1. First step is to do some prep work on our dependent variable, second contract percent of cap. The tables where that data lived treated injured players strangely, so these steps help address that so we can have a value for injured players.
   1. This will likely become a moot point from now on since we are changing the way to calculate that percent of cap variable and how it will be stored.
2. This step does some cleanup work on the other dependent variable we used to use, Pro grades.
   1. This step is irrelevant now because we don’t project to pro grades anymore.
3. Next we add bane player id to our season position algorithm tables. This makes it easier to join to the measurables data later on since those use bane player id, not nfl player id like the position tables do.
   1. We added this step because we considered trying to model “Slot WR” and “Slot Corners” separately from other WRs and CBs but there haven’t been enough of them in the past for it to really work yet.
4. We need to get each player’s Draft Board Position because that is what position we model them for. Since players can move around on the Draft Board throughout the process, we order the different board types so that we can pull their Draft Board position from as late in the process as possible.
5. This is a series of steps where we try to find any extreme outlier 40 times and then throw them out. We like to use electronic times as much as possible, but for some reason even the electronic times are “hand-started” so sometimes one can be way off.
6. Create a master measurables table. This is the table that everything following will work off of. There are a few where clauses to pull out extreme outliers in some of the drills.
7. This gets us an average age to plug in for players whose birthdate we don’t have.
   1. This step is irrelevant now because the age part of the model lives outside of the measurables model.
8. Pivot the non-run measurables so that you have one row for every player for every workout.
   1. Basically you are turning a long, skinny table into a short, wide table.
9. Pivot the run measurables so that you have one row for every player for every workout.
   1. You do the non run ones separately because you always want the HIGHEST value for these measurements, but you want the LOWEST value for runs.
10. This is a step to help handle workouts where a player doesn’t run. You need this because workout priority on runs is what you use to fill in priority for other measurables.
    1. Like if a player runs at Indy, his workout priority is for electronic time, but you don’t have electronic height and weight. So you put the electronic priority on the whole row.
11. Here you join the run measurables to the non run measurables so you have one row for every player workout with all of his measurables for that workout.
    1. Basically what we did was split runs and non runs during the pivot so we could order them separately, then rejoin them together after they were ordered. This may be an overly complicated way of doing it, but it works for now.
12. Get a player’s highest priority height to fill in for where height is missing.
    1. We don’t expect height to change much workout to workout so making sure height comes from the same workout isn’t as important as the others.
13. This step creates a table to feed in to R so you can generate “expected measurables” for every player.
    1. We do this so we can account for players running at different sizes. For example, running a 40 at 4.30 is more impressive at 230 than at 180. So we look at times in a “vs. expected” lens.
14. Run the R program to generate expected measurables
    1. Github>/corey-ravens/draft\_model/expected\_measurables\_score\_data
    2. This program does the expected measurables AND the adjusted measurables.
15. Join the outputs from the R program to the input table.
    1. This way you can subtract actual time from expected time. And can adjust the times for players who don’t run at Indy.
    2. This brings up another controversial part of the model since we make adjustments for non-electronic times (usually count them as slower than they are timed, based on a model). Joe doesn’t like that we do this. It’s up for discussion.
16. Unpivot the measurables. This way it is easier to do the ordering and sorting to choose which measurable to use in the player’s final measurables row.
    1. The earlier pivot was so you could keep the same workout together, now you unpivot so you can order by best measurables regardless of workout and then re-pivot back to one row in the end.
17. Order the measurables by adjusted priority and then best time.
    1. You split up runs and measures again because for runs you want LOWEST, for measures you want HIGHEST.
18. Append the measures to the runs so they are all in one table again.
19. Pivot the previous table to you now have one row for every player with the measurables you use for modeling.
20. Join the measurables for modeling table from above step to the dependent variables and identifying data.
    1. There are some players whose data came in late or just didn’t get matched up or who you manually filled in. They are added in individually after step 20.
21. Run the R program to build the measurables models
    1. Github>/corey-ravens/draft\_model/ measurables\_build\_model
    2. We used this program to build a separate model for every position.
22. Run the R program to score the measurables data
    1. Github>/corey-ravens/draft\_model/measurables\_score\_data
    2. Generate measurables scores for the players. Sometimes we run players through multiple positions, so someone can get an OB and DE score.
23. Run the R program to generate measurables similarity scores
    1. Github>/corey-ravens/draft\_model/measurables\_similarity\_scores
    2. Finds the difference between the target player and every other player in all of the variables that the model deemed important for that position, multiply the differences by the coefficients corresponding to those variables, and sum the scores.