Missense Atlas Notes:

2 tane Route var. HomePage, search icin; ProteinPage, belirtilen proteini göstermek icin

ÖNEMLI !! :

Su an backend'i görebilmek icin VPN ile sabancı networkunden bağlanıp sonra API'nın ana sayfasına gidip https://10.3.2.13:8080/ , güveniyorum demek gerekiyor, yoksa browser SSL sertifikasını dogrulayamadığından, API'yı reddediyor.

Overview:

Config.js dosyasından gelen constant'lar => hem styling ile alakalı, heatmap Renkleri ve sayıları, boyutu boşlukların boyutu vs. Metadata nin gösterildiği kısımın renkleri vs. Ayrıca yeni tool eklemek gerekince sadece buraya ekleme kolaylığı sağlıyor. (all_predictions_tools_array'e eklenmeli, çünkü onu export ediyorum, tool'ları bireysel olarak export etmek yerine).

SearchBox Component:

Aranılacak Protein sekansı, MD5Sum, Fasta Formatında Sekans, Uniprot Protein ID, ya da Uniprot Gene ID ile aranabiliyor. Fasta Format secildiyse "helper_protein_sequence_trimmer" fonksiyonu ile userin girmek istediği sekansı alıyorum, ve MD5Sum'a çevirip öyle aratıyorum. MD5Sum girdiyse user direkt MD5Sum ile aratıyorum. Uniprot protein ID ve Gene ID icin direkt API var, büyük harfler ile yolluyorum user inputunu trim'leyip.

Aratma seçeneği değiştirince, bir önceki input field'a yazdığı değer kaybolmasın diye, ayrı variable'lar ile tutuyorum user inputu.

Protein Page:

Proteinin annotation tool'larındaki skorlarını her tool icin backend'den fetchliyorum, metadata kısmını ise backend'den fetchlemek yerine (?q=1 opsiyonuyla) direkt ebi.ac.uk sitesinden çekiyorum, cunku backend de aynı siteden fetchleyip front ende yolluyordu.

Metadata icinden sadece insan protein'lerine denk gelen metadata'ları kullanıyorum (findMetadataHumanAccAndIndices fonksiyonu yardımıyla).

Metadata'yı parselayıp, proteinNameJSX, geneNameJSX, uniprotIDJSX variable'larına cevirip header'da gosteriyorum.

Tool secimi kısmı ise 'selectorProteinTools' variable'ı.

Tool değiştirirken, switchTool fonksiyonu kullanılıyor. Config'den gelen prediction tool parametresinin score ranges kısmında degisiklik yapılıyor anlık gösterilen protein sekansına göre:

- Aggregator seciliyorsa, bu protein için kaç tane tool olduguna gore score range'in sonu olacak değer belirtiliyor (tool sayısı).
- Provean için, deleterious range'inin minimumu ve benign range'inin maximumu belirtiliyor, ve skorların dağılımına göre her score range (deleterious, benign) icin, score range icindeki median skorun, score range'in başlangıç ve bitiş renginin %50 %50 karışımı, olacak sekilde renk geçişinin olmasını saglayacak, 'gradient_ratio' variable'ını belirtiyor (https://gka.github.io/chroma.js/#scale-domain).
- Geri Kalan tool'lar icin ise, score range'lerinin başlangıç ve bitiş değerleri önceden belirtilmiş olduğundan, o değerler değiştirilmiyor, 'get_new_score_ranges' fonksiyonu yardımıyla, her score range icin, eğer score range'in median skoru, score range'in en soldaki pozisyonunu 0, en sağdaki pozisyonunun 100 olarak düşünürsek, 25 ile 75 pozisyonları arasındaysa, median skor, başlangıç ve bitiş renklerinin %50 %50 karışımı olucak şekilde renk geçisinin olmasını sağlayacak 'gradient_ratio' variable'ının belirtiyor. Eğer, median değer 25 ile 75 pozisyonları arasında değilse, median değer sanki 25 ya da sanki 75 pozisyonundaymış diye varsayarak (hangisine daha yakınsa onu seçerek) 'gradient_ratio' variable'ı belirtiliyor.

Not: 'gradient_ratio' variable'ı color_lists_array' i yaratırken kullanılıyor. ChromaJs'in şu örneğindeki gibi (https://gka.github.io/chroma.js/#scale-domain), scale'in başlangıç, %50 %50 karışımı, ve bitiş renkleri, ve bunlara denk gelicek sayıları belirtiliyor. 'gradient_ratio' 0 ile 1 arasında değer alıp, ortadaki rengin pozisyonunu belirlenmesini sağlıyor (0 en sol, 1 en sağ).

Seçili olan 'prediction tool' değişince, o tool'un sonuçlarını ifade edicek olan renkleri, yani colro_lists_array'i , hesaplıyor (seçili olan tool'un her score range'i için 'number_of_colors' parametresine ve seçilen tool'un 'start_color', 'end_color' ve 'gradient_ratio' parametrelerine göre). Bu color_lists_array ColorRangesLegend ve Heatmap componentlerine prop olarak gönderiliyor.

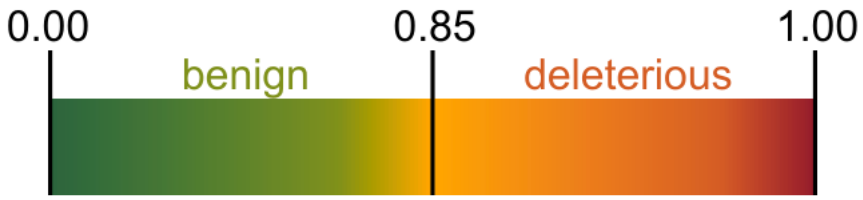
ColorRangesLegend, secili olan tool'un renklerinin ne ifade ettiğini gösteriyor.

Heatmap, secili tool'un datasına göre çiziliyor, aggregator secili ise butun tool'ların datasını kullanıyor.

Birden fazla metadata var ise ebi.ac.uk API'sinin sonucunda, Selector cıkıyor, ve altında Metadata'yı görselleştiren kulvarlar çiziliyor.

Heatmap ve Metadata componentlerinde, Zoom ve Pan değerleri aynı olması icin, Zoom ve Pan state'leri Protein Page sayfasında tutuluyor.

Color Ranges Legend:



Color Ranges Legend componenti heatmap'in renklerinin ne anlama geldiğini gösteren legend oluyor.

Input propları olarak Şu an seçili olan tool'un parametrelerini ve ProteinPage'de hesaplanmış, color_lists_array'i alıyor. HTML <canvas> kullanılarak color_lists_array üzerinden loop'lanılarak çiziliyor. Config dosyasından gelen her range için 'vw' ye göre belirlenen width'e göre size veriliyor, ve her ekran resize landığında tekrar çizdirmeye zorluyorum, event Listener ekleyerek. HTML Canvas kullanmak biraz karışık bir koda sebep verdi.

Heatmap:

Heatmap'in solundaki legend'i, heatmap'in üstündeki anlık gözüken pozisyonu gösteren şeridi, ve heatmap'in altındaki 'position average' şeridi bu component tarafından çiziliyor. Ayrıca Tooltip de bu component'in içinde çiziliyor. Bu component içinde gözüken her şey HTML Canvas kullanılarak çiziliyor.

Aldığı input proplar:

- anlık seçili olan tool,
- proteinin datası (seçili olan toola göre bütün data ProteinPage'de ayıklanılarak gönderiliyor), ColorRangesLegend'a gönderilen renkler,
- renk sayısı (artık buna ihtiyaç yok, çıkartılmalı),
- user tarafından ayarlanmış, anlık zoom ve pan değeri, ve bu değeri değiştirmek için fonksiyon

'DrawHeatmap2' fonksiyonu ana heatmap'ı ve alttaki 'position average' kısmını çiziyor, heatmap'te anlık gözüken aminoasit sayısı config dosyasından gelen 'heatmap_grid_draw_threshold'dan küçükse, hücreleri ayırt etmek kolaylaşsın diye grid görünümü verecek çizgiler de çiziliyor. Pozisyon averagelar cizildikten sonra en sonunda fonksiyonun, pozisyonlara gelen aminoasitlerin harfleri çiziliyor, eğer harfler taşmadan çizilebilicekse.

‘DrawAminoAcidLegend’ fonksiyonu heatmap’in solundaki amino asitlerin hangi satırlara denk geldiğini, ve ‘position average’ satırını belirten legend’ı çiziyor.

‘DrawHeatmapPositions’ fonksiyonu ana heatmap ile ‘position average’ kısmının arasındaki bir canvasa index’leri çiziyor, anlık gözüken amino asit sayısına göre de cetvel benzeri bir görünüm olacak şekilde çizim yapıyor.

DrawCurrentViewWindow, heatmap’ın üstündeki su anda gözüken kısım nereye denk geliyor bilgisini çiziyor

Performansı iyileştirmek için: şu an her zoom ya da görseli kaydırma işleminde canvas temizlenip tekrardan bütün gözüken amino asitler tekrardan çizildiği için, her pozisyona denk gelecek renkler useMemo ile cache’lenip heatmapColors variable’ında saklanıyor, HeatmapMeanColors variable’ı ise, Position average ve anlık gözüken kısmı gösteren heatmap’in üstündeki şeritteki renkler için. Çizim yapılırken Canvas üzerinde, fillColor’u değiştirme sayısını ve çizim işlemi sayısını minimize etmek için bitişik ve aynı renkte olan aminoasit hücrelerini tek seferde çizmeye çalışıyorum. Saniyede yapılabilecek maksimum zoom sayısını, 30 civarı bir sayıya kısıtlıyorum, ‘prevTime’ variable’ını kullanarak.

Aggregator ile çizerken bütün tool’lar için olan data’dan oluşuyor proteinData prop’u, ve aggregator seçiliyken hücrelerin renklerinin hesaplanması ve tooltip’in çizimi herhangi bir tool seçiliyken olana göre biraz daha farklı bir kod kullanıyor, available_tools_list variable’ı bu yüzden var.

‘Mouse Wheel’ eventinin listener’ını useEffect kullanarak vermemim sebebi, event.preventDefault()’un onWheel prop’uyla ekleyince çalışmaması. Ayrıca, maksimum zoom ya da minimum zoom değerlerine ulaşılırsa (yani mouse wheel eventi zoom in ya da zoom out ile sonuçlanmayacaksa), en son zoom eventinin üzerinden belirli bir süre geçtikten sonra ‘mouse wheel’ eventi engellenmiyor ve sayfa aşağı ya da yukarı kaydırılabiliyor.

Tekrar çizimler için de useEffect ile dependency array’e bakılarak yapılıyor, ama useEffect kaldırılabilir tekrar çizimler için.

‘calculateMedianOfPosition', ‘calculateRiskAssessment', ‘helperGetPositionScore' ve ‘helperGetPosition’ fonksiyonları helper fonksiyon olarak kullanılıyor, ve API’den dönen data’da bazen eksik değerler ya da NaN değerleri olabiliyor, o durumlara uygun calısıcak sekilde uygun fonksiyonlar.

Metadata Features Table:

Sırasıyla her feature kategorisi icin metadata feature lane component çağırılarak gösteriliyor ve metadata feature lane component'ine kullancağı renk paleti veriliyor (Config dosyasına renk paletleri array'i oluşturdum.).

Tooltip çiziliyor.

Metadata Feature Lane:

Yine Canvas kullanarak, Metadata'daki feature'ları gösteriyorum, yukarıdaki heatmap'le aynı pozisyon denk gelecek sekilde. Feature'ların üstüne tıklandığında tooltip acılıyor ve eğer feature çok kucuk gozukuyorsa tooltip feature'ın yakınına tıklanınca da acılıyor.

Burada da heatmap'teki gibi zoom özelliği var, heatmau'le senkronize calışıyor.

Çizime baslamdan once kaç sub lane (yani çakışan kaç tane feature oldugunu) hesaplıyorum, burada memoize etmedim bu fonksiyonun sonucunu, performans icin memoize edilebilir ama cunku değişmiyor hiç bir zaman değer. Bu hesabı yaparken hangi feature'ın hangi sub lane'de çizileceğini de o an hesaplayıp extendedFeaturesArray diye bir variable oluşuyorum ki çizerken kolaylık sağlasın

Sorunlar / Eksikler / Öneriler:

- ## Genel
  - Camel case, Snake case karışık, standartlaştırmak daha iyi olabilir.,
  - HTML vs CSS kodları biraz çirkin, gözden geçirilebilir.
  - MUI gibi bir library kullanılabilir.
  - Bazı commentler, kod değiştikten sonra da silinmediği için, gerceği yansıtmıyor hepsi. Bazıları da gereksiz.
  - Github vulnerabilities var diye uyarıya kullanılan package'larda, o konuda bir aksiyon almak lazım.
  - A route for error, if the url doesn't match any routes.
- ## HomePage:
  - Return kısmındaki JSX
- ## SearchBox:
  - Fasta format parserından (helper_protein_sequence_trimmer) emin degilim, fasta format icin butun case'leri kapsamıyor olabilir, hazır bir fasta format parserı kullanmadım, gördüğüm örneklerdeki case'leri kapsayıcak bir sekilde yazdım,
  - Examples kısmında Fasta Format örneği eksik, config.js'in en altındaki examples_lookup' a Fasta Format örneği eklenmeli
  - Search tuşuna tıklandığında MD5Sum veya Fasta kullanılarak aratırken, API'den cevap geç gelirse, network request'in cevabının beklendiği belli olmuyor, sanki tuş hiçbir şey yapmamış gibi gözüküyor.
  - MD5Sum'a çevirirken CryptoJS library'si yerine baska bir şey kullanılabilir.
  - Gene ID search'ünde direkt bir proteine gitmek yerine GeneID'ye uyan bütün sekansları listelemek
  - Uniprot protein ID ve Gene ID ile aratırken user inputuna auto complete yapmak
- ## ProteinPage:
  - Number of colors parameter doesn't need to be passed onto heatmap component, as it can be imported from config file
- ## ColorRangesLegend:
  - Html canvas yerine CSS kullanılarak aynı gradient yaratılırsa, daha sade bir koda sahip olur.

- **Heatmap:**
  - Yüksek sayıda aminoasit içeren protein'lerde performans sorunları oluyor. HTML Canvas kullanarak her değişiklikte tekrardan çizmek dışında bir çözüm'e gitmek gerekebilir (Projenin sonlarında farkettiğim için alternatifleri deneyecek zamanım kalmamıştı).
  - Bazı font değerleri browser'ın height ına göre çiziliyor, telefonlarda bu sorun olabilir.
  - Mobile ekranlarda da kullanılabilir olmasını istiyorsak icin degisiklikler gerekebilir. Belki MUI kullanmak bu konuda yardımcı olabilir.
  - UseEffect kullanılarak tekrar çizme kısmı değiştirilebilir, useEffect'e gerek yok gibi gözüküyor, her re-renderladığında bu component tekrar çizilebilir. setTooltip tekrar tekrar render yaratmasın diye bir önlem almak gerekir ama.
  - Sürekli useCallback ile memoize lanan fonksiyonlar, dışarı taşınırsa componenten, belki daha temiz durabilir kod.
  - Canvas'lara width, ve height verme ve CSS'ler biraz karışık.
- **MetadataFeatureTable:**
  - Metadata daha iyi parselanabilir. Ve daha güzel çizilebilir tooltip
- **MetadataFeatureLane:**
  - Burdaki wheelzoom ve pan fonksiyonları aslında heatmap'tekilerle aynı ya da neredeyse aynı, code duplication olmuş oldu burda tekrar tanımlanarak, ortak bi yerde tanımlayıp oradan import edilse hem kod daha temiz hem de daha maintainable olur.
  - Feature çok kucuk gozukuyosa en yakını secme logici belki değişebilir.
  - Art arda olan feature'ların korunumu bitişik olabiliyor, bu yüzden ayırt etmek zor olabiliyor. Bunların arasına 2 tane ayrı feature oldugunu belirten bir ayıraç eklenebilir.