# PSB Pipeline V2.2, 2019-02-25 update

**Use this documentation after README.md and performing setups in the Manifest.txt file**

## Quick Rampup

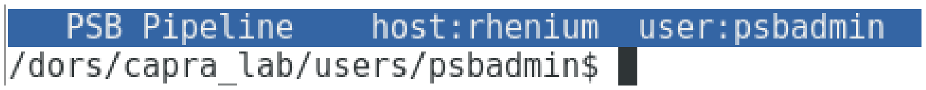
### Prep, Parse, Plan, Launch, Monitor, Report

0) Customize your shell environment (*UDN*,PATH, $PYTHONPATH, perl5) with the command:

```
source /dors/capra_lab/users/psbadmin/psb_prep.bash
```

(psb_prep.**csh** is alternately available for tcsh/csh shell users)

The modified environment settings will be shown, and your prompt changes to a distinctive blue:



```
   PSB Pipeline    host:rhenium  user:psbadmin
/dors/capra_lab/users/psbadmin$
```

Figure 1: Blue Prompt Indicates Pipeline prep'd

Note the convenient *UDNvariable, therootdirectoryforallourcases.Currently, forproductionwork, weset*UDN
to /dors/capra_lab/projects/psb_collab/UDN

> **ENSURE** that all pipeline commands are run with the blue prompt active. If you accidentally close your shell, repeat the psb_prep.bash source command upon opening a new shell.

1) Create a case directory from the UDN124356 case identifier:

```
mkdir $UDN/UDN124356
cd $UDN/UDN124356
```

2) Copy and rename the UDN supplied .xlsx file to $UDN/UDN124356/UDN123456.xlsx

3) Copy your user-specific pipeline yourUserId.config file to the *UDNdirectory, fromwhereitwillprovidedefaultconfigurationoverridesforthiscase, andfuturecasesyo*
Then, edit the contents of the file to reflect your email address.

4) Parse the complex UDN .xlsx file to pipeline-ready missense mutation list, and gene text files:

```
cd $UDN/UDN124356
parse_udn_report.py
```

Compare both the created .csv file of missense mutations, and the gene list, to the original .xlsx.

4b) User-supplied models: You may add one user-supplied model to any mutation line in the missense.csv file. This model will supplement Swiss and Modbase models, and PDB srtuctures, found in the pipeline's SQL database. Place your model .pdb file in the case directory, and append a comma, and the .pdb filename, at the right. ALSO, you must add the text ",user_model" to the right of the first header line of the missense.csv file

**Your user model must have amino acids numbered to match transcript numbering. It can contain only one chain at this time.**

For example, your file would be hand-edited to look something like this

```
,gene,refseq,mutation,unp,user_model
0,SCN11A,NM_014139,R60S,Q9UI33-1,test_SCN11A.pdb
1,ABCD,NM_012435,A59H,Q12345-2
```

5) Plan the work (jobs to be run for each mutation, based on available structures)

```
psb_plan.py
```

6) Launch the jobs

```
psb_launch.py
```

7) Monitor progress at intervals

```
psb_monitor.py
```

8) When Pathprox jobs have completed, run the final reports. Ddg results will be included as they are available

```
psb_rep.py
```

You may move the final .zip or .tar.gz files to extract to a laptop, or public website.

## Detailed Instructions

The Personal Structural Biology Pipeline reads a table of clinic-supplied mutations for a given patient (also known as project, or case) and launches a variety of analysis algorithms (pathprox, ddG, sequence) on each mutation.

A typical patient run involves ~50 mutations, and runs of ~500 indepent programs. Currently, the Pipeline runs on a "slurm" cluster, though this requirement could easily be circumvented in a future release, if requested.

The final outputs are per-mutation structural analysis reports.

Most software is housed in a write-protected area of /dors, writeable only by user "psbadmin"

It is convenient to reference the root directory for all UDN patient cases as $UDN:

bash shell example: `export UDN=/dors/capra_lab/projects/psb_collab/UDN`

### Configuration

The PSB Pipeline requires the compute environment (PATH, PYTHONPATH, PERL5LIB) to be unambiguously configured. Your first comamnd on the pipeline will be:

**`source /dors/capra_lab/users/psbadmin/psb_prep.bash`**

to prepare these settings. (psb_prep.csh serves the same purpose for tcsh preparation)

Once you "source" psb_prep.bash, you run the psb_*.py application programs like any other linux binary, with no regard to their specific location.

The PSB Pipeline integrates a wide variety of external genomic data, external structural models, MySQL database queries, PDBMap library support files, and other resources. The locations of these resources, as well as tuning parameters for slurm requests, of two configuration files: global and local

The *global* configuration file defaults to /dors/capra_lab/users/psbadmin/config/global.config (technically, it is ../../config/global.config, relative to the psb_*.py scripts directory).
However, that default can be over-ridden with the "-c" (or "–config") command line parameter.

Here is an example of the current global.config file:

## Global configuraton settings copied from the old v13.config file

```
[Genome_PDB_Mapper]
dbhost = chgr2.accre.vanderbilt.edu
dbname = pdbmap_v13
dbuser = psb_access
dbpass = psb-access
pdb_dir = /dors/capra_lab/data/rcsb
swiss_dir = /dors/capra_lab/data/swissmodel/SWISS-MODEL_Repository/
swiss_summary = /dors/capra_lab/data/swissmodel/SWISS-MODEL_Repository/INDEX_JSON
modbase2013_dir = /dors/capra_lab/data/modbase/ModBase_H_sapiens_2013_GRCh37.70.pep.all/models/
modbase2013_summary = /dors/capra_lab/data/modbase/ModBase_H_sapiens_2013_GRCh37.70.pep.all/H_
modbase2016_dir = /dors/capra_lab/data/modbase/H_sapiens_2016/Homo_sapiens_2016/model/
modbase2016_summary = /dors/capra_lab/data/modbase/H_sapiens_2016/Homo_sapiens_2016.summary.tx
idmapping = /dors/capra_lab/users/mothcw/mydata/idmapping/HUMAN_9606_idmapping_sprot.dat.gz
interpro_dir=/dors/capra_lab/data/interpro/
# idmapping = /dors/capra_lab/data/uniprot/idmapping/HUMAN_9606_idmapping_sprot.dat.gz
sec2prim = /dors/capra_lab/data/uniprot/idmapping/uniprot_sec2prim_ac.txt
sprot = /dors/capra_lab/data/uniprot/swissprot/uniprot_sprot_human.dat
pfam = pfam/pdb_pfam_mapping.txt
sifts = data/sifts/xml
create_new_db = False
vep = /dors/capra_lab/opt/ensembl-tools-release-87/scripts/variant_effect_predictor/variant_effect_predictor.
chimera_headless = /dors/capra_lab/users/mothcw/chimera/bin/chimera

# config_dict parameters specific to udn_pipeline.py
dssp_exe = /dors/capra_lab/projects/psb_collab/psb_pipeline/data/dssp/dssp_local.exe
chimera_headless = /dors/capra_lab/users/mothcw/chimera/bin/chimera
output_rootdir = /dors/capra_lab/projects/psb_collab
collaboration = UDN

[SlurmParametersAll]
account = capra_lab_csb
ntasks = 1

[SlurmParametersUDNSequence]
# These are for ddg and sequence analysis (udn_pipeline.py configs)
time = 12:00:00
mem = 10GB

[SlurmParametersUDNStructure]
# These are for ddg and sequence analysis (udn_pipeline.py configs)
time = 96:00:00
mem = 10GB

[SlurmParametersReport]
# These are used when the psb_rep.py program is called with –slurm to make
```

a .slurm run
# to simultaneously run all the reports Usually only done for BIG mutation sets.
time = 1:00:00
mem = 3GB

User-specific parameter overrides are essential for configuration of slurm emails, and to use test data sets. You should create a user-specific config file as shown below, and place it in your pipeline working directory. By default, pipeline modules look for this file in the parent directory of the case, typically $UDN. However, you may override this default with the -u command line option.

% cd $UDN …… Create the file as you like… % cat mothcw.config
[UserSpecific]
idmapping = /dors/capra_lab/users/mothcw/mydata/idmapping/HUMAN_9606_idmapping_sprot.dat.gz
[SlurmParametersAll]
mail-user=chris.moth@vanderbilt.edu
mail-type=end
[SlurmParametersPathProxCOSMIC]
time = 1-0
mem = 30GB
[SlurmParametersPathProxClinvar]
time = 1-0
mem = 30GB

These two config files default for every pipeline application. (As shown in psb_plan.py –help) You may find it helpful to place both the -c global.config and -u local.config command line parameters into a single short environment variable.

Finally, a third config file, defaulting to the casename.config (overrideable with -g) is searched in the case directory. Ths file is typically unnecessary, but it is extremely convenient for when a case should be procssed with variant lists other than the default exac/clinvar/COSMIC

Here is an example case-config file showing three changes from the defaults: * neutral variants for pathrpox are switched from default exac in the sql database to gnomad in the sql database * disease1 variants for pathprox reports are changed from clinvars in the sql database, to a custom list in a .txt file.
Disease2 variants are switched from the COSMIC cancer set to tcga in the sql database

$ cd $UDN/UDN$123456 cat UDN123456.config
[PathProx]
neutral_variant_short_description=gnomad
neutral_variant_sql_label=gnomad
disease1_variant_filename=P10636-1_13clinvars.txt
disease1_variant_short_description=13clinvars
disease1_variant_sql_label=13clinvars

disease2_variant_short_description=TCGA
disease2_variant_sql_label=tcga

## Preparation of input

### Optional step 0: Run parse_udn_report.py Parse the UDN-supplied .xlsx (Excel) file to create a pipeline-ready .csv file

Each case from the Undiagnosed Disease Network (UDN) is identified by a six digit number. Prepend this number with a string like "UDN" or "TEST" and copy (and rename) the UDN-supplied .xlsx file to your $UDNdirectoryas$UDN/UDN123456/UDN123456.xlsx or $UDN/Test123456/Test123456.xlsx.

Parse the complex xlsx file to csv with:

```
cd ~/psbwork
parse_udn_report.py -u mothcw.config UDN124356
```

Because of the config file parameters, parse_udn_report.py will locate UDN123456.xlsx as $UDN/UDN123456/UDN123456.xlsx.

parse_udn_report.py will create a final "UDN124356_missense.csv" output file name (also UDN123456_genes.json and UDN124356_genes.txt for other analyses)

The Excel .xlsx file will not be used again for any other pipeline processing.

The format of the .csv file is precise, but simple. It is typically "easy" to create or edit these files by hand if a source .xlsx is not available, or is badly mis-formatted.

Each row contains an index value (ignored), a gene name, a refseq identifier, a transcript-referenced amino acid mutation, and a uniprot ID for the refseq ID (likely optional - need to check).

```
/capra_lab/projects/psb_collab/UDN/Test501234% cat Test501234.csv
,gene,refseq,mutation,unp
0,KCNA2,NM_001204269,R300C,P16389-2
1,FRAS1,NM_025074,H3285Y,Q86XX4-2
2,FRAS1,NM_025074,P214L,Q86XX4-2
3,AP3B2,NM_004644.4,E465K,Q13367-1
4,ATP6V0A1,NM_001130020.1,R741Q,Q93050-3
5,RAPGEF6,NM_016340,N1075S,Q8TEU7-1
6,ALG11,NM_001004127,Q213P,Q2TAA5
```

## Pipeline step 1: Create a work plan from the csv file

**(Don't stress over remembering command line arguments. Use the standard '-h' option for a helpful reminder of the default inputs)**

Be sure to specify the global and local config files, as well as your job name. Your command sequence will look like:

% cd ~/psbwork $ psb_plan.py

Here is what I see on my system with the above command:

```
140 ~/psbwork % psb_plan.py Test123456 -u mothcw.config
psb_plan.py: Pipeline execution plan generator.  -h for detailed help
Collaboration-wide  psb_plan log file is /dors/capra_lab/projects/psb_collab/UDN/Test123456/
Loading swiss model JSON metadata from /dors/capra_lab/data/swissmodel/SWISS-MODEL_Reposito
Loading idmapping file from /dors/capra_lab/mothcw/mydata/idmapping/HUMAN_9606_idmapping_sp
Retrieving project mutations from /dors/capra_lab/projects/psb_collab/UDN/Test123456/Test12
Work for 2 mutations will be planned
Planning  11,SPRY3,NM_001304990,R242C,O43610
    2 structures retained     0 dropped.    6 jobs will run.  See: $UDN/Test123456/SPRY3_NM_
Planning  16,TPRN,NM_001128228,P39L,Q4KMQ1-1
    3 structures retained     2 dropped.    8 jobs will run.  See: $UDN/Test123456/TPRN_NM_
```

### Logging

psb_plan.py creates a master psb_plan.log file for the entire run, as well as an additional log file for each mutation that echoes the running log information. Usually, these logged details are too dense for screen display. Occasionally, they can be very helpful to sort through problems.

**To add log entries to your screen display, additionally include "–verbose" when running any of the psb pipeline applications. There is an additional –debug option that could include more messages. By default, only the less common WARNING, ERROR, FATAL, EXCEPTION, and CRITICAL log messages are displayed to the screen.**

Typically, you will not want to inspect fine details of the workplan.csv files that are generated for each mutation. The locations of these files are well-documented in the .log files that are listed in the above output. You can review all of the log file, or just "grep" for what interests you:

```
git hub markdown force line break % grep "workplan.csv" /dors/capra_lab/projects/psb_collab/
14:34:22 INFO [ psb_plan.py:730] Workplan written to /dors/capra_lab/projects/psb_collab/UDN
14:34:22 INFO [ psb_plan.py:730] Workplan written to /dors/capra_lab/projects/psb_collab/UDN
```

The tab-delimited workplan.csv files are easily reviewed in libreoffice or excel. The first row is a header with column names. The second and following list specific details about each job. Depending on the available structures, a workplan.csv file could have 1 job or 100. The average seems to be just under 10. For purpose of explanation, I have transposed the first two rows of a workplan.csv file (row 1 is shown as the left column below):

```
uniquekey   SPRY3_NM_001304990_R242C_ENSP00000302978_1_A_PathProxCOSMIC
chain       A
command     pathprox2.py
config      /dors/capra_lab/users/psbadmin/config/global.config
cwd     /dors/capra_lab/projects/psb_collab/psb_pipeline/bin
flavor      PathProxCOSMIC
gene        SPRY3
method      MTALL
mutation    R242C
options     -c /dors/capra_lab/users/psbadmin/config/global.config -u mothcw.config ENSP000
outdir      /dors/capra_lab/projects/psb_collab/UDN/Test123456/SPRY3_NM_001304990_R242C/ENS
pdbid       ENSP00000302978_1
project     Test123456
refseq      NM_001304990
unp     O43610
userconfig  mothcw.config
```

**Structure Reports**

Another important output from psb_plan.py is the . . . structure_report.csv and . . . dropped_models.csv.
(**Need to change names of these files**) These files, in combination with the details in the log file itself, document the consideration of all candidate 3D structures that are incorporated into (or dropped from) the workplan. Typical reasons for dropping a structure include the availability of higher resoltion .pdb files, higher sequence identity models, or duplication of models between swiss, modbase16, and modbase13, etc.

## Pipeline step 2: Launch jobs

Login to your slurm cluster head node, and launch all the jobs with

```
% cd ~/psbwork
% psb_launch UDN123456 -u mothcw.config
```

For each of the jobs in the workplan.csv files, a slurm file will be created, and submitted with the "sbatch" command

Here is a sample run:

```
~/psbwork$ psb_launch.py TestJED -u mothcw.config --relaunch
psb_launch.py: Pipeline launcher.  Run after psb_plan.py.   -h for detailed help
Retrieving project mutations from /dors/capra_lab/projects/psb_collab/UDN/TestJED/TestJED.cs
Launching all jobs for 2 mutations
Launching SPRY3      NM_001304990 R242C
       24753955:SPRY3_NM_001304990_R242C_ENSP00000302978_1_A_PathProxCOSMIC
       24753956:SPRY3_NM_001304990_R242C_ENSP00000302978_1_A_PathProxClinvar
       24753957:SPRY3_NM_001304990_R242C_ENSP00000302978_2_A_PathProxCOSMIC
       24753958:SPRY3_NM_001304990_R242C_ENSP00000302978_2_A_PathProxClinvar
       24753959:SPRY3_NM_001304990_R242C_ENSP00000302978_2_A_ddG
       24753960:SPRY3_NM_001304990_R242C_SequenceAnnotation
Recording all jobids to /dors/capra_lab/projects/psb_collab/UDN/TestJED/SPRY3_NM_001304990_F
Launching TPRN       NM_001128228 P39L
       24753961:TPRN_NM_001128228_P39L_ENSP00000387100.4_1_A_PathProxCOSMIC
       24753962:TPRN_NM_001128228_P39L_ENSP00000387100.4_1_A_PathProxClinvar
       24753963:TPRN_NM_001128228_P39L_ENSP00000387100_1_A_PathProxCOSMIC
       24753964:TPRN_NM_001128228_P39L_ENSP00000387100_1_A_PathProxClinvar
       24753965:TPRN_NM_001128228_P39L_ENSP00000387100_1_A_ddG
       24753966:TPRN_NM_001128228_P39L_ENSP00000387100_2_A_PathProxCOSMIC
       24753967:TPRN_NM_001128228_P39L_ENSP00000387100_2_A_PathProxClinvar
       24753968:TPRN_NM_001128228_P39L_SequenceAnnotation
Recording all jobids to /dors/capra_lab/projects/psb_collab/UDN/TestJED/TPRN_NM_001128228_P3
```

In the above output, the slurm job ids are shown to the left of the job names (which are simply the 'uniquekey' components of the original work plan)

## Pipeline step 3: Monitor jobs until completion

Every job has its own "outdir" where its results are stored. Under that directory is a 'status' directory which is cleared by psb_launch.py. This directory contains up to 3 files that are updated as each job runs:

- status/complete The presence of this empty file informs the monitor that the job has exited with status code 0, indicating that it completed satisfactorily, and that its output is ready for processing
  OR
- status/FAILED The presence of this file tells the monitor that the job has exited with "code 1" which indicates
  ALSO
- status/info The contents of the file tell the monitor how the job is doing. It is set to "Submitted" on initial submit by psb_launch.py
- status/progress Each time the info file is updated, the progress file is updated with the line number being execited in the .py file

As soon as the monitor sees the "complete" or "FAILED" marker files, the workstatus.csv row for the job is updated with status/info, and no further checks are performed.

(In previous version, psb_monitor.py would additionally inspect the output of "scontrol show job jobid" to help fill in the workstatus.csv file with the most complete information possible. However, this had dramatically slow runtime performance, and that section of code was commented out)

IMPORTANT: Final recording of Exit Code is done from the definitive presence of the status/complete or status/FAILED flags. It is not necessary for slurm to "remember" a job in order to have its exit status recorded

Here is an example of a psb_monitor.py run on my system

```
$ cd ~/psbwork
~/psbwork $ psb_monitor.py TestJED -u mothcw.config
./psb_monitor.py: Pipeline monitor for launched jobs.  -h for detailed help.
Retrieving project mutations from /dors/capra_lab/projects/psb_collab/UDN/TestJED/TestJED.cs
Monitoring all jobs for 2 mutations
Monitoring SPRY3      NM_001304990 R242C
Recording all updates to /dors/capra_lab/projects/psb_collab/UDN/TestJED/SPRY3_NM_001304990_
4 of 6 jobs still incomplete:
          Jobid:Flavor                Info
       24753955:SPRY3_NM_001304990_R242C_ENSP00000302978_1_A_PathProxCOSMIC    Submitted
       24753956:SPRY3_NM_001304990_R242C_ENSP00000302978_1_A_PathProxClinvar    Submitted
       24753957:SPRY3_NM_001304990_R242C_ENSP00000302978_2_A_PathProxCOSMIC    Submitted
       24753958:SPRY3_NM_001304990_R242C_ENSP00000302978_2_A_PathProxClinvar    Submitted
Monitoring TPRN       NM_001128228 P39L
Recording all updates to /dors/capra_lab/projects/psb_collab/UDN/TestJED/TPRN_NM_001128228_P
6 of 8 jobs still incomplete:
          Jobid:Flavor                Info
       24753961:TPRN_NM_001128228_P39L_ENSP00000387100.4_1_A_PathProxCOSMIC    Submitted
       24753962:TPRN_NM_001128228_P39L_ENSP00000387100.4_1_A_PathProxClinvar    Submitted
       24753963:TPRN_NM_001128228_P39L_ENSP00000387100_1_A_PathProxCOSMIC    Submitted
       24753964:TPRN_NM_001128228_P39L_ENSP00000387100_1_A_PathProxClinvar    Submitted
       24753966:TPRN_NM_001128228_P39L_ENSP00000387100_2_A_PathProxCOSMIC    Submitted
       24753967:TPRN_NM_001128228_P39L_ENSP00000387100_2_A_PathProxClinvar    Submitted
```

**Reminder to document J'ns method for repeatedly calling something until it is done. We need this**

You are welcome to independently inspect your slurm job queue with slurm commands. However, psb_monitor.py brings the advantage of updating your workstatus.csv files. It also reports much more specifically on the nature of failed jobs.

Because the job names are quite long, the standard formatting of the slurm squeue command can be unsatisfactory. You may find an alias with full-formatting to be helpful. I use this to review all my running, pending, and recently exited jobs on our cluster:

alias sq='squeue -o "%.9i %65j %.2T %.9M %.9l %R" -u mothcw'

Full documentation of the slurm squeue command is available here

## Pipeline step 4: Running reports

When jobs are complete, you may generate reports.

```
% cd $UDN/UDN123456
% psb_rep.py
4 of 4  ./psb_rep.py Pipeline report generator.  -h for detailed help
Retrieving project mutations from /dors/capra_lab/projects/psb_collab/UDN/TestJED/TestJED.cs
Reporting on all 2 project TestJED mutations
Reporting on SPRY3      NM_001304990 R242C
Generating html and pdf final reports for 2 of 6 complete jobs:
```

A complete portable website fileset is created, in both .zip and .tar.gz file formats.

### Running a report for only one mutation

A nice thing about all the psb_*.py programs is that you can somewhat bypass the *entire* list of mutations, and run a report (or launch or monitor) just one mutation.

Once initialized, psb_rep.py can take a few minutes per mutation to report. For long mutation lists, this can be tedious, and you may want to parallelize the run. Simply add the –slurm option to do this as in:

```
psb_rep.py TestJED -u mothcw.config --slurm
```

You must manually submit the slurm file to the scheduler with the sbatch command. It is listed at the end:

> ~/psbwork $ psb_rep.py UDN525217 -u mothcw.config -c ../config/v13.config –slurm 4 of 4 ./psb_rep.py Pipeline report generator. -h for detailed help Retrieving project mutations from /dors/capra_lab/users/mothcw/UDNtests/UDN525217/UDN525217.csv Slurm script to run 13 reports for UDN525217 is /dors/capra_lab/users/mothcw/UDNtests/UDN525217/sl Generating slurm entry for DENND5B NM_144973 D849E Generating slurm entry for GOLGA6L2 NM_001304388 G567E Generating slurm entry for TMEM37 NM_183240 T70P Generating slurm entry for CLUH NM_015229 P227L Generating

> slurm entry for FRMD4A NM_018027 T286M Generating slurm entry for MED13L NM_015335 M323I Generating slurm entry for RAI1 NM_030665 I898V Generating slurm entry for PCDHGB4 NM_003736 P15L Generating slurm entry for TPRA1 NM_001136053 V196A Generating slurm entry for ATP2C2 NM_014861 S81W Generating slurm entry for C3orf62 NM_198562 G205S Generating slurm entry for C3orf62 NM_198562 N142D Generating slurm entry for ARHGAP6 NM_013427 L75F Created slurm script to launch all psb_rep.py processes for this case: /dors/capra_lab/users/mothcw/UDNtests/UDN525217/slurm/psb_reps.slurm

The -h option to psb_rep.py reminds you that you can also run or rerun a report for only one of the mutations.

$ psb_rep.py -u mothcw.config $UDN/UDN525217/ARHGAP6_NM_013427_L75F/ARHGAP6_NM_013427_L75F_st$