README

Jessica Bonnie

10/06/2015

HumanCoreExome

Throughout this **README**, the calls to the code in this directory will be proceeded with `${hcefolder}` which refers to the path to this folder, currently set to the following location with this line:

```
hcefolder=/h2/t74/cphgdesk/share/cphg_OnengutLab/Jessica/Farewell/HumanCoreExome
```

Throughout this **README** I will assume that you have a project folder where you are working. If I need to create folders I will do that with reference to ${projectfolder}, which you will need to set according to where you are working.

QC of data run on HumanCoreExome Chips **requires** that you know which **version of the array** was used. There are hard-coded paths to the manifests included in multiple scripts. If your manifest files are not in the same location as mine, you will have *issues*, so make sure you check/change them.

Cleaning HCE data requires the following scripts:

- `check_export.sh`
- `makedata.sh`
- `pheno_inc.sh`
- `qc_gen.sh`
- `qcZ_gen.sh`
- `gender_gen.R`
- `rawrel_relat.R`
- `projpca_plus.R`
- `projpca_gen.R`
- `gender_basic.R`
- `EURprojpca_gen.R`
- `samplecallrateZ_gen.R`
- `snpqcZ_gen.R`

I have also included two scripts from other HCE projects which call all of the scripts described:

- `teddy`
- “

Here are the steps for cleaning data from HumanCoreExome arrays. I usually perform these script in a subdirectory of the `${projectfolder}` called raw. I have not done that here, but I really recommend that you do that.

Preparing the Raw Data

1. **check_export.sh** – this script checks the integrity of the genome studio export. (NOTE: This is EXACTLY the same step and script as for IMCHIP.) It is called like so:

```
cd ${projectfolder}
bash ${hcefolder}/check_export.sh ${DATAFILE} ${tracking} "Top Alleles"
```

where `${DATAFILE}` is the path to the raw export in the cphgcore, `${tracking}` is the path to the sample tracking sheet, and "Top Alleles" refers to the suffix of the columns from the export which will be used to create the plink files.

2. **makedata.sh** – this script produces a non-phenotyped raw plink file from the genome studio export. It is called like so:

```
bash ${hcefolder}/makedata.sh ${DATAFILE} ${plink_nopheno} "Top Alleles" > ${logfile}
```

where `${DATAFILE}` is the path to the raw export in the cphgcore, `${plink_nopheno}` is the name/path where the output PLINK file should be written, and "Top Alleles" refers to the suffix of the columns from the export which will be used to create the plink files. If you wish to record a log of the process you can designate a `${logfile}`, otherwise remove the redirection `>` to it in every command.

3. If there are duplicate individual IDs PLINK will complain when it is called during the last script. If that happens, PLINK's log file will provide a list of those IDs. The duplicates *must* be renamed before proceeding. Bad things will happen otherwise.

```
grep "Duplicate individual found:" ${plink_nopheno}.log | sed 's/Duplicate individual found: \[ //g' |\
     sed 's/ \]//g' | cut -d' ' -f1 | awk '{print $1,$1,$1"_2",$1"_2"}' > duplicate_update.txt

plink --update-ids duplicate_update.txt --bfile ${plink_nopheno} --out ${plink_nophenoB} --make-bed --n
```

4. Now the phenotype information must be added. This script will only add STATUS and SEX information to the plink file which it is given (make sure the column names are recognizable in the phenotype file.) The "covariable count" is a number indicating how many columns will be of interest for coloring graphs later (Cohort is the first one, and is standard; the next is Race. The titles are hardcoded into the script, but can be easily changed to match the title of the columns in the phenotype table.)

```
bash ${hcefolder}/pheno_inc.sh ${phenofile} ${plink_nophenoB} ${plink_raw} ${covariablecount} T >> ${lo
```

where `${phenofile}` is the phenotype file (with "Sex" and "Status" and "SampleID" columns – n.b. both columns must contain numeric values, `${plink_nophenoB}` is the input PLINK file, `${plink_raw}` is the name of the output PLINK file, `${covariablecount}` is the number of covariables in the file (recall that the list of titles is hardcoded, so if there is a covariable other than "Cohort", the column title must be put into the list), and the "T" indicates that there is status information to be incorporated. This script will produce a covariable file that will be used during graphing later on: `${plink_raw}.covariable.names`

5. Phenotype information that was not included through the **pheno_inc.sh** script can be added using PLINK. See example scripts for more detail.

QC - Pre zCall

Note This script expects that it will be called from a *tempdata* folder. The results will be written to a `QC` directory created by the script on the same level as tempdata. I would recommend putting *tempdata* in your `${projectfolder}` or a subfolder therin.

1. **qc_gen.sh** Basic sample QC is performed prior to zCall in order to better inform that process. The script to run this is called qc_gen.sh. Because this script was written by Wei-Min Chen for a specific project, it currently requires certain hard coded values be altered at the top of the script in order for the graphs to be titled and labeled appropriately. The QC will run just fine without changing these values, but the graphs will be labelled with the wrong array version and study title. This script calls upon R scripts also included in this folder. If they fail, check to makes sure that the “scripts” folder is properly assigned to this folder.

```
cd ${projectfolder}
mkdir tempdata
cd tempdata
bash ${hcefolder}/qc_gen.sh ${nickname} ${plink_raw} ${chipversion} ${overall_title} ${sampletoberemoved
```

where `${nickname}` is a short alias that will be used to name the files, `${plink_raw}` is the raw PLINK file with all phenotype information included (if you followed my advice, it is in `${projectfolder}/raw`, which will need to be included when it is passed to the script), `${chipversion}` refers to the version of the array, and `${overall_title}` refers to the project name (to go in the plot headers). The script also takes an *optional* argument `${sampletoberemoved}`. This is not usually used during preliminary QC. (How do you already know which samples need to be removed if you haven’t done any QC?)

2. Kinship Checking If family information was provided with the phenotypic information, then we need to check that the kinship between those samples matches what is expected for those relationships. In order to check kinship, you can use the following commands:

```
cd ${projectfolder}
mkdir kinship
cd kinship
plink --noweb --make-bed --exclude ${snptoberemoved} --remove ${sampletoberemoved} --bfile ${plinkraw} -

cd kinship

king -b ${nickname}_preZclean.bed --related --degree 2 --prefix ${preZclean}

#Now add a relationship column to kin and kin0

head -n1   ${preZclean}.kin0 | awk 'OFS="\t"{print $0,"Relationship"}'>   ${preZclean}kin0_relationship
awk 'NR > 1 && $8 > 0.4' ${preZclean}.kin0 | awk 'OFS="\t"{print $0, "Duplicate"}' >>  ${preZclean}kin0_
awk 'NR > 1 && $8 < 0.4 && $8 > 0.177 && $7<0.005' ${preZclean}.kin0| awk 'OFS="\t"{print $0, "PO"}' >>
awk 'NR > 1 && $8 < 0.4 && $8 > 0.177 && $7>0.005' ${preZclean}.kin0 | awk 'OFS="\t"{print $0, "FS"}' >
awk 'NR > 1 && $8 <=0.177 && $8 > 0.0884' ${preZclean}.kin0| awk 'OFS="\t"{print $0, "2nd"}' >>  ${preZ

head -n1 ${preZclean}.kin | awk 'OFS="\t"{print $0,"Relationship"}'>   ${preZclean}kin_relationship.txt
awk 'NR > 1 && $9 > 0.4' ${preZclean}.kin | awk 'OFS="\t"{print $0, "Duplicate"}' >>  ${preZclean}kin_r
awk 'NR > 1 && $9 < 0.4 && $9 > 0.177 && $8<0.005' ${preZclean}.kin| awk 'OFS="\t"{print $0, "PO"}' >>
awk 'NR > 1 && $9 < 0.4 && $9 > 0.177 && $8>0.005' ${preZclean}.kin | awk 'OFS="\t"{print $0, "FS"}' >>
awk 'NR > 1 && $9 <=0.177 && $9 > 0.0884' ${preZclean}.kin| awk 'OFS="\t"{print $0, "2nd"}' >>  ${preZc
awk 'NR > 1 && $9 <=0.0884' ${preZclean}.kin| awk 'OFS="\t"{print $0, "LessThan2nd"}' >>  ${preZclean}ki

# Create a file with all relationships with non-Zero values in the Error column
awk '$10>0' ${preZclean}kin_relationship.txt > ${preZclean}_KIN_nonZeroError.txt

#Draw a graph
R CMD BATCH "--args ${preZclean}.kin0 ${outgraph}" ${hcefolder}/rawrel_relat.R
ps2pdf ${outgraph}.pdf ${outgraph}.pdf
```

where `${sampletoberemoved}`, `${snptoberemoved}` and `${updatesex}` are the files in the *QC* directory produced during **qc_gen.sh**, `${plink_raw}` is the raw PLINK file with all phenotype information included, `${preZclean}` is the name that you want to give the KING output, and `${outgraph}` is the name that you want your graph to receive.

zCall

Note The zCall script expects that it will be called from a *tempdata* folder. The results will be written to a `zCalled` directory created by the script on the same level as tempdata. I would recommend putting *tempdata* in your `${projectfolder}` or a subfolder therin.

1. **check_export.sh** – this script checks the integrity of the genome studio export. (NOTE: This is EXACTLY the same script as earlier.) It is called like so:

```
cd ${projectfolder}
bash ${hcefolder}/check_export.sh ${ZDATAFILE} ${tracking} "GType"
```

```
where `${ZDATAFILE}` is the path to the raw export in the cphgcore expressly exported for zCall (see ins
```

2. **zCall_master.sh** – this script recalls the genotypes of rare SNPs with MAF <1% and Call Rate > 99%. It produces a new raw plink file, which it writes to a *zCalled* folder on the same level as the working directory.

```
cd ${projectfolder}
mkdir tempdata
cd tempdata
bash ${hcefolder}/zCall_master.sh ${ZDATAFILE} ${chipversion} ${sampletoberemoved} \
${plink_raw}  > ${logfile3}
```

where `${ZDATAFILE}` is the path to the raw export in the cphgcore expressly exported for zCall (see instructions to core in this folder),`${plink_raw}` is the raw PLINK file with all phenotype information included (if you followed my advice, it is in `${projectfolder}/raw`, which will need to be included when it is passed to the script), `${chipversion}` refers to the version of the array (in this format: "12v1.0", "12v1.1", etc.).

3. **zCalibrationLoop** – this script uses intermediate files from **zCall_master.sh** to calculate different thresholds for different Z Values to create a table to allow users to select a different Z value for the **zCall_master.sh** process, if it's merited. If it is, you will have to rerun **zCall_master.sh** after changing the hardcoded value (7) in Step II.2.

```
cd ${projectfolder}
bash ${hcefolder}/zCalibrationLoop.sh tempdata/zcallreport_QCed.txt tempdata/BETAS.txt
```

QC - Post zCall

Note This script expects that it will be called from a *tempdata* folder. The results will be written to a `QC_Z` directory created by the script on the same level as tempdata. I would recommend putting *tempdata* in your `${projectfolder}` or a subfolder therin.

1. **qcZ_gen.sh** – this script takes the output from **zCall_master.sh** and a list of samples including the results from the pre-zCall **qc_gen.sh** and any additional samples that may need to be removed (due to Mendelian Errors perhaps). It produces a new *snptoberemoved.txt*, new *sampletoberemoved.txt*, and a new *sexupdate.txt* which are to be applied to the zCall output. This script calls upon R scripts also included in this folder. If they fail, check to makes sure that the "scripts" folder is properly assigned to this folder.

```
cd ${projectfolder}
mkdir tempdata
cd tempdata
bash ${hcefolder}/qcZ_gen.sh ${nickname} ${zCalledraw} ${chipversion} ${overall_title} \
${sampletoberemoved} > ${logfile5}
```

where `${nickname}` is a short alias that will be used to name the files, `${zCalledraw}` is the PLINK file output from **zCall_master.sh** (it will be in the *zCalled/* folder where the final results are written), `${chipversion}` refers to the version of the array, `${overall_title}` refers to the project name (to go in the plot headers), and `${sampletoberemoved}` is a sample list that includes the output from **qc_gen.sh** as well as any other samples that were identified from removal during kinship analysis.

2. **projpca_plus.R** – in addition to the PCA projection graph that are created during QC, another PCA graph can be drawn.

```
R CMD BATCH "--args tempdata/${nickname}6apc.ped  ${overall_title} ${chipversion}\
${covfile} ${tag}" ${hcefolder}/projpca_plus.R
ps2pdf projpca_${tag}.ps projpca_${tag}.pdf
```

where `${nickname}` is a short alias that will be used to name the files, `${nickname}6apc` is a projection file created during the post-zCall QC, `${tag}` is a word to add to the title of the graph.

Prepare for Release

There has only been one instance of taking a study from A to Z for QC and then splitting it up and running a percohort QC. I wrote the scripts to split the files, run the perCohort QC, prepare zipped releases. . . . but they are not at all generalized for use with other projects, and I did not actually write the actual per-cohort analysis scripts. . . so, I'm only including the scripts that I wrote. . . . with names of cohorts redacted. Also there are a bunch of hard coded things here that I didn't change.

1. Kinship and Splitting If there was more than one array used during the project, there will need to be some inter-array kinship calculated. This is done in **split_cohorts.sh**.
2. **per_cohort.sh** – this script calls scripts that are not present in this folder which perform "per-cohort" QC. It calls graphing scripts which are described in more detail in the *ScatterPlots* folder.

```
cd tempdata
bash ${hcefolder}/per_cohort.sh ${chipversion}
```

where `${chipversion}` is in this format: "12-1.0","12-1.1", etc.

3. **release_zip.sh** – this script takes the QC_Final folders produced during **per_cohort.sh**, zips them up with a list of passwords.

```
cd tempdata
bash ${hcefolder}/release_zip.sh ${freezenum}
```

where `${freezenum}` refers to which dataFreeze the cohort is from in order to put it in the correct release folder.