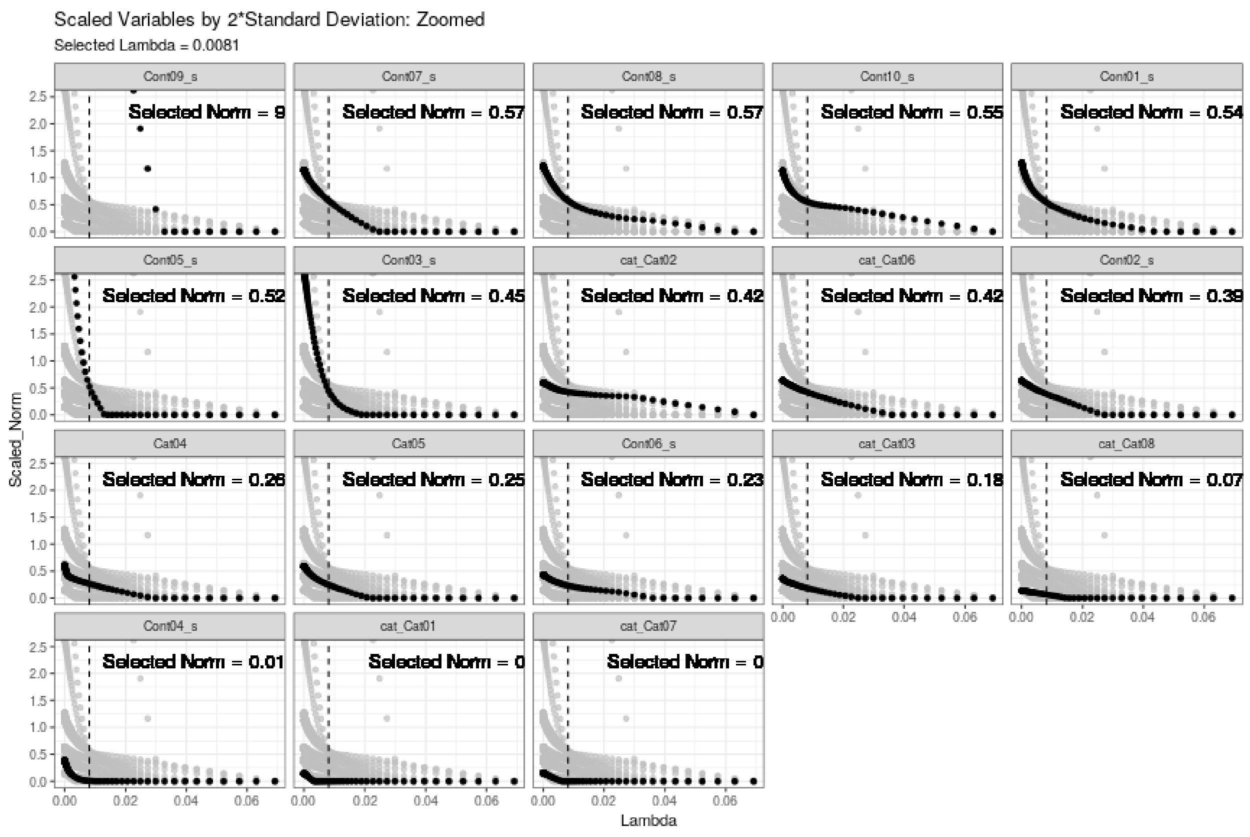
```r
# Zoomed In
ggplot(lambda_group_l, aes(x = Lambda, y = Scaled_Norm, color = Biomarker)) +
  scale_shape_manual(values = seq(65, length.out = (length(unique(lambda_group_l$Biomarker))))) +
  # guides(shape = F) +
  # scale_x_reverse() +
  geom_vline(xintercept = min_lambda, linetype = "dashed") +
  geom_point(aes(shape = Biomarker), size = 3) +
  coord_cartesian(ylim = c(0, 2.5)) +
  ggtitle("Scaled Variables by 2*Standard Deviation: Zoomed",
          subtitle = paste0("Selected Lambda = ", round(min_lambda, 4))) +
  theme_bw(18)
```



```r
# Faceted version
ggplot(lambda_group_l, aes(x = Lambda, y = Scaled_Norm)) +
  scale_shape_manual(values = seq(65, length.out = length(unique(lambda_group_l$Biomarker)))) +
  geom_point(data = lambda_group_l %>%
               dplyr::select(-Biomarker), colour = "grey", alpha = 0.7) +
  geom_point(aes(color = Biomarker), color = "black") +
  facet_wrap(reorder(Biomarker, Rank) ~ ., nrow = 4, ncol = 5) +
  geom_text(aes(x = Inf, y = Inf, label = paste0("Selected Norm = ",
                                   round(Selected_Norm_s, 2)),
                size = 2, hjust = 1, vjust = 1)) +
  theme_bw() +
  theme(legend.position = "none") +
  ggtitle("Scaled Variables by 2*Standard Deviation", subtitle = paste0("Selected Lambda = ", round(min_
  geom_vline(xintercept = min_lambda, linetype = "dashed")
```



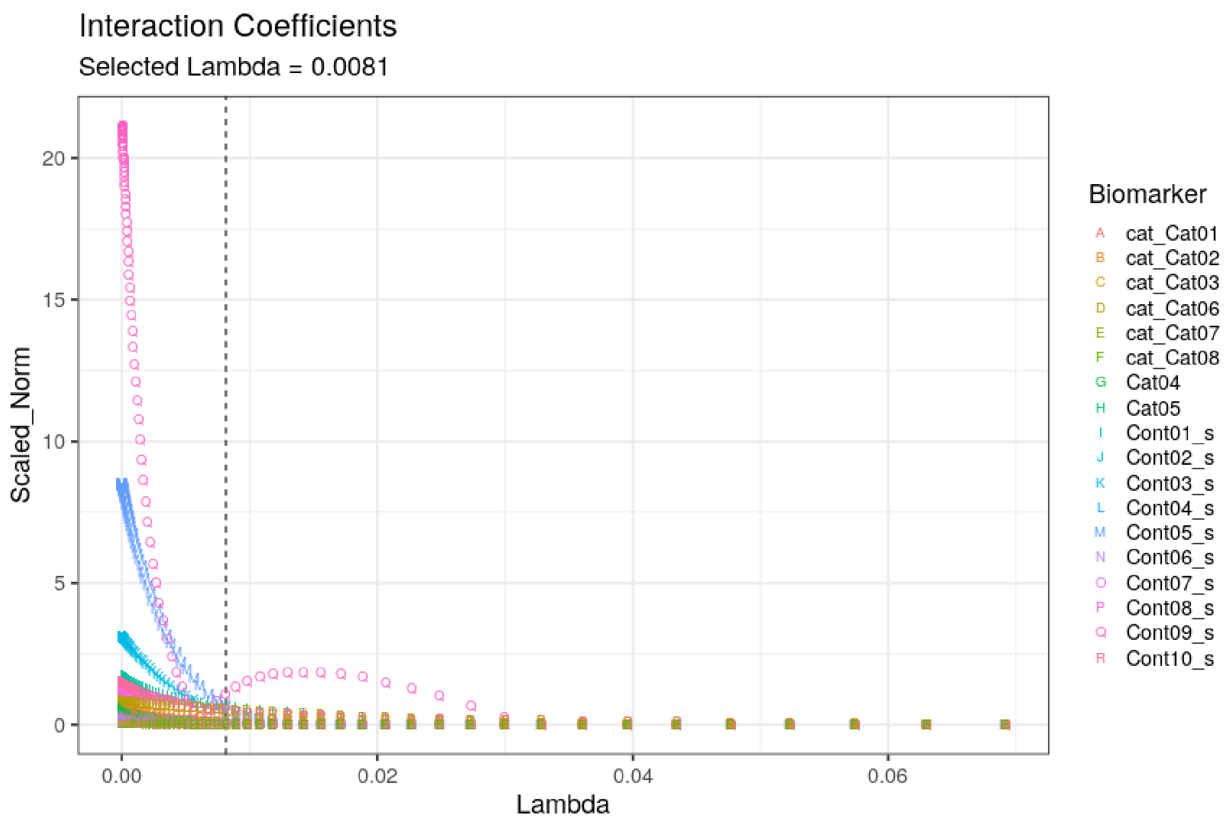
```r
# Faceted version Zoomed In
ggplot(lambda_group_l, aes(x = Lambda, y = Scaled_Norm)) +
  scale_shape_manual(values = seq(65, length.out = length(unique(lambda_group_l$Biomarker)))) +
  geom_point(data = lambda_group_l %>%
               dplyr::select(-Biomarker), colour = "grey", alpha = 0.7) +
  geom_point(aes(color = Biomarker), color = "black") +
  facet_wrap(reorder(Biomarker, Rank) ~ ., nrow = 4, ncol = 5) +
  geom_text(aes(x = Inf, y = Inf, label = paste0("Selected Norm = ", round(Selected_Norm_s, 2)),
                size = 2, hjust = 1, vjust = 2)) +
  theme_bw() +
  theme(legend.position = "none") +
  ggtitle("Scaled Variables by 2*Standard Deviation: Zoomed", subtitle = paste0("Selected Lambda = ", r
  geom_vline(xintercept = min_lambda, linetype = "dashed") +
  coord_cartesian(ylim = c(0, 2.5))
```



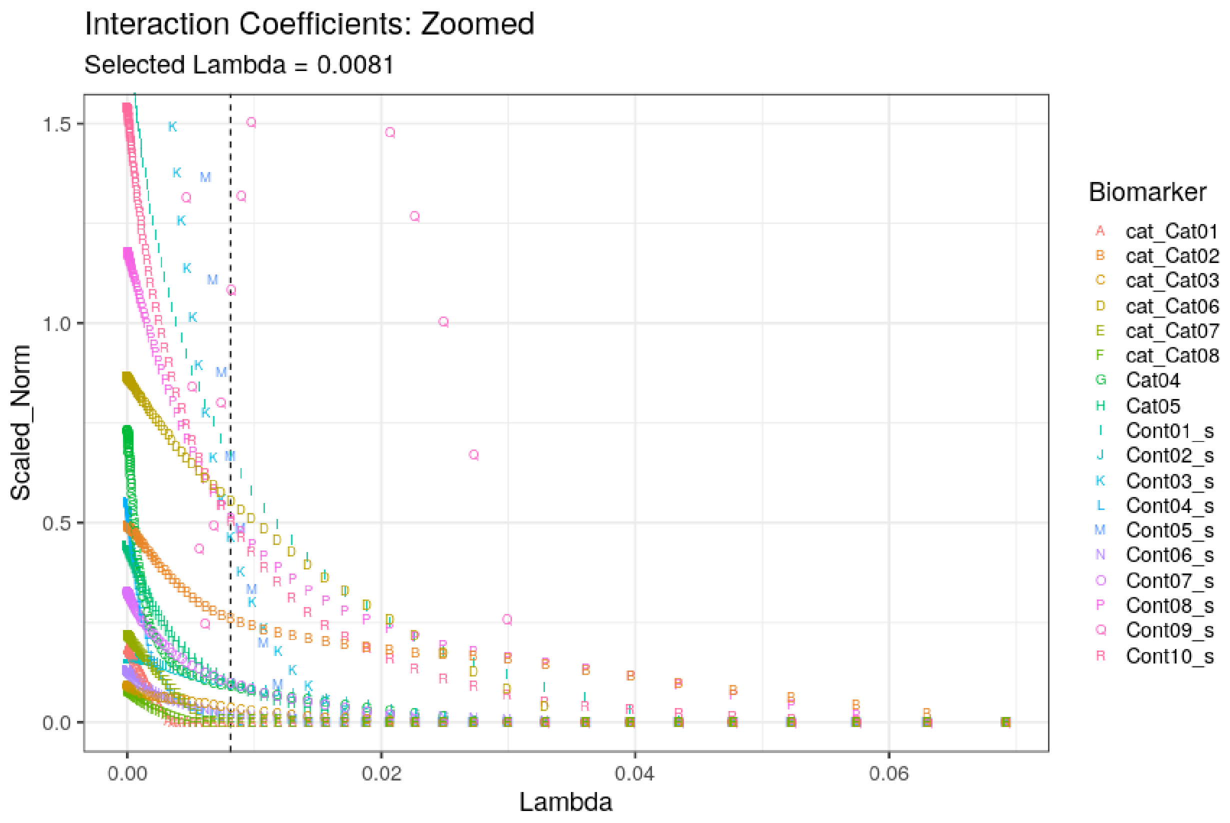
5.2.2 Interaction

```
# Set up data to make lambda trajectory plot for only Interaction coefficients

mult_cat <- c("Cat04", "Cat05")
cat_var <- c("cat", mult_cat)

# Create vector of all categorical variable names used in model (to be used later when making PE plots)
cat_mod <- c(clin, Cat04, Cat05)

# Get coefficients across all lambdas
coef_lambda <- predict(full_PR, X = X, type = "coefficients")
coef_lambda <- as.data.frame(coef_lambda)

# Fix column names so that lambda values aren't in scientific notation
colnames(coef_lambda) <- round(full_PR$lambda, 7)

coef_lambda <- rownames_to_column(coef_lambda, var = "Var")

coef_lambda <- mutate(coef_lambda, Type = ifelse(str_detect(Var, pattern = "INT"),
                                                 "Interaction", "Main_Effect"))

coef_lambda <- filter(coef_lambda, Var != "TRT2")
coef_lambda <- filter(coef_lambda, Var != "(Intercept)")
coef_lambda <- mutate(coef_lambda,
                      CAT = ifelse(str_detect(Var, pattern = paste(cat_var, collapse = "|")), 1, 0 ))

# Simplify Variable Names such that single name covers all columns for givevn variable
coef_lambda <- mutate(coef_lambda,
                      BM = ifelse(CAT == 0 & Type == "Main_Effect" & !str_detect(Var, pattern = "_Ind")
                                  str_sub(Var, 1, -2),
                                 ifelse(CAT == 0 & Type == "Main_Effect" & str_detect(Var, pattern = "_
                                       str_sub(Var, 1, -5),
                                       ifelse(CAT == 0 & Type == "Interaction" & !str_detect(Var, patte
                                              str_sub(Var, 1, -9),
                                              ifelse(CAT == 0 & Type == "Interaction" & str_detect(Var
                                                     str_sub(Var, 1, -12),
                                                     ifelse(CAT == 1 & Type == "Interaction" & !str_de
                                                            str_sub(Var, 1, -8),
                                                            ifelse(CAT == 1 & Type == "Interaction" & s
                                                                   str_sub(Var, 1, -10), Var)))))))

# Remove prefixes of continuous variables
coef_lambda <- mutate(coef_lambda, BM2 = ifelse(CAT == 0, str_sub(BM, 5), BM))


# Only keep interaction coefficients
coef_Int <- filter(coef_lambda, Type == "Interaction")

# Remove unnecessary columns
coef_Int <- dplyr::select(coef_Int, -CAT)

# Collapse Multilevel categorical variables into one name
coef_Int <- mutate(coef_Int, group = ifelse(str_detect(Var, pattern = "Cat04"), "Cat04",
                                           ifelse(str_detect(Var, pattern = "Cat05"), "Cat05", BM2)))

# Calculate number of terms in each group for scaling purposes
coef_Int <- coef_Int %>% group_by(group) %>%
  mutate(k = length(BM)) %>%
  ungroup()

# Remove unnecessary columns
coef_Int <- dplyr::select(coef_Int, -BM, -BM2, -Type)


# Transpose data to have column for lambda
coef_Int_l <- pivot_longer(coef_Int, -c(Var, group, k), names_to = "Lambda", values_to = "coef")
coef_Int_l$Lambda <- as.numeric(coef_Int_l$Lambda)
coef_Int_l <- mutate(coef_Int_l, log_lambda = log(Lambda))

# Calculate scaled norms
coef_Int_l <- coef_Int_l %>%
  group_by(Lambda, group) %>%
  mutate(Scaled_Norm = sqrt(sum(coef^2)/k)) %>%
  distinct(group, .keep_all = T) %>% # Remove duplicate rows of variable within a group
  arrange(Lambda, group, Var) %>%
  ungroup()
```

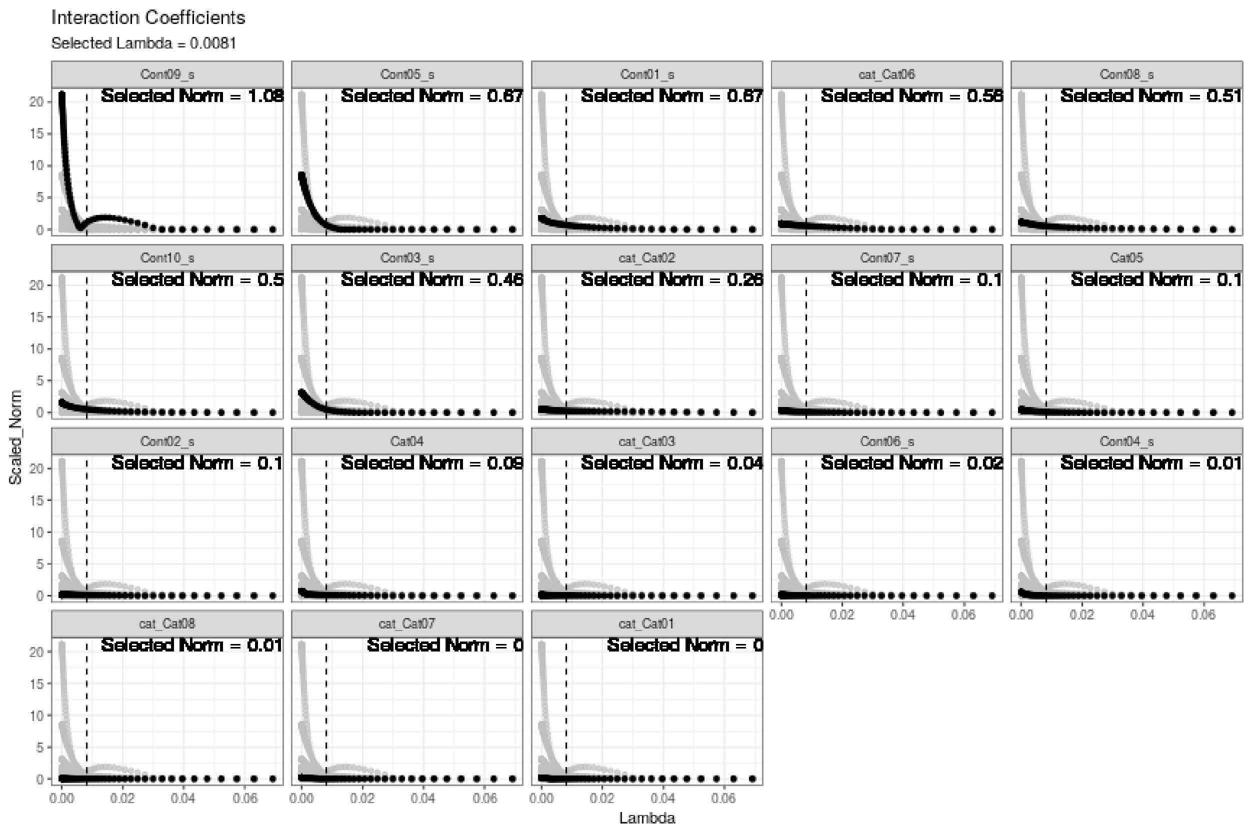
```r
# Rename group to more intuitive name
coef_Int_l <- rename(coef_Int_l, Biomarker = group)

# Rank biomarkers by Scaled Norm at selected lambda
Int_rank <- filter(coef_Int_l, Lambda == round(min_lambda, 7))

# Reversing order of rank so that largest is first
Int_rank <- mutate(Int_rank,
                   Rank = nrow(Int_rank) - rank(Scaled_Norm, ties.method = "random"))
Int_rank <- mutate(Int_rank, Selected_Norm = Scaled_Norm)
Int_rank <- dplyr::select(Int_rank, Biomarker, Rank, Selected_Norm)
coef_Int_l <- inner_join(coef_Int_l, Int_rank)

# Make plot with Lambda on x-axis
ggplot(coef_Int_l, aes(x = Lambda, y = Scaled_Norm, color = Biomarker)) +
  scale_shape_manual(values = seq(65, length.out = (length(unique(coef_Int_l$Biomarker))))) +
  # guides(shape = F) +
  # scale_x_reverse() +
  geom_vline(xintercept = min_lambda, linetype = "dashed") +
  geom_point(aes(shape = Biomarker), size = 3) +
  ggtitle("Interaction Coefficients", subtitle = paste0("Selected Lambda = ", round(min_lambda, 4))) +
  theme_bw(18)
```



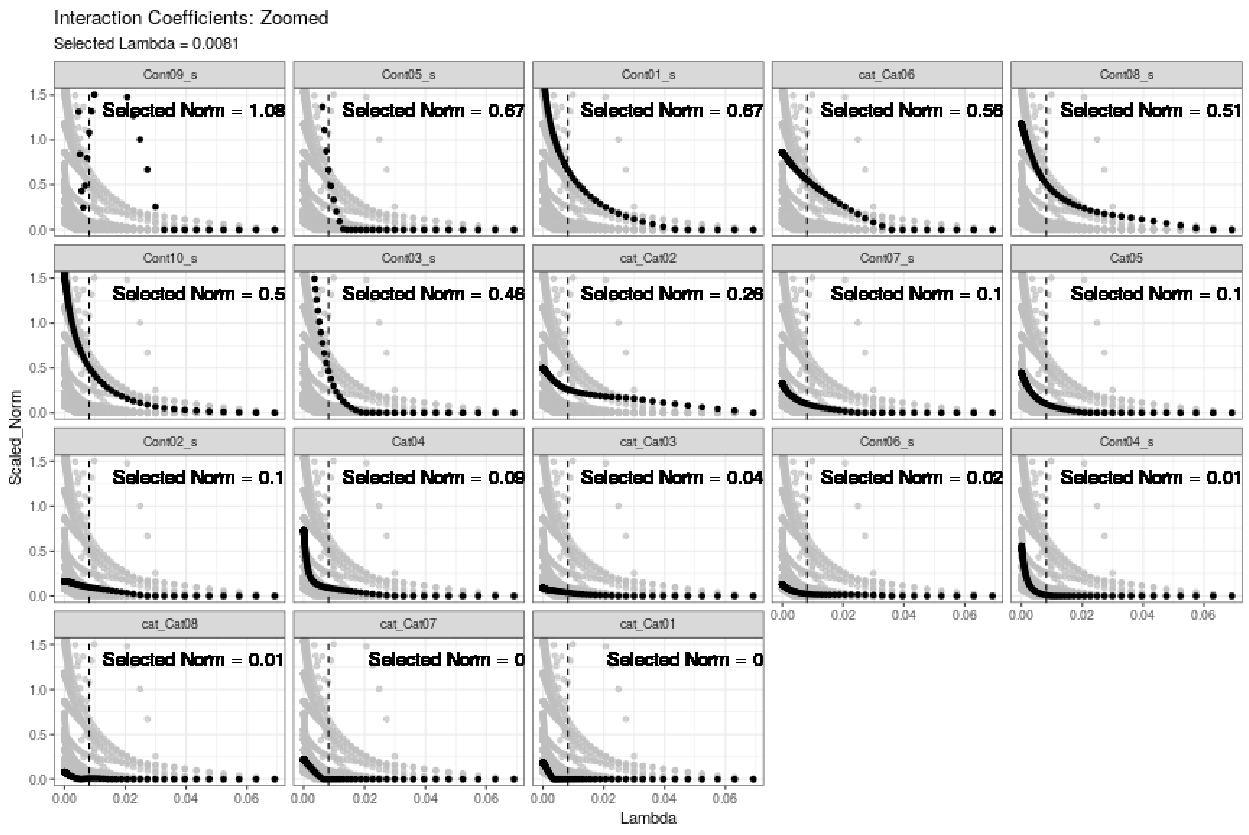
```r
# Zoomed In
ggplot(coef_Int_l, aes(x = Lambda, y = Scaled_Norm, color = Biomarker)) +
  scale_shape_manual(values = seq(65, length.out = (length(unique(coef_Int_l$Biomarker))))) +
  # guides(shape = F) +
  # scale_x_reverse() +
  geom_vline(xintercept = min_lambda, linetype = "dashed") +
  geom_point(aes(shape = Biomarker), size = 3) +
  coord_cartesian(ylim = c(0, 1.5)) +
  ggtitle("Interaction Coefficients: Zoomed", subtitle = paste0("Selected Lambda = ", round(min_lambda,
  theme_bw(18)
```



```
# Faceted version
ggplot(coef_Int_l, aes(x = Lambda, y = Scaled_Norm)) +
  scale_shape_manual(values = seq(65, length.out = length(unique(coef_Int_l$Biomarker)))) +
  geom_point(data = coef_Int_l %>%
               dplyr::select(-Biomarker), colour = "grey", alpha = 0.7) +
  geom_point(aes(color = Biomarker), color = "black") +
  facet_wrap(reorder(Biomarker, Rank) ~ ., nrow = 4, ncol = 5) +
  geom_text(aes(x = Inf, y = Inf, label = paste0("Selected Norm = ", round(Selected_Norm, 2)),
                size = 2, hjust = 1, vjust = 1)) +
  theme_bw() +
  theme(legend.position = "none") +
  ggtitle("Interaction Coefficients", subtitle = paste0("Selected Lambda = ", round(min_lambda, 4))) +
  geom_vline(xintercept = min_lambda, linetype = "dashed")
```



```r
# Faceted version Zoomed
ggplot(coef_Int_l, aes(x = Lambda, y = Scaled_Norm)) +
  scale_shape_manual(values = seq(65, length.out = length(unique(coef_Int_l$Biomarker)))) +
  geom_point(data = coef_Int_l %>%
               dplyr::select(-Biomarker), colour = "grey", alpha = 0.7) +
  geom_point(aes(color = Biomarker), color = "black") +
  facet_wrap(reorder(Biomarker, Rank) ~ ., nrow = 4, ncol = 5) +
  geom_text(aes(x = Inf, y = Inf, label = paste0("Selected Norm = ", round(Selected_Norm, 2)),
                size = 2, hjust = 1, vjust = 2)) +
  theme_bw() +
  theme(legend.position = "none") +
  ggtitle("Interaction Coefficients: Zoomed",
          subtitle = paste0("Selected Lambda = ", round(min_lambda, 4))) +
  geom_vline(xintercept = min_lambda, linetype = "dashed") +
  coord_cartesian(ylim = c(0, 1.5))
```



5.3 Partial Effects Plots

In conjunction to seeing the strength of association between each variable and outcome, it is useful to see what is the shape of the association. Due to the implementation of splines we cannot rely on conventional forest plots for continuous variables as they assume a consistent effect across all values of the variable of interest.

5.3.1 Categorical Variables

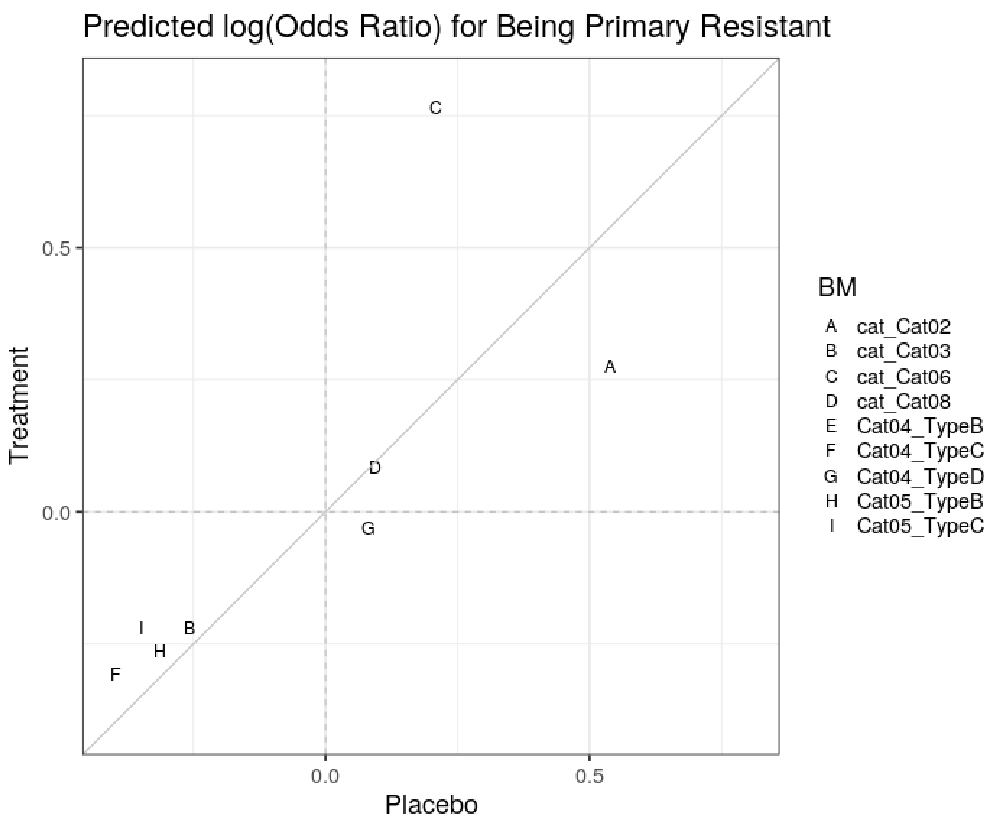
The associations for Categorical variables on the other hand can be summarized by the more familiar forest plots. We also demonstrate a scatter plot which provides similar information.

```
# Partial Effects Plots


# Get prediction matrix for selected lambda to use in PE plots
coef <- predict(full_PR, X = X, type = "coefficients", which = index)
coef <- rownames_to_column(as.data.frame(coef), var = "Var")
coef <- rename(coef, coef = as.character(round(full_grpreg$best_lambda, 4))) # The selected lambda
coef <- mutate(coef, Type = ifelse(str_detect(Var, pattern = "INT"), "Interaction", "Main_Effect"))
coef <- filter(coef, Var != "TRT2")
coef <- filter(coef, Var != "(Intercept)")
coef <- mutate(coef, CAT = ifelse(str_detect(Var, pattern = paste(cat_var, collapse = "|")), 1, 0 ))

# Simplify Variable Names such that single name covers all columns for givevn variable
coef <- mutate(coef, BM = ifelse(CAT == 0 & Type == "Main_Effect" &
```

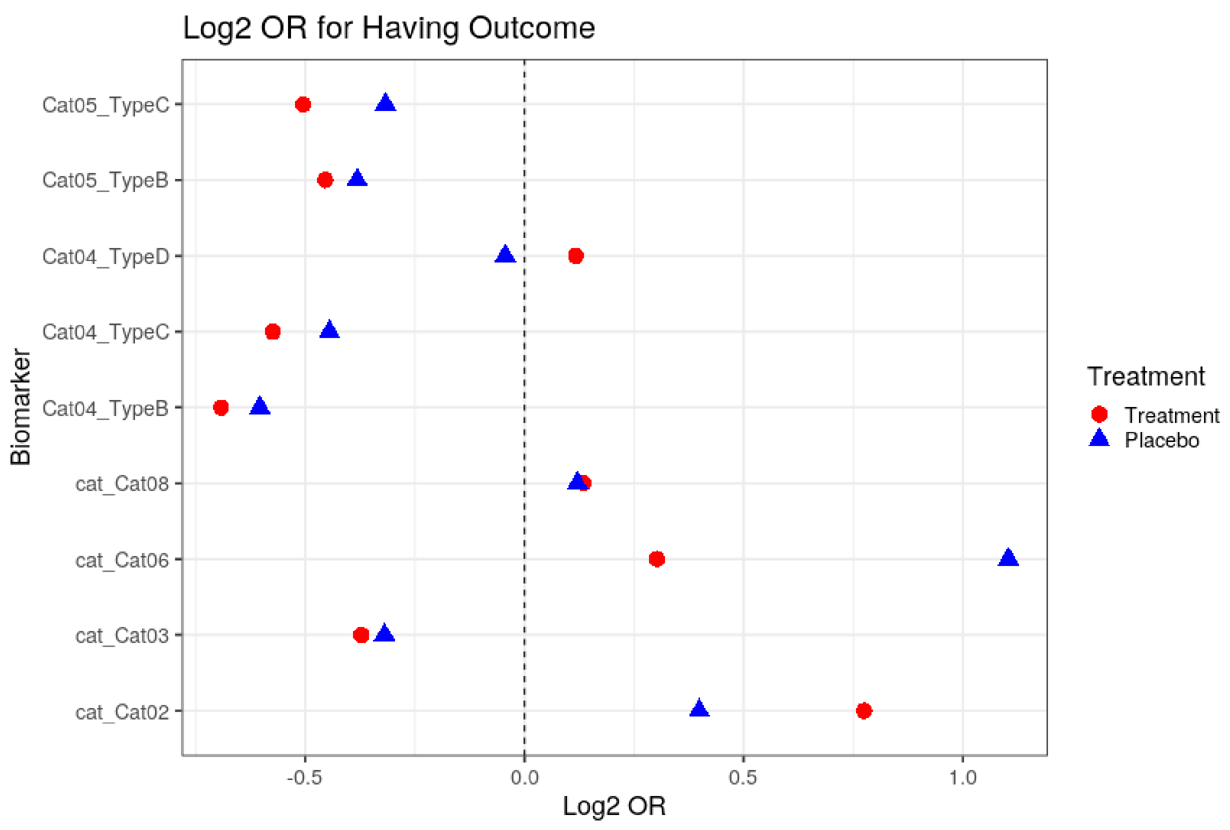
```r
                                   !str_detect(Var, pattern = "_Ind"), str_sub(Var, 1, -2),
                                 ifelse(CAT == 0 & Type == "Main_Effect" &
                                          str_detect(Var, pattern = "_Ind"), str_sub(Var, 1, -5),
                                        ifelse(CAT == 0 & Type == "Interaction" &
                                                 !str_detect(Var, pattern = "_Ind_"),
                                               str_sub(Var, 1, -9),
                                               ifelse(CAT == 0 & Type == "Interaction" &
                                                        str_detect(Var, pattern = "_Ind_"),
                                                      str_sub(Var, 1, -12),
                                                      ifelse(CAT == 1 & Type == "Interaction" &
                                                               !str_detect(Var, pattern = paste(mult_ca
                                                             str_sub(Var, 1, -8),
                                                             ifelse(CAT == 1 & Type == "Interaction" &
                                                                      str_detect(Var, pattern = paste(mu
                                                                    str_sub(Var, 1, -10), Var)))))))

# Remove variables which were filtered out
coef <- filter(coef, coef != 0)


###########################
# Categorical


# Set up data for scatter plot of categorical coefficients
coef_CAT <- filter(coef, CAT == 1) %>% dplyr::select(-CAT)
coef_CAT <- mutate(coef_CAT, group = ifelse(str_detect(Var, pattern = "Cat04"), "Cat04",
                                            ifelse(str_detect(Var, pattern = "Cat05"), "Cat05", BM)))

# Calculate scaled norms
coef_CAT <- coef_CAT %>% group_by(group, Type) %>%
  mutate(k = length(coef)) %>%
  mutate(Norm = sqrt(sum(coef^2)/k)) %>%
  ungroup()

# Transpose to calculate contrasts
coef_cat_w <- pivot_wider(coef_CAT, id_cols = c(BM, group), names_from = Type, values_from = coef)

# Calculate contrasts

if(simon == "Yes"){
  coef_cat_w <- mutate(coef_cat_w, Placebo = Main_Effect - Interaction)
  coef_cat_w <- mutate(coef_cat_w, Treatment = Main_Effect + Interaction)
  coef_cat_w <- mutate(coef_cat_w, Placebo_HR = exp(Placebo))
  coef_cat_w <- mutate(coef_cat_w, Treatment_HR = exp(Treatment))

} else if(simon == "No") {

  coef_cat_w <- mutate(coef_cat_w, Placebo = Main_Effect)
  coef_cat_w <- mutate(coef_cat_w, Treatment = Main_Effect + Interaction)
  coef_cat_w <- mutate(coef_cat_w, Placebo_HR = exp(Placebo))
  coef_cat_w <- mutate(coef_cat_w, Treatment_HR = exp(Treatment))
}
```

```r
ggplot(coef_cat_w, aes(x = Placebo, y = Treatment, shape = BM, thick = 5)) +
  # facet_grid(Model ~ Test) +
  # geom_hline(yintercept = -log10(0.05), linetype = "dashed", color = "grey") +
  # annotate("text", x = -0.5, y = -log10(0.05) + 0.2, label = "p = 0.05", color = "grey") +
  # annotate("text", angle = 270, x = -log10(0.05) + 0.2, y = 3.5, label = "p = 0.05", color = "grey")
  # geom_vline(xintercept = -log10(0.05), linetype = "dashed", color = "grey") +
  geom_hline(yintercept = 0, linetype = "dashed", color = "grey") +
  geom_vline(xintercept = 0, linetype = "dashed", color = "grey") +
  geom_point(size = 3, stroke = 2) +
  scale_shape_manual(values=c(65:(64+length(unique(coef_cat_w$BM))))) +
  ggtitle("Predicted log(Odds Ratio) for Being Primary Resistant") +
  ylim(-0.4, 0.8) +
  xlim(-0.4, 0.8) +
  coord_fixed() +
  geom_abline(color = "grey") +
  theme_bw(18)
```

```
## Warning: Removed 1 rows containing missing values (`geom_point()`).
```



```r
coef_groups <- predict(full_PR, X = X, type = "norm", which = index)
names(coef_groups) <- mod_var
coef_groups <- rownames_to_column(as.data.frame(coef_groups), var = "Var")

# Set up data for Table
coef_cat_table <- dplyr::select(coef_CAT, group, Type, k, Norm)
coef_cat_table <- unique(coef_cat_table)
coef_cat_table <- pivot_wider(coef_cat_table, id_cols = c(group, k), names_from = Type, values_from = Ne
coef_cat_table <- inner_join(coef_cat_table, coef_groups, by = c("group" = "Var"))
coef_cat_table <- rename(coef_cat_table, Group = coef_groups)
coef_cat_table <- mutate(coef_cat_table, Group = Group/sqrt(k))


# Make dot plot of OR for categorical variables
coef_cat_w <- pivot_longer(coef_cat_w, c(Placebo, Treatment), names_to = "Treatment",
                           values_to = "log_OR")
coef_cat_w <- mutate(coef_cat_w, log2_OR = log_OR*log2(exp(1)))


ggplot(coef_cat_w, aes(y = BM, x = log2_OR, color = Treatment, shape = Treatment)) +
  geom_point(size = 5) +
  scale_shape_discrete(labels=c("Treatment", "Placebo")) +
  scale_linetype_discrete(labels=c("Treatment", "Placebo")) +
  scale_color_manual(values = c("red", "blue"), labels = c("Treatment", "Placebo")) +
  geom_vline(xintercept = 0, linetype = "dashed") +

  ylab("Biomarker") +
  xlab("Log2 OR") +
  theme_bw(18) +
  ggtitle("Log2 OR for Having Outcome")
```



5.3.2 Continuous Variables

Partial Effects (PE) plots are particularly important when using splines in the model for continuous variables since the relationship can no longer be summarized simply by the sign of the coefficient and its slope. Rather, since there is no linearity assumption we must visualize the whole relationship through Partial Effects plots.

In order to make PE plots, we have to make predictions across a range of values for each biomarker for each treatment while keeping everything else constant. In general we do not recommend predicting the entire range of observed values because outliers will distort the x-axis.In this case study we make predictions for the middle 90% of the observed data.

When working with splines, it is important to apply the knots at the same locations as they were in the training model.

In the plot below we combine both continuous and categorical variables onto a single plot. We found that it can be confusing for some audiences to visualize the categorical and continuous variables on different figures and therefore this was a way to avoid that confusion. See the grpLasso Vignette with Survival Outcome (https://github.com/apfela2/Vignettes/Vignette_grpLasso_Survival) for an example of a PE plot with only continuous variables.

NOTE: One very important bug to be wary of when making predictions from grpreg models is that it assumes the variables in the newdata are in the same order as they were in the training data. If the order changes you will get wrong predictions WITHOUT WARNING!

```
# Continuous

# Set up data for scatterplot of continuous coefficients
coef_cont <- filter(coef, CAT == 0) %>% dplyr::select(-CAT)
coef_cont <- mutate(coef_cont, BM = str_sub(BM, 5, -1))

# Calculate Scaled Norms
coef_cont <- coef_cont %>% group_by(BM, Type) %>%
  mutate(k = length(coef)) %>%
  mutate(Norm = sqrt(sum(coef^2)/k)) %>%
  ungroup()

# Create Table
cont_coef_table <- coef_cont
cont_coef_table <- dplyr::select(cont_coef_table, Var, BM, coef, Type)
cont_coef_table <- pivot_wider(cont_coef_table, id_cols = c(Var, BM),
                               names_from = Type, values_from = coef)
cont_coef_table <- mutate(cont_coef_table, Main_Effect = round(Main_Effect, 3))
cont_coef_table <- mutate(cont_coef_table, Interaction = round(Interaction, 3))

# tmp <- dplyr::select(cont_coef_table, Interaction)

coef_cont <- dplyr::select(coef_cont, BM, Type, Norm, k)
coef_cont <- unique(coef_cont)

# Set up for PE plot
coef_cont_w <- pivot_wider(coef_cont, id_cols = c(BM, k), names_from = Type, values_from = Norm)
coef_cont_w <- inner_join(coef_cont_w, coef_groups, by = c("BM" = "Var"))
coef_cont_w <- rename(coef_cont_w, Group = coef_groups)
coef_cont_w <- mutate(coef_cont_w, Group = Group/sqrt(k))
```

```
###########################################
# Set up data for partial effects plots
###########################################

# Identify knot locations for each continuous covariate
# cont_dat_s1 <- select(cont_dat, str_subset(colnames(cont_dat), "s1"))

# Sanity checks
attributes(cont_dat$mod_Cont01_s1)$parms
```

```
## [1] -0.3851069 -0.1084560  0.4320535
```

```
attributes(cont_dat$Ind_Cont09_s1)$parms
```

```
## [1] -0.2459660 -0.2212533  0.4954145
```

```
# Interpolate, and apply model knot locations to create splines
cont_raw <- dplyr::select(imp_dat, cont_bm, Ind_bm)
```

```
## Warning: Using an external vector in selections was deprecated in tidyselect 1.1.0.
## i Please use `all_of()` or `any_of()` instead.
##   # Was:
##   data %>% select(Ind_bm)
##
##   # Now:
##   data %>% select(all_of(Ind_bm))
##
## See <https://tidyselect.r-lib.org/reference/faq-external-vector.html>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
meds <- sapply(cont_raw, median)

nintp <- 200
pred_dat <- list()
pred_dat_names <- list()


# Make same names as in model data
for(i in seq_len(length(cont_bm))){
  pred_dat_names[[i]] <-  c(paste0("mod_", cont_bm[i], "1"), paste0("mod_", cont_bm[i], "2"))
}


# Start with only cont versions of Cont09 and we'll add Ind versions later
for(i in seq_len(length(Ind_bm))){
  pred_dat_names[[i + length(cont_bm)]] <-  c(paste0("Ind_", Ind_bm[i], "1"),
                                               paste0("Ind_", Ind_bm[i], "2"))
}

pred_dat_names <- unlist(pred_dat_names)

# Create vector of ALL cont bm
all_bm <- c(cont_bm, Ind_bm)
```

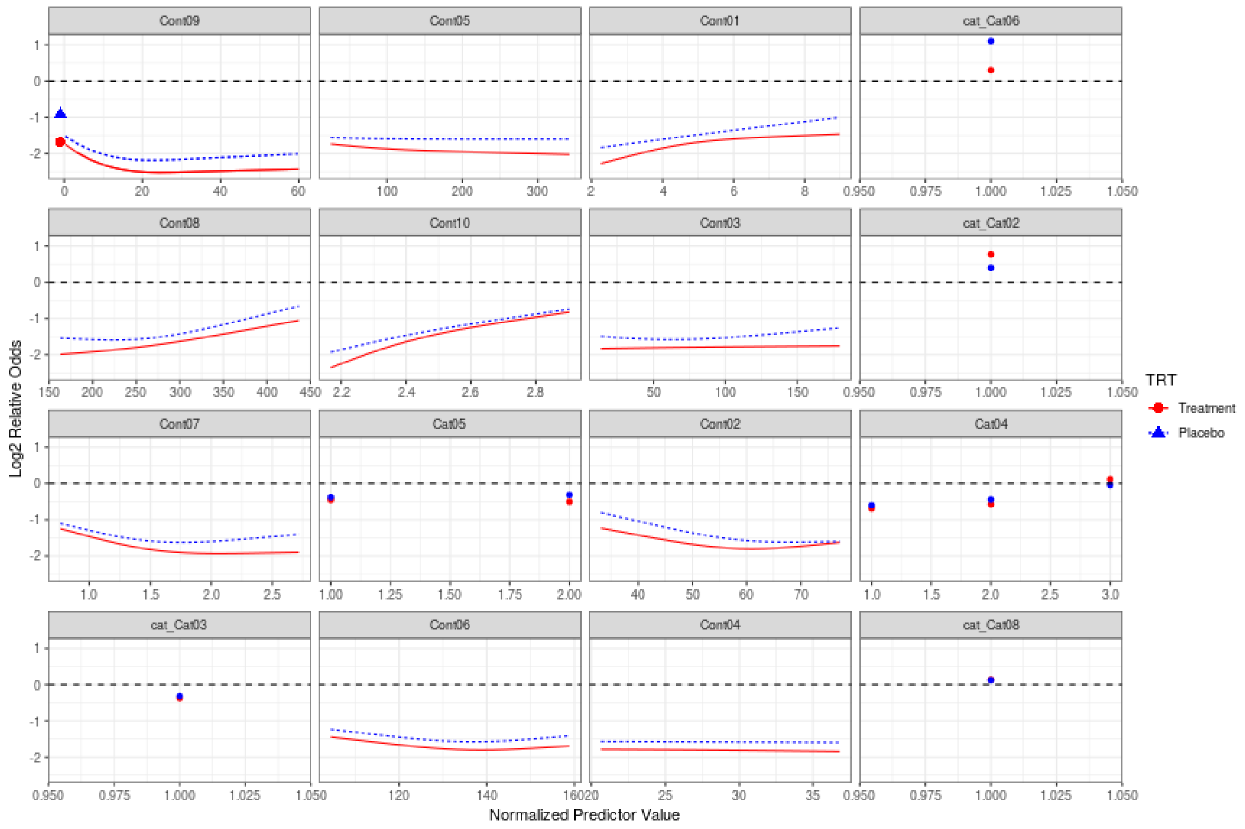
```
# For 1st BM we create 2 (one for each ARM) sequences from 10% to 90% followed by median for remaining 
# For 2nd BM and on we start with median for preceding BM followed by sequence
for(i in seq_len(length(cont_bm))){
  if(i == 1) {
    pred_dat[[cont_bm[i]]] <- c(rep(seq(quantile(imp_dat[[cont_bm[i]]], 0.05),
                                        quantile(imp_dat[[cont_bm[i]]], 0.95),
                                        length.out = nintp), 2),
                                rep(meds[i], (length(all_bm)-i)*nintp*2))
  } else {
    pred_dat[[cont_bm[i]]] <- c(rep(meds[i], (i-1)*nintp*2),
                                rep(seq(quantile(imp_dat[[cont_bm[i]]], 0.05),
                                        quantile(imp_dat[[cont_bm[i]]], 0.95),
                                        length.out = nintp), 2), rep(meds[i],
                                                                     (length(all_bm)-i)*nintp*2))
  }

  # Create 2 basis functions from interpolated data using same knots as in model
  # NOTE: This code assumes there were only 3 knots in model for all continuous variables
  pred_dat[[paste0("mod_", cont_bm[i], "1")]] <- rcs(pred_dat[[cont_bm[[i]]]],
                                                     attributes(cont_dat[[paste0("mod_", cont_bm[i], "1
                                                     name = pred_dat_names[(2*i) - 1])[,1]
  attr(pred_dat[[paste0("mod_", cont_bm[i], "1")]], "dimnames") <- NULL
  pred_dat[[paste0("mod_", cont_bm[i], "2")]] <- rcs(pred_dat[[cont_bm[[i]]]],
                                                     attributes(cont_dat[[paste0("mod_", cont_bm[i], "2
                                                     name = pred_dat_names[2*i])[,2]
  attr(pred_dat[[paste0("mod_", cont_bm[i], "2")]], "dimnames") <- NULL
}

# For Cont09 treat as 2nd BM or later from previous chunk
for(i in seq_len(length(Ind_bm))){
  j <- length(cont_bm) + i # Convenience to recalibrate Ind_bm to cont_bm
  pred_dat[[Ind_bm[i]]] <- c(rep(meds[j], (j - 1)*nintp*2),
                             rep(seq(quantile(imp_dat[[Ind_bm[i]]], 0.05),
                                     quantile(imp_dat[[Ind_bm[i]]], 0.95),
                                     length.out = nintp), 2),
                             rep(meds[j], (length(all_bm)-j)*nintp*2))

  pred_dat[[paste0("Ind_", Ind_bm[i], "1")]] <- rcs(pred_dat[[Ind_bm[[i]]]],
                                                    attributes(cont_dat[[paste0("Ind_", Ind_bm[i], "1")]
                                                    name = pred_dat_names[(2*j) - 1])[,1]
  attr(pred_dat[[paste0("Ind_", Ind_bm[i], "1")]], "dimnames") <- NULL
  pred_dat[[paste0("Ind_", Ind_bm[i], "2")]] <- rcs(pred_dat[[Ind_bm[[i]]]],
                                                    attributes(cont_dat[[paste0("Ind_", Ind_bm[i], "2")]
                                                    name = pred_dat_names[2*j])[,2]
  attr(pred_dat[[paste0("Ind_", Ind_bm[i], "2")]], "dimnames") <- NULL
}


pred_dat <- as.data.frame(pred_dat)

# Create Indicator functions for Cont09
pred_dat <- mutate(pred_dat, Ind_Cont09_s_Ind = ifelse(Cont09_s == min(imp_dat$Cont09_s), 1, 0))
```

```r
# Keep only terms used in model
pred_dat <- dplyr::select(pred_dat, str_subset(colnames(pred_dat), "mod"),
                          str_subset(colnames(pred_dat), "Ind"))

# Create TRT2
if(simon == "Yes") {
pred_dat$TRT2 <- rep(c(1, -1), each = nintp, times = length(all_bm))
} else if(simon == "No"){
  pred_dat$TRT2 <- rep(c(1, 0), each = nintp, times = length(all_bm))
}


# Create categorical variables set to reference levels
for(var in cat_mod){
  pred_dat[[var]] <- 0
}

# Create interactions for all variables
pred_dat <- mutate_at(pred_dat, bm, list(modINT = ~. * TRT2))
pred_dat <- mutate_at(pred_dat, Cat04, list(Cat04INT = ~. * TRT2))
pred_dat <- mutate_at(pred_dat, Cat05, list(Cat05INT = ~. * TRT2))
pred_dat <- mutate_at(pred_dat, clin, list(catINT = ~. * TRT2))
pred_dat <- mutate_at(pred_dat, Ind, list(IndINT = ~. * TRT2))


#################################
# Get predictions
###################################

# !!!! BUG IN predict.grpsurv !!!!
# Must have prediction matrix IN SAME ORDER as training data! If not, results will be INCORRECT WITH NO

# Ensure prediction matrix is same order as training data
pred_dat <- dplyr::select(pred_dat, PARAMS$x)
pred_mat <- as.matrix(pred_dat)

# Get Predictions
pred_dat$preds <- predict(full_PR, X = pred_mat, type = "link", which = index)

# Identify which BM is getting predicted
pred_dat$BM <- rep(all_bm, each = 2*nintp)

#################################
# Make partial effects plots
##################################

raw_bm <- sub("_s$", "", all_bm)
plot_dat <- list()

# Create dataframe of variables on original scale, but mapping predictions
for(i in seq_len(length(raw_bm))) {
  plot_dat[[raw_bm[i]]] <- rep(seq(quantile(red_dat[[raw_bm[i]]], 0.05, na.rm = T),
                                   quantile(red_dat[[raw_bm[i]]], 0.95, na.rm = T),
                                    length.out = nintp), 2)
}


plot_dat <- as.data.frame(plot_dat)

plot_dat$TRT <-  rep(c("Treatment", "Placebo"), each = nintp)

plot_dat <- pivot_longer(plot_dat, -TRT, names_to = "Biomarker", values_to = "Value")

# Rearrange in order to match up order with pred_dat
plot_dat <- arrange(plot_dat, match(Biomarker, raw_bm), TRT, Value)

# Since everything matches, we transfer preds
plot_dat$preds <- pred_dat$preds

plot_dat <- mutate(plot_dat, log2_preds = preds*log2(exp(1)))

# Identify filtered BM
filtered <- mod_var[!(mod_var %in% mod_var[predict(full_PR, X = X, type = "groups", which = index)])]
filtered <- sub("_s$", "", filtered)

# Filter out biomarkers removed
plot_dat_filtered <- filter(plot_dat, !(Biomarker %in% filtered))




# Separate out point of discontinuity for better visualization
nuggets <- filter(plot_dat_filtered, Biomarker == "Cont09" & Value == min(red_dat$Cont09, na.rm = T))
nuggets <- mutate(nuggets, Value2 = Value - 1)


no_nugget <- filter(plot_dat_filtered, (Biomarker == "Cont09" &
                                          Value != min(red_dat$Cont09, na.rm = T)) | Biomarker != "Cont0


# Create dataframe for categorical variables with corresponding columns
coef_cat_w <- mutate(coef_cat_w,
                     BM2 = ifelse(str_detect(BM, pattern = "cat|TypeB|TypeB"), 1,
                                  ifelse(str_detect(BM, pattern = "TypeC|TypeC"), 2, 3)))

cat_plot_dat <- data.frame(TRT = coef_cat_w$Treatment, Biomarker = coef_cat_w$group,
                           Value = coef_cat_w$BM2, preds = coef_cat_w$log_OR,
                           log2_preds = coef_cat_w$log2_OR)


# Combine with no_nuggets dataframe
combined_plot_dat <- rbind(no_nugget, cat_plot_dat)

# Set up ranking to combine
Int_rank2 <- Int_rank
Int_rank2 <- mutate(Int_rank2, Biomarker = gsub("_s", "", Biomarker))
```

```r
combined_plot_dat <- inner_join(combined_plot_dat, Int_rank2)
combined_plot_dat$Biomarker <- reorder(combined_plot_dat$Biomarker, combined_plot_dat$Rank)
nuggets <- inner_join(nuggets, Int_rank2)
nuggets$Biomarker <- reorder(nuggets$Biomarker, nuggets$Rank)
cat_plot_dat <- inner_join(cat_plot_dat, Int_rank2)
cat_plot_dat$Biomarker <- reorder(cat_plot_dat$Biomarker, cat_plot_dat$Rank)

all_bm2 <- gsub("_s", "", all_bm)


tmp <- ggplot(combined_plot_dat, aes(x = Value, y = log2_preds, color = TRT, linetype = TRT)) +
  geom_line(data = subset(combined_plot_dat, Biomarker == all_bm2)) +
  geom_line(data = subset(combined_plot_dat, Biomarker == "Cont09")) + # To get around bug where some o
  facet_wrap(~Biomarker, scales = "free_x") +
  ylab("Log2 Relative Odds") +
  xlab("Normalized Predictor Value") +
  scale_shape_discrete(labels=c("Treatment", "Placebo")) +
  scale_linetype_discrete(labels=c("Treatment", "Placebo")) +
  scale_color_manual(values = c("red", "blue"), labels = c("Treatment", "Placebo")) +
  theme_bw()
```

```
## Warning in `==.default`(Biomarker, all_bm2): longer object length is not a
## multiple of shorter object length

## Warning in is.na(e1) | is.na(e2): longer object length is not a multiple of
## shorter object length
```

```r
tmp +
geom_point(data=nuggets,
                 aes(x=Value2,y=log2_preds, color = TRT, shape = TRT), size = 3) +
geom_point(data = cat_plot_dat,
           aes(x = Value, y = log2_preds, color = TRT), inherit.aes = F) +
  geom_hline(yintercept = 0, linetype = "dashed")
```



```r
# scale_x_discrete()
```

6 References

Gelman, A., 2008. Scaling regression inputs by dividing by two standard deviations. Statistics in medicine 27, 2865–2873.

Tian, L., Alizadeh, A.A., Gentles, A.J., Tibshirani, R., 2014. A simple method for estimating interactions between a treatment and a large number of covariates. Journal of the American Statistical Association 109, 1517–1532.

7 System Information

Time required to process this report: *1.359608 mins*

R session information:

```
## R version 4.3.1 (2023-06-16)
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 18.04.6 LTS
##
## Matrix products: default
## BLAS:   /opt/tbio/domino_202310/binaries/R-4.3.1/lib/R/lib/libRblas.so
## LAPACK: /opt/tbio/domino_202310/binaries/R-4.3.1/lib/R/lib/libRlapack.so;  LAPACK version 3.11.0
##
## locale:
```

```
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## time zone: America/New_York
## tzcode source: system (glibc)
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] lubridate_1.9.3  forcats_1.0.0    purrr_1.0.2      readr_2.1.4
##  [5] tidyverse_2.0.0  grpreg_3.4.0     rmsb_1.0-0       rms_6.7-1
##  [9] Hmisc_5.1-1      corrplot_0.92    bmsPURR_0.1.34   tidyr_1.3.0
## [13] tibble_3.2.1     stringr_1.5.0    ggplot2_3.4.3    dplyr_1.1.3
## [17] conflicted_1.2.0
##
## loaded via a namespace (and not attached):
##   [1] rstudioapi_0.15.0   jsonlite_1.8.7      CatMisc_1.1.5
##   [4] magrittr_2.0.3      TH.data_1.1-2       farver_2.1.1
##   [7] rmarkdown_2.25      fs_1.6.3            vctrs_0.6.3
##  [10] memoise_2.0.1       askpass_1.2.0       base64enc_0.1-3
##  [13] htmltools_0.5.6     usethis_2.2.2       polspline_1.1.23
##  [16] Formula_1.2-5       StanHeaders_2.26.28 htmlwidgets_1.6.2
##  [19] desc_1.4.2          sandwich_3.0-2      zoo_1.8-12
##  [22] cachem_1.0.8        DominoR_0.0.2       mime_0.12
##  [25] lifecycle_1.0.3     pkgconfig_2.0.3     Matrix_1.6-1.1
##  [28] R6_2.5.1            fastmap_1.1.1       shiny_1.7.5
##  [31] digest_0.6.33       pingr_2.0.2         colorspace_2.1-0
##  [34] ps_1.7.5            rprojroot_2.0.3     pkgload_1.3.3
##  [37] labeling_0.4.3      timechange_0.2.0    fansi_1.0.4
##  [40] httr_1.4.7          compiler_4.3.1      remotes_2.4.2.1
##  [43] withr_2.5.1         htmlTable_2.4.1     backports_1.4.1
##  [46] inline_0.3.19       QuickJSR_1.0.6      pkgbuild_1.4.2
##  [49] MASS_7.3-60         quantreg_5.97       openssl_2.1.1
##  [52] sessioninfo_1.2.2   myRepository_1.34.0 loo_2.6.0
##  [55] tools_4.3.1         foreign_0.8-85      httpuv_1.6.11
##  [58] nnet_7.3-19         glue_1.6.2          callr_3.7.3
##  [61] nlme_3.1-163        promises_1.2.1      grid_4.3.1
##  [64] checkmate_2.2.0     getPass_0.2-2       cluster_2.1.4
##  [67] generics_0.1.3      ParamSetI_1.17.0    gtable_0.3.4
##  [70] EventLogger_1.15.3  tzdb_0.4.0          data.table_1.14.8
##  [73] hms_1.1.3           utf8_1.2.3          pillar_1.9.0
##  [76] later_1.3.1         splines_4.3.1       dynamictable_1.5
##  [79] lattice_0.21-9      survival_3.5-7      SparseM_1.81
##  [82] tidyselect_1.2.0    miniUI_0.1.1.1      knitr_1.44
##  [85] gridExtra_2.3       stats4_4.3.1        xfun_0.40
##  [88] devtools_2.4.5      matrixStats_1.0.0   rstan_2.26.23
##  [91] stringi_1.7.12      yaml_2.3.7          evaluate_0.22
##  [94] codetools_0.2-19    cli_3.6.1           RcppParallel_5.1.7
##  [97] rpart_4.1.19        xtable_1.8-4        munsell_0.5.0
## [100] processx_3.8.2      Rcpp_1.0.11         MatrixModels_0.5-2
## [103] ellipsis_0.3.2      prettyunits_1.2.0   profvis_0.3.8
## [106] urlchecker_1.0.1    mvtnorm_1.2-3       scales_1.2.1
## [109] crayon_1.5.2        rlang_1.1.1         multcomp_1.4-25
```

7.1 Domino environment details

Domino User: apfela

Domino Project:

Domino Run ID:

Domino Run #: