2018-09-12 14:46:38

Here's how we currently grab the results:

```
echo "target,pheno,var,nfeat,model,metric,val" > auto_summary.csv;
for y in diag_group2 random_HI_slope random_total_slope random_inatt_slope
\
    OLS_inatt_slope OLS_HI_slope OLS_total_slope \
    group_HI3 group_total3 group_INATT3; do
    for f in `grep -l \"${y} *o`; do
        phen=`head -n 2 $f | tail -1 | awk '{FS=" "; print $6}' | cut -
d"/" -f 4 | cut -d"_" -f 1,2 -`;
        var=`head -n 2 $f | tail -1 | awk '{FS=" "; print $5}' | cut -d"/"
-f 4 | sed -e "s/\.R//g"`;
        model=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $2}'
| cut -d"_" -f 1`;
        acc=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $3}'`;
        metric=`grep -A 0 model_id $f | awk '{FS=" "; print $2}'`;
        nfeat=`grep -e "26. " $f | tail -1 | awk '{ print $3 }'`;
        echo $y,$phen,$var,$nfeat,$model,$metric,$acc >> auto_summary.csv;
    done;
done
```

It looks like uni01 is not doing much. For the ones that it actually run (sometimes there were no variables p < .01)), the results were not better than when using uni. That could indicate overfitting, but also there could be features that interact well and hacving so few features (As in uni01) means shooting ourselves in the foot.

2018-09-14 12:34:50

I updated the code to use all the improvements/enhancements we have in the limited version, except that it's not limited to algorithms or time. In fact, I think the filtering to the structural data puts the number of variables in a manageable number, so we don't need to run the limited code anymore. And I have alreayd noticed that the raw code breaks because it runs out of memory... let's run it anyways, and let it die.

```
rm swarm.automl
for var in raw pca uni pca_uni uni_pca uni01; do
    for m in area thickness volume; do
        for sx in inatt HI total; do
            for sl in OLS random; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/struct_${m}_08312018_260timeDiff12mo.RData.gz
~/data/baseline_prediction/struct_gf_09052018_260timeDiff12mo.csv
${sl}_${sx}_slope ~/tmp/${m}_${sl}_${sx}" >> swarm.automl;
            done;
        done;
        for sx in INATT HI total; do
```

```
            echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/struct_${m}_08312018_260timeDiff12mo.RData.gz
~/data/baseline_prediction/struct_gf_09052018_260timeDiff12mo.csv
group_${sx}3 ~/tmp/${m}_${sx}" >> swarm.automl;
        done;
        echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/struct_${m}_08312018_260timeDiff12mo.RData.gz
~/data/baseline_prediction/struct_gf_09052018_260timeDiff12mo.csv
diag_group2 ~/tmp/${m}_diag_group2" >> swarm.automl;
    done;
done;
sed -i -e "s/^/unset http_proxy; /g" swarm.automl;
swarm -f swarm.automl -g 40 -t 32 --time 1-00:00:00 --logdir trash_bin --
job-name struc -m R --gres=lscratch:10
```

We can also run the snp code. I also don't think there's much value in running the PCA code there, so it's just the raw code.

```
rm swarm.automl
var=raw;
for m in {4..9}; do
    for sx in inatt HI total; do
        for sl in OLS random; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/geno3_snps1e0${m}_09132018.RData.gz
~/data/baseline_prediction/geno3_gf_09142018.csv ${sl}_${sx}_slope
~/tmp/${m}_${sl}_${sx}" >> swarm.automl;
        done;
    done;
    for sx in INATT HI total; do
        echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/geno3_snps1e0${m}_09132018.RData.gz
~/data/baseline_prediction/geno3_gf_09142018.csv group_${sx}3
~/tmp/${m}_${sx}" >> swarm.automl;
    done;
    echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/geno3_snps1e0${m}_09132018.RData.gz
~/data/baseline_prediction/geno3_gf_09142018.csv diag_group2
~/tmp/${m}_diag_group2" >> swarm.automl;
done;
m='prs'
for sx in inatt HI total; do
    for sl in OLS random; do
        echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/geno3_${m}_09142018.RData.gz
~/data/baseline_prediction/geno3_gf_09142018.csv ${sl}_${sx}_slope
~/tmp/${m}_${sl}_${sx}" >> swarm.automl;
    done;
done;
for sx in INATT HI total; do
    echo "Rscript --vanilla ~/research_code/automl/${var}.R
```

```
~/data/baseline_prediction/geno3_${m}_09142018.RData.gz
~/data/baseline_prediction/geno3_gf_09142018.csv group_${sx}3
~/tmp/${m}_${sx}" >> swarm.automl;
done;
echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/geno3_${m}_09142018.RData.gz
~/data/baseline_prediction/geno3_gf_09142018.csv diag_group2
~/tmp/${m}_diag_group2" >> swarm.automl;
sed -i -e "s/^/unset http_proxy; /g" swarm.automl;
swarm -f swarm.automl -g 40 -t 32 --time 1-00:00:00 --logdir trash_bin --
job-name geno -m R --gres=lscratch:10
```

And since we're running all for comparison, let's attach neuropsych as well:

```
for var in raw pca uni pca_uni uni_pca uni01; do
    for m in all cpt wiscraw wiscstd wj iq; do
        for sx in inatt HI total; do
            for sl in OLS random; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/cog_${m}_09142018.RData.gz
~/data/baseline_prediction/cog_gf_09142018.csv ${sl}_${sx}_slope
~/tmp/${m}_${sl}_${sx}" >> swarm.automl;
            done;
        done;
        for sx in INATT HI total; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/cog_${m}_09142018.RData.gz
~/data/baseline_prediction/cog_gf_09142018.csv group_${sx}3
~/tmp/${m}_${sx}" >> swarm.automl;
        done;
        echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/cog_${m}_09142018.RData.gz
~/data/baseline_prediction/cog_gf_09142018.csv diag_group2
~/tmp/${m}_diag_group2" >> swarm.automl;
        done;
    done;
done;
```

2018-09-18 10:03:19

Then we collect all results again, but with a small change to the script so we can correctly capture the number of features used:

```
echo "target,pheno,var,nfeat,model,metric,val" > auto_summary.csv;
for y in diag_group2 random_HI_slope random_total_slope random_inatt_slope
\
    OLS_inatt_slope OLS_HI_slope OLS_total_slope \
    group_HI3 group_total3 group_INATT3; do
    for f in `grep -l \"${y} *o`; do
        phen=`head -n 2 $f | tail -1 | awk '{FS=" "; print $6}' | cut -
d"/" -f 4 | cut -d"_" -f 1,2 -`;
        var=`head -n 2 $f | tail -1 | awk '{FS=" "; print $5}' | cut -d"/"
-f 4 | sed -e "s/\.R//g"`;
        model=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $2}'
| cut -d"_" -f 1`;
        acc=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $3}'`;
        metric=`grep -A 0 model_id $f | awk '{FS=" "; print $2}'`;
        if [[ $var == 'raw' ]] && [[ $phen == 'dti_ALL' ]]; then
            nfeat=36066;
        elif [[ $var == 'raw' ]] && [[ $phen = *'dti_'* ]]; then
            nfeat=12022;
        elif grep -q "Running model on" $f; then  # newer versions of the
script
            nfeat=`grep "Running model on" $f | awk '{FS=" "; print $5}'`;
        else # older versions were less verbose
            nfeat=`grep -e "26. *[1-9]" $f | grep "\[1\]" | tail -1 | awk
'{ print $3 }'`;
        fi
        echo $y,$phen,$var,$nfeat,$model,$metric,$acc >> auto_summary.csv;
    done;
done
```

So I saved auto_summary_09182018.csv in ~/Dcouments/baseline_prediction.

2018-09-19 10:22:19

Now we run the resting state variables:

```
for var in raw pca uni pca_uni uni_pca uni01; do
    for m in aparc aparc.a2009s aparc_trimmed aparc.a2009s_trimmed; do
        for sx in inatt HI total; do
            for sl in OLS random; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/${m}_n215_09182018.RData.gz
~/data/baseline_prediction/rsfmri_gf_09182018.csv ${sl}_${sx}_slope
~/tmp/${m}_${sl}_${sx}" >> swarm.automl;
            done;
        done;
        for sx in INATT HI total; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/${m}_n215_09182018.RData.gz
~/data/baseline_prediction/rsfmri_gf_09182018.csv group_${sx}3
~/tmp/${m}_${sx}" >> swarm.automl;
        done;
        echo "Rscript --vanilla ~/research_code/automl/${var}.R
~/data/baseline_prediction/${m}_n215_09182018.RData.gz
~/data/baseline_prediction/rsfmri_gf_09182018.csv diag_group2
~/tmp/${m}_diag_group2" >> swarm.automl;
```

```
    done;
done;
sed -i -e "s/^/unset http_proxy; /g" swarm.automl;
cp swarm.automl swarm.automl_rsfmri;
swarm -f swarm.automl -g 40 -t 32 --time 4:00:00 --partition quick --
logdir trash_bin --job-name rsfmri -m R --gres=lscratch:10
```

Subgroup analysis

I then added some prep code to the functions to handle the different subgroups we're analyzing. But before I could do that, in order to use a single gf file across modalities, I had ot make a change to our older datasets, so that they included the MRN. The merge takes care of only including the MRNs for which we have that specific dataset...

```
> library(gdata)
> clin = read.xls('~/data/baseline_prediction/long_scans_08072018.xlsx',
'dti')
>
load('~/data/baseline_prediction/dti_ad_voxelwise_n263_08152018.RData.gz')
> clin = clin[, c(1,4)]
> colnames(clin) = c('MRN', 'mask.id')
> data = merge(clin, data, by='mask.id')
> dim(data)
[1]   263 12024
> save(data,
file='~/data/baseline_prediction/dti_ad_voxelwise_n263_09192018.RData.gz',
compress=T)
>
load('~/data/baseline_prediction/dti_rd_voxelwise_n263_08152018.RData.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/dti_rd_voxelwise_n263_09192018.RData.gz',
compress=T)
>
load('~/data/baseline_prediction/dti_fa_voxelwise_n263_08152018.RData.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/dti_fa_voxelwise_n263_09192018.RData.gz',
compress=T)
>
load('~/data/baseline_prediction/dti_ALL_voxelwise_n263_08152018.RData.gz'
)
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/dti_ALL_voxelwise_n263_09192018.RData.gz'
, compress=T)
>
load('~/data/baseline_prediction/dti_tracts_voxelwise_n263_08152018.RData.
gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/dti_tracts_voxelwise_n263_09192018.RData.
gz', compress=T)

> clin = read.xls('~/data/baseline_prediction/long_scans_08072018.xlsx',
'mprage')
> clin = clin[, c(1,4)]
> colnames(clin) = c('MRN', 'mask.id')
>
load('~/data/baseline_prediction/struct_rois_09062018_260timeDiff12mo.RDat
a.gz')
> data = merge(clin, data, by='mask.id')
> dim(data)
[1] 260 273
> save(data,
file='~/data/baseline_prediction/struct_rois_09192018_260timeDiff12mo.RDat
a.gz', compress=T)
>
load('~/data/baseline_prediction/struct_area_08312018_260timeDiff12mo.RDat
a.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/struct_area_09192018_260timeDiff12mo.RDat
a.gz', compress=T)
>
load('~/data/baseline_prediction/struct_thickness_08312018_260timeDiff12mo
.RData.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/struct_thickness_09192018_260timeDiff12mo
.RData.gz', compress=T)
>
load('~/data/baseline_prediction/struct_volume_08312018_260timeDiff12mo.RD
ata.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/struct_volume_09192018_260timeDiff12mo.RD
ata.gz', compress=T)

> clin =
read.csv('/Volumes/Shaw/prs/FINAL_FILES_08242018/REGRESSION/PRS2017_geno3_
1KG_noFlip_genop05MAFbtp01rsbtp9.csv')
> clin = clin[,1:2]
> load('~/data/baseline_prediction/geno3_prs_09142018.RData.gz')
> colnames(clin) = c('mask.id', 'MRN')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/geno3_prs_09192018.RData.gz', compress=T)
> load('~/data/baseline_prediction/geno3_snps1e04_09132018.RData.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/geno3_snps1e04_09192018.RData.gz',
compress=T)
> load('~/data/baseline_prediction/geno3_snps1e05_09132018.RData.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/geno3_snps1e05_09192018.RData.gz',
compress=T)
> load('~/data/baseline_prediction/geno3_snps1e06_09132018.RData.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/geno3_snps1e06_09192018.RData.gz',
compress=T)
> load('~/data/baseline_prediction/geno3_snps1e07_09132018.RData.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/geno3_snps1e07_09192018.RData.gz',
compress=T)
> load('~/data/baseline_prediction/geno3_snps1e08_09132018.RData.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/geno3_snps1e08_09192018.RData.gz',
compress=T)
> load('~/data/baseline_prediction/geno3_snps1e09_09132018.RData.gz')
> data = merge(clin, data, by='mask.id')
> save(data,
file='~/data/baseline_prediction/geno3_snps1e09_09192018.RData.gz',
compress=T)
```

Now, we need to construct a super swarm file. I'll also called the job differently so we can compile results just within these new log files:

```
for var in raw pca uni pca_uni uni_pca uni01; do
    for f in `/bin/ls -1 ~/data/baseline_prediction/*0919*gz
~/data/baseline_prediction/cog*gz ~/data/baseline_prediction/aparc*gz`;do
        for nn in nonew_ ''; do
            for g in ADHDonly_ ''; do
                for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope; do
                    echo "Rscript --vanilla
~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/tmp/${nn}${g}${t}" >> swarm.automl_subgroup;
                done;
            done;
            for g in ADHDNOS_ ADHDNOS_group; do
                for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope; do
                    echo "Rscript --vanilla
~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/tmp/${nn}${g}${t}" >> swarm.automl_subgroup;
                done;
            done;
        done;
    done;
```

```
done;
sed -i -e "s/^/unset http_proxy; /g" swarm.automl_subgroup;
swarm -f swarm.automl_subgroup -g 40 -t 32 --time 4:00:00 --logdir
trash_bin --job-name subgroups -m R --gres=lscratch:10
```

2018-09-20 09:18:45

SLURM was bundling my swarms, which screwed up my output files. So, I had to split the runs:

```
split -l 500 swarm.automl_subgroup
for f in `/bin/ls -1 xa?`; do swarm -f $f -g 40 -t 32 --time 4:00:00 --
partition quick --logdir trash_bin --job-name subgroups2 -m R --
gres=lscratch:10; done
```

While we wait on results, let's collect the rsfmri results to add to the previous summary:

```
echo "target,pheno,var,nfeat,model,metric,val" > auto_summary.csv;
for y in diag_group2 random_HI_slope random_total_slope random_inatt_slope
\
    OLS_inatt_slope OLS_HI_slope OLS_total_slope \
    group_HI3 group_total3 group_INATT3; do
    for f in `grep -l \"${y} rsfmri*o`; do
        phen=`head -n 2 $f | tail -1 | awk '{FS=" "; print $6}' | cut -
d"/" -f 4 | cut -d"_" -f 1,2 -`;
        var=`head -n 2 $f | tail -1 | awk '{FS=" "; print $5}' | cut -d"/"
-f 4 | sed -e "s/\.R//g"`;
        model=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $2}'
| cut -d"_" -f 1`;
        acc=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $3}'`;
        metric=`grep -A 0 model_id $f | awk '{FS=" "; print $2}'`;
        if [[ $var == 'raw' ]] && [[ $phen == 'dti_ALL' ]]; then
            nfeat=36066;
        elif [[ $var == 'raw' ]] && [[ $phen = *'dti_'* ]]; then
            nfeat=12022;
        elif grep -q "Running model on" $f; then  # newer versions of the
script
            nfeat=`grep "Running model on" $f | awk '{FS=" "; print $5}'`;
        else # older versions were less verbose
            nfeat=`grep -e "26. *[1-9]" $f | grep "\[1\]" | tail -1 | awk
'{ print $3 }'`;
        fi
        echo $y,$phen,$var,$nfeat,$model,$metric,$acc >> auto_summary.csv;
    done;
done
```

2018-09-21 10:15:52

Now we compile the results from the subgroup analysis:

```
echo "target,pheno,var,nfeat,model,metric,val" > auto_summary.csv;
for y in diag_group2 OLS_inatt_slope OLS_HI_slope OLS_total_slope; do
    for f in `grep -l \"${y} subgroups2*o`; do
        phen=`head -n 2 $f | tail -1 | awk '{FS=" "; print $6}' | cut -
d"/" -f 6 | cut -d"_" -f 1,2 -`;
        target=`head -n 2 $f | tail -1 | awk '{FS=" "; print $8}'`;
        var=`head -n 2 $f | tail -1 | awk '{FS=" "; print $5}' | cut -d"/"
-f 4 | sed -e "s/\.R//g"`;
        model=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $2}'
| cut -d"_" -f 1`;
        acc=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $3}'`;
        metric=`grep -A 0 model_id $f | awk '{FS=" "; print $2}'`;
        if [[ $var == 'raw' ]] && [[ $phen == 'dti_ALL' ]]; then
            nfeat=36066;
        elif [[ $var == 'raw' ]] && [[ $phen = *'dti_'* ]]; then
            nfeat=12022;
        elif grep -q "Running model on" $f; then  # newer versions of the
script
            nfeat=`grep "Running model on" $f | awk '{FS=" "; print $5}'`;
        else # older versions were less verbose
            nfeat=`grep -e "26. *[1-9]" $f | grep "\[1\]" | tail -1 | awk
'{ print $3 }'`;
        fi
        echo $target,$phen,$var,$nfeat,$model,$metric,$acc >>
auto_summary.csv;
    done;
done
```

Results too good to be true? getting very close to 100% accuracy for subgroups, but that's true even when using leaderboard scoring, and forced 5-fold CV... look at uni01.R for tests... Also, nice that loading saved models bring up the cross-validation scores. Maybe plot it in the brain to see if it makes sense, implement some dummy scoring, and calculate univariate filters only in training set? Maybe I could get the leaderboard also through Kfold, setting it form the outside, and then I can accumulate several test results that can give us a distribution...

The issue there is that we could potentially be succeptible to different models. Something to think about...

2018-09-24 10:02:38

Just noticed that I didn't change the correct variables in the cog files... see that specific notes for new files. Now, I'm just re-running those cog files, after changing the date.

```
split -l 600 swarm.automl_subgroupCog
for f in xaa xab; do swarm -f $f -g 40 -t 32 --time 4:00:00 --partition
```

```
quick --logdir trash_bin --job-name subgroupsCog -m R --gres=lscratch:10;
done
```

Let's also collect the results of using DTI with the new data:

```
echo "target,pheno,var,nfeat,model,metric,val" > auto_summary.csv;
for y in diag_group2 random_HI_slope random_total_slope random_inatt_slope
\
    OLS_inatt_slope OLS_HI_slope OLS_total_slope \
    group_HI3 group_total3 group_INATT3; do
    for f in `grep -l \"${y} subgroupDTI*o`; do
        phen=`head -n 2 $f | tail -1 | awk '{FS=" "; print $6}' | cut -
d"/" -f 6`;
        target=`head -n 2 $f | tail -1 | awk '{FS=" "; print $8}'`;
        var=`head -n 2 $f | tail -1 | awk '{FS=" "; print $5}' | cut -d"/"
-f 4 | sed -e "s/\.R//g"`;
        model=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $2}'
| cut -d"_" -f 1`;
        acc=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $3}'`;
        metric=`grep -A 0 model_id $f | awk '{FS=" "; print $2}'`;
        if [[ $var == 'raw' ]] && [[ $phen == 'dti_ALL' ]]; then
            nfeat=36066;
        elif [[ $var == 'raw' ]] && [[ $phen = *'dti_'* ]]; then
            nfeat=12022;
        elif grep -q "Running model on" $f; then  # newer versions of the
script
            nfeat=`grep "Running model on" $f | awk '{FS=" "; print $5}'`;
        else # older versions were less verbose
            nfeat=`grep -e "26. *[1-9]" $f | grep "\[1\]" | tail -1 | awk
'{ print $3 }'`;
        fi
        echo $target,$phen,$var,$nfeat,$model,$metric,$acc >>
auto_summary.csv;
    done;
done
```

I then changed uni01_limited.R to use a leaderboard (test) set, fix the CV folds, output a proxy for dummy predictions, and also to calculate the univariate filter using only the training data. Let's run a few iterations of it, using only the DTI_all dataset, just to see what's the spread in the test set.

```
var=uni01_limited
f=~/data/baseline_prediction/dti_ALL_voxelwise_n272_09212018.RData.gz
for target in nonew_ADHDonly_diag_group2 diag_group2
nonew_ADHDonly_OLS_inatt_slope; do
    for i in {1..100}; do
        echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${target}
~/data/baseline_prediction/models/${target} $RANDOM" >>
swarm.automl_LBtest;
```

```
    done;
done;
sed -i -e "s/^/unset http_proxy; /g" swarm.automl_LBtest;
swarm -f swarm.automl_LBtest -g 40 --partition quick -t 32 --time 4:00:00
--logdir trash_bin --job-name lb -m R --gres=lscratch:10
```

2018-09-25 09:57:37

Let's see how the spread looks like...

```
echo "target,pheno,var,nfeat,model,metric,val,dummy" > auto_summary.csv;
for y in nonew_ADHDonly_diag_group2 diag_group2
nonew_ADHDonly_OLS_inatt_slope; do
    for f in `grep -l \"${y} lb_*o`; do
        phen=`head -n 2 $f | tail -1 | awk '{FS=" "; print $6}' | cut -
d"/" -f 6`;
        target=`head -n 2 $f | tail -1 | awk '{FS=" "; print $8}'`;
        var=`head -n 2 $f | tail -1 | awk '{FS=" "; print $5}' | cut -d"/"
-f 4 | sed -e "s/\.R//g"`;
        model=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $2}'
| cut -d"_" -f 1`;
        acc=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $3}'`;
        metric=`grep -A 0 model_id $f | awk '{FS=" "; print $2}'`;
        if [[ $var == 'raw' ]] && [[ $phen == 'dti_ALL' ]]; then
            nfeat=36066;
        elif [[ $var == 'raw' ]] && [[ $phen = *'dti_'* ]]; then
            nfeat=12022;
        elif grep -q "Running model on" $f; then  # newer versions of the
script
            nfeat=`grep "Running model on" $f | awk '{FS=" "; print $5}'`;
        else # older versions were less verbose
            nfeat=`grep -e "26. *[1-9]" $f | grep "\[1\]" | tail -1 | awk
'{ print $3 }'`;
        fi
        if grep -q "Class distribution" $f; then
            dummy=`grep -A 1 "Class distribution" $f | tail -1 | awk
'{FS=" "; {for (i=2; i<=NF; i++) printf $i ";"}}'`;
        else
            dummy=`grep -A 1 "MSE prediction mean" $f | tail -1 | awk
'{FS=" "; print $2}'`;
        fi
        echo $target,$phen,$var,$nfeat,$model,$metric,$acc,$dummy >>
auto_summary.csv;
    done;
done
```

So, the results were quite good (auto_summary_leaderboard_09252018). For the binary group decision between persistent and remitted (nonew_ADHDonly_diag_group2), we get a mean over different seeds of about 90%. The class majority would be around 70%. Note that I had the training seed fixed at 42, but that's not an issue.

I also did some research on what happens to the data in H2O autoML. By looking at the confusion matrix of the best models, it's clear that the actual training data is (about) 80% of the data sent in the training argument, with the rest used for validation. Note that the 5-fold cross-validation happens only within the training data, so it's 5-fold on the 80%. For example, if df2 is sent for training, and it has 110 observations, 81 observations were used for training and 29 for validation. The 5-fold is done only within the 81. So, in this case, we're actualing fitting a combination of algorithm and grid parameters to .8*81, doing it for all parts of the grid until we see no potential improvement in the 29 observations in the validation set, then predicting the remaining .2*81.

This might be overkill, but it definitely keeps the procedure clean. Especially if we use a leaderboard set later.

Now, keep in mind this is only true for autoML (or any grid search). If we were to simply train a givel model, there would be no validation set. In other words, the training part would be N (110 in the case above), which would also be the number of cross-validation predictions as each cross validation would predict independent .2*N observations.

So, given everything we've seen so far, I think it's fair to run the entire experiment using a single seed, without specifying the leaderboard. Then, for the "interesting" results, we run multiple seeds (still without the leaderboard). The idea there is that we'll use Wendy's search as the test set eventually. But then, for the best results in the CV approach, we also run them through the leaderboard approach, just to we have some backup results.

What I might do is specify the validation myself, so I have more control of how much data goes into it, and also compute univariate filter only in training data.

Of course, we should also start plotting in the brain where all these good variables are.

```
for var in raw pca uni pca_uni uni_pca uni01; do
    for f in `/bin/ls -1 ~/data/baseline_prediction/cog*0924*gz \
        ~/data/baseline_prediction/dti*0921*gz
~/data/baseline_prediction/aparc*gz \
        ~/data/baseline_prediction/struct*gz
~/data/baseline_prediction/geno3*gz \
        ~/data/baseline_prediction/aparc*gz`;do
        for nn in nonew_ ''; do
            for g in ADHDonly_ ''; do
                for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope; do
                    echo "Rscript --vanilla
~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/data/baseline_prediction/models/${nn}${g}${t} 42" >>
swarm.automl_valFixed;
                done;
            done;
            for g in ADHDNOS_ ADHDNOS_group; do
                for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope; do
                    echo "Rscript --vanilla
~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/data/baseline_prediction/models/${nn}${g}${t} 42" >>
swarm.automl_valFixed;
```

```
                done;
            done;
        done;
        for y in random_HI_slope random_total_slope random_inatt_slope \
            group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${y}
~/data/baseline_prediction/models/${y} 42" >> swarm.automl_valFixed;
        done;
    done;
done;
sed -i -e "s/^/unset http_proxy; /g" swarm.automl_valFixed;
split -l 1000 swarm.automl_valFixed vf;
for f in `/bin/ls vf??`; do
    echo "ERROR" > swarm_wait
    while grep -q ERROR swarm_wait; do
        echo "Trying $f"
        swarm -f $f -g 40 -t 32 --time 4:00:00 --partition quick --logdir
trash_bin --job-name valFixed -m R --gres=lscratch:10 2> swarm_wait;
        if grep -q ERROR swarm_wait; then
            echo -e "\tError, sleeping..."
            sleep 10m;
        fi;
    done;
done
```

I have also been thinking that we should run the subgroups in the latent categories as well. My reasoning here is that the latent variables were obtained in the complete group, but we run it for subsamples of it (e.g. not everyone had DTI). So, if we are removing some people because of being NV, or late onset, we should still be able to use those labels under the same logic.

```
rm swarm.automl_valFixed;
function print_commands () {
    for nn in nonew_ ''; do
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/data/baseline_prediction/models/${nn}${g}${t} 42" >>
swarm.automl_valFixed;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/data/baseline_prediction/models/${nn}${g}${t} 42" >>
swarm.automl_valFixed;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/data/baseline_prediction/models/${nn}${g}${t} 42" >>
swarm.automl_valFixed;
        done;
    done;
}

for f in `/bin/ls -1 ~/data/baseline_prediction/cog*0924*gz \
    ~/data/baseline_prediction/dti*0921*gz
~/data/baseline_prediction/aparc*0918*gz \
    ~/data/baseline_prediction/struct*0919*gz`; do
    for var in raw pca uni pca_uni uni_pca uni01; do
        print_commands
    done;
    var=raw;
    for f in `/bin/ls -1 ~/data/baseline_prediction/geno3*0919*gz`; do
        print_commands
    done;
done;

sed -i -e "s/^/unset http_proxy; /g" swarm.automl_valFixed;
wc -l swarm.automl_valFixed;
split -l 1000 swarm.automl_valFixed vf;
for f in `/bin/ls vf??`; do
    echo "ERROR" > swarm_wait
    while grep -q ERROR swarm_wait; do
        echo "Trying $f"
        swarm -f $f -g 40 -t 32 --time 4:00:00 --partition quick --logdir
trash_valFixed --job-name valFixed -m R --gres=lscratch:10 2> swarm_wait;
        if grep -q ERROR swarm_wait; then
            echo -e "\tError, sleeping..."
            sleep 10m;
        fi;
    done;
done
```

2018-09-26 12:28:10

Note that all tests above were conducted removing all variables that had any NAs in them. Today I changed the scripts to keep NAs so we could run the social variables a bit better, and also to better deal in the future when combining datasets.

Not sure how keeping the NAs would affect the results, as I don't have a good quantification of how many of the single-domain datasets have NAs in them.

At this moment the cluster is swamped with all the stuff I'm running above. So, I'm trying to run the different tests with the socioeconomic variables locally:

```
function print_commands () {
    var=$1;
    f=$2;
    for nn in nonew_ ''; do
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/data/baseline_prediction/models/${nn}${g}${t} 42 | tee
social_${var}_${nn}${g}${t}.log" >> swarm.automl_allSocial;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/data/baseline_prediction/models/${nn}${g}${t} 42 | tee
social_${var}_${nn}${g}${t}.log" >> swarm.automl_allSocial;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
~/data/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
~/data/baseline_prediction/models/${nn}${g}${t} 42 | tee
social_${var}_${nn}${g}${t}.log" >> swarm.automl_allSocial;
        done;
    done;
}
f=~/data/baseline_prediction/social_09262018.RData.gz;
for var in raw pca uni pca_uni uni_pca uni01; do
    print_commands $var $f
done;
```

2018-09-27 10:51:48

Let's run some stuff under Philip's account...

```
rm swarm.automl_valFixed;
function print_commands () {
    for nn in nonew_ ''; do
```

```
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >>
swarm.automl_valFixed;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >>
swarm.automl_valFixed;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >>
swarm.automl_valFixed;
        done;
    done;
}

for f in `/bin/ls -1 /data/NCR_SBRB/baseline_prediction/cog*0924*gz \
    /data/NCR_SBRB/baseline_prediction/dti*0921*gz
/data/NCR_SBRB/baseline_prediction/aparc*0918*gz \
    /data/NCR_SBRB/baseline_prediction/struct*0919*gz`; do
    for var in raw pca uni pca_uni uni_pca uni01; do
        print_commands
    done;
    var=raw;
    for f in `/bin/ls -1
/data/NCR_SBRB/baseline_prediction/geno3*0919*gz`; do
        print_commands
    done;
done;

sed -i -e "s/^/unset http_proxy; /g" swarm.automl_valFixed;
wc -l swarm.automl_valFixed;
split -l 1000 swarm.automl_valFixed vf;
for f in `/bin/ls vf??`; do
    echo "ERROR" > swarm_wait
    while grep -q ERROR swarm_wait; do
        echo "Trying $f"
        swarm -f $f -g 40 -t 32 --time 4:00:00 --partition quick --logdir
trash_valFixed --job-name valFixed -m R --gres=lscratch:10 2> swarm_wait;
```

```
            if grep -q ERROR swarm_wait; then
                echo -e "\tError, sleeping..."
                sleep 10m;
            fi;
        done;
    done
```

Note that the tests I started running in Philip's account (starting on vfai split file) do have the NA variables still in there, because it checked the newer code. It won't affect any data domains that don't have NAs, though.

I'm also getting a bit uneasy with doing the univariate analysis in the cross-validation set. I know the results were still quite good in the leaderboard analysis, but I think it'd be more fair if we did no cross-validation, specified the training, validation, and test sets, and ran that hundreds of times, combining the leaderboard results across runs. Let's see how that goes (nocv_ code). But the issue there is that it's not a true implementation of the internal CV, which only predicts the held out data once. So, I'll have to implement the CV manually, and do the predictions as we go.

At this point, the nocv_uni01 code works, but we still need to combine the different models. Howe would we do that? In other words, what's the best model? We need to recreate the cross-validation leaderboard. But that's not possible because automl only exposes the leader, which is a model that has been trained on 4/5 of the data already. So, it's not fair to have it predict that same dataset...

So, we're back at where we started. We can use the train+valid split, doing the univariate test on the entire thing, and report the best model for further testing, including variable selection and testing using Wendy's set. We could potentially randomize it with different seeds, so that we're playing with different validation sets, and we then can assess the stability of the best model selected, as well as the features used.

The other options is to use train+valid+test, repeating it several times. That's what we did for the best results (DTI) so far, which gives us a nice test distribution, uncorrupted by the univariate test that we're using for feature selection. Just remember that in this case the leaderboard (i.e. best model) is based on the test data, not the CV held-out data.

2018-09-28 18:33:21

Maybe the best approach is to use the fixed seed approach to find the best model, then we just permute for different seeds that best model, without autoML. We'd do it to get the test estimate, but also the error estimate using nuisance variables?

2018-10-01 12:17:15

Collecting the main results:

```
echo "target,pheno,var,nfeat,model,metric,val,dummy" > auto_summary.csv;
for f in `ls trash_valFixed/*o`; do
    phen=`head -n 2 $f | tail -1 | awk '{FS=" "; print $6}' | cut -d"/" -f
6`;
    target=`head -n 2 $f | tail -1 | awk '{FS=" "; print $8}'`;
    var=`head -n 2 $f | tail -1 | awk '{FS=" "; print $5}' | cut -d"/" -f
4 | sed -e "s/\.R//g"`;
    model=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $2}' |
cut -d"_" -f 1`;
    acc=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $3}'`;
    metric=`grep -A 0 model_id $f | awk '{FS=" "; print $2}'`;
    if [[ $var == 'raw' ]] && [[ $phen == 'dti_ALL' ]]; then
        nfeat=36066;
    elif [[ $var == 'raw' ]] && [[ $phen = *'dti_'* ]]; then
        nfeat=12022;
    elif grep -q "Running model on" $f; then  # newer versions of the
script
        nfeat=`grep "Running model on" $f | awk '{FS=" "; print $5}'`;
    else # older versions were less verbose
        nfeat=`grep -e "26. *[1-9]" $f | grep "\[1\]" | tail -1 | awk '{
print $3 }'`;
    fi
    if grep -q "Class distribution" $f; then
        dummy=`grep -A 1 "Class distribution" $f | tail -1 | awk '{FS=" ";
{for (i=2; i<=NF; i++) printf $i ";"}}'`;
    else
        dummy=`grep -A 1 "MSE prediction mean" $f | tail -1 | awk '{FS="
"; print $2}'`;
    fi
    echo $target,$phen,$var,$nfeat,$model,$metric,$acc,$dummy >>
auto_summary.csv;
done
```

It turns out that there are too many errors in the output... I think I'll just re-run them individually, even though it might take some extra time. This way we know for sure we're running everything we can.

```
fname=swarm.automl_dti
rm $fname;
function print_commands () {
    for nn in nonew_ ''; do
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
        done;
    done;
}

for f in `/bin/ls -1 /data/NCR_SBRB/baseline_prediction/dti*0921*gz`; do
    for var in raw pca uni pca_uni uni_pca uni01; do
        print_commands
    done;
done;

sed -i -e "s/^/unset http_proxy; /g" $fname;
wc -l $fname;
split -l 1000 $fname dti;
for f in `/bin/ls dti??`; do
    echo "ERROR" > swarm_wait
    while grep -q ERROR swarm_wait; do
        echo "Trying $f"
        swarm -f $f -g 40 -t 32 --time 4:00:00 --partition quick --logdir
trash_dti --job-name valFixed -m R --gres=lscratch:10 2> swarm_wait;
        if grep -q ERROR swarm_wait; then
            echo -e "\tError, sleeping..."
            sleep 10m;
        fi;
    done;
done
```

Then, repeat it for the remaining phenotypes. Note that now I'm running all phenotypes in /data/NCR_SBRB, to make it easier to compile the results and make sure we're always using the same data source.

```
fname=swarm.automl_rsfmri
rm $fname;
function print_commands () {
    for nn in nonew_ ''; do
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
        done;
    done;
}

for f in `/bin/ls -1 /data/NCR_SBRB/baseline_prediction/aparc*0918*gz`; do
    for var in raw pca uni pca_uni uni_pca uni01; do
        print_commands
    done;
done;

sed -i -e "s/^/unset http_proxy; /g" $fname;
wc -l $fname;
split -l 1000 $fname rsfmri;
for f in `/bin/ls rsfmri??`; do
    echo "ERROR" > swarm_wait
    while grep -q ERROR swarm_wait; do
        echo "Trying $f"
        swarm -f $f -g 40 -t 32 --time 4:00:00 --partition quick --logdir
trash_rsfmri --job-name valFixed -m R --gres=lscratch:10 2> swarm_wait;
        if grep -q ERROR swarm_wait; then
            echo -e "\tError, sleeping..."
            sleep 10m;
        fi;
    done;
done
```

```
fname=swarm.automl_struct
rm $fname;
function print_commands () {
    for nn in nonew_ ''; do
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
        done;
    done;
}

for f in `/bin/ls -1 /data/NCR_SBRB/baseline_prediction/struct*0919*gz`;
do
    for var in raw pca uni pca_uni uni_pca uni01; do
        print_commands
    done;
done;

sed -i -e "s/^/unset http_proxy; /g" $fname;
wc -l $fname;
split -l 1000 $fname struct;
for f in `/bin/ls struct??`; do
    echo "ERROR" > swarm_wait
    while grep -q ERROR swarm_wait; do
        echo "Trying $f"
        swarm -f $f -g 40 -t 32 --time 4:00:00 --partition quick --logdir
trash_struct --job-name valFixed -m R --gres=lscratch:10 2> swarm_wait;
        if grep -q ERROR swarm_wait; then
            echo -e "\tError, sleeping..."
            sleep 10m;
        fi;
    done;
done
```

```
fname=swarm.automl_cog
rm $fname;
function print_commands () {
    for nn in nonew_ ''; do
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
        done;
    done;
}

for f in `/bin/ls -1 /data/NCR_SBRB/baseline_prediction/cog*0924*gz`; do
    for var in raw pca uni pca_uni uni_pca uni01; do
        print_commands
    done;
done;

sed -i -e "s/^/unset http_proxy; /g" $fname;
wc -l $fname;
split -l 1000 $fname cog;
for f in `/bin/ls cog??`; do
    echo "ERROR" > swarm_wait
    while grep -q ERROR swarm_wait; do
        echo "Trying $f"
        swarm -f $f -g 40 -t 32 --time 4:00:00 --partition quick --logdir
trash_cog --job-name valFixed -m R --gres=lscratch:10 2> swarm_wait;
        if grep -q ERROR swarm_wait; then
            echo -e "\tError, sleeping..."
            sleep 10m;
        fi;
    done;
done
```

```
fname=swarm.automl_geno
rm $fname;
function print_commands () {
    for nn in nonew_ ''; do
```

```
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
        done;
    done;
}

var=raw;
for f in `/bin/ls -1 /data/NCR_SBRB/baseline_prediction/geno3*0919*gz`; do
    print_commands
done

sed -i -e "s/^/unset http_proxy; /g" $fname;
wc -l $fname;
swarm -f $fname -g 40 -t 32 --time 4:00:00 --partition quick --logdir
trash_geno --job-name valFixed -m R --gres=lscratch:10;
```

2018-10-02 10:38:10

I'm splitting allSocial to run in the cluster, into 4 pieces, so I can run each in an interactive machine between Philip and I. Hopefully that will speed things up, as the desktop running is dying frequently. I've noticed lots of errors coming out of the socioeconomic variables and the different pipelines. It's unclear whether we should be running them just in raw as well. In any case, it might be worth considering a closer look at them, if any of the good results look promising.

2018-10-03 10:09:31

LEt's recode the social work just for the raw pipeline to better compare with something like the geno phenotype:

```
fname=swarm.automl_soc
rm $fname;
function print_commands () {
    for nn in nonew_ ''; do
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
        done;
    done;
}

var=raw;
f=/data/NCR_SBRB/baseline_prediction/social_09262018.RData.gz
print_commands

sed -i -e "s/^/unset http_proxy; /g" $fname;
wc -l $fname;
swarm -f $fname -g 40 -t 32 --time 4:00:00 --partition quick --logdir
trash_soc --job-name social -m R --gres=lscratch:10;
```

But because we split them into different trash directories, we need to make a small change to the compilation code:

```
echo "target,pheno,var,nfeat,model,metric,val,dummy" > auto_summary.csv;
old_phen=''
for f in `ls trash_*/*o`; do
    phen=`head -n 2 $f | tail -1 | awk '{FS=" "; print $6}' | cut -d"/" -f
5`;
    if [[ $phen != $old_phen ]]; then
```

```
            echo $phen;
            old_phen=$phen;
        fi;
        target=`head -n 2 $f | tail -1 | awk '{FS=" "; print $8}'`;
        var=`head -n 2 $f | tail -1 | awk '{FS=" "; print $5}' | cut -d"/" -f
    4 | sed -e "s/\.R//g"`;
        model=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $2}' |
    cut -d"_" -f 1`;
        acc=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $3}'`;
        metric=`grep -A 0 model_id $f | awk '{FS=" "; print $2}'`;
        if [[ $var == 'raw' ]] && [[ $phen == 'dti_ALL' ]]; then
            nfeat=36066;
        elif [[ $var == 'raw' ]] && [[ $phen = *'dti_'* ]]; then
            nfeat=12022;
        elif grep -q "Running model on" $f; then  # newer versions of the
    script
            nfeat=`grep "Running model on" $f | awk '{FS=" "; print $5}'`;
        else # older versions were less verbose
            nfeat=`grep -e "26. *[1-9]" $f | grep "\[1\]" | tail -1 | awk '{
    print $3 }'`;
        fi
        if grep -q "Class distribution" $f; then
            dummy=`grep -A 1 "Class distribution" $f | tail -1 | awk '{FS=" ";
    {for (i=2; i<=NF; i++) printf $i ";"}}'`;
        else
            dummy=`grep -A 1 "MSE prediction mean" $f | tail -1 | awk '{FS="
    "; print $2}'`;
        fi
        echo $target,$phen,$var,$nfeat,$model,$metric,$acc,$dummy >>
    auto_summary.csv;
    done
```

Looking at the file auto_Summary_10032018, using uni is almost always the best approach. For DTI, it's a competition between ALL and AD, but 223 is always on top. OK, so for DTI we'll run ALL 223.

For rsfmri, it looks like aparc2009s_trimmed does best. For structural, we'll stick with thickness for now, even though volume would be a close second.

For SNPs, it's hardly ever significant. We might as well just create a new one using genome-wide significance and call it the ay. But if anything, SNPs are better than PRS.

But keep in mind that these results come from a single seed. So, it's possible that when moving to a seed distribution, using a test set, things will fall apart, so we might need to try other options. For example, using the 272 DTI set, or combining the volume/thickness/area estimates to best mimic DTI.

Also, need a more careful look at the code running for socioeconomic, cognitive and clinical variables... lots of them seemed to die.

2018-10-04 10:15:44

Also, it looks like the tracts file doesn't have correct variable names? So, add it to the list of phenotypes to double-check.

cog_all was also not run, so we'll need to add that to the set of things to run, along with dti_tracts.

Given the small number of variables, I could potentially just run raw and uni for those 2 sets.

It also turns out that social variables ran fine... I only ran them using raw, and the results are abismal, but they did run. Same for clinics (only ran using raw), but the results there are better.

So, let's fix and re-run dti_tracts and cog_all.

```
job_name=missing
fname=swarm.automl_${job_name}
rm $fname;
function print_commands () {
    for nn in nonew_ ''; do
        for g in ADHDonly_ ''; do
            for t in diag_group2 OLS_inatt_slope OLS_HI_slope
OLS_total_slope \
                random_HI_slope random_total_slope random_inatt_slope \
                group_HI3 group_total3 group_INATT3; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        # diag_group2 doesn't apply to ADHD_NOS
        for g in ADHDNOS_ ADHDNOS_group; do
            for t in OLS_inatt_slope OLS_HI_slope OLS_total_slope \
            random_HI_slope random_total_slope random_inatt_slope; do
                echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
            done;
        done;
        g=ADHDNOS_;
        for t in group_HI3 group_total3 group_INATT3; do
            echo "Rscript --vanilla ~/research_code/automl/${var}.R $f
/data/NCR_SBRB/baseline_prediction/long_clin_0918.csv ${nn}${g}${t}
/data/NCR_SBRB/baseline_prediction/models/${nn}${g}${t} 42" >> $fname;
        done;
    done;
}

for f in /data/NCR_SBRB/baseline_prediction/cog_all_09242018.RData.gz \
    /data/NCR_SBRB/baseline_prediction/dti_tracts_n223_10042018.RData.gz \
    /data/NCR_SBRB/baseline_prediction/dti_tracts_n272_10042018.RData.gz;
do
    for var in raw uni; do
        print_commands
    done;
done;
```

```
sed -i -e "s/^/unset http_proxy; /g" $fname;
wc -l $fname;
swarm -f $fname -g 18 -t 24 --time 4:00:00 --partition quick --logdir
trash_${job_name} --job-name ${job_name} -m R --gres=lscratch:10;
```

I'm also running the ADHD-200 variables using raw. I'll likely have to make a raw_test as well, for those datasets with too few variables, but for consistency let's keep it as is for now.

```
job_name=adhd200
fname=swarm.automl_${job_name}
rm $fname;
var=raw;
f=/data/NCR_SBRB/baseline_prediction/adhd200_10042018.RData.gz
print_commands

sed -i -e "s/^/unset http_proxy; /g" $fname;
wc -l $fname;
swarm -f $fname -g 24 -t 19 --time 4:00:00 --partition quick --logdir
trash_${job_name} --job-name ${job_name} -m R --gres=lscratch:10;
```

2018-10-05 10:22:20

Let's compile the results for the missing datasets and also for adhd200:

```
echo "target,pheno,var,nfeat,model,metric,val,dummy" > auto_summary.csv;
old_phen=''
for dir in adhd200 missing; do
    echo $dir
    for f in `ls trash_${dir}/*o`; do
        phen=`head -n 2 $f | tail -1 | awk '{FS=" "; print $6}' | cut -
d"/" -f 5`;
        target=`head -n 2 $f | tail -1 | awk '{FS=" "; print $8}'`;
        var=`head -n 2 $f | tail -1 | awk '{FS=" "; print $5}' | cut -d"/"
-f 4 | sed -e "s/\.R//g"`;
        model=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $2}'
| cut -d"_" -f 1`;
        acc=`grep -A 1 model_id $f | tail -1 | awk '{FS=" "; print $3}'`;
        metric=`grep -A 0 model_id $f | awk '{FS=" "; print $2}'`;
        if [[ $var == 'raw' ]] && [[ $phen == 'dti_ALL' ]]; then
            nfeat=36066;
        elif [[ $var == 'raw' ]] && [[ $phen = *'dti_'* ]]; then
            nfeat=12022;
        elif grep -q "Running model on" $f; then  # newer versions of the
script
            nfeat=`grep "Running model on" $f | awk '{FS=" "; print $5}'`;
        else # older versions were less verbose
            nfeat=`grep -e "26. *[1-9]" $f | grep "\[1\]" | tail -1 | awk
```

```
'{ print $3 }'`;
        fi
        if grep -q "Class distribution" $f; then
            dummy=`grep -A 1 "Class distribution" $f | tail -1 | awk
'{FS=" "; {for (i=2; i<=NF; i++) printf $i ";"}}'`;
        else
            dummy=`grep -A 1 "MSE prediction mean" $f | tail -1 | awk
'{FS=" "; print $2}'`;
        fi
        echo $target,$phen,$var,$nfeat,$model,$metric,$acc,$dummy >>
auto_summary.csv;
    done;
done
```

2018-11-21 15:51:25

Before I move on to more descriptive results, I just want to make a plot of our best ML results for the 4 different groupwise comparisons. I'll also include the random data prediction next to it.

Note that I'm still playing with fMRI ICASSO on connectivity matrices and MELODIC, so there might be something there still. But the current results reflect what we can do best using rawCV and data transforms.

I'm doing this based only on AUC, and then I add the acc plot just for illustration. My strategy here was to look for all _summary.csv files for any specific domain, in that target, and pick the bets we had. Then, it's easy to find the matching seed < 0 for comparison.

nvVSadhd: DTI: DTI_PCA-kaiser (AD) or DTI_PCA-elbow (AD), both at .67 struct: thickness_subjScalePCA-kaiser at .62 rsFMRI: aparc_pcorr_kendal_trimmed with PCA-elbow is by far the best transform at .725

nvVSper: DTI: struct: rsFMRI:

nvVSrem: DTI: struct: rsFMRI:

perVSrem: DTI: .66 DTI_PCA-kaiser (AD) struct: .77 struct_subjScale:PCA-kaiser (volume) rsFMRI: .71 aparc_pcorr_pearson PCA-kaiser

So, let's create the CSV and make the barplots in R.

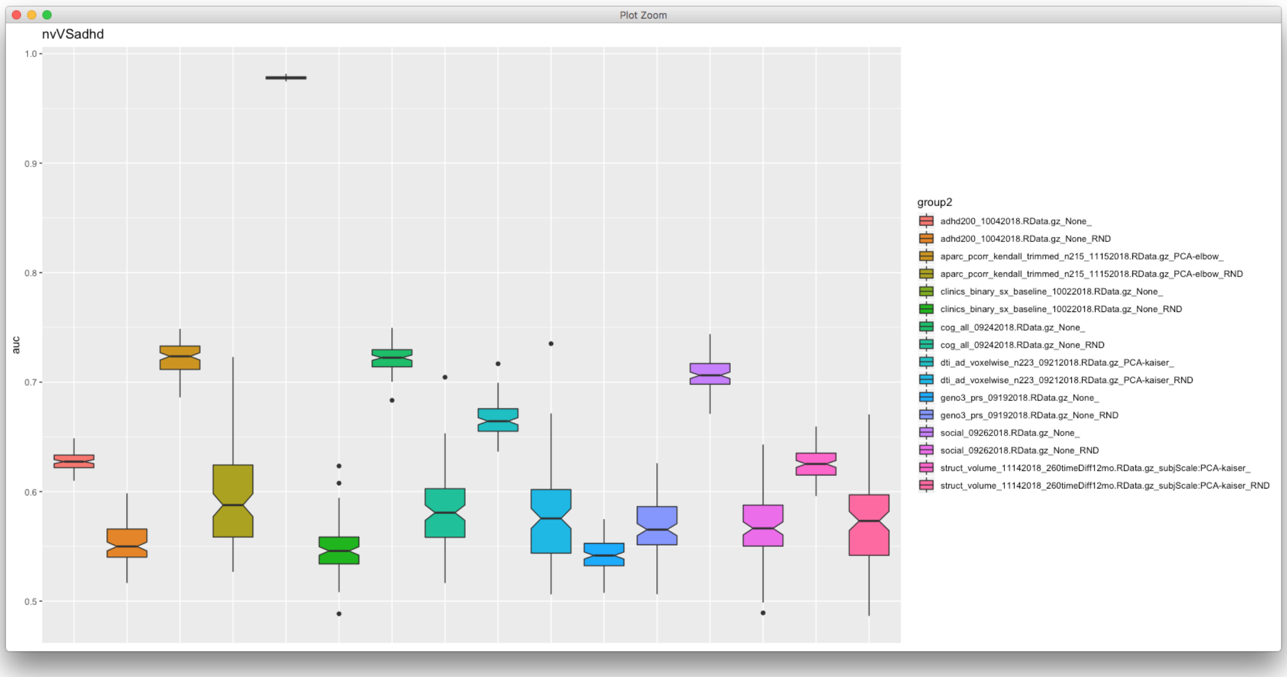
2018-11-27 14:29:09

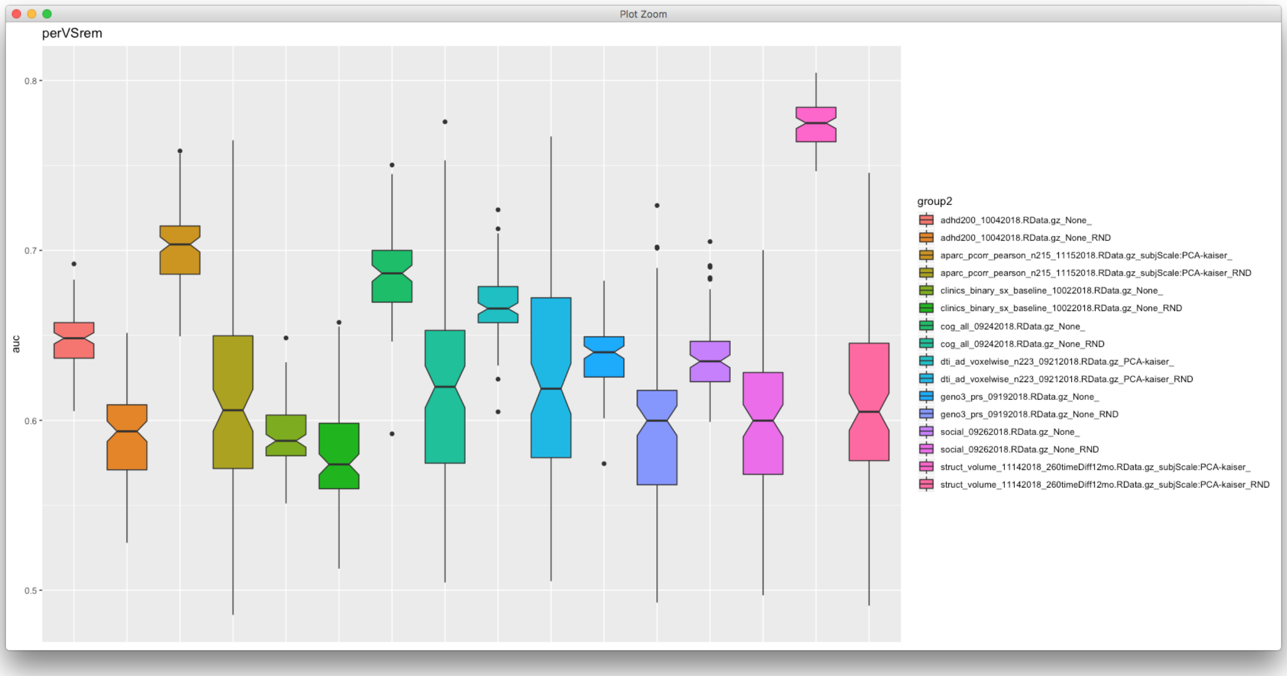
Can we make a massive CSV to plot these best results with the crappy domains results?

```
myfile=summary_for_philip_11272018.csv
head -n 1 dataTransformsFMRIPCA_summary.csv > $myfile;
grep ,PCA-elbow dataTransformsFMRIPCA_summary.csv | grep adhd | grep
aparc_pcorr_kendall_trimmed >> $myfile;
grep subjScale:PCA-kaiser dataTransformsFMRIPCA_summary.csv | grep
perVSrem | grep aparc_pcorr_pearson >> $myfile;
```

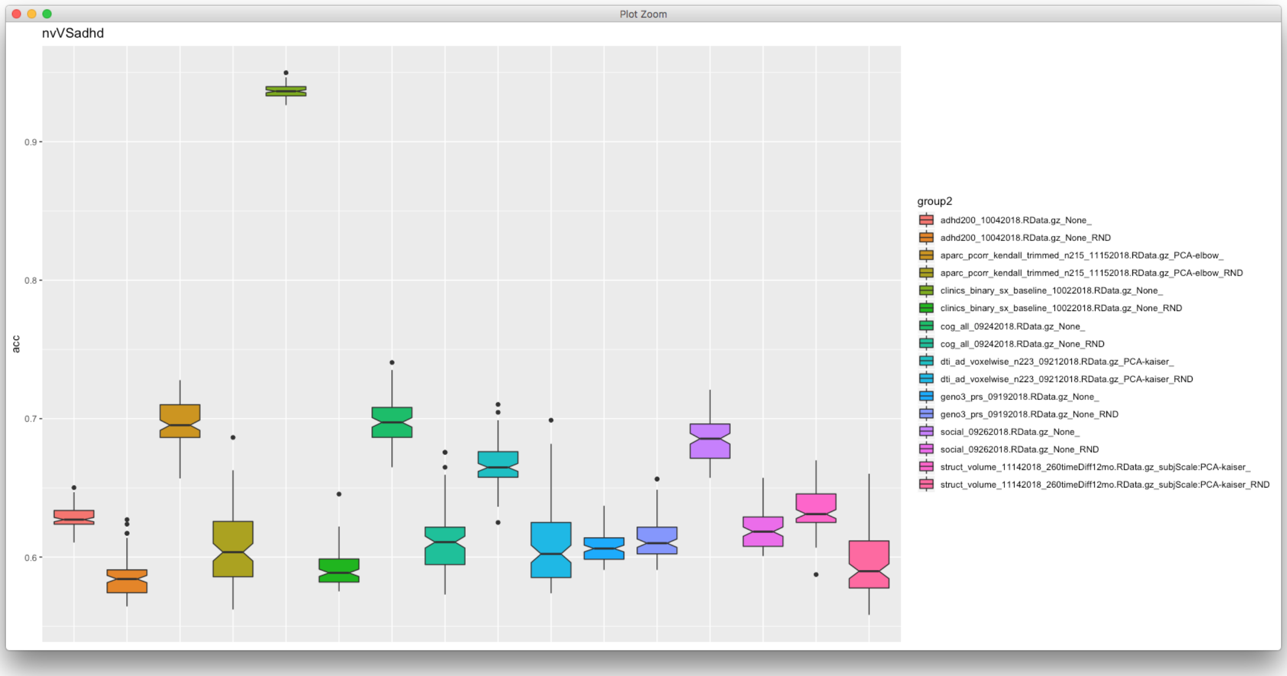
```
grep ,PCA-kaiser dataTransformsPCA_summary.csv | grep nvVSadhd | grep
/dti_ad >> $myfile;
grep ,PCA-kaiser dataTransformsPCA_summary.csv | grep perVSrem | grep
/dti_ad >> $myfile;
grep subjScale:PCA-kaiser dataTransformsPCA_summary.csv | grep nvVSadhd |
grep /struct_volume | grep -v dti >> $myfile;
grep subjScale:PCA-kaiser dataTransformsPCA_summary.csv | grep perVSrem |
grep /struct | grep -v dti >> $myfile;
grep -v target crappyDomains_summary.csv >> $myfile;
```

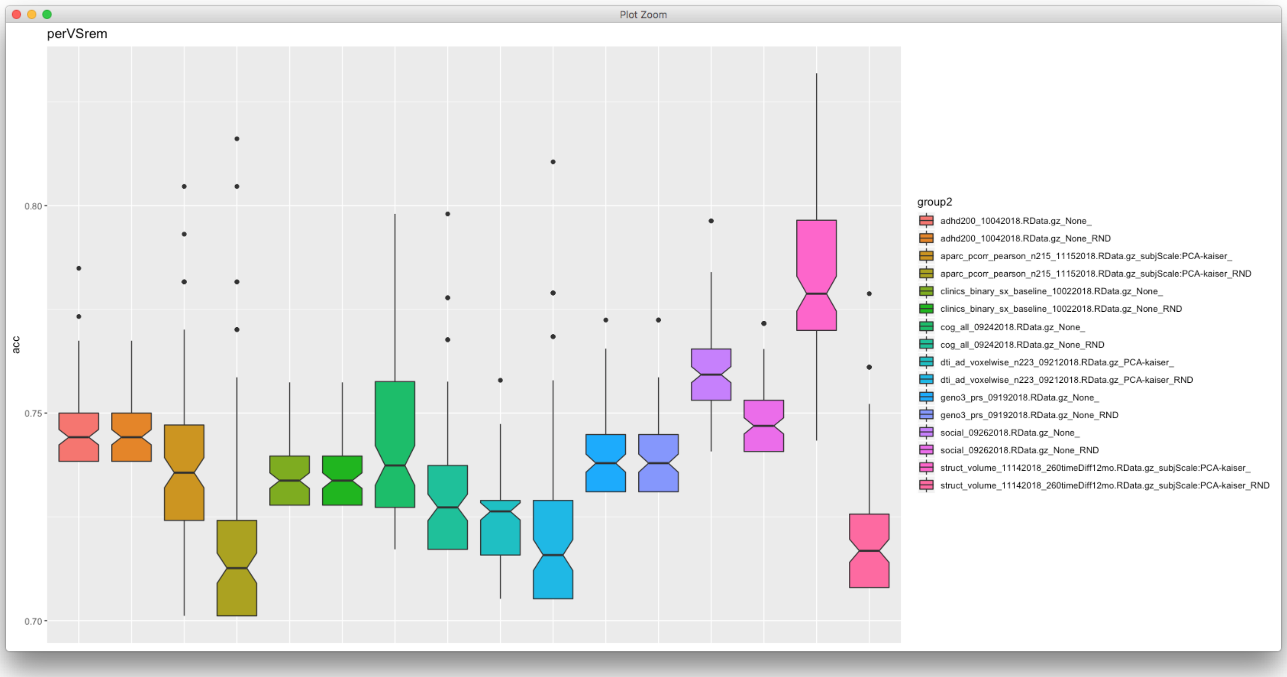
```
data = read.csv('~/tmp/summary_for_philip_11272018.csv')
data$pheno = gsub('/data/NCR_SBRB/baseline_prediction//', '', data$pheno)
data$group = ''
data[data$seed<0,]$group = 'RND'
data$group2 = sapply(1:nrow(data), function(x) { sprintf('%s_%s_%s',
data$pheno[x], data$var[x], data$group[x])} )
# then, for each target
idx = data$target=='nvVSadhd'
p1<-ggplot(data[idx,], aes(x=group2, y=auc, fill=group2))
print(p1+geom_boxplot(notch=T) + ggtitle(unique(data[idx,]$target)) +
theme(axis.title.x=element_blank(),
        axis.text.x=element_blank(),
        axis.ticks.x=element_blank())
```





And now we do accuracy as well:





For these boxplots (from ggplot's documentation): The lower and upper hinges correspond to the first and third quartiles (the 25th and 75th percentiles). The whiskers extend from the hinge to the largest/smallest value no further than 1.5*IQR from the hinge (where IQR is the inter-quartile range, or distance between the first and third quartiles). Data beyond the end of the whiskers are called "outlying" points and are plotted individually. The notches extend 1.58*IQR / sqrt(n). This gives a roughly 95% confidence interval for comparing medians. See McGill et al. (1978) for more details. If notches don't overlap, it suggests that the medians are significantly different.

So, here's a list of the things I'm waiting / working on:

- waiting on decoding results using ICASSO output for ROI connectivity (rsFMRI)
- waiting on computation of full and partial correlation matrices using movement residuals (rsFMRI)
- work on MELODIC outputs (rsFMRI)
- work on individual symptom changes
- work on simple descriptives for the different data domains
- work on diagnostic results as a "filter" for prognostic analysis
- work on our previous published results as a "filter" for prognostic analysis
- try domain voting approach instead of loose feature interactions
- work on age prediction model, using error magnitude as proxy for ADHD