Webinar_Example1

Introduction:

We are not magicians. A working example to best illustrate what an API is and how we can 'talk' to a server. In our case we are talking with the cBIO web interface. Let's try to pull data from a specific study and redo the figures which pop up when we are online.

Why should one ever use an API when we basically get all the information digested online in a pretty nice format?

Well the answer is easy :) 1. We save time (compare to searching a study online, pressing download button, importing in any IDE, etc.) 2. Flexibility (Fetched data can be further analyzed in any conceivable way) 3. Automate processess (We can fetch data automatically at any time and directly implement data in processing pipelines) 4. Avoiding frustration with data formats (Fetched data is ready to use in R)

Example 1: Get data from Prostate Adenocarcinoma (TCGA, Cell 2015)

Requirements: load neccessary packages and create API object

```
## load packages:
library(cBioPortalData)
library(httr)
library(dplyr)
library(stringr)
library(biomaRt)
library(ggplot2)

## GENERAL: create the actual API
cbio <- cBioPortal()
```

The central object is the **cbio** object:

Now, let's have a look which studies are available, and choose the desired

```
studies = as.data.frame(getStudies(cBioPortal())) # getStudies is a convenient shortcut;
head(studies) # print the top rows and check all the studies
```

```
##                                                                    name
## 1 Cholangiocarcinoma (National Cancer Centre of Singapore, Nat Genet 2013)
## 2                   Cutaneous T Cell Lymphoma (Columbia U, Nat Genet 2015)
## 3                Esophageal Squamous Cell Carcinoma (UCLA, Nat Genet 2014)
## 4           Oral Squamous Cell Carcinoma (MD Anderson, Cancer Discov 2013)
## 5                       Hepatocellular Carcinomas (INSERM, Nat Genet 2015)
## 6                                  Uveal Melanoma (QIMR, Oncotarget 2016)
```

```
##                   shortName
## 1 Cholangiocarcinoma (NCCS)
## 2      CTCL (Columbia 2015)
## 3          ESCC (UCLA 2014)
## 4         Head & neck (MDA)
## 5        HCC (Inserm, 2015)
## 6                 UM (QIMR)
##
## 1
## 2 Whole-Exome Sequencing (WXS) of tumor-normal sample pairs from 25 patients with Sezary Syndrome and
## 3                                  Whole exome sequencing (WXS) or targeted deep sequencing (TDS)
## 4                                                                        Comprehensive profiling of
## 5                                                                                             Whole
## 6                                                                  Whole-genome or whole-exome sequer
##   publicStudy     pmid                                citation groups status
## 1        TRUE 24185513          Chan-on et al. Nat Genet 2013 PUBLIC      0
## 2        TRUE 26551667 Da Silva Almeida et al. Nat Genet 2015             0
## 3        TRUE 24686850              Lin et al. Nat Genet 2014 PUBLIC      0
## 4        TRUE 23619168    Pickering et al. Cancer Discov 2013             0
## 5        TRUE 25822088          Schulze et al. Nat Genet 2013 PUBLIC      0
## 6        TRUE 26683228       Johansson et al. Oncotarget 2016 PUBLIC      0
##            importDate allSampleCount              studyId cancerTypeId
## 1 2019-02-14 00:00:00             15       chol_nccs_2013         chol
## 2 2019-02-15 00:00:00             43   ctcl_columbia_2015          nhl
## 3 2019-02-19 00:00:00            139       escc_ucla_2014         escc
## 4 2019-02-19 00:00:00             40 hnsc_mdanderson_2013         hnsc
## 5 2019-02-19 00:00:00            243   hcc_inserm_fr_2015          hcc
## 6 2019-02-19 00:00:00             28         um_qimr_2016           um
##   referenceGenome
## 1            hg19
## 2            hg19
## 3            hg19
## 4            hg19
## 5            hg19
## 6            hg19
```

```
## now let's refine our search to list all prostate cancer studies available
studies[grep('Prostate.*', studies$name), c('name', 'studyId')] # list all Prostate cancer studies with
```

```
##                                                                name
## 93               Metastatic Prostate Adenocarcinoma (MCTP, Nature 2012)
## 97                         Prostate Adenocarcinoma (MSKCC, PNAS 2014)
## 98                      Prostate Cancer (MSKCC, JCO Precis Oncol 2017)
## 127                 Prostate Adenocarcinoma (Broad/Cornell, Cell 2013)
## 144                                    Prostate Cancer (MSK, 2019)
## 146                   Prostate Adenocarcinoma (TCGA, Firehose Legacy)
## 157             Prostate Adenocarcinoma (Broad/Cornell, Nat Genet 2012)
## 158      Neuroendocrine Prostate Cancer (Multi-Institute, Nat Med 2016)
## 170          Prostate Adenocarcinoma (MSKCC/DFCI, Nature Genetics 2018)
## 185                   Prostate Adenocarcinoma (MSKCC, Cancer Cell 2010)
## 186         Prostate Adenocarcinoma (Fred Hutchinson CRC, Nat Med 2016)
## 190                      Prostate Adenocarcinoma (SMMU, Eur Urol 2017)
## 199                     Prostate Adenocarcinoma (CPC-GENE, Nature 2017)
## 209 The Metastatic Prostate Cancer Project (Provisional, November 2019)
```

```
## 241                     Prostate Adenocarcinoma (TCGA, PanCancer Atlas)
## 272         Metastatic Prostate Cancer (SU2C/PCF Dream Team, Cell 2015)
## 273                           Prostate Adenocarcinoma (TCGA, Cell 2015)
## 274 Metastatic Prostate Adenocarcinoma (SU2C/PCF Dream Team, PNAS 2019)
## 275                            Prostate Cancer (DKFZ, Cancer Cell 2018)
## 276                Prostate Adenocarcinoma Organoids (MSKCC, Cell 2014)
## 283                        Prostate Adenocarcinoma (MSK, Eur Urol 2020)
##                               studyId
## 93                           prad_mich
## 97                     prad_mskcc_2014
## 98                     prad_mskcc_2017
## 127                    prad_broad_2013
## 144                      prad_msk_2019
## 146                          prad_tcga
## 157                        prad_broad
## 158                     nepc_wcm_2016
## 170                        prad_p1000
## 185                        prad_mskcc
## 186                        prad_fhcrc
## 190                  prad_eururol_2017
## 199                     prad_cpcg_2017
## 209               prad_mpcproject_2018
## 241     prad_tcga_pan_can_atlas_2018
## 272                    prad_su2c_2015
## 273                     prad_tcga_pub
## 274                    prad_su2c_2019
## 275                 prostate_dkfz_2018
## 276 prad_mskcc_cheny1_organoids_2014
## 283              prad_cdk12_mskcc_2020
```

```
## The 'studyId' column will be our key to pull some data
## We go for Prostate Adenocarcinoma (TCGA, Cell 2015); prad_tcga_pub
```

Now we look into study participants; get all the ID's from this particular study First, we look into every single step; then we introduce a shortcut (convenient function)

```
## list all study (ID) participants
all.patients = cbio$getAllPatientsInStudyUsingGET(studyId = 'prad_tcga_pub')
all.patients = httr::content(all.patients, as = 'parsed') ## parsed might be the most suitable argument
patients_dataframe = data.frame(matrix(
  unlist(all.patients),
  nrow = length(all.patients),
  byrow = T
)) ## convert to data frame; easier downstream handling
colnames(patients_dataframe) = c('Identifier', 'Patient_ID', 'Study_ID')

head(patients_dataframe) ## Again print the first rows
```

```
##                                Identifier      Patient_ID      Study_ID
## 1 VENHQS1ISSO3MTY5LTAxOnByYWRfdGNnYV9wdWI TCGA-HI-7169-01 prad_tcga_pub
## 2     VENHQS1ISSO3MTY5OnByYWRfdGNnYV9wdWI    TCGA-HI-7169 prad_tcga_pub
## 3     VENHQS1FSi01NTAyOnByYWRfdGNnYV9wdWI    TCGA-EJ-5502 prad_tcga_pub
## 4     VENHQS1IQy03MjA5OnByYWRfdGNnYV9wdWI    TCGA-HC-7209 prad_tcga_pub
```

```
## 5     VENHQS1IQy03NzQ4OnByYWRfdGNnYV9wdWI    TCGA-HC-7748 prad_tcga_pub
## 6     VENHQS1KNC1BODNOOnByYWRfdGNnYV9wdWI    TCGA-J4-A83N prad_tcga_pub
```

```
################################################################################
## here is a convenient shortcut, which basically do all the steps above in one line
all_samples = allSamples(cbio, studyId = 'prad_tcga_pub')
head(all_samples)
```

```
## # A tibble: 6 x 6
##   uniqueSampleKey    uniquePatientKey    sampleType   sampleId patientId studyId
##   <chr>              <chr>               <chr>        <chr>    <chr>     <chr>
## 1 VENHQS1ISS03MTY5L~ VENHQS1ISS03MTY5On~ Primary Sol~ TCGA-HI~ TCGA-HI-~ prad_t~
## 2 VENHQS1FSi01NTAyL~ VENHQS1FSi01NTAyOn~ Primary Sol~ TCGA-EJ~ TCGA-EJ-~ prad_t~
## 3 VENHQS1IQy03MjA5L~ VENHQS1IQy03MjA5On~ Primary Sol~ TCGA-HC~ TCGA-HC-~ prad_t~
## 4 VENHQS1IQy03NzQ4L~ VENHQS1IQy03NzQ4On~ Primary Sol~ TCGA-HC~ TCGA-HC-~ prad_t~
## 5 VENHQS1KNC1BODNOL~ VENHQS1KNC1BODNOOn~ Primary Sol~ TCGA-J4~ TCGA-J4-~ prad_t~
## 6 VENHQS0yQS1BOFZWL~ VENHQS0yQS1BOFZWOn~ Primary Sol~ TCGA-2A~ TCGA-2A-~ prad_t~
```

Now we look into clinical attributes (data);

We can fetch clinical data for one specific patient, or for a whole study. Let's start with one particular study and compare the output to the cbio portal web interface

```
## remember: we fetch all the data from our mother API cbio
## first we look at one particular patient:
patient.x.clinics = cbio$getAllClinicalDataOfPatientInStudyUsingGET(studyId = 'prad_tcga_pub', ## selec
                                                                    patientId = 'TCGA-VP-A87C') ## sele
patient.x.clinics = httr::content(patient.x.clinics)
patient.x.clinics = data.frame(matrix(
  unlist(patient.x.clinics),
  nrow = length(patient.x.clinics),
  byrow = T)) ## convert again, easier handling

patient.x.clinics$X1 = NULL ## delete first (unesseccary) column
colnames(patient.x.clinics) = c('Patient_ID', 'Study_ID', 'Attribute', 'Value') # change colnames
head(patient.x.clinics) ## have a look and compare to cBIO portal format (under clinics)
```

```
##     Patient_ID      Study_ID  Attribute                     Value
## 1 TCGA-VP-A87C prad_tcga_pub        AGE                        67
## 2 TCGA-VP-A87C prad_tcga_pub  BRCA1_CNA                   hetloss
## 3 TCGA-VP-A87C prad_tcga_pub CDKN1B_MUT                         0
## 4 TCGA-VP-A87C prad_tcga_pub ERG_STATUS                    fusion
## 5 TCGA-VP-A87C prad_tcga_pub       RACE BLACK OR AFRICAN AMERICAN
## 6 TCGA-VP-A87C prad_tcga_pub    RB1_MUT                         0
```

```
## now we can take a closer look to a specific sample
## in this case we use the same patient and sample '-01'
patient.x.sample = cbio$getAllClinicalDataOfSampleInStudyUsingGET(studyId = 'prad_tcga_pub',
                                                                  sampleId = 'TCGA-VP-A87C-01') ## comp
patient.x.sample = httr::content(patient.x.sample)
patient.x.sample = data.frame(matrix(
  unlist(patient.x.sample),
```

```
  nrow = length(patient.x.sample),
  byrow = T)) ## convert again, easier handling

patient.x.sample$X1 = NULL
patient.x.sample$X2 = NULL
patient.x.sample$X3 = NULL

colnames(patient.x.sample) = c('Patient_ID', 'Study_ID', 'Attribute', 'Value')
head(patient.x.sample) ## print again and compare to cBIOportal output
```

```
##     Patient_ID      Study_ID                 Attribute  Value
## 1 TCGA-VP-A87C prad_tcga_pub   ABSOLUTE_EXTRACT_PLOIDY   1.93
## 2 TCGA-VP-A87C prad_tcga_pub   ABSOLUTE_EXTRACT_PURITY    0.4
## 3 TCGA-VP-A87C prad_tcga_pub ABSOLUTE_GENOME_DOUBLINGS      0
## 4 TCGA-VP-A87C prad_tcga_pub             AKT1_MUTATION      0
## 5 TCGA-VP-A87C prad_tcga_pub                  AR_MRNA 548.02
## 6 TCGA-VP-A87C prad_tcga_pub                 AR_SCORE -10.12
```

```
## compare the output with the patient/sample view in cBIO online
## we are no magicians :)
```

And now we want to retrieve all the clinical data from all study participants

```
## get all the clinical data for all patients
all.clinics = cbio$getAllClinicalDataInStudyUsingGET(studyId = 'prad_tcga_pub')
all.clinics = httr::content(all.clinics)
all.clinics = data.frame(matrix(
  unlist(all.clinics),
  nrow = length(all.clinics),
  byrow = T)) ## convert again, easier handling

all.clinics[, c(1, 2, 3)] = NULL
head(all.clinics) # now we have one big data frame where clinical attributes are listed according to sa
```

```
##             X4            X5                   X6                      X7
## 1 TCGA-HI-7169 prad_tcga_pub          CANCER_TYPE         Prostate Cancer
## 2 TCGA-HI-7169 prad_tcga_pub CANCER_TYPE_DETAILED Prostate Adenocarcinoma
## 3 TCGA-HI-7169 prad_tcga_pub        ONCOTREE_CODE                    PRAD
## 4 TCGA-HI-7169 prad_tcga_pub          SAMPLE_TYPE                 Primary
## 5 TCGA-EJ-5502 prad_tcga_pub        AKT1_MUTATION                       0
## 6 TCGA-EJ-5502 prad_tcga_pub              AR_MRNA                 1021.46
```

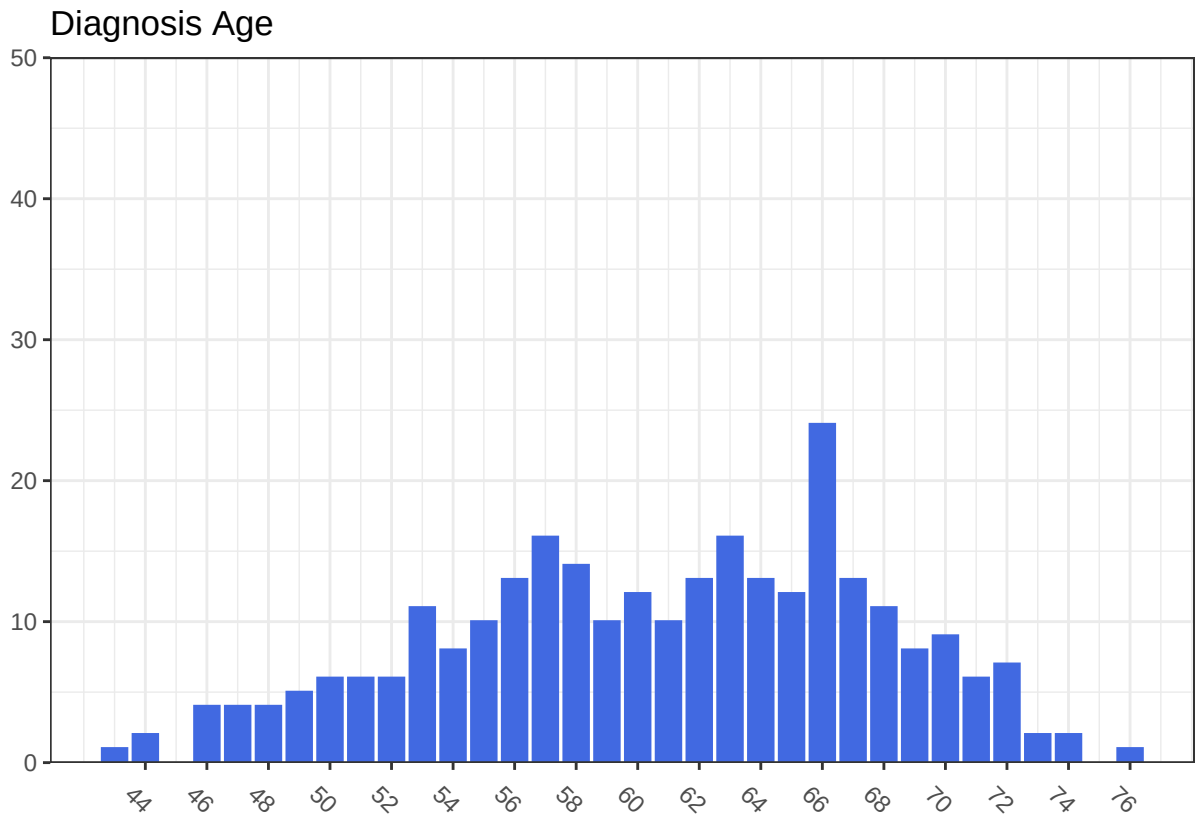
```
################################################################################
## here is a shortcut, which basically summarizes all the efforts above in one single line
all_clinical_data = clinicalData(cbio, studyId = 'prad_tcga_pub')
head(all_clinical_data)
```

```
## # A tibble: 6 x 14
##   uniquePatientKey patientId studyId AGE   BRCA1_CNA CDKN1B_MUT ERG_STATUS
```

```
##   <chr>            <chr>     <chr>   <chr> <chr>     <chr>      <chr>
## 1 VENHQS0yQS1BOFc~ TCGA-2A-~ prad_t~ 54    hetloss   0          none
## 2 VENHQS0yQS1BOFc~ TCGA-2A-~ prad_t~ 69    diploid   0          none
## 3 VENHQS0yQS1BOFZ~ TCGA-2A-~ prad_t~ 51    hetloss   0          fusion
## 4 VENHQS0yQS1BOFZ~ TCGA-2A-~ prad_t~ 57    diploid   0          none
## 5 VENHQS0yQS1BOFZ~ TCGA-2A-~ prad_t~ <NA>  hetloss   0          fusion
## 6 VENHQS0yQS1BOFZ~ TCGA-2A-~ prad_t~ 52    diploid   0          fusion
## # ... with 7 more variables: PREOPERATIVE_PSA <chr>, RACE <chr>, RB1_MUT <chr>,
## #   RESIDUAL_TUMOR <chr>, SAMPLE_COUNT <chr>, SUBTYPE <chr>, ZMYM3_MUT <chr>
```

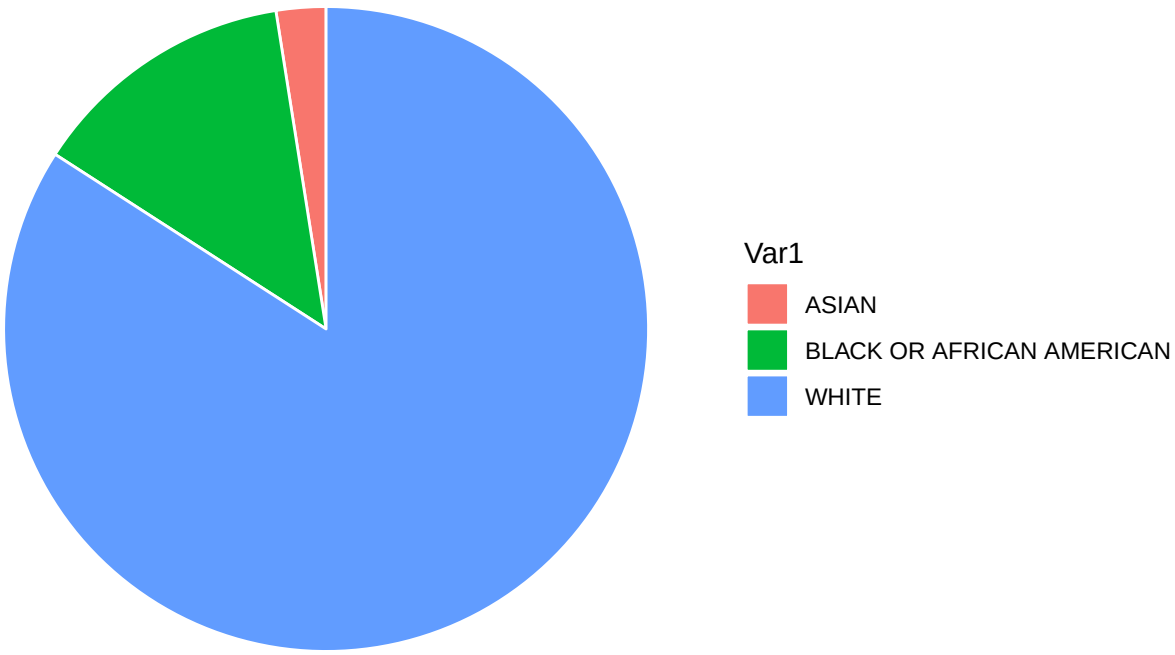
Make some metadata information plots; which we already know from cBIO

```
## Just plot Age at diagnosis
ggplot() +
  geom_bar(
    data = all_clinical_data,
    aes(x = as.numeric(AGE)),
    color = 'royalblue',
    fill = 'royalblue',
    na.rm = T,
    width = 0.8
  ) +
  theme_bw() +
  scale_x_continuous(breaks = seq(44, 76, 2)) +
  scale_y_continuous(expand = c(0, 0), limits = c(0, 50)) +
  theme(axis.text.x = element_text(angle = -45)) +
  labs(x = '', y = '', title = 'Diagnosis Age')
```




```
## race category
race = as.data.frame(table(all_clinical_data$RACE))

ggplot(race, aes(x = "", y = Freq, fill = Var1)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) + theme_void()
```



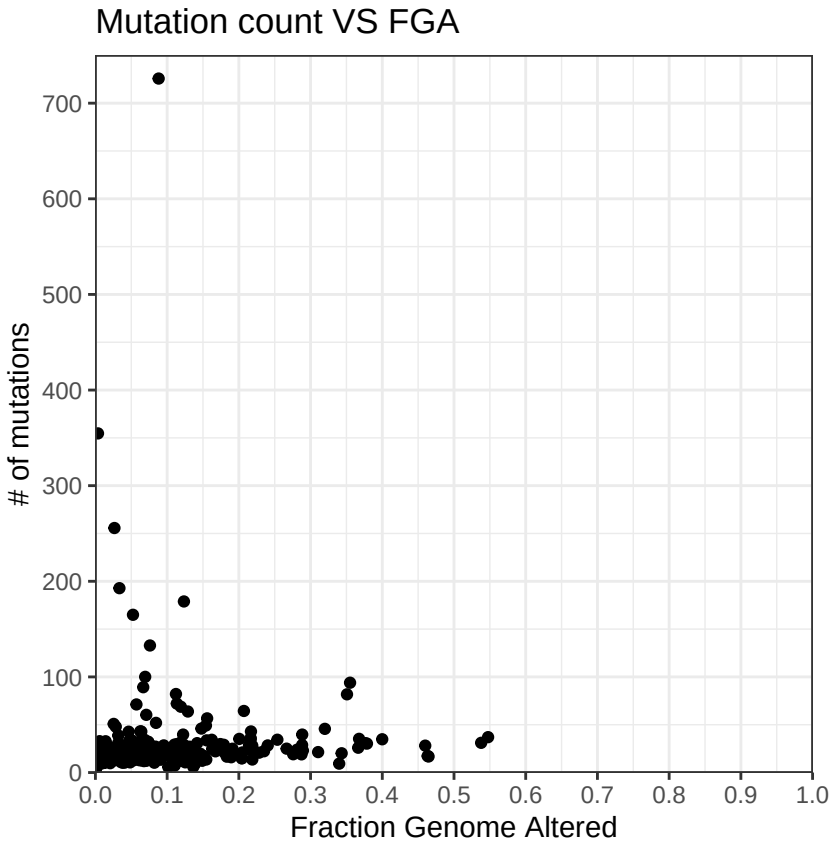
Let us reproduce some figures with the clinical data we just pulled from the API

```
## all_clinics
## look at number of mutations vs fraction genome altered:
FGA = all.clinics[all.clinics$X6 == 'FRACTION_GENOME_ALTERED', ] ## subset dataframe
mutation.count = all.clinics[all.clinics$X6 == 'MUTATION_COUNT', ] ## subset dataframe

data.figure1 = merge(FGA[,c(1, 4)],
                     mutation.count[,c(1,4)],
                     by.x = 'X4',
                     by.y = 'X4',
                     all = T) ## merge the two data frame to prepare for plot
colnames(data.figure1) = c('patient', 'FGA', 'mutation.count')


## now let's plot, using the data modified with ggplot
ggplot(data.figure1, aes(x = as.numeric(FGA),
                         y = as.numeric(mutation.count))) +
  geom_jitter() +
  theme_bw() +
  theme(legend.position = 'none', aspect.ratio = 1) +
  scale_y_continuous(limits = c(0, 750), expand = c(0,0), breaks = seq(0, 800, by = 100)) +
  scale_x_continuous(limits = c(0, 1), expand = c(0, 0), breaks = seq(0, 1, by = 0.1)) +
  labs(x = 'Fraction Genome Altered', y = '# of mutations', title = 'Mutation count VS FGA')
```

```
## Warning: Removed 12 rows containing missing values (geom_point).
```



Turning into molecular data now

Which 'measurements' are actually available and what can we retrieve?

```
genomic_parameters = molecularProfiles(cbio,
                                       studyId = 'prad_tcga_pub',
                                       projection = c('SUMMARY')) ## SUMMARY is most basic (but) compre

head(genomic_parameters)
```

```
## # A tibble: 6 x 7
##   molecularAltera~ datatype name  description showProfileInAn~ molecularProfil~
##   <chr>            <chr>    <chr> <chr>       <lgl>            <chr>
## 1 PROTEIN_LEVEL    LOG2-VA~ Prot~ Protein ex~ FALSE            prad_tcga_pub_r~
## 2 COPY_NUMBER_ALT~ DISCRETE Puta~ Putative c~ TRUE             prad_tcga_pub_g~
## 3 MRNA_EXPRESSION  CONTINU~ mRNA~ mRNA gene ~ FALSE            prad_tcga_pub_r~
## 4 MRNA_EXPRESSION  Z-SCORE  mRNA~ mRNA z-Sco~ TRUE             prad_tcga_pub_r~
## 5 COPY_NUMBER_ALT~ CONTINU~ Rela~ Relative l~ FALSE            prad_tcga_pub_l~
## 6 METHYLATION      CONTINU~ Meth~ Methylatio~ FALSE            prad_tcga_pub_m~
## # ... with 1 more variable: studyId <chr>
```

```
## now we see that we have 9 different measurements on this cohort (Protein, mutation, methylations, et
## quite a lot :)
## now we can retrieve specific datasets
```

Now, we get a little more concise: We want to retrieve mutational data; for this purpose we need a new 'key' column; called 'molecularProfileId'. Let's look which options we have

```
print(genomic_parameters[, 'molecularProfileId'])
```

```
## # A tibble: 9 x 1
##   molecularProfileId
##   <chr>
## 1 prad_tcga_pub_rppa
## 2 prad_tcga_pub_gistic
## 3 prad_tcga_pub_rna_seq_v2_mrna
## 4 prad_tcga_pub_rna_seq_v2_mrna_median_Zscores
## 5 prad_tcga_pub_linear_CNA
## 6 prad_tcga_pub_methylation_hm450
## 7 prad_tcga_pub_mutations
## 8 prad_tcga_pub_fusion
## 9 prad_tcga_pub_rna_seq_v2_mrna_median_all_sample_Zscores
```

we start by looking into one specific patient and retrieve the mutational spectrum

```
## now let's recall mutations; starting with a specific patient

## we need a little diversion here; in order to obtain all mutations we first need to get all entrez_ge
## for this purpose we use the bioMaRt package
mart.object <- useMart("ensembl",
                dataset = "hsapiens_gene_ensembl") # we are searching for human data

genes_ids <- getBM(mart = mart.object,
                   attributes = c("hgnc_symbol", "entrezgene_id"))
## make a vector which we will use later
genes_ids = unique(as.vector(genes_ids$entrezgene_id), na.rm = T)

all.patient.mutations = molecularData(
  api = cbio,
  molecularProfileId = 'prad_tcga_pub_mutations', ## check out the identifier here (recall from functio
  entrezGeneIds = genes_ids,
  sampleIds = paste0(patients_dataframe$Patient_ID, '-01'),
  check = T
)

head(all.patient.mutations) ## now let's look into this patient
```

```
## # A tibble: 6 x 36
##   uniqueSampleKey uniquePatientKey molecularProfil~ sampleId patientId
##   <chr>           <chr>            <chr>            <chr>    <chr>
## 1 VENHQS1DSC01Nz~ VENHQS1DSC01Nzk~ prad_tcga_pub_m~ TCGA-CH~ TCGA-CH-~
## 2 VENHQS1FSi01NT~ VENHQS1FSi01NTE~ prad_tcga_pub_m~ TCGA-EJ~ TCGA-EJ-~
## 3 VENHQS1FSi01NT~ VENHQS1FSi01NTE~ prad_tcga_pub_m~ TCGA-EJ~ TCGA-EJ-~
```

```
## 4 VENHQS1FSi03Nz~ VENHQS1FSi03Nzg~ prad_tcga_pub_m~ TCGA-EJ~ TCGA-EJ-~
## 5 VENHQS1HOS02Mz~ VENHQS1HOS02MzY~ prad_tcga_pub_m~ TCGA-G9~ TCGA-G9-~
## 6 VENHQS1KNC04Mj~ VENHQS1KNC04MjA~ prad_tcga_pub_m~ TCGA-J4~ TCGA-J4-~
## # ... with 31 more variables: entrezGeneId <int>, studyId <chr>, center <chr>,
## #   mutationStatus <chr>, validationStatus <chr>, tumorAltCount <int>,
## #   tumorRefCount <int>, normalAltCount <int>, normalRefCount <int>,
## #   startPosition <int>, endPosition <int>, referenceAllele <chr>,
## #   proteinChange <chr>, mutationType <chr>, functionalImpactScore <chr>,
## #   fisValue <dbl>, linkXvar <chr>, linkPdb <chr>, linkMsa <chr>,
## #   ncbiBuild <chr>, variantType <chr>, keyword <chr>, driverFilter <chr>,
## #   driverFilterAnnotation <chr>, driverTiersFilter <chr>,
## #   driverTiersFilterAnnotation <chr>, chr <chr>, variantAllele <chr>,
## #   refseqMrnaId <chr>, proteinPosStart <int>, proteinPosEnd <int>
```

```
## now we look into one specific patient:
all.patient.mutations[all.patient.mutations$patientId == 'TCGA-VP-A87C', ]
```

```
## # A tibble: 18 x 36
##    uniqueSampleKey uniquePatientKey molecularProfil~ sampleId patientId
##    <chr>           <chr>            <chr>            <chr>    <chr>
##  1 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
##  2 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
##  3 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
##  4 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
##  5 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
##  6 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
##  7 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
##  8 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
##  9 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## 10 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## 11 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## 12 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## 13 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## 14 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## 15 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## 16 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## 17 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## 18 VENHQS1WUC1BOD~ VENHQS1WUC1BODd~ prad_tcga_pub_m~ TCGA-VP~ TCGA-VP-~
## # ... with 31 more variables: entrezGeneId <int>, studyId <chr>, center <chr>,
## #   mutationStatus <chr>, validationStatus <chr>, tumorAltCount <int>,
## #   tumorRefCount <int>, normalAltCount <int>, normalRefCount <int>,
## #   startPosition <int>, endPosition <int>, referenceAllele <chr>,
## #   proteinChange <chr>, mutationType <chr>, functionalImpactScore <chr>,
## #   fisValue <dbl>, linkXvar <chr>, linkPdb <chr>, linkMsa <chr>,
## #   ncbiBuild <chr>, variantType <chr>, keyword <chr>, driverFilter <chr>,
## #   driverFilterAnnotation <chr>, driverTiersFilter <chr>,
## #   driverTiersFilterAnnotation <chr>, chr <chr>, variantAllele <chr>,
## #   refseqMrnaId <chr>, proteinPosStart <int>, proteinPosEnd <int>
```